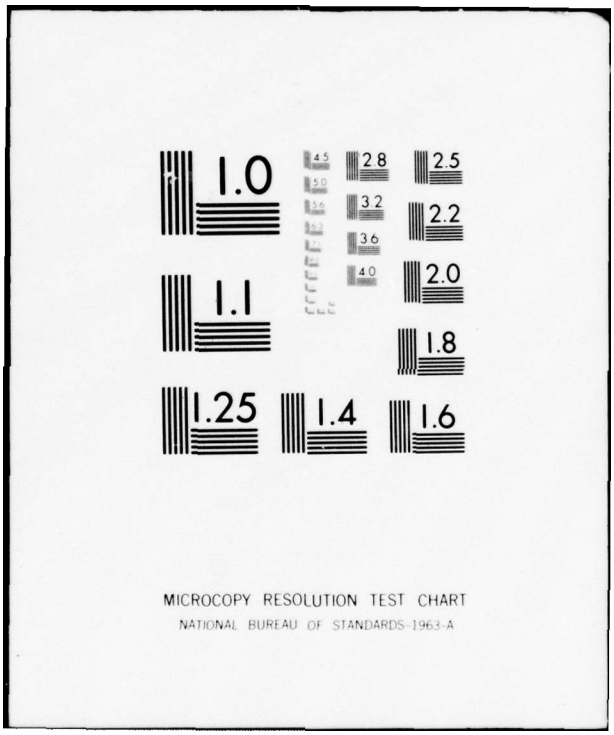
AD-A037 436 WHARTON SCHOOL OF FINANCE AND COMMERCE PHILADELPHIA P--ETC F/G 9/2
WAND USER'S GUIDE.(U)
APR 76 R GERRITSEN, R CORTES, J RIBEIRO N00014-75-C-0462
UNCLASSIFIED 76-01-03 NL

| OF |
AD
A037436

END

DATE
FILMED
5-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)
B.S.

WAND USER'S GUIDE

Rob Gerritsen,
Ricardo Cortes,
Jim Ribeiro,
and Ruth Zowader

76-01-03

Department of Decision Sciences

The Wharton School

University of Pennsylvania

Draft #3

April 15, 1976

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 76-01-03
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*: WAND User's Guide
5. TYPE OF REPORT & PERIOD COVERED: Final Rept.
6. PERFORMING ORG. REPORT NUMBER: 76-01-03

7. AUTHOR(s): Rob Gerritsen, Ricardo Cortes, Jim Ribeiro, Ruth Zowader
8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0462

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Decision Sciences Department
University of PA/Wharton School
Philadelphia, PA 19104

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
15 Apr 76
Technical report.

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Information Systems
Arlington, Virginia 22217

12. REPORT DATE: 4/76
13. NUMBER OF PAGES: 47 (32) 40p.

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Unlimited

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

408 757

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Network structures
Plex structures
DBTG
DBMS

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
This guide presents the basic concepts of WAND, a Database Management System. The first section presents the major concepts and terminology of WAND. It also introduces the Data Description Language (DDL) and the Data Manipulation Language (DML). The second section describes in detail the DDL and the procedures used to create the schema. The third section contains the DML statements and several examples illustrating their use.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

PREFACE

The WAND (Wharton Alerting Network Database) system is a partial implementation of the DBTG (Data Base Task Group) April 1971 Report. The implementation was begun as a class project of IS202 when I taught it in the spring of 1975. All students worked hard, but the special contributions made by Don Siever, John Schniepp, Bradley Ross, and Ruth Zowader should be noted. Other students that participated were: Louis DeBasio, Fulvio Flori, Eric Foradori, Bruno Gargiulo, Julie Gerdts, Petro Kozak, Michael Veale and Barry Weinstein.

Ruth Zowader continued with the project, turning it into the product it now is. Those who know the difference between "project" and "product" appreciate the contribution of Ms. Zowader.

An initial version of this user's guide was also prepared as a class project, in Tom Johnson's IS201, by Ricardo Cortes and Jim Ribeiro.

This research was supported in part by the Office of Naval Research under Contract No. N00014-75-C-0462.

I am indeed grateful to all of the people who helped make this a success.

Rob Gerritsen

Table of Contents

1.0 INTRODUCTION

This guide presents the basic concepts of WAND, a Data Base Management System (DBMS), and the statements available in Version A.04 of the system. It is divided into three sections.

The first section presents the major concepts and terminology of WAND. It also introduces the Data Description Language (DDL), which is used for describing the database, and the Data Manipulation Language (DML), which acts as an interface between the user and the Data Base Management System. The second section describes in detail the DDL and the procedures used to create the schema. The third section contains the DML statements and several examples illustrating their use. There are also seven Appendixes which present in handy reference format the DDL sentences, DML statements, error diagnostics, descriptions of the three utility routines DBINIT, DBDUMP, and DBLOOK, and a list of reserved words.

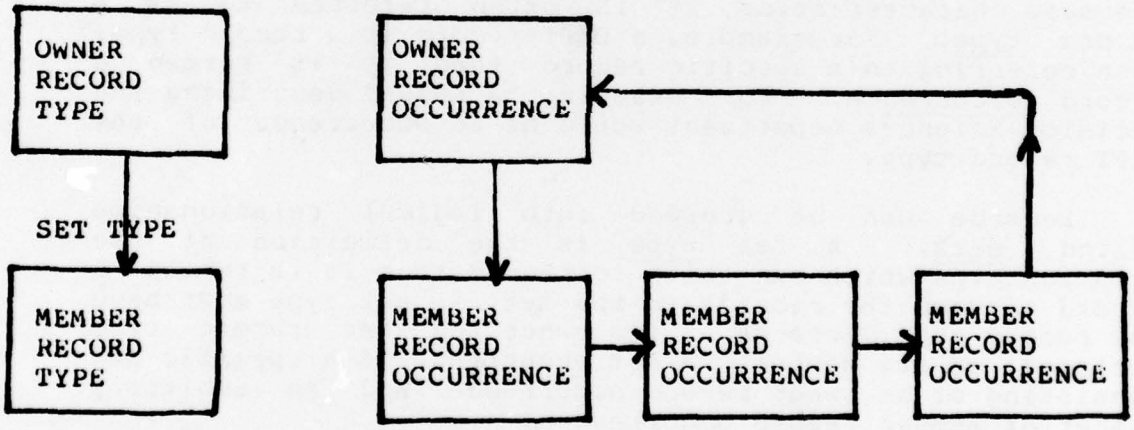


Figure 1. DS diagram and a corresponding set occurrence implemented as a chain.

2.0 MAJOR CONCEPTS

The WAND system consists of a compiler, some utility programs, and a set of routines (callable from COBOL, FORTRAN, and MACRO programs) which provide facilities for the management of databases, including database design, update, retrieval, and attendant functions. The conceptual design of this system is based on the design proposed by CODASYL'S Data Base Task Group in their 1971 report[1]. The following sections introduce the major concepts of the WAND system.

2.1 Records And Sets

A record is the largest physically contiguous entity within the database. It is a collection of data-items, which are the smallest units of named data in the database. When a record defines an arbitrary number of records with the same characteristics, it is often referred to as a record type. For example, a DEPT record is a record type. When referring to a specific record then it is termed a record occurrence. For example, a record describing the Decision Sciences Department would be an occurrence of the DEPT record type.

Records can be grouped into logical relationships called sets. A set type is the definition of the relationships which can exist ic record then it is termed a record between the records in the set. A set type must have one record type declared as its owner and one record type declared as its member. A set occurrence is a specific set consisting of an owner record occurrence and an arbitrary number of member record occurrences.

Figure 1 illustrates a Data Structure(DS) diagram, a notation used to describe data structures, and a particular occurrence of the structure described (e.g., as it might be found in the database). Figure 1 shows how a chain of pointers establishes a set occurrence -- one owner record occurrence and any number (in this case three) member record occurrences.

