A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community nd federal, state and local governments.

Although portions of this report are not reproducible, it is being made available in microfiche to facilitate the availability of those parts of the document which are legible.

LA-UR-83-2980

CONF-8310182--3

LA-UR--83-2980

DE84 001707

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: MULTIDOMAIN, MULTIRECORD-TYPE DATATRIEVE-11 DATABASES

AUTHOR(S) Robert R. Horning

NOTICE
PORTIONS OF THIS REPORT ARE ILLEGIBLE.
It has been reproduced from the best available copy to permit the broadest possible availability.

MN ONLY

SUBMITTED TO 1983 Fall DECUS U. S. Symposium, Las Vegas, NV, October 24-28, 1983

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

MASTER

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO 836 R4
ST NO 2629 5 81

MULTIDOMAIN, MULTIRECORD-TYPE DATATRIEVE-11 DATA BASES

Robert R. Horning
Los Alamos National Laboratory
Los Alamos, New Mexico

ABSTRACT

Data bases consisting of multiple domains and multirecord-type domains present special problems in design, loading, maintenance, and report generation. The logical association of records is a fundamental concern in all these problem areas. This paper describes techniques for dealing with this and other specifics using Datatrieve-11, Sort-11, FORTRAN 77, and the RSX-11M Indirect Command File Processor.

INTRODUCTION

Hierarchical data bases can be simulated using the OCCURS* clause and its variant the OCCURS DEPENDING. However, these will result in much wasted disk space if the number of OCCURences has a wide range and if the lengths and numbers of fields are large. This difficulty can be largely overcome by defining data bases consisting of single domains with several record redefinitions or multiple domains whose records are linked by pointers.

Retrieval of single records from moderate-sized files (5000-10000 records) using Datatrieve-11 is generally rapid enough to avoid operator fatigue, especially if keyed access is used. Similarly, reports made from these files usually can be produced within several minutes. However, creating comprehensive reports from multiple files or multirecord-type files may take unacceptable amounts of time --- ranging into many hours. (We have had some reports that took more than a day.) Compiled programs to replace Datatrieve procedures can provide multiple-order-of-magnitude reductions in the time required to create reports from such data bases and also overcome the Datatrieve-11 pool problems associated with simultaneously readying several domains.

MULTIRECORD DOMAINS

A hierarchical data base can be simulated using the OCCURS and OCCURS DEPENDING clauses to store the "children" data in the same record as the "parent". If the number of children fields actually used is fairly consistent from record to record, this method provides a convenient and efficient data organization.

However, if the number of children varies considerably from parent to parent, but the number of bytes required for each child is about the same as for the parent, more efficient use of mass storage devices can be achieved using the REDEFINES clause to create separate children records for each parent within the same file. Now the number of children can vary from zero to an arbitrarily large number with little or no penalty in wasted disk space.

Figures 1 and 2 illustrate the use of the multirecord type domain. Figure 1 shows the front and back of a worksheet used to record data needed to install digital data communications services for one of our computer users. The worksheet is broken down into several sections. Not all sections need be used, and some of those that are may occur more than once.

Figure 2 shows the complete domain definition used to specify the organization of the worksheet's data in a file. This domain is called CWR for Communications Work Request. Note first that all records associated with a specific worksheet are assigned the same value in one field, CWR_NO, thus associating them with one another. To speed access to all records for a particular worksheet we make this field the primary key for the file corresponding to this domain.

The GENERAL record may be regarded as the parent in this hierarchical arrangement; all other records are children. Note that all children do not share the same record definition. They are all more or less similar in length; some may occur more than 20 times for one parent.

---

* Words appearing in UPPER CASE are either Datatrieve key words or field names within the data bases discussed in this paper.

A method of associating a record redefinition with a specific record is needed. The REC_TYP field performs this function. The record associated with the GENERAL section of the worksheet is assigned a REC_TYP value of 1; other sections are also given unique numeric REC_TYP entries.

