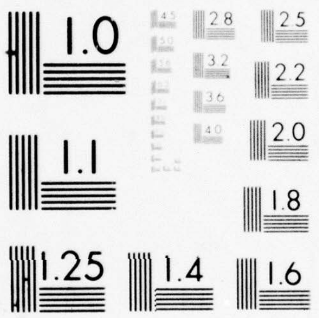
AD-A037 335 WHARTON SCHOOL OF FINANCE AND COMMERCE PHILADELPHIA P--ETC F/G 9/2
WAND USER'S GUIDE. ADDENDUM. (U)
MAY 76 M D ZISMAN N00014-75-C-0462
UNCLASSIFIED 76-05-08 NL

1 OF 1
AD
A037335

END
DATE
FILMED
4-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12
B.S.

WAND USER'S GUIDE ADDENDUM

Michael D. Zisman

Working Paper

76-05-08

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA. 19174

May 21, 1976
~~Draft #1~~

This document will be integrated with the next version of the Wand User's Guide, Working Paper 76-01-03.

NR

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 76-05-08	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 WAND User's Guide. Addendum.		5. TYPE OF REPORT & PERIOD COVERED 9 Final rept.,
		6. PERFORMING ORG. REPORT NUMBER 14 76-Ø5-Ø8 ✓
7. AUTHOR(s) 10 Michael D./Zisman		8. CONTRACT OR GRANT NUMBER(s) 15 NØØØ14-75-C-Ø462 ✓
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 Technical report
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Information Systems Arlington, Virginia 22217		12. REPORT DATE 5/76
		13. NUMBER OF PAGES 22
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 11 21 May 76		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited 12 21p.

D D C
MAR 25 1976
C

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

Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Network structures
Plex structures
DBTG (Database Task Group)
DBMS (Database Management Systems)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A description of the COBOL based data manipulation language (DML) of the WAND system. Includes descriptions of all commands as well as examples of their use.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE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

408 757

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1.0 COBOL DATA MANIPULATION LANGUAGE

1.1 General Description

The DML for COBOL is specified in the April '71 DBTG report. WAND supports a subset of this specification. The DML specifies additions to the Identification division, DATA division, and PROCEDURE division of standard COBOL.

A COBOL program with the DML statements inserted is first processed by a WAND system program named WNDCBL. This program translates the DML statements to COBOL call statements to the WAND system. Upon completion of this translation, the user has the option of automatically linking to the COBOL compiler and then the loader. Examples of using the COBOL/WAND implementation can be found in the next section of this document.

In the present implementation of WAND, all DML statements must be on a line by themselves in COBOL programs and not intermixed with non DML statements. For example, the following would be improper:

```
IF A=B GET STUREC
```

whereas the following would be correct:

```
IF A=B
GET STUREC.
```

DML statements need not be entirely specified on one line, however.

1.2 IDENTIFICATION Division

If a privacy key is specified in the schema definition, then this privacy key must be specified in all programs which interact with the database. This is communicated to WAND via the PRIVACY KEY statement in the IDENTIFICATION division.

[<u>PRIVACY</u> <u>KEY</u> IS priv-key].

Priv-key must be either a literal enclosed in quotes or a variable which contains the privacy key. If a variable is specified, it must be at least 10 characters in length and have usage display-7. Examples are:

```
PRIVACY KEY 'GOAWAY'.
PRIVACY KEY IS KEY-AREA.
```

1.3 DATA Division

A new section named the SCHEMA section must be included in all COBOL programs which interact with WAND. Furthermore, the SCHEMA section must be the first section in the DATA division. Syntax is as follows:

```
DATA DIVISION.
SCHEMA SECTION.
INVOKE SCHEMA schema-name.
```

Schema-name is the name of the schema and of the database. WNDCBL will expect the database to be in the file named schema-name.DB, the schema itself to be in schema-name.SCH, and the user work area to be in schema-name.WKC. WNDCBL will automatically insert a linkage section at the end of the DATA division and copy in the user work area from schema-name.WKC.

1.4 PROCEDURE Division

To communicate with WAND, the user writes DML statements into the PROCEDURE divsion of his program. These statements will be discussed individually.

1.4.1 Opening The Database -

Before any other DML statements can be executed, the database must be opened. This is done with the OPEN AREA statement.

```
OPEN AREA schema-name [USAGE-MODE is {UPDATE}]
                                    {RETRIEVAL}
```

If USAGE-MODE is not specified, retrieval is assumed.