2.2 Schema

The schema is a description of the database. This description includes the names and characteristics of areas, sets, records and data-items. The Schema is created by the File Definition Processor (FDP) from a definition of the

- - - - - - - - -

1. CODASYL, CODASYL Data Base Task Group April 71 Report, available from ACM, New York City.

database written by the user in the Data Definition Language (DDL).

Schemas are often drawn in DS diagram form. Figure 2 depicts the schema which is used in examples throughout this guide.

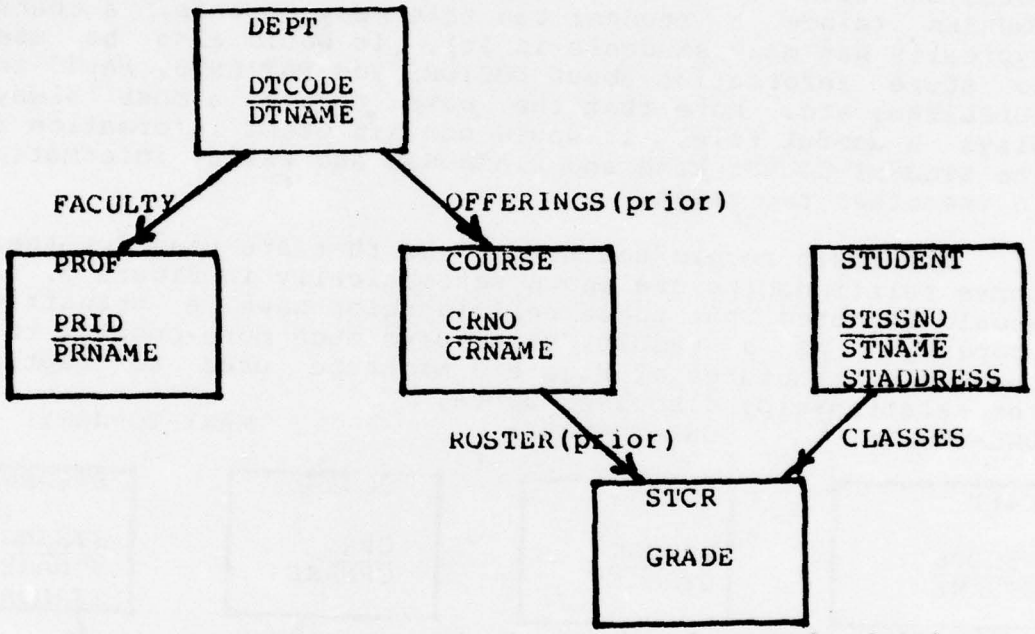


Figure 2. DS diagram of structure of example database.

The Schema is used by various WAND components, including the utility routines and the Data Manipulation Language (DML) library.

2.3 Relationships

WAND is capable of handling the following relationships (sometimes called functional dependencies) between any two data-items.

-One-to-one: Items which are contained in the same record have a mutual one-to-one relationship. A data-item which is contained in a record that is a member of a set has a one-to-one relationship to any item contained in the record that owns the set.

-One-to-many: A data-item in the owner record of a set is in a one-to-many relationship to any data-item that is contained in the member record of the set.

-Many-to-many: Two data-items are said to be many-to-many when they are contained in different records, each record an owner of a set that has a third record, called a base record, as a member of both such sets. The three record structure used for a many-to-many relationship is called a confluent hierarchy. This is the structure that would be used to store information about STUDENTs and COURSEs (since a student can take many courses; a course typically has many students in it). It would also be used to store information about DOCTORS and PATIENTS, PARTS and SUPPLIERS, etc. Note that the base record almost always plays a useful role. It would contain GRADE information in the STUDENT-COURSE case and DIAGNOSIS and PRICE information in the other two cases.

The basic record/set structures that are used for these three relationships are shown schematically in Figure 3. It should be noted that these relationships have a transitive property. As a result, structures much more complex than the basic structures of Figure 3 might be used to capture the relationships discussed above.

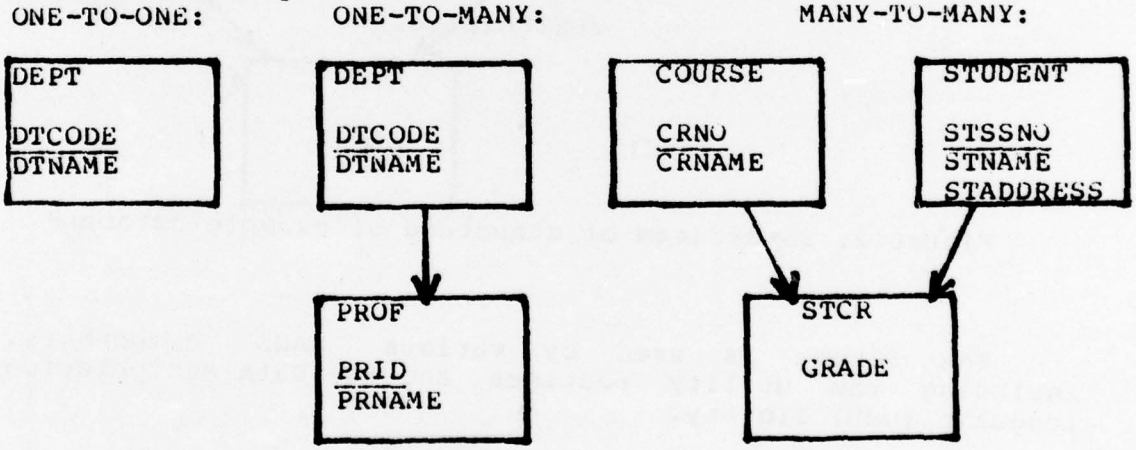


Figure 3. Relationships are captured in certain structures.

2.4 Database Key

Every record occurrence in a WAND database has associated with it a unique database key. A database key consists of a page or block number and a line number. Line numbers are assigned to records in a manner that insures a unique database key for each record occurrence. Database keys are used as pointers in chain structures, and the currency indicators always contain database key values (or zeros).

For example, the database key for the DEPT record for Decision Sciences might be 1203 indicating it was stored as the third record on the 12th page of the database.

This implementation differs from the DBTG specification to the extent that WAND does not guarantee that the database key of a record occurrence remains unchanged for the life of the record occurrence. This flexibility simplifies restructuring. As a result, the user is urged not to use the LOCATION MODE IS DIRECT clause in the DDL, nor to use database key values as data.

The last restriction can be enforced by not using the FINDD command (or by only using it to re-establish currency of a particular record type).

To sum up, the unsophisticated user should ignore database keys.

2.5 WAND Structure

This section describes the four main components of the WAND system. Figure 4 illustrates how these components interact with the user's program and the DBMS.

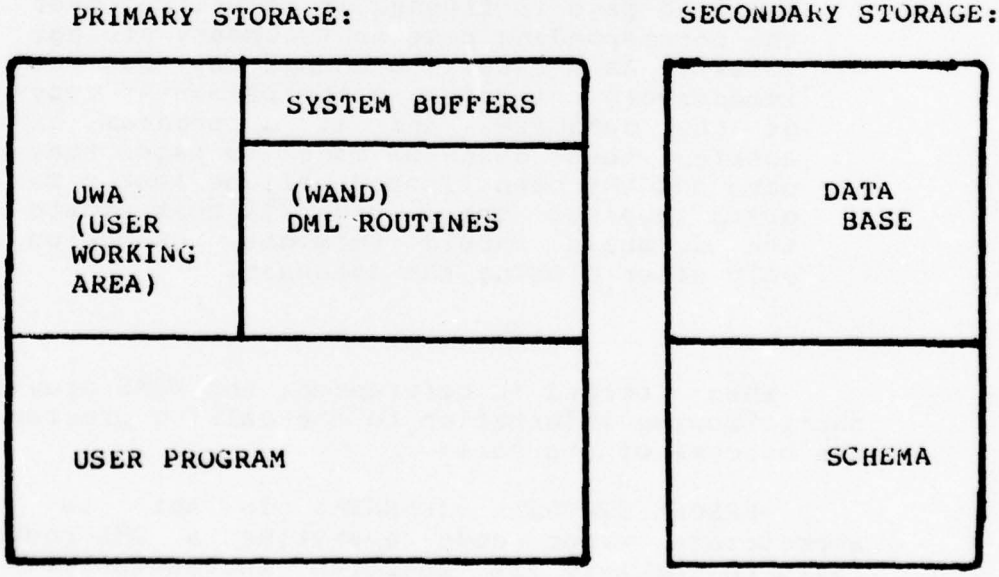


Figure 4. Structure of memory in WAND.