Also needed is a way to assure that all fields in a new record are initialized appropriately even though the user may not explicitly enter data in them. Datatrieve will initialize fields according to the first record definition it finds, not according to the remaining field definitions in the current REDEFINES clause. The DUMMY_A1 field defined in figure 2 assures that all unused fields beyond SITE will be set to ASCII blanks. Of course this technique is inappropriate for numeric fields, but it provides a reasonable compromise.

Other, more effective, ways to initialize all fields include creating separate domain definitions for each record type within the file and using these exclusively for data entry, and/or defining data entry procedures that explicitly initialize all fields for a given record type. Either of these methods is, by itself, sufficient to guarantee appropriate initialization.

Separate domain definitions for each record in a multirecord type file can also help alleviate the well-known Datatrieve-11 pool problem because, individually, they are much shorter than the complete domain definition. They can be used individually, also, with corresponding data entry procedures and appropriate data validation tables, where the complete domain definition would cause difficulty. Figure 3 shows the domain definition for the GENERAL section alone, and figure 4 illustrates one data entry procedure used with this domain.

It is prudent to let Datatrieve, and not the user, assign values to fields that are used to maintain the internal organization of the data. In figures 2 and 3, these fields are CWR_NO, which identifies the parent and its children, and REC_TYP, which identifies the redefinition to be used for the specific record. Each set consisting of parent and children must be distinguished from all other sets. One way to maintain a unique identifier, in this case CWR_NO, is to store a number in a separate file with its corresponding domain definition. Each time a new parent is created, the number can be fetched, used, incremented, and replaced. The first few lines of the procedure in figure 4 do this. The record type identifier can simply be set in the data entry procedure, as in line 27 of figure 4.

Users often find it helpful to have a minimum number of procedures with which to deal while entering and updating data. One way of achieving this end is to combine STORE and MODIFY processes in the same procedure. Figure 4 illustrates how this can be done for the parent record. In this case, the user is assumed to know whether he wishes to modify an existing record or create a new one. However, he may not know whether a specific record exists and may not wish to bother finding out. Figure 5 illustrates one way of letting Datatrieve search for a record and, if the record is found, display its contents so the user can modify it; otherwise the record is created. Note the use of nested FOR clauses; they are very handy in applications such as this.

Often a user wishes to enter data for a parent and several children in one session. Ideally, one would include all the necessary code in a single procedure. Unfortunately, the Datatrieve-11 pool problem is likely