1.4.2 CLOSING The Database -

When all processing is done, the database should be closed with the CLOSE statement. This will direct WAND to empty the incore buffers and terminate processing.

```
CLOSE AREA schema-name
```

1.4.3 STORING Records Into The Database -

Records are stored into the database with the STORE statement. Linkage in all sets is performed automatically by WAND.

<u>STORE</u> record-name

1.4.4 DELETING Record -

A records is deleted from the database with use of the DELETE statement. A record type must be specified in the delete statement; the current of that record type is deleted from the database.

<u>DELETE</u> record-name.

1.4.5 MODIFYING Records -

To replace an existing record in the database, the MODIFY statement is used. When a MODIFY statement is executed, the record type specified is moved from the UWA into the WAND buffers for storage into the database.

<u>MODIFY</u> record-name

1.4.6 RETRIEVING A Record -

After a record is found via a FIND statement, it is not automatically moved into the UWA. To accomplish this, a GET statement is executed.

<u>GET</u> record-name

After the GET statement is executed, the record is available for user processing.

1.4.7 The FIND Statement -

Navigating through the database is done with the FIND statement. Using the FIND statement, it is possible to access records by their physical location, via calculated keys, via set relationships, or by processing the database sequentially. Each of these methods is discussed below.

1.4.7.1 Access Via Calculated Key -

In order to locate a record via calc key, the record must be defined in the schema with a LOCATION MODE CALC clause. Before executing the FIND statement, the user must place the value of the key of the desired record into the key location field of the proper record in the UWA.

FIND [NEXT DUPLICATE WITHIN] record-name RECORD

If the next duplicate clause is not present, WAND will search the database for the first record with the appropriate key field value. If the next duplicate clause is present, WAND will search the database for the next record in a group of records with the same key. The next duplicate clause cannot be used if NO DUPLICATES ALLOWED was specified in the schema definition.

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

1.4.7.2 Location Via Set Relationships -

A set consists of an owner and one or more members. It is possible to navigate from an owner to a member and along the path of members; it is also possible to navigate from a member record to its owner. There exists a form of the FIND statement for each of these possibilities.

1.4.7.2.1 Navigation From Member To Owner -

To find an owner of a record which participates in a set relationship, currency must be established at the proper member record. The following FIND statement is then issued to find the owner:

FIND OWNER RECORD OF set-name SET

This will establish currency of the owning record type at the owning record of set-name set.

1.4.7.2.2 Navigating From Owner To Member -

To navigate from an owner record to a member record of a set, the following form of the FIND is used:

FIND position [record-name] RECORD OF set-name SET

Position must be one of the following:

1. FIRST : The database will be searched for the first record of set-name set.

2. LAST: The database will be searched for the last record type of set-name set.

3. NEXT: The database will be searched for the next record after the current record of set-name set. If current of set-name is the last record of the set, error status will be set to 0307.

4. PRIOR: The database will be searched for the record prior to the current record of set-name set. If the current of set-name is the first record in the set, error status will be set to 0307.

5. Integer constant(e.g. +5 or -2): The database will be searched for the +Nth or -Nth record of set-name set relative to the current record of set-name set. An error will occur if no such record exists.

6. Variable name: If a COBOL variable is specified here, it must be either a one word COMP item or a display-7 5 character item. If it is character, it's value must be one of "FIRST", "LAST", "NEXT", or "PRIOR". If it is a COMP item it's value is treated the same way as 5 above.

1.4.7.3 Sequential Processing Of The Database. -

To process the database sequentially for all records or records of a specific record type, the following form of the FIND is used:

FIND position [record-name] RECORD OF schema-name AREA

Position must have one of the values described in the previous sub-section. If record-name is specified, only records of that type will be found. If record-name is not specified, all record in the database are eligible for selection. Schema-name is required and must be the name of the schema. The user should not assume that the records are in any particular order. This form of the FIND is useful when all records of a given type must be processed and order is not important.

1.4.7.4 Access Through Database Key -

This form of the FIND statement requires that the user know the exact physical location of the desired record. It should only be used by experienced programmers to establish or re-establish currency.

```
FIND [record-name] USING dbid
```

Record-name is optional and not required. Dbid is required and must be an integer (COMP) variable. Its value must be of the form PLL where P is database page number and LL is line number within the page.