1. SCHEMA: The schema describes the database in terms of the characteristics of the data and the implicit and explicit relationships between data-items. The schema is created from the description of the database written in the DDL by the user. It governs access to the database made by the DML routines.

2. DATABASE: The database consists of all the record occurrences and set occurrences which are controlled by the schema. A database is structured to allow records to be accessed without regard to their physical location within the database.

3. SYSTEM BUFFERS: Up to four pages(blocks) of the database are kept in high-speed (core) memory by the DML library routines. These buffers are managed in a LRU manner, that is, the page that contains the most recently referenced record occurrence will remain in the buffers longer than any of the other three pages, unless one of these pages is referenced before it is flushed.

NOTE

Database modifications are first made in the pages stored in the buffers. When a modified page is flushed it is written over the corresponding page on secondary storage (disk). As a result, a change may not be immediately reflected in the permanent copy of the database, and if a program is aborted then changes made to pages that have not yet been flushed will be lost. To guard against this, programs that update the database should terminate execution only after closing the database.

When a record is referenced, the DBMS provides the following information to the calling program on the outcome of its call:

-ERROR STATUS: (ERRSTA) is set to the appropriate error code everytime a DML routine fails to complete its function successfully. It will retain this value until the user resets it (to zero). If ERRSTA is not zero then WAND will not perform any data management for the user. Hence, the user must be sure to 'field' all error codes.

-CURRENT RECORD OCCURRENCE OF RUN-UNIT: (CRRUNU) contains the database key of the record most recently selected by the DBMS. s to complete its

-CURRENT RECORD TYPE: (CRRECT) contains a number identifying the type of the record that is current of run-unit. Identifying numbers are assigned sequentially by the FDP so that the first record that is defined has number 1, the second has number 2, etc.

-CURRENT CALC RECORD OCCURRENCE: (CRCALC) contains the database key of the record most recently accessed with FINDC or FINDCM.

-CURRENT OCCURRENCE OF RECORD TYPE: Associated with each record type is the database key of the last record accessed of that type. The CURRNT function will return this value for any record.

-CURRENT RECORD OCCURRENCE OF SET TYPE: Associated with each set type is the database key of the last record accessed that is either an owner or a member of that set type. The CURRNT function will return this value for any set.

4. User Working Area: Records in the system buffers are not available to the user. Only when they are transferred to the User Working Area (UWA) can the information contained in the record be accessed.

2.6 Location Mode

Each record must have a LOCATION MODE designated for it. Location mode specifies the method in which the record will be stored. The location mode is specified with the DDL and can be one of the following three types:

1. DIRECT: The record is stored directly by its address (a database key) which must be given by the user. It is suggested that this location mode not be used. It will make restructuring impossible.

2. CALC: This location mode is used to store and retrieve records into and from the database using an item within the record as a key. The system transforms the key value into a page address and stores or retrieves the record on the basis of that address. This location mode is used for direct

access.

3. VIA SET-NAME: The record must be a member of the *named* set. Record occurrences are stored on the same page as the owner record occurrence of the selected set occurrence. If there is no room on that page, then the next page in sequence is used, and so on. This location mode should be used if the record in question need not be directly accessed. Proper use of this location mode can increase database efficiency. Since all of the records on a page can be obtained in one physical access, the number of physical accesses is reduced if memeber records are stored VIA the sets through which they are most frequently accessed.

2.7 Set Mode

Set mode characterizes the way in which records in a set are related. Only CHAIN mode is supported by WAND. In this mode, a forward or NEXT pointer is contained in each record in the set. This allows for serial access of all the records in a set. A record contains one next pointer for each set type in which it is declared as a member or owner. Efficient processing of sets in a backward direction is possible if a PRIOR pointer is included in the member and owner records of the set. This pointer is optional and has to be declared by the user for each set type in which processing in the backward direction is desired. Member records of a set can also be linked to the owner of the set. This is done by declaring an OWNER pointer for the member record of the set. This pointer is also optional. All pointers are database keys.

Note that inclusion of optional PRIOR and OWNER linkages only affects processing efficiency. WAND will permit backwards processing, or access to owners, even if such linkages are not present. Unless the physical size of the database is a concern, it is usually best to include both prior and owner pointers in all sets.

2.8 Set Order

The order of a set type determines where a new member occurrence is inserted in the set. The following types of order exist in WAND:

1. *FIRST:* The new record is logically stored as the immediate successor to the owner record of that set occurrence.

2. LAST: The new record is logically stored as the immediate predecessor to the owner record of that set occurrence.

3. NEXT: The new record is logically stored as the immediate successor to the current record of the given set occurrence.

4. PRIOR: The new record is logically stored as the immediate predecessor to the current record of the set occurrence.

NOTE

Next and prior orderings should not be used unless absolutely necessary. Using these orderings will make your programming more complex and will make it more difficult to change the database structure at a later date.

3.0 DATA DESCRIPTION LANGUAGE

WAND provides the user with a means to define the schema by using the Data Description Language (DDL).

The definition of the schema, written in the DDL, consists of three types of entry which serve to:

-Identify the schema (Schema Entry)

-Define records (Record Entry)

-Define sets (Set Entry)

Entries, or sentences, must always end with a period.

Sections 3.1, 3.2, and 3.3 of this chapter describe the Schema Entry, Record Entry, and Set Entry respectively. Each description contains the general skeleton of the entry, followed by a set of rules applicable for each case and some relevant examples.

In these descriptions the following notation is used:

1. Words that must be replaced with a user-defined name or value are in lower case.

2. ___ (underline) word or character that must appear.

3. () Encloses a phrase that may be ommitted.

4. {} Encloses lines from which only one may be used.

5. !! Encloses phrases that may be repeated.

In section 3.4, the overall structure of the DDL and the procedure for creating the schema, are described. Section 3.4.1 gives a complete example of a schema definition while section 3.4.2 contains the corresponding User Working Area (UWA) generated by the FDP for inclusion in the user's FORTRAN programs.

User defined names for records, sets and items must be unique. The first character of a name must be a letter, the remaining characters must be alphanumeric. Although names of up to 10 characters are allowed, if FORTRAN is to be used to access the database then the first 6 characters of a name must be unique. Take heed -- neither WAND nor FORTRAN will check if a name is unique in the first six characters! Reserved words (page 1) may not be used as a name.

Clauses must be ordered as in the skeleton.

Commas, semi-colons and tabs are treated like spaces.

All lower case characters in the input to the FDP are transformed to upper case (except for the password).

3.1 Schema Entry

he first 6 characters

The function of the Schema Entry is to identify the schema of a database, and to define general database characteristics. The skeleton of the Schema Entry is as follows:

```
SCHEMA NAME IS schema-name
       (PRIVACY LOCK IS password)
       (DATABASE SIZE IS integer PAGES)
       (PAGE SIZE IS integer WORDS).
```

'Schema-name' is a user-defined name which will be used to identify all files associated with the particular schema (see section 3.4).

'Schema-name' must be unique and is from 1 to 10 characters in length.