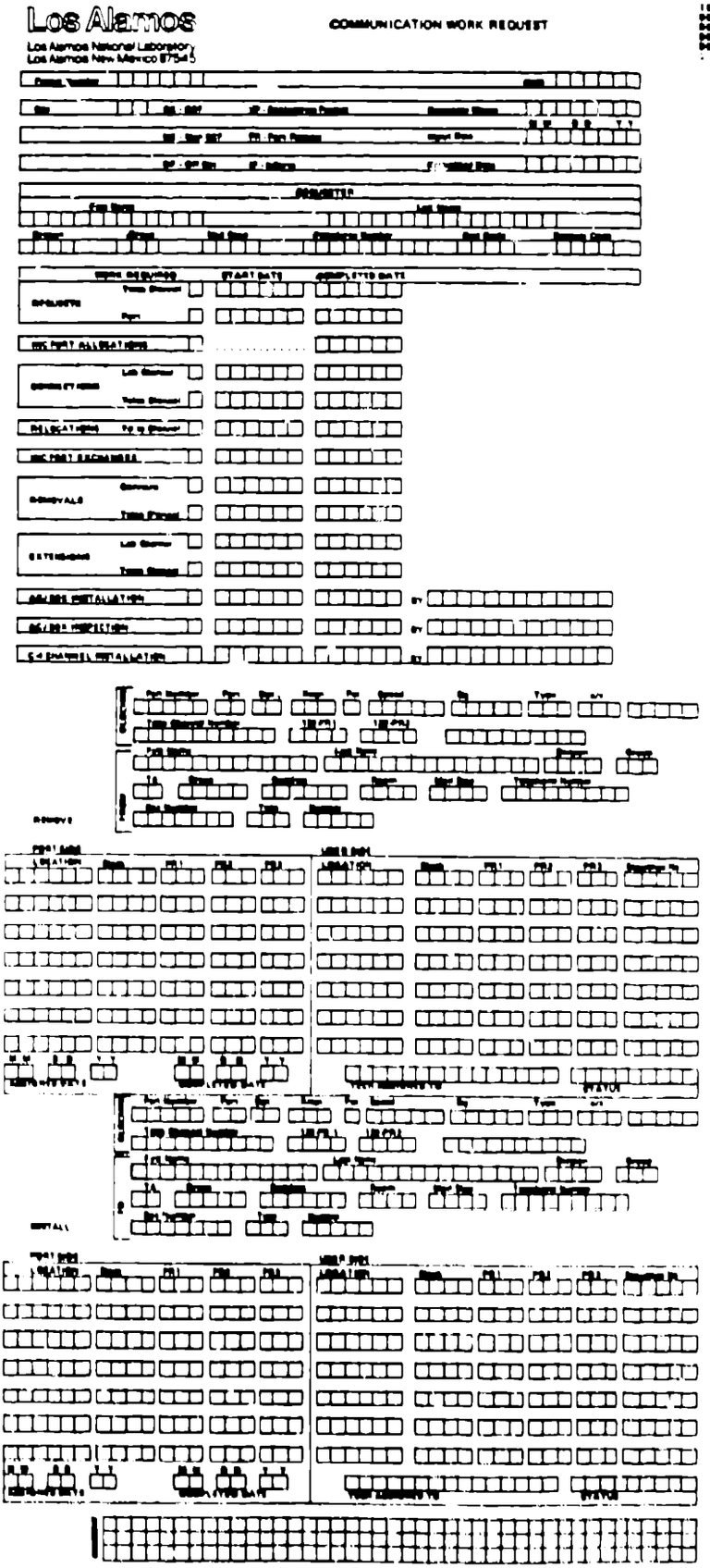

Figure 1. Communications Work Request

```
$DISK:[270.33CWR.CMD
DELETE CWR;
DELETE CWRREC;
DEFINE DOMAIN CWR USING CWRREC ON DUA1:[270.33CWR.DAT;
DEFINE RECORD CWRREC ORIGIN
;  CWR-REC
       05  CWR-NO          PIC IS 9(6)  USAGE IS INTEGER.
       05  REC-TYP         PIC IS 9(2)  USAGE IS INTEGER.
           10REC-TYP = 1      (++) GENERAL SECTION OF CWR ENDING SHEET
           10REC-TYP = 2      (++) SIGSTA SECTION OF CWR ENDING SHEET
           10REC-TYP = 3      (++) WRFREQ SECTION OF CWR ENDING SHEET
           10REC-TYP = 4      (++) DESCRPN SECTION OF CWR ENDING SHEET
           10REC-TYP = 5      (++) DESCRPN SECTION OF CWR ENDING SHEET
           10REC-TYP = 6      (++) TECH SECTION OF CWR ENDING SHEET
           10REC-TYP = 7      (++) TO SECTION OF CWR ENDING SHEET
           10REC-TYP = 8,9    (++) REMOVE SECTION OF CWR ENDING SHEET
           10REC-TYP = 10,11(++) INSTAL SECTION OF CWR ENDING SHEET
           10REC-TYP = 12     (++) COMMENT SECTION OF CWR ENDING SHEET
       05  BITE            PIC IS X(2).
       05  DUMMY01         PIC IS X(72).
       05  GENERAL REDEFINES DUMMY01.
           10  DEV-PTR         PIC IS 9(6)  USAGE IS INTEGER.
           10  FWD-PTR         PIC IS 9(6)  USAGE IS INTEGER.
           10  PROJECT-NO      PIC IS 9(6).
           10  MEMO-DATE       USAGE IS DATE.
           10  LOG-DATE        USAGE IS DATE.
           10  COMPL-DATE      USAGE IS DATE.
           10  INPUT-DATE      USAGE IS DATE.
           10  TOPO-UPDATE     USAGE IS DATE.
           10  AFER            PIC IS X.
           10  FILLER          PIC IS X(22).
       05  SIG-PER REDEFINES DUMMY01.
           10  SEGMENTION
               15  NAME
                   20  LAST-NAME   PIC IS X(15).
                   20  FIRST-NAME  PIC IS X(12).
               15  ORG.
                   20  DIV         PIC IS X(4).
                   20  DEP         PIC IS X(3).
                   20  FILLER      PIC IS X.
               15  LOCATION
                   20  TA          PIC IS X(2)  USAGE IS INTEGER.
                   20  BLDG        PIC IS X(6).
                   20  ROOM        PIC IS X(6).
                   20  MAIL-STOP   PIC IS X(4).
                   20  PHONE       PIC IS X(11).
                       EDIT-STRING IS 7,777,777 BXXX.
           10  CHARGE
               15  COST-CTR    PIC IS 9(6)  USAGE IS INTEGER.