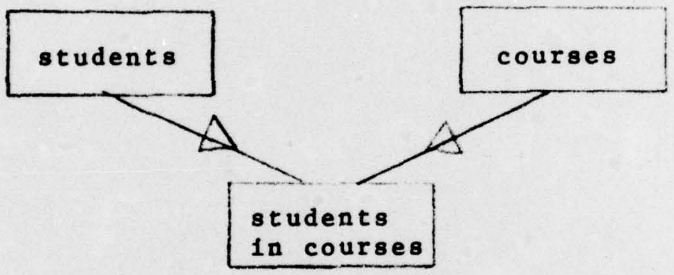
If the record is found all currency updates are made.

2.0 COMPREHENSIVE EXAMPLE

2.1 Introduction

In this section an entire WAND application will be presented as a tutorial and example on how to use WAND.

Consider a simple student records system. We wish to maintain data about students, data about courses being offered, and data about students taking courses. The proper data structure for this application is a confluent hierarchy:



The STU record will contain personal data about each student and the key would normally be social security number or student number. This record would contain all of the personal data about the student such as name, academic status, etc. The COURSE record will contain data describing each course, such as course number, title, credits, instructor, etc. The STUCOR record would contain data about a student taking a particular course. This would include grade, pass/fail indicator, etc. For our example we will only use a small number of data elements. The STU record will contain last name (LNAME) and title (TITLE). The course record will contain course number (CNO), credit hours (CREDIT), and course title (CTITLE). The student-course record (STUCOR) will contain only grade (GRADE).

Both the STU record and COR record will be accessable via calculated keys. The key of the STU record will be LNAME and the key to the COR record will be CNO.

2.2 Schema Creation

The first step in building the database is to write a schema definition using the data definition language described is section 1 of this document. The database will be called STUREC and the schema is shown in figure 1.

```
SCHEMA STUREC
PRIVACY KEEPOUT.

RECORD STU LOCATION CALC USING LNAME DUPLICATES NOT ALLOWED
          LNAME TYPE CHARACTER 10
          TITLE TYPE CHARACTER 5.

RECORD COR LOCATION CALC USING CNO DUPLICATES NOT ALLOWED
          CNO TYPE CHARACTER 5
          CREDIT TYPE FIXED
          CTITLE TYPE CHARACTER 20.

RECORD STUCOR LOCATION VIA SC
          GRADE TYPE CHARACTER 2.

SET SC MODE CHAIN ORDER LAST
          OWNER STU
          MEMBER STUCOR.

SET CS MODE CHAIN ORDER LAST
          OWNER COR
          MEMBER STUCOR.
```

Figure 1 - schema for STUREC

After the schema is entered into file STUREC.DDL (extension must be DDL), the WAND schema processor is invoked. The name of this program is FDP and it is invoked at monitor level by:

```
X FDP .
```

FDP will ask for the name of the schema and then process the file schema-name.DDL . It will produce the following files:

1. schema-name.WRK - the FORTRAN work area

2. schema-name.WKC - the COBOL work area

3. schema-name.SCH - used by WAND itself.

4. schema-name.ERR - list of errors found by FDP.

FDP will print on the terminal the number of errors which were found in the schema definition. If this is not zero, the user should examine the ERR file and correct the schema definition. If there are no errors, the schema has been accepted by WAND and the user can now initialize the database. This is accomplished by executing DBINIT as follows:

```
X DBINIT .
```

DBINIT will ask for the name of the schema and then proceed to initialzie the database.

The size of the database created by DBINIT will be determined by the number of pages and page size specified in the schema definition. For this reason, the FDP program must be run successfully before the database can be created with DBINIT. DBINIT will initialize a database that already exists. If DBINIT is asked to initialize a database that already exists, it will print a message at the terminal informing the user that he is attempting to destroy an existing database. The user must then confirm his desire for DBINIT to proceed.

2.3 Loading The Database

After the database is created, we can write programs to load the database and subsequently retrieve data from the database. WAND supports both FORTRAN and COBOL as host languages. To this end, FDP produces both a FORTRAN work area definition (.WRK file) and a COBOL work area definition (.WKC file, see figure 2). For the rest of this example, we will use COBOL as the host language; hence we will utilize the .WKC file.

Data Manipulation Language for COBOL is fully specified in the DBTG Report of April, 1971. WAND supports a subset of this specification. The user communicating with WAND through a COBOL program writes DML statements as described in section 4 of this document. The COBOL program with these DML statements is then processed by a WAND system program named WNDCBL. This program accecpts the user's COBOL program as input and produces as output a COBOL program with the DML statements translated to call statements to the WAND system.