If a privacy lock is specified, then 'password' must be provided everytime the database is opened (see DBOPEN, section 4.1).

The default database size is 10 pages and the default page size is 128 words. Optimal page sizes are integer multiples of 128.

EXAMPLES:

```
SCHEMA NAME IS SCHOOL
 PRIVACY LOCK IS DBTG
 DATABASE 5
 PAGE SIZE IS 256 WORDS.

SCHEMA IS HOSPITAL.
```

```
SCHEMA IS ORGNTN PRIVACY SECRET.
```

3.2 Record Entry

The function of a Record Entry is to name and give certain characteristics to record types and their subordinate data-items. The skeleton of a Record Entry is as follows:

```
RECORD NAME IS record-name

   LOCATION MODE IS
      {VIA set-name                                              }
      {CALC USING item-name-1 DUPLICATES ARE (NOT) ALLOWED}
      {DIRECT                                                    }

   !item-name-2 TYPE IS
      {CHARACTER integer}
      {FIXED            }
      {REAL             }!.
```

'Record-name' is a unique user-defined name which identifies all the occurrences of a record type.

If VIA set-name is specified as location mode, then 'record-name' must be a member of 'set-name'.

If 'CALC' mode is specified, then a data item named 'item-name-1' must be contained within the record type being defined (e.g., also defined as an 'item-name-2').

It is suggested that DIRECT location mode not be used.

Any occurrence of a given record type can be retrieved VIA any set in which it is a member, regardless of the location mode defined for the record type.

EXAMPLES:

```
RECORD NAME IS DEPT
 LOCATION MODE IS CALC USING DTCODE DUPLICATES NOT
ALLOWED
DTCODE TYPE IS CHARACTER 5
DTNAME TYPE IS CHARACTER 20.


RECORD NAME PROF
```

```
 LOCATION MODE CALC USING PRID DUPLICATES
PRID TYPE IS FIXED
PRNAME TYPE IS CHARACTER 20.


RECORD STCR
 LOCATION VIA ROSTER
SCGRADE TYPE CHARACTER 2.
```

3.3 Set Entry

The function of a Set Entry is to name and give certain characteristics to set types within a database. The skeleton of a Set Entry is as follows:

```
SET NAME IS set-name
        MODE IS CHAIN
                (LINKED TO PRIOR)

        ORDER IS
                {FIRST}
                {LAST }
                {NEXT }
                {PRIOR}

        OWNER IS record-name-1

        MEMBER IS record-name-2
                (LINKED TO OWNER).
```

'Set-name' is a user-defined name which will identify all the occurrences of a set type.

'Record-name-1' and 'record-name-2' must be different and must correspond to record types previously defined in their respective Record Entries.

EXAMPLES:

```
SET NAME IS FACULTY MODE IS CHAIN
 ORDER IS NEXT
 OWNER IS DEPT
 MEMBER IS PROF.


SET ROSTER MODE CHAIN LINKED TO PRIOR
 ORDER NEXT
```

```
OWNER COURSE
MEMBER STCR LINKED TO OWNER.
```

3.4 Creation Of The Schema

In defining the database, the following rules apply:

1. There must be only one Schema Entry in the schema definition.

2. For each record type and each set type in the database, a separate entry is required.

3. The Schema Entry must be the first entry in the definition.

4. All the Record Entries must precede all the Set Entries in the definition.

The creation of the schema for a database is achieved by means of a simple procedure consisting of the following two steps:

1. The definition of the schema is put into a file named schema-name.DDL. This file may contain line numbers. (Not yet implemented.)

2. The File Definition Processor (FDP) is run by issuing the following command at the monitor level.

```
RU FDP[4010,51]
```

The FDP will respond with SCHEMA NAME:. You should type in the name of the schema. When processing is complete, the number of errors encountered by the FDP is typed on the user's terminal. The FDP generates several files. All file names are taken from the name of the schema defined in response to the SCHEMA NAME prompt. If this name is different from the name in the Schema Entry, a warning message is typed.

1. schema-name.ERR: This file contains any errors encountered by the FDP while parsing the input from file schema-name.DDL. This file contains the lines which caused errors. The place in the line where the error was detected is marked with ^. Any error encountered by FDP will cause a skip to the next sentence (e.g., following the next period) and

processing will continue from there.

2. schema-name.WRK: This file is created only if no errors are detected and contains all the common, equivalence and integer statements that every FORTRAN program using the DML routines will need to access the database. Therefore, the appropriate FORTRAN .WRK file should be included in all programs that the user writes to access the database. This can be accomplished without merging files (see page 20).

3. schema-name.SCH: This file is created only if no errors are detected and contains the description of the database used by the DML routines.

3.4.1 Schema Definition -

The following listing is a complete example of the Data Description Language. This text would be contained in the file SCHOOL.DDL and corresponds to the schema diagram of Figure 1.3.

```
SCHEMA NAME IS SCHOOL
PRIVACY LOCK IS DBTG.


RECORD NAME IS DEPT
LOCATION MODE IS CALC USING DTCODE DUPLICATES NOT
DTCODE TYPE IS CHARACTER 5
DTNAME TYPE IS CHARACTER 20.


RECORD NAME IS PROF
LOCATION MODE IS CALC USING PRID DUPLICATES NOT
PRID TYPE IS FIXED
PRNAME TYPE IS CHARACTER 20.


HEMA NAME IS SCHOOL
RECORD NAME IS COURSE
LOCATION MODE IS CALC USING CRNO DUPLICATES NOT
CRNO TYPE IS CHARACTER 7
CRNAME TYPE IS CHARACTER 20.


RECORD NAME IS STUDENT
LOCATION MODE IS CALC USING STSSNO DUPLICATES NOT
STSSNO TYPE IS FIXED
STNAME TYPE IS CHARACTER 20
```

```
        STADDRESS TYPE IS CHARACTER 20.


        RECORD NAME IS STCR
        LOCATION MODE IS VIA ROSTER
        GRADE TYPE IS CHARACTER 2.


        SET NAME IS FACULTY MODE IS CHAIN
        ORDER IS NEXT
        OWNER IS DEPT
        MEMBER IS PROF.


        SET NAME IS OFFERINGS MODE IS CHAIN LINKED TO PRIOR
        ORDER IS FIRST
        OWNER IS DEPT
        MEMBER IS COURSE.


        SET NAME IS ROSTER MODE IS CHAIN LINKED TO PRIOR
        ORDER IS NEXT
        OWNER IS COURSE
        MEMBER IS STCR LINKED TO OWNER.


        SET NAME IS CLASSES
        MODE IS CHAIN
        ORDER IS LAST
        OWNER IS STUDENT
        MEMBER IS STCR
        LINKED TO OWNER.
```