               15  PROJ-CDE    PIC IS 9(6).
               15  FILLER      PIC IS X(7).
           10  TERMINAL MODIFIERS CHARGE.
               15  ISM-DEP     PIC IS X(7).
               15  FILLER      PIC IS X(6).
       05  WORK-REQUESTED REDEFINES DUMMY01.
           10  WORK-REQUIRED       PIC IS X(2).
           10  TELCO-SP-RQ         PIC IS 9(4)  USAGE IS INTEGER.
           10  START-DATE          USAGE IS DATE.
           10  COMPLETED-DATE      USAGE IS DATE.
           10  DUE-DATE            USAGE IS DATE.
           10  WRI-BY              PIC IS X(15).
           10  FILLER              PIC IS X(130).
       05  CIRCUIT REDEFINES DUMMY01.
           10  CHARGE(
               15  PRT-RQ      PIC IS X(6).
               15  PRT-STA     PIC IS X.
               15  PRT-SYS     PIC IS X(5).
               15  PRT-TYP     PIC IS X(3).
               15  PAR         PIC IS X(2).
               15  BRSR        PIC IS X(3).
               15  FORMAT      PIC IS X.
               15  SIGNALLING  PIC IS X(5).
               15  CHARGE CODE PIC IS X(3).
               15  DEV TYP     PIC IS X(3).
               15  DEV RQ      PIC IS X(6).
               15  DEV STA     PIC IS X(1).
           10  TELCO CHAR
               15  SP RQ       PIC IS X(6).
               15  DIAL UP UC  PIC IS X(1).
                   EDIT STRING IS BXX.
               15  FIRST PAIR  PIC IS X.X.
               15  SECOND PAIR PIC IS X.X.
           10  FILLER          PIC IS X(90).
       05  TERMINATIONS REDEFINES DUMMY01.
           10  PORT SIDE
               15  PRT SIG LOC PIC IS X.X.
               15  PRT SIG DIR PIC IS X.X.
               15  PRT SIG P01 PIC IS X(3).
               15  PRT SIG P02 PIC IS X(3).
               15  PRT SIG P03 PIC IS X(3).
           10  USER SIDE
               15  USR SIG LOC PIC IS X(6).
               15  USR SIG DIR PIC IS X.X.
               15  USR SIG P01 PIC IS X(3).
               15  USR SIG P02 PIC IS X(3).
               15  USR SIG P03 PIC IS X(3).
           10  SEQ NUM PIC IS 9(4) USAGE IS INTEGER EDIT STRING IS ZZZ.
           10  ASSIGNMENT
               15  ASSIGNED DATE       USAGE IS DATE.
               15  COMPLETION DATE     USAGE IS DATE.
               15  CUT BY              PIC IS X(12).
               15  CUT STATUS          PIC IS X(4).
       05  COM-REC REDEFINES DUMMY01.
           10  COMMENT             PIC IS X(72).
```