When WNDCBL is executed, it will ask for an input file and output file. It is recommended that input files have the extension WCB and that output files have the standard CBL extension. When the input file is requested by WNDCBL, an extension of WCB is assumed if no extension is given. For this reason, files with no extension cannot be processed. When the output file is requested, the user can specify any valid DEC-10 file name or simply depress the carriage-return. In the latter case (i.e. no filename is entered) the output file will have the same filename as the input file and the extension CBL. To summarize, it is recommended that the user write his/her program into a file with extension WCB. When running WNDCBL, the user should enter only the filename (and not the extension) when asked for input file. When asked for the output file name the user should <CR>. The output file will then have the name of the input file and the extension CBL.

It is possible to compile the WNDCBL output file and exectute it directly from WNDCBL. When WNDCBL has created an output file suitable for the COBOL compiler, it will ask if the file is to be compiled. If the user response to "COMPILE ?" is "N", WNDCBL will terminate and the user may compile the program when convenient. If the user replies "Y" to the above request, he will then be asked if the compiled program should be loaded and executed. If the user replies "Y", the program will be compiled, loaded with the proper Wand subroutines, And execution will commence. If the user replies no to this question the program will be compiled but execution will not be attempted. A sample session is as follows:

```
.X WNDCBL
INFILE: BUILD
OUTFILE:
COMPILE ? Y
EXECUTE ? Y
COBOL: BUILD    [BUILD.CBL]
LINK:  LOADING
[LNKXCT MAINLE EXECUTION]
..execution follows
```

The BUILD program loads the database (see figure 3). It allows the user to load student records, course records, and stucor records. The output of WNDCBL, which has the DML COBOL statements converted to WAND calls is shown in figure 4. Note the entry statement immediately after the PROCEDURE DIVISION statement. This is because BUILD will actually be a COBOL subroutine which is called by a program named CBSTRT (for COBOL start). This program must be loaded with all COBOL programs which are to access WAND databases. CBSTRT is a very simple program which establishes the linkage between your COBOL program and the WAND FORTRAN programs. If the user replies yes to the compile and execute questions in WNDCBL then he need not be concerned with CBSTRT or other details of WAND. For this reason, that is the recommended action.

If the user does not have WNDCBL initiate execution, or if the COBOL program does not have to be compiled, then the user must load his program and the other necesary routines which his program needs to execute successfully. This is accomplished by executing the program WANDGO. WANDGO will ask for the name of your program, and then load it along with the necessary WAND routines. This would be done As follows:

```
.X WANDGO
PROGRRAM NAME: BUILD
LINK:  LOADING
[LNKXCT EXWAND EXECUTION]
..execution follows
```

WANDGO can be used to initiate execution of WAND programs (COBOL or FORTRAN) as long as there is only one user program to be loaded. If the user must load more than one program (i.e. several subprograms) then he must load the proper WAND routines himself. This is done by executing the following command string at monitor level:

```
.EX CBSTRT[4010,51],your-programs,DML[4010,51]/SEA
```

CBSTRT is alway required for COBOL programs and MUST be the first program specified in the EX command. Your-programs is, in this case, the name of the user program to be executed. DML is the library which contains all of the WAND routines which will be required by BUILD. The /SEA is a switch which tells the loader (LINK-10) that not all programs in DML are to be loaded, but only those requested by your-programs. If the user is debugging the COBOL program, the DEBUG command should be used instead of EX.

2.4 Retrieval Of Data

After executing BUILD, data has been stored in the database STUREC.DB. We are now prepared to retrieve data from the database. A common requirement would be an online program which, given a student name as input, will print data about all of his/her courses. RETR (see figure 5) is an example of such a program. It accepts as input a student's last name. The last name is put in LNAME of STU and then a FIND statement is performed to establish "currency" at the proper STU record. By retrieving all members in the SC set, we can get all of the STUCOR records owned by this particular STU record in the SC set. For each STUCOR record we FIND, we can FIND OWNER in the CS set and thus GET the COR record which has the course number and, title. WE can then print the course number title, and grade for each STUCOR record. This technique for processing confluent hierarchies is a very common practice. The unfamiliar reader should carefully study figure 5. The output of WNDCBL for RETR is shown in figure 6. Notice at the end of the program several entry statements have been added. These have no effect on the execution of the program and are inserted only to control the action of LINK-10.