3.4.2 Working Area File -

After processing SCHOOL.DDL the file containing the necessary equivalence and integer statements is created. This file, SCHOOL.WRK, is shown below. Notice that the last four elements of array DB contain historical information about the creation of the schema.

```
C      FROM SCHEMA CREATED ON 14-Apr-76  10:35      .
C      CREATED BY FDP  A.04 SCHEMA GENERATION      0.
       INTEGER DB
       COMMON/DBASE/DB(   45)
       DATA (DB(K),K= 40,45)/'14-Apr-76 10:35      A.04',0/
       EQUIVALENCE(ERRSTA     ,DB(     1))
       EQUIVALENCE(CRRECT     ,DB(     2))
       EQUIVALENCE(CRRUNU     ,DB(     3))
```

```
      EQUIVALENCE(CRCALC     ,DB(    4))
C DEPT      RECORD.
      EQUIVALENCE(DEPT       ,DB(    6))
      EQUIVALENCE(DTCODE     ,DB(    6))
      EQUIVALENCE(DTNAME     ,DB(    7))
C PROF      RECORD.
      EQUIVALENCE(PROF       ,DB(   12))
      EQUIVALENCE(PRID       ,DB(   12))
      EQUIVALENCE(PRNAME     ,DB(   13))
C COURSE    RECORD.
      EQUIVALENCE(COURSE     ,DB(   18))
      EQUIVALENCE(CRNO       ,DB(   18))
      EQUIVALENCE(CRNAME     ,DB(   20))
C STUDENT   RECORD.
      EQUIVALENCE(STUDENT    ,DB(   25))
      EQUIVALENCE(STSSNO     ,DB(   25))
      EQUIVALENCE(STNAME     ,DB(   26))
      EQUIVALENCE(STADDRESS  ,DB(   30))
C STCR      RECORD.
      EQUIVALENCE(STCR       ,DB(   35))
      EQUIVALENCE(GRADE      ,DB(   35))
C *** SETS ***
      EQUIVALENCE(ROSTER     ,DB(   36))
      EQUIVALENCE(FACULTY    ,DB(   37))
      EQUIVALENCE(OFFERINGS  ,DB(   38))
      EQUIVALENCE(CLASSES    ,DB(   39))
      INTEGER ERRSTA
      INTEGER CRRECT
      INTEGER CRRUNU
      INTEGER CRCALC
      INTEGER DEPT       (    5)
      INTEGER DTCODE
      INTEGER DTNAME     (    4)
      INTEGER PROF       (    5)
      INTEGER PRID
      INTEGER PRNAME     (    4)
      INTEGER COURSE     (    6)
      INTEGER CRNO       (    2)
      INTEGER CRNAME     (    4)
      INTEGER STUDENT    (    9)
      INTEGER STSSNO
      INTEGER STNAME     (    4)
      INTEGER STADDRESS  (    4)
      INTEGER STCR       (    1)
      INTEGER GRADE
```

4.0 DATA MANIPULATON LANGUAGE

Once the schema has been defined, the user will want to write programs that interact with the database. This can be accomplished by use of the Data Manipulation Language (DML) within a host language, such as FORTRAN.

The following sections list the DML statements available in WAND. The general form is presented along with a description of each statement's functional use. Several examples are given which illustrate the use of many of the statements.

Whenever a record name or set name is required in a DML statement it can be entered in one of three ways.

1. directly as the name of the set or record (not in quotes as a literal),

2. as a double precision variable or single precision array having as value the name,

3. as a literal or variable with value zero, which is usually interpreted to indicate the last record or set previously referenced.

Method 1 is preferred.

The DML commands are:

DBOPEN, DBCLOS open and close the database

STORE, MODIFY, DELETE update the database

FINDC, FINDPO, FINDAP, FINDO, FINDD find records in the database

GET moves data into the UWA

CURRNT, SETCUR retrieve and establish record and set currency status indicators

GARBAGE cleans up the database

The commands will be described in an order that will facilitate some simple examples. The first example follows a description of DBOPEN, STORE and DBCLOS. The second example follows FINDC, FINDPO, FINDAP and GET, and a final example follows a description of the remaining commands.

4.1 DBOPEN

The DBOPEN statement is used to open the database for processing.

CALL DBOPEN(schema-name,password,mode)

The password and schema-name must be as defined in the Data Description Language. If the mode is 1 then the database will be open for update. When mode is 0, the database can only be read.

4.2 STORE

The STORE statement adds new records to the database.

CALL STORE(record-name)

STORE establishes a new record occurrence in the database by storing the values of the data-items of the specified record in the UWA into the database. An error status of 1208 is returned if the database has not been opened for update (mode=0) and a STORE is attempted.

The record will be linked to the current occurrences of all set types in which it is a member. The specific insertion point in the set is determined by the order clause of the set. For example, if ORDER IS FIRST, then a new member record will always be logically inserted as the first record in the set.

New set occurrences will be created for all set types in which the record is an owner.

All currency updates are made once the record has been successfully loaded into the database.

4.3 DBCLOS

DBCLOS closes the database.

CALL DBCLOS

It is most important that routines which update the database (MODE=1) close the database prior to termination. If this is not done, the database may not contain all of the changes made and may even become unusable.

4.4 Example 1

The DML statements discussed so far are sufficient for building a simple database as illustrated in the following example.

```
DOUBLE PRECISION SCHOOL
DATA SCHOOL/'SCHOOL'/
CALL DBOPEN(SCHOOL,'DBTG',1)
DTCODE='DESCI'
DTNAME(1)='DECIS'
DTNAME(2)='ION S'
DTNAME(3)='CIENC'
DTNAME(4)='ES    '
CALL STORE(DEPT)
CALL DBCLOS
STOP
END
```

or building a simple database as illustrated in the following example.

Note that DTNAME was defined in the DDL as a data-item of the record DEPT and DTCODE was defined as the key of the record.

Under the second method of passing record names, it is possible to assign record and set names to variables, e.g.:

```
DATA ADEPT/'DEPT'/
CALL STORE(ADEPT)
```

Assuming that this program is in file PROG1.F4, then it can be executed by issuing the following monitor command:

```
EX SCHOOL.WRK+PROG1,DML[4010,51]/LIBRARY
```

4.5 FINDC

The FINDC statement is used to find a record with a key.

```
CALL FINDC(record-name,position)
```

Position may only have values of 'FIRST' or 'NEXT'. NEXT is used for records which have DUPLICATES allowed and permits a program to find all occurrences of record-name with the same value for the CALC key. Error status is set to 0307 if no more duplicates exist.

The key field of the record must first be initialized in the UWA with the value of the key of the sought record. For example, if we wanted to find the Decision Sciences record then DTCODE would first be initialized with 'DESCI'.

The use of this statement is restricted to record types that have a LOCATION MODE of CALC.

All currency updates are made once the record is located in the database.

4.6 FINDPO

To find a record positionally within a set or area, the FINDPO statement is used. If the record is found all currency updates are made.

CALL FINDPO(position,set-name,record-name)

Position may have one of the following five values:

1. 'FIRST': The database will be searched for the first record of the occurrence of the given set type.

2. 'LAST': The database will be searched for the last record of the current occurrence of the given set type.

3. 'NEXT': The database will be searched for the next record after the current record of the given set (current of set type). If current of set type is the last record of the given set, then ERRSTA is set to 0307.

4. 'PRIOR': The database will be searched for the record prior to the current record of the given set. If current of set type is the first record of the indicated set, then ERRSTA is set to 0307.

5. N (Integer or integer variable): The database will be searched for the Nth record of the given set relative to the owner of that set. N may be negative. Error status 307 will occurr if the owner is encountered while searching for the Nth record.

4.7 FINDAP

Find in area, positional.

FINDAP can be used to treat the entire database as a sequential file, or to treat all occurrences of a particular record type as a sequential file. It is useful to process all records of a particular type.

CALL FINDAP(position,0,record-name)

Position may have one of the following five values:

1. 'FIRST': The database will be searched for the first record of the given record type.

2. 'LAST': The database will be searched for the last record of the given record type.

3. 'NEXT': The database will be searched for the next record after the current record of the same type. If the current record of the same type is the last record of that type in the database, then ERRSTA is set to 0307.

4. 'PRIOR': The database will be searched for the record prior to the current record of the same type. If the current record of that type is the first record of that type in the database, then ERRSTA is set to 0307.

5. N (Integer or integer variable): The database will be searched for the Nth record of the given type relative to the current record of that type. N may be negative. Error status 307 will occur if the beginning or end of the database is encountered while searching for the Nth record.

If zero (0) is specified as the record type, then the appropriately positioned record of any type is found (all types qualify). The desired position is counted from current of record-name if a record-name is given, otherwise it is counted from current of run-unit.

When no record type is specified, the type of the record that was actually found can be determined from the value of CRRECT in the UWA.