Figure 2. The communications work request data base

```
DELETE NEWTEMP1;
DEFINE DOMAIN NEWTEMP1
 USING NEWTEMP1REC ON DUA1:[270.33CWR.DAT;
DELETE NEWTEMP1REC;
DEFINE RECORD NEWTEMP1REC
 USING
1  NEWTEMP1-REC
       05  CWR NO          PIC IS 9(6)  USAGE IS INTEGER.
       05  REC-TYP         PIC IS 9(2)  USAGE IS INTEGER.

       05  BITE            PIC IS X(2).
       05  GENERAL.
           10  DEV-PTR         PIC IS 9(6)  USAGE IS INTEGER.
           10  FWD-PTR         PIC IS 9(6)  USAGE IS INTEGER.
           10  PROJECT-NO      PIC IS 9(6).
           10  MEMO-DATE       USAGE IS DATE.
           10  LOG-DATE        USAGE IS DATE.
           10  COMPL-DATE      USAGE IS DATE.
           10  INPUT-DATE      USAGE IS DATE.
           10  TOPO-UPDATE     USAGE IS DATE.
           10  AFER            PIC IS X.
           10  FILLER          PIC IS X(22).
```

Figure 3. The domain and record definitions for record type 1

```
!DU1:[270,3]LOGGENRAL.CMD
DELETE LOGGENRAL;
DEFINE PROCEDURE LOGGENRAL
FINISH
SET ABORT
DECLARE BCK-PTR PIC IS 9(6) USAGE IS INTEGER.
DECLARE CUR-CWR PIC IS 9(6) USAGE IS INTEGER.
DECLARE CUR-REV PIC IS 9(6) USAGE IS INTEGER.
DECLARE CUR-FWD PIC IS 9(6) USAGE IS INTEGER.
DECLARE PRV-FWD PIC IS 9(6) USAGE IS INTEGER.
DECLARE NEW-REC PIC IS X.
NEW-REC = "F"
BCK-PTR = 0
READY NEXTPSR WRITE
IF *."C if you wish to create a new CWR or M to modify an old one" EQ "C" THEN
        FOR NEXTPSR BEGIN
                NEW-REC = "T"
                CUR-CWR = NEXT-PSR
                PRINT COL 20, "This CWR's number is", CUR-CWR(-)
                MODIFY USING NEXT-PSR = CUR-CWR + 1
        END ELSE
                CUR-CWR = *."number of the CWR you wish to modify"
FINISH
READY CWR SHARED WRITE
BCK-PTR = *."previous CWR, if any (6 digits)"
IF NEW-REC = "T" THEN STORE CWR USING BEGIN
        CWR-NO          = CUR-CWR
        REC-TYP         = 1
        REV-PTR         = CUR-CWR
        FWD-PTR         = CUR-CWR
        COMPL-DATE      = " "
        INPUT-DATE      = "TODAY"
        PROJECT-NO      = *."project number (6 chars.)"
        SITE            = *."site (2 chars.)"
        MEMO-DATE       = *."requestor's memo date (mm/dd/yy)
        LOG-DATE        = *."log book date (mm/dd/yy)"
        TOPO-UPDATE     = " "
        XFER            = " "
        END ELSE
IF *."Y if you wish to modify the GENERAL section" EQ "Y" THEN BEGIN
        FOR CWR WITH (CWR-NO=CUR-CWR) AND (REC-TYP=1) MODIFY USING BEGIN
                PRINT SKIP, COMPL-DATE, XFER, SKIP
                COMPL-DATE = *."completed date (mm/dd/yy)"
                XFER = *."Y if this CWR has been verified for transfer to TOPO"
                END
        FOR CWR WITH (CWR-NO=CUR-CWR) AND (REC-TYP=1)
                MODIFY USING BEGIN
                PRINT SKIP, PROJECT-NO, SITE, MEMO-DATE, LOG-DATE, SKIP
                PROJECT-NO = *."project number (6 chars.) or <tab>"
                SITE = *."site (2 chars.) or <tab>"
                MEMO-DATE = *."requestor's memo date (mm/dd/yy) or <tab>"
                LOG-DATE = *."Log Book Date (mm/dd/yy) or <tab>"
                END
        END
FOR CWR WITH (CWR-NO=CUR-CWR) AND (REC-TYP=1)
        WHILE SITE NOT IN SITETBL MODIFY USING BEGIN
                PRINT SKIP, "Bad site code (", SITE(-), "); please try again.", SKIP
                SITE    = *."site (2 chars.)"
        END
RELEASE SITETBL
END-PROCEDURE;
```