2.5 Sample Output And Programs

```
01 DB-LINKAGE.
        05 FLAGS.
                10 ERRSTA PIC S9(5) COMP.
                10 FILLER PIC X(20) USAGE DISPLAY-7.
        05 STU.
                10 LNAME PIC X(10) USAGE DISPLAY-7.
                10 TITLE PIC X(5) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 COR.
                10 CNO PIC X(5) USAGE DISPLAY-7.
                10 CREDITS PIC S9(5) COMP.
                10 CTITLE PIC X(20) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 STUCOR.
                10 GRADE PIC X(2) USAGE DISPLAY-7.
        05 SETS.
                10 SC PIC S9(5) COMP.
                10 CS PIC S9(5) COMP.
```

Figure 2 - STUREC.WKC file produced by FDP

```
IDENTIFICATION DIVISION.
PROGRAM-ID.BUILD.
PRIVACY KEY IS "KEEPOUT".
ENVIRONMENT DIVISION.
DATA DIVISION
SCHEMA SECTION.
INVOKE SCHEMA STUREC.
WORKING-STORAGE SECTION.
77 TRAN-CODE PIC X USAGE DISPLAY-7.
PROCEDURE DIVISION.
        OPEN AREA STUREC USAGE-MODE UPDATE.
        IF ERRSTA NOT = 0
                DISPLAY 'OPEN ERROR ' ERRSTA
                STOP RUN.
        PERFORM PROCESS UNTIL TRAN-CODE = '0'.
        CLOSE AREA STUREC.
        IF ERRSTA NOT = 0
        DISPLAY 'CLOSE ERROR ' ERRSTA.
        STOP RUN.
PROCESS.
        DISPLAY 'TRANSACTION CODE: ' WITH NO ADVANCING.
        ACCEPT TRAN-CODE.
        IF TRAN-CODE ='1' PERFORM TRAN1
        ELSE IF TRAN-CODE ='2' PERFORM TRAN2
        ELSE IF TRAN-CODE='3' PERFORM TRAN3
        ELSE IF TRAN-CODE NOT ='0' DISPLAY 'INVALID TRANSACTION
CODE!!'.
TRAN1.
        DISPLAY 'NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        DISPLAY 'TITLE: ' WITH NO ADVANCING.
        ACCEPT TITLE.
        STORE STU.
        IF ERRSTA NOT = 0
                DISPLAY 'TRAN1 STORE ERROR ' ERRSTA.
TRAN2.
        DISPLAY 'COURSE NUMBER: ' WITH NO ADVANCING.
        ACCEPT CNO.
        DISPLAY 'CREDIT HOURS: ' WITH NO ADVANCING.
        ACCEPT CREDITS.
        DISPLAY 'COURSE TITLE: ' WITH NO ADVANCING.
        ACCEPT CTITLE.
        STORE COR.
        IF ERRSTA NOT = 0
                DISPLAY 'TRAN2 STORE ERROR ' ERRSTA.
TRAN3.
        DISPLAY 'NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        DISPLAY 'COURSE NUMBER: ' WITH NO ADVANCING.
        ACCEPT CNO.
        FIND STU.
        IF ERRSTA NOT = 0
```

```
                DISPLAY 'TRAN3 FIND STU ERROR ' ERRSTA
        ELSE
                FIND COR
                IF ERRSTA NOT = 0
                        DISPLAY 'TRAN3 FIND COR ERROR ' ERRSTA
                ELSE
                        DISPLAY 'GRADE: ' WITH NO ADVANCING
                        ACCEPT GRADE
                        STORE STUCOR
                        IF ERRSTA NOT = 0
                                DISPLAY 'TRAN3 STORE ERROR '
ERRSTA.
```