4.8 GET

FIND statements only load the record into the system buffers and perform currency updates. The GET statement will retrieve the current record occurrence of a record type and load y found can be determined from the value of CRRECT in the UWA. it into the appropriate data-iems in the UWA.

```
CALL GET(record-name)
```

or

```
CALL GET(0)
```

If the first format is used, a record of the given type must be current. In the second case, the record that is moved to the UWA is the current record occurrence of the run-unit (CRRUNU), usually the last record that was found.

4.9 Example 2

Suppose we want to list all the faculty members in a particular department. There is a set named FACULTY, that has owner DEPT and member PROF. This is a one-to-many relationship or hierarchy, since for each department there are many professors within that department and each professor only belongs to one department. The following program will list all the professors within a department.

```
                .
                .
                .
        DTCODE='DESCI'
        CALL FINDC(DEPT,'FIRST')
1       CALL FINDPO('NEXT',FACULTY,PROF)
        IF(ERRSTA.NE.0) GO TO 99
        CALL GET(PROF)
        WRITE(5,2) (PRNAME(I), I=1,4)
        GO TO 1
2       FORMAT(' ',4A5)
99              .
                .
                .
```

Since ERRSTA will be set to the appropriate error code (307) when FINDPO attempts to access the next record when the current record is the last record in FACULTY, the loop is properly terminated when all professors have been found.

If the program continues to process the database at statement 99, then it should first reset ERRSTA to 0 to indicate to WAND that the error has been recognized. WAND

will not proceed until this is done.

4.10 FINDD

FINDD will find a record directly by its address within the database.

CALL FINDD(address)

The address is an integer of the form PLL where P is the page number and LL is the two digit line number of the record within that page.

If the record is found all currency updates are made.

The FINDD statement is included for completeness and should only used by experienced programmers. Its primary purpose is to re-establish currency for a particular record occurrence.

4.11 FINDO

FINDO finds the owner of the given set type.

CALL FINDO(set-name)

There must be a current record of the given set type or an error will occur. It is not required that the member records of the set have owner pointers. If there are no owner pointers, WAND will simply 'walk' through the chain to find the owner. Note that if there are many members in the set, this can take a long time, especially if the members are not stored VIA this set.

All currency updates are made if the owner of the set is found.

4.12 MODIFY

It is likely that the user will want to modify records which already exist in the database. his is accomplished by use of the MODIFY statement. An error status of 0809 will be returned if the database is not opened for update (mode=0) when a MODIFY is attempted.

CALL MODIFY(record-name)

The current occurrence of the named record type is re-written. The data-items of the record in the database are modified with the values of the data-items in the UWA.

If the record has a key and this key is changed, then the record may be moved to another page in the database. In this case, all necessary relinking will be performed. This differs from the DBTG report which specifies that records cannot be moved in this manner.

4.13 DELETE

Records can be deleted from the database by using the DELETE statement.

CALL DELETE(record-name)

DELETE will delete the current record and all records which are hierarchically below that record (i.e. Any record occurrence in a set which the deleted record owns, all records in sets which those records own, etc.). An error status of 0209 will be returned if the database is not opened for update (mode=0) when a DELETE is attempted.

4.14 GARBAGE

WAND provides a garbage collection routine which zeroes out those records which have previously been deleted.

CALL GARBAGE

The records in the database are moved up over the space made available through execution of the GARBAGE statement.

4.15 CURRNT

CURRNT Is an integer function that returns the database key of the current named record or of the current record of the named set.

variable-name=CURRNT(record-or-set-name)

If record-or-set-name is not the name of a record or set, ERRSTA is set to 8840.

4.16 SETCUR

SETCUR is used to change the currency status of the named record or set.

CALL SETCUR(record-or-set-name,value)

Value should be an integer constant or variable containing a database key. If record-or-set-name is not a record or set name, ERRSTA is set to 8840.

Only experienced programmers should use this DML command. Its primary use is in association with recursive structures, such as Bill-of-Material.

4.17 Example 3

Suppose that for a given course we would like to determine the roster of students in that course. Also, for a given student we would like to determine that student's classes. There is a set named ROSTER that has owner COURSE and member STCR. Also, there is a set named CLASSES that has owner STUDENT and member STCR. This three record structure captures a many-to-many relationship or confluent hierarchy. In a given course there are many students in that course, and a particular student may be enrolled in many courses. The following example prints out a list of the classes of a given student.

```
                .
                .
                .
C       FIND THE CLASSES THAT JANET GLEASON
C       IS TAKING(KEY=FIXED=STSSNO=950000001)
        STSSNO=950000001
        CALL FINDC(STUDENT,'FIRST')
1       CALL FINDPO('NEXT',CLASSES,STUDENT)
        IF(ERRSTA.NE.0)GO TO 2
        CALL FINDO(ROSTER)
        CALL GET(COURSE)
        WRITE(5,3) (CRNAME(I), I=1,4)
        GO TO 1
3       FORMAT(' ',4A5)
2               .
                .
                .
```

APPENDIX A

DDL

SYMBOL	MEANING
__ (underline)	WORD MUST APPEAR
()	PHRASE MAY BE OMMITTED
{}	ONLY ONE OF THE LINES MAY BE USED
!!	PHRASE MAY APPEAR MULTIPLE TIMES

Lower case words must be replaced by a user-defined name or value.

```
SCHEMA NAME IS schema-name
        (PRIVACY LOCK IS password)
        (DATABASE SIZE IS integer PAGES)
        (PAGE SIZE IS integer WORDS).

RECORD NAME IS record-name
    LOCATION MODE IS
        {VIA set-name                                                    }
        {CALC USING item-name-1 DUPLICATES ARE (NOT) ALLOWED}
        {DIRECT                                                          }

    !item-name-2 TYPE IS
        {CHARACTER integer}
        {FIXED            }
        {REAL             }!.

SET NAME IS set-name
        MODE IS CHAIN
                (LINKED TO PRIOR)

        ORDER IS
                {FIRST}
                {LAST }
                {NEXT }
                {PRIOR}

        OWNER IS record-name-1

        MEMBER IS record-name-2
                (LINKED TO OWNER).
```

APPENDIX B

DML

CALLING SEQUENCES

```
CALL DBOPEN(schema-name,password,mode)
     mode = 0 for read only
     mode = 1 for update

CALL DBCLOS


CALL STORE(record-name)
CALL MODIFY(record-name)
CALL DELETE(record-name)


CALL FINDC(record-name,position)
     position = 'FIRST' or 'NEXT'

CALL FINDD(address)
CALL FINDO(set-name)
CALL FINDPO(position,set-name,record-name)
     position = 'FIRST', 'LAST', 'NEXT',
                'PRIOR', or integer
CALL FINDAP(position,0,record-name)
     position = 'FIRST', 'LAST', 'NEXT',
                'PRIOR', or integer


CALL GET(record-name)


variable-name = CURRNT(record-or-set-name)
CALL SETCUR(record-or-set-name,value)

CALL GARBAGE
```

APPENDIX C

ERROR CODES AND CONDITIONS

In WAND an error condition produced by a DML statement will not cause the user's program to be aborted. The user is responsible for providing facilities in his programs for error detection and error recovery. Every error condition produced by a DML statement causes the contents of a variable (ERRSTA) in the UWA to be set to a value indicating the error encountered.

The successful completion of a DML statement will not alter the value of ERRSTA, which is initialized to zero. After an error condition has been produced, the value of ERRSTA should be reinitialized by the user's program, since this is not done by the system.

All error codes are four-digit numbers of the form NNMM, where NN indicates the DML routine which produced the error (Table 1), and MM describes the type of error produced (Table 2).

TABLE 1. DML ERROR CODES (MAJOR).

CODE	Originating routine
01	DBCLOS
02	DELETE
03	FINDD, FINDC, FINDO, FINDPO
05	GET
08	MODIFY
09	DBOPEN
12	STORE
88	CURRNT, SETCUR
99	File I/O

TABLE 2. ERROR STATUS CONDITIONS.

CODE	DESCRIPTION.
01	Database not open.
04	Set type indicated not found for record.
06	No current record in set type or no current of given record type. Most common when attempting to STORE a record and currency has not been established in all sets of which it is to be a member.
07	Attempt to find 'NEXT' or 'PRIOR' and current record is the last or first respectively in the set.
08	Invalid record name.
09	Database not in update mode.
10	Invalid set name.
11	Wrong password.
12	Record type indicated does not have a database key.
13	No current record of run-unit.
14	No current of indicated record type
17	Record not found or logically deleted.
19	Invalid record type
20	Record types do not match.
21	Chain error; no pointer to record.
22	Record type is zero.
23	Trying to store existing record.
24	Not enough room to store direct.
25	Not enough room to store anywhere.
26	Record name not in schema, no record found with matching key or no member records in set.
28	Trying to open area which is already open.

29	Duplicate CALC key not allowed for record.
30	Current of CALC different than expected from hashing algorithm.
40	Name not a record or set name.
51	Deleted record involved.
52	Owner of set has been deleted.
53	Owner of set has been deleted.
55	Record type not a member of given set or owner and member record types do not match schema specification.
56	Record must be a MEMBER of some set.
57	Record type not member/owner of a given set.
58	System error; Logically deleted record with NEXT and PRIOR pointers.
59	Record with no data (zero length).
60	Two owner records in set.
90	Schema is out of date, rerun the FDP.
91	System error reading schema file.
92	Unable to open schema file (*.SCH).
93	System error writing database.
94	System error reading database.
95	Database cannot be opened for output.
96	No device available for database.
97	Insufficient core to store schema.
98	Database does not exist (use DBINIT first).
99	Block does not exist (BUFFEX).

APPENDIX D

DBINIT

DBINIT is a utility routine that initializes WAND databases. It must be used before data can be stored in the database and cannot be used until a schema has been created. To initialize a database, type at monitor level:

```
RU DBINIT[4010,51]
```

DBINIT will ask for the database name and password, and if no database presently exists, a database will be initialized. If a database already exists, the user is warned that initialization will destroy all data presently stored in the database. To proceed in this instance the user must type YES.

In either case, successful initialization is indicated with the message:

```
DATABASE OPENED -- ERRSTA = 0
```

APPENDIX E

DBDUMP

DBDUMP is a utility for printing the contents of selected database pages in various formats, either on the user's terminal or on the line printer. To execute DBDUMP, type at monitor level:

```
RU DBDUMP[4010,51]
```

DBDUMP will then respond by asking for the database name and the password. If the database is successfully opened, DBDUMP will ask for an output destination. A response of F will send the dump output to FOR01.DAT (for subsequent printing), any other response will cause dumping on the user's terminal.

Subsequently, the user is asked to specify a line width in characters. If none is specified the width defaults to 80 if output is to a terminal and 132 if output is to a file.

Dumping is always done in two formats: numeric and ASCII. The user may select octal or decimal radix for the numeric format. In response to the prompt, O selects octal, any other response selects decimal radix.

Finally, the user is asked to specify the first block (or page) and last block of the range to be dumped. When dumping of the specified pages has been completed, DBDUMP cycles and again asks for an output device specification. Execution is terminated with control-c.

APPENDIX F

DBLOOK

F.1 INTRODUCTION

DBLOOK is part of the WAND system. It was written with several purposes in mind. First of all, it provides an interactive DML. In other words, any DML command can be given to DBLOOK directly. Secondly, DBLOOK gives information about the sets, records and items in a database. A HELP facility is provided to instruct the novice in DML programming.

DBLOOK is useful to the Data Base Administrator; it will tell him about the structure of selected database components. It is useful to the programmer for the same reason. In addition, the programmer can use it as a database editor, to check if programs are running correctly, to fix minor errors in the database, or to create a test database.

The indirect file capability permits execution of a file of commands and makes it possible for a user to construct a simple query facility with DBLOOK.

Finally, because of the extensive HELP routines, DBLOOK can be used by the student to learn about WAND and how to program in DML.

F.2 CONVENTIONS

DBLOOK has been designed for ease of use. Commands can be abbreviated, default values are assumed whenever possible, and other default actions are frequently taken. The user can examine and control such defaults with the STATUS and TO commands (see page 5).

Commands may be entered in upper or lower case or any combination of cases. Where a shortened form of the command is permitted, it is the first n letters (n>=1) of the command that results in a unique command storing. For the

FIND commands the 'IND' is ommitted from the shortened form of the command.

Multiple commands may be entered on a line, separated by semi-colons (;). The commands on a line are not executed until a carriage return is typed. A command should be contained entirely on a single line.

Several types of literals are recognized by the system: integers, real numbers, and character strings. An integer is a string of numeric characters optionally preceeded by plus (+) or minus (-). A real number is a string of numeric characters with a decimal point, optionally preceded by a plus or minus sign, or in standard FORTRAN 'E' format (e.g. 12.436E-5). A character string is a string of alphanumeric characters or any string of characters enclosed in quotes ('). To include a quote character as part of the string, it must be repeated.

A string in quotes will always be interpreted as a literal. But, to save a user the inconvenience of typing the quotes, the system will try to determine whether or not a string is a literal from the context. This is successful in all cases as long as the string does not contain blanks, commas, parenthesis, quotes or semi-colons. In the special case of an assignment statement or when the user is responding to a DBLOOK prompt for data, a literal will be successfully recognized (without delimiting quotes) as long as it does not contain a quote or a semi-colon.

Case (upper or lower) is preserved in literals.

ERRSTA. As a result of executing a DML command, WAND may set ERRSTA to a non-zero value. (See WAND User's Guide). DBLOOK prints the value of ERRSTA whenever it is non-zero. In addition, a REPEAT loop is stopped if ERRSTA becomes non-zero. ERRSTA will remain non-zero until the user resets it to zero with ERRSTA=0. As long as ERRSTA is not zero, none of the DML commands will function, and ERRSTA will remain unchanged.

Arguments. Many DML commands have record, set, area, or data item names as arguments. One can refer to the last referenced record, set, area, or data-item simply by typing zero(0) for the argument. This does not apply to commands, such as FINDAP, where 0 has a special meaning.

When an error is detected in a line of commands, the erroneous command and all following it are ignored. A vertical arrow is printed to point to the probable error and an explanatory diagnostic is also printed.

F.3 USING DBLOOK

To run DBLOOK type

```
RU DBLOOK[4010,51]
```

DBLOOK signals that it is ready for a command by typing an asterisk (*).

At this point any of the HELP commands may be entered. To find out what these are, type HELP.

To be able to successfully enter any of the other commands, the database must first be opened. The command for this is:

```
CALL DBOPEN (database, password, mode)
```

For example:

```
CALL DBOPEN(SCHOOL,XTRA,0)
```

or, since 'CALL', the parens and the commas may be omitted for convenience:

```
DBOPEN NAMFIL 'MY PAS' 1
```

If mode is 1, then the database is opened for update. If any changes have been made to a database that has been opened for update, the database must be closed before execution is terminated (with DBCLOS). The EXIT command will automatically close the database before terminating execution.

In addition to the standard DML commands, DBLOOK has several commands that assist in data access: DISPLAY, REPEAT, assignment, and indirect file.

1. The format of DISPLAY is

```
DISPLAY arg1 arg2 arg3 . . .
```

where argn is a data-item name or a record name. The value in working storage of argn is printed if it is a data-item name. If argn is a record name the working storage values of all data-items contained in the record are printed. If no arguments are given, then DBLOOK assumes the same set of arguments as was specified on the last DISPLAY command (this default does not apply in LABEL mode). The printed value of each data-item is preceded by a blank. The formats used to print the values of integers, reals and character strings of length n are I10, E15.8 and nA1 respectively.