Figure 3 - BUILD program with DML statements

```
IDENTIFICATION DIVISION.
PROGRAM-ID.BUILD.
*PRIVACY KEY IS "KEEPOUT".
ENVIRONMENT DIVISION.
DATA DIVISION
*SCHEMA SECTION.
*INVOKE SCHEMA STUREC.
WORKING-STORAGE SECTION.
77 TRAN-CODE PIC X USAGE DISPLAY-7.
LINKAGE SECTION.
01 DB-LINKAGE.
        05 FLAGS.
                10 ERRSTA PIC S9(5) COMP.
                10 FILLER PIC X(20) USAGE DISPLAY-7.
        05 STU.
                10 LNAME PIC X(10) USAGE DISPLAY-7.
                10 TITLE PIC X(5) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 COR.
                10 CNO PIC X(5) USAGE DISPLAY-7.
                10 CREDITS PIC S9(5) COMP.
                10 CTITLE PIC X(20) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 STUCOR.
                10 GRADE PIC X(2) USAGE DISPLAY-7.
        05 SETS.
                10 SC PIC S9(5) COMP.
                10 CS PIC S9(5) COMP.
PROCEDURE DIVISION.
        ENTRY COB USING DB-LINKAGE.
*       OPEN AREA STUREC USAGE-MODE UPDATE.
        CALL WANDA USING "DBOPEN","STUREC    ","KEEPOUT   ",1.
        IF ERRSTA NOT = 0
                DISPLAY 'OPEN ERROR ' ERRSTA
                STOP RUN.
        PERFORM PROCESS UNTIL TRAN-CODE = '0'.
*       CLOSE AREA STUREC.
        CALL WANDA USING "DBCLOS".
        IF ERRSTA NOT = 0
        DISPLAY 'CLOSE ERROR ' ERRSTA.
        STOP RUN.
PROCESS.
        DISPLAY 'TRANSACTION CODE: ' WITH NO ADVANCING.
        ACCEPT TRAN-CODE.
        IF TRAN-CODE ='1' PERFORM TRAN1
        ELSE IF TRAN-CODE ='2' PERFORM TRAN2
        ELSE IF TRAN-CODE='3' PERFORM TRAN3
        ELSE IF TRAN-CODE NOT ='0' DISPLAY 'INVALID TRANSACTION
CODE!!'.
TRAN1.
        DISPLAY 'NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        DISPLAY 'TITLE: ' WITH NO ADVANCING.
        ACCEPT TITLE.
*       STORE STU.
        CALL WANDA USING "DBSTOR",STU.
        IF ERRSTA NOT = 0
                DISPLAY 'TRAN1 STORE ERROR ' ERRSTA.
TRAN2.
        DISPLAY 'COURSE NUMBER: ' WITH NO ADVANCING.
        ACCEPT CNO.
        DISPLAY 'CREDIT HOURS: ' WITH NO ADVANCING.
        ACCEPT CREDITS.
        DISPLAY 'COURSE TITLE: ' WITH NO ADVANCING.
        ACCEPT CTITLE.
*       STORE COR.
        CALL WANDA USING "DBSTOR",COR.
        IF ERRSTA NOT = 0
                DISPLAY 'TRAN2 STORE ERROR ' ERRSTA.
TRAN3.
        DISPLAY 'NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        DISPLAY 'COURSE NUMBER: ' WITH NO ADVANCING.
        ACCEPT CNO.
*       FIND STU.
        CALL WANDA USING "FINDC",STU,"FIRST".
        IF ERRSTA NOT = 0
                DISPLAY 'TRAN3 FIND STU ERROR ' ERRSTA
        ELSE
*               FIND COR
                CALL WANDA USING "FINDC",COR,"FIRST"
                IF ERRSTA NOT = 0
                        DISPLAY 'TRAN3 FIND COR ERROR ' ERRSTA
                ELSE
                        DISPLAY 'GRADE: ' WITH NO ADVANCING
                        ACCEPT GRADE
*                       STORE STUCOR
                        CALL WANDA USING "DBSTOR",STUCOR
                        IF ERRSTA NOT = 0
                                DISPLAY 'TRAN3 STORE ERROR '
ERRSTA.
        ENTRY DBGET.
        ENTRY DBMODI.
        ENTRY DBDELE.
        ENTRY FINDPO.
        ENTRY FINDD.
        ENTRY FINDO.
        ENTRY FINDAP.
```