Several modes and defaults (see page 5) affect display processing. The default formats described above can be changed by the user. The user can set DISPLAY mode which will result in the automatic display of any record that is found with any of the FIND commands. The user can also set LABEL mode which will result in automatic labelling of item values with item names. In label mode, only one item is displayed on a line with its label.

2. The REPEAT command repeats all commands to the left of the REPEAT a specified number of times. For example, the line of commands:

```
FINDPO NEXT STUDNS STUDNT;GET STUDNT;DISPLAY STUDNT;REPEAT 5
0
```

will cause up to 50 student records to be printed. The REPEAT terminates when ERRSTA is non-zero (e.g., when all members of the STUDNS set have been found). The maximum number of repeats is 10 if a value is not specified.

Up to five nested REPEAT commands may be specified on one command line.

3. Assignment is accomplished by any statement of the form.

```
data-item-name = literal
```

For example:

```
STUNAM='JONES A.  R.'
```

or

```
AGE=21
```

This command can be used to initialize CALC key values or to set data-item values when updating the database.

In PROMPT mode (see below) DBLOOK will print the data-item-names of all items that need a value for the indicated action; the user accomplishes assignment by typing the literal value he wishes assigned.

4. The user can tell DBLOOK to read commands from a file. This is accomplished with the @ command. For example,

```
@FIL.CMD
```

will cause DBLOOK to read from a file named

FIL.CMD. DBLOOK assumes that the extent of the file is .DAT unless another extent is specified. If the file has no extent, a blank extent must be specified. e.g.,

@ XYZ.

is the only valid way to specify a file named XYZ. DBLOOK will print all commands that are read from the file. The processing of commands on the file cannot be interrupted. Commands can be entered from the terminal after the last command from the file has been processed.

F.4 DBLOOK DEFAULTS -- STATUS AND TO

The user can change and interrogate DBLOOK's status.

The following 8 modes (default underlined) and 10 generics with their initial values determine DBLOOK status:

GET	NOGET	
PROMPT	NOPROMPT	
DISPLAY	NODISPLAY	
LABEL	NOLABEL	
QUOTE	'	(delimits literals)
SEMIC	;	(separates commands)
ASTERISK	*	(prompts for commands)
REPEAT#	10	(default number of repeats)
USER	!	(next word is not a command)
ASSIGN	=	(used for assignment)
ATSIGN	@	(selects indirect file)
COMMENT	*	(ignore input line)
FIXFMT	1X,I10,	(integer FORMAT)
REALFMT	1X,F8.3,	(real number FORMAT)

In GET mode, any record that is found is automatically moved to the UWA. In NOGET mode, the user must move data to the UWA by executing a GET command. Most users will want to operate in GET and PROMPT modes. However, if it is desired to duplicate the WAND commands exactly, then NOGET and NOPROMPT modes should be invoked.

PROMPT/NOPROMPT mode affects processing when storing or modifying any record, or when finding a record with a calculated key (FINDC). In PROMPT mode, the user is prompted with the name of every data-item in the record being STOREd or MODIFYd, or with the name of the data-item that is the key of the record being found. The user should respond with the value that he wants assigned to each data-item as he is promted with its name. A carriage return without typing in any value will leave the value of the data-item unchanged from what it is in the UWA. In NOPROMPT

mode, the user must set values with the assignment statement prior to execution of the STORE, MODIFY or FINDC.

DISPLAY/NODISPLAY mode controls automatic display of the values of all data-items in a record whenever a record is found (FINDC, FINDD, FINDPO, FINDAP, FINDO). For example, the example that was used to illustrate the REPEAT command can be reduced (in GET and DISPLAY mode) to:

```
FINDPO NEXT STUDNS STUDNT;REPEAT 50
```

LABEL/NOLABEL mode controls the format of the DISPLAY. In NOLABEL mode, the values of all data-items that are to be displayed are contained on a single line. In LABEL mode, the name of each data-item is displayed, followed by the value of that data-item. Each data item so displayed takes one line.

STATUS. The status of DBLOOK can be displayed at anytime by typing STATUS.

TO. The status can be changed with the TO command.

```
TO mode
```

or

```
TO generic value
```

where mode is one of the eight modes, generic one of seven generic character names, one of two format names, or REPEAT , and value is an appropriate value for the generic. For example:

```
TO NOPROMPT
```

or

```
TO QUOTE /
```

or

```
TO ASTERISK  :
```

F.5 DBLOOK COMMANDS AND ABBREVIATIONS

The following can also be obtained by typing COMMAND.

COMMAND	CO	CURRNT	CU	DBCLOS	DBC	DBOPEN	DBO
DELETE	DE	DISPLAY	DI	ERRSTA	ER	EXIT	EX
FINDAP	FAP	FINDC	FC	FINDD	FD	FINDO	FO
FINDPO	FPO	GET	G	HELP	H	ITEMS	I
MODIFY	M	RECORDS	REC	REPEAT	REP	SETCUR	SETC
SETS	SETS	STATUS	STA	STORE	STO	TO	T
@	@						

APPENDIX G

WAND RESERVED WORDS

THE FOLLOWING MAY NOT BE USED FOR USER DEFINED NAMES.

ACTUAL	ALIAS	ALL	ALLOWED
ALTER	ALWAYS	ARE	AREA
AREA-CODE	AREA-ID	ASC	ASCENDING
AUTOMATIC	AUTO	BIN	BINARY
BIT	BY	CALC	CALL
CHAIN	CHAR	CHARACTER	CHECK
CLOSE	COMPLEX	CURRENT	DATABASE
DATABASE-KEY	DBKEY	DEC	DECIMAL
DECODING	DELETE	DESC	DESCENDING
DIRECT	DISPLAY	DUP	DUPLICATES
DYNAMIC	ENCODING	EXCL	EXCLUSIVE
FIND	FIRST	FIXED	FLOAT
FOR	GET	IN	INDEX
INDEXED	INSERT	IS	KEY
LAST	LINKED	LOC	LOCACTION
LOCK	LOCKS	MAND	MANDATORY
MANUAL	MEMBER	MEMBERS	MODE
MODIFY	NAME	NEXCL	NEXT
NON-EXCLUSIVE	NOT	OCCURRENCE	OCCURS
OF	ON	ONLY	OPEN
OPT	OPTIONAL	OR	ORDER
OWNER	PAGE	PAGES	PIC
PICTURE	POINTER-ARRAY	PTR	PRIOR
PRIVACY	PROC	PROCEDURE	PROT
PROTECTED	RANGE	REAL	RECORD
RECORD-NAME	REMOVE	RESULT	RETR
RETRIEVAL	SCHEMA	SEARCH	SELECTION
SELECTIVE	SET	SIZE	
SORTED	SOURCE	STORE	SYSTEM
TEMP	TEMPORARY	THRU	TIMES
TO	TYPE	UPDATE	USAGE
USING	VALUE	VIA	VIRTUAL
WITHIN	WORDS		

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation Center (12)
Cameron Station
Alexandria, VA 22314

Office of Naval Research (6)
Arlington, VA 22217

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Illinois 60605

New York Area Office
715 Broadway - 5th Floor
New York, NY 10003

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington, DC 20380

Office of Naval Research
Code 458
Arlington, VA 22217

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation & Mathematics Dept.
Bethesda, MD 20084

Mr. Kim B. Thompson
Technical Director
Information Systems Division
(OP-911G)
Office of Chief of Naval Operations
Washington, DC 20350

Professor Omar Wing
Columbia University
Dept of Electrical Engineering
and Computer Science
New York, NY 10027

Office of Naval Research (2)
Information Systems Program
Code 437
Arlington, VA 22217

Office of Naval Research
Code 1021P Branch Office, Boston
495 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106

Naval Research Laboratory (6)
Technical Information Division
Code 2627
Washington, DC 20375

Office of Naval Research
Code 455
Arlington, VA 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
(OP-916D)
Office of Chief of Naval Operations
Washington, DC 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick, NJ 08903
Attn: Dr. Henry Voos