Figure 4 - BUILD programs with DML statements converted to calls

```
IDENTIFICATION DIVISION.
PROGRAM-ID.RETR.
PRIVACY KEY IS "KEEPOUT".
ENVIRONMENT DIVISION.
DATA DIVISION.
SCHEMA SECTION.
INVOKE SCHEMA STUREC.
PROCEDURE DIVISION.
        OPEN AREA STUREC USAGE-MODE RETRIEVAL.
        IF ERRSTA NOT = 0
                DISPLAY 'OPEN ERROR ' ERRSTA
                STOP RUN.
        DISPLAY 'STUDENT NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        PERFORM PROCESS UNTIL LNAME=SPACES.
        CLOSE AREA STUREC.
        IF ERRSTA NOT = 0
        DISPLAY 'CLOSE ERROR ' ERRSTA.
        STOP RUN.
PROCESS.
        FIND STU RECORD.
        IF ERRSTA NOT = 0
                DISPLAY 'STUDENT FIND ERROR ' ERRSTA
                MOVE 0 TO ERRSTA
        ELSE PERFORM GETCOR UNTIL ERRSTA NOT = 0
                IF ERRSTA NOT = 307
                        DISPLAY 'GET COURSE ERROR ' ERRSTA.
        DISPLAY 'STUDENT NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        MOVE 0 TO ERRSTA.
GETCOR.
        FIND NEXT STUCOR RECORD SC SET.
        IF ERRSTA  = 0
                GET STUCOR
                IF ERRSTA  = 0
                        FIND OWNER CS SET
                        IF ERRSTA=0
                                GET COR
                                IF ERRSTA =0 DISPLAY
CNO,CTITLE,GRADE.
```

Figure 5 - RETR programs with DML statements

```
IDENTIFICATION DIVISION.
PROGRAM-ID.RETR.
*PRIVACY KEY IS "KEEPOUT".
ENVIRONMENT DIVISION.
DATA DIVISION.
*SCHEMA SECTION.
*INVOKE SCHEMA STUREC.
LINKAGE SECTION.
01 DB-LINKAGE.
        05 FLAGS.
                10 ERRSTA PIC S9(5) COMP.
                10 FILLER PIC X(20) USAGE DISPLAY-7.
        05 STU.
                10 LNAME PIC X(10) USAGE DISPLAY-7.
                10 TITLE PIC X(5) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 COR.
                10 CNO PIC X(5) USAGE DISPLAY-7.
                10 CREDITS PIC S9(5) COMP.
                10 CTITLE PIC X(20) USAGE DISPLAY-7.
                10 FILLER PIC X(5) USAGE DISPLAY-7.
        05 STUCOR.
                10 GRADE PIC X(2) USAGE DISPLAY-7.
        05 SETS.
                10 SC PIC S9(5) COMP.
                10 CS PIC S9(5) COMP.
PROCEDURE DIVISION.
        ENTRY COB USING DB-LINKAGE.
*       OPEN AREA STUREC USAGE-MODE RETRIEVAL.
        CALL WANDA USING "DBOPEN","STUREC    ","KEEPOUT   ",0.
        IF ERRSTA NOT = 0
                DISPLAY 'OPEN ERROR ' ERRSTA
                STOP RUN.
        DISPLAY 'STUDENT NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        PERFORM PROCESS UNTIL LNAME=SPACES.
*       CLOSE AREA STUREC.
        CALL WANDA USING "DBCLOS".
        IF ERRSTA NOT = 0
        DISPLAY 'CLOSE ERROR ' ERRSTA.
        STOP RUN.
PROCESS.
*       FIND STU RECORD.
        CALL WANDA USING "FINDC",STU,"FIRST".
        IF ERRSTA NOT = 0
                DISPLAY 'STUDENT FIND ERROR ' ERRSTA
                MOVE 0 TO ERRSTA
        ELSE PERFORM GETCOR UNTIL ERRSTA NOT = 0
                IF ERRSTA NOT = 307
                        DISPLAY 'GET COURSE ERROR ' ERRSTA.
        DISPLAY 'STUDENT NAME: ' WITH NO ADVANCING.
        ACCEPT LNAME.
        MOVE 0 TO ERRSTA.
GETCOR.
*       FIND NEXT STUCOR RECORD SC SET.
        CALL WANDA USING "FINDPO","NEXT ",SC,STUCOR.
        IF ERRSTA  = 0
*               GET STUCOR
                CALL WANDA USING "DBGET",STUCOR
                IF ERRSTA  = 0
*                       FIND OWNER CS SET
                        CALL WANDA USING "FINDO",CS
                        IF ERRSTA=0
*                               GET COR
                                CALL WANDA USING "DBGET",COR
                                IF ERRSTA =0 DISPLAY
CNO,CTITLE,GRADE.
        ENTRY DBSTOR.
        ENTRY DBMODI.
        ENTRY DBDELE.
        ENTRY FINDD.
        ENTRY FINDAP.
```

Figure 6 - RETR program with DML statements coverted to calls

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 102IP 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

FILM

4