Figure 4. Data entry procedure for record type 1

```
        DELETE LOGREQSTR;
        DEFINE PROCEDURE LOGREQSTR
        IF *."Y if you wish to enter data for REQUESTOR" EQ "Y" THEN BEGIN
        NEW-REC = "F"
        FOR CWR WITH (CWR-NO = CUR-CWR) AND (REC-TYP E) 2) MODIFY USING BEGIN
                NEW-REC = "T"
                PRINT SKIP, NAME, ORG, MAIL-STOP, PHONE, CHARGE, SKIP
                FIRST-NAME = *."requestor's first name (13 chars.) or <tab>"
                LAST-NAME = *."requestor's last name (15 chars.) or <tab>"
                DIV = *."requestor's division (4 chars.) or <tab>"
                GRP = *."requestor's group (3 chars.) or <tab>"
                MAIL-STOP= *."requestor's mail stop (4 chars.) or <tab>"
                PHONE = *."requestor's phone (11 digits) or <tab>"
                COST-CTR = *."cost center (4 digits) or <tab>"
                PROG-COD = *."program code (4 chars.) or <tab>"
                END
        IF NEW-REC NE "T" THEN STORE CWR USING BEGIN
                CWR-NO  = CUR-CWR
                REC-TYP = 2
                FIRST-NAME = *."requestor's first name (13 chars.)"
                LAST-NAME  = *."requestor's last name (15 chars.)"
                DIV = *."requestor's division (4 chars.)"
                GRP = *."requestor's group (3 chars.)"
                TA = " "
                BLDG = " "
                ROOM = " "
                MAIL-STOP = *."requestor's mail stop (4 chars.)"
                PHONE = *."requestor's phone (11 digits)"
                COST-CTR = *."cost center (4 digits)"
                PROG-COD = *."program code (4 chars.)"
                END
        END
        END-PROCEDURE
```

Figure 5. Data entry procedure for record type 2

to prevent this. A straightforward "workaround" for this is to write an indirect command file that creates and passes instructions to Datatrieve for execution. Figure 6 shows one such command file. Unfortunately, this technique is slow and tedius because of the time required to initialize Datatrieve with each call. We are experimenting with ways to reduce this initialization time.

Data retrieval and report generation from multirecord type domains are considerably more complex than from non hierarchical single-record-type domains. Because no user-written logic can be included in Datatrieve Report Writer procedures, the data to be contained in a report must first be transferred to a "flat" file. A procedure to do this is shown in Figure 7, and the domain and record definitions are shown in Figure 8. Note the use of nested FOR clauses to produce the work file. These can cause Datatrieve to make millions of disk accesses taking many hours if a unique key is not used in describing the record to be selected in the inner FOR loop. For each record in the outer FOR loop, about one-half the domain indicated in the inner FOR loop will be searched for the correct record. Consider the 5000-plus record multirecord-type file used in the above illustrations. When the work file was created with no keys, the procedure of Figure 7 took over 20 h of clock time. This was true for Datatrieve running under PSXIIM or RSXIIM+ on a PDP11/70 with no other tasks active, except system overhead. After we modified the procedure of Figure 7 to create the file PRELOG with CWR-NO as the primary key with no duplications, the procedure took about 1 h of clock time.

Interactive use of similar nested FOR clauses to retrieve single parent children sets from within a single multirecord type domain takes only seconds or tens of seconds. If keys are used in specifying the desired record, thus, operator fatigue is not excessive provided the multiline nested FOR statements are stored in "canned" procedures and need not be entered by the operator.

```
        .!DU1:[27@,3]UPDATECWR.CMD
        .ENABLE QUIET
        .ENABLE SUBSTITUTION
        .SETS CLI <CLI>
        .IF CLI EQ "DCL" SET MCR=TI:
        PIP [27@,3]*.DAT/UN/NM
.1@@:   .ASKS [2:3::3@S] SECT Enter section you wish to
         update
        .IF SECT EQ "GEN" .GOTO 2@@
        .IF SECT EQ "REQ" .GOTO 2@@
        .IF SECT EQ "WRK" .GOTO 2@@
        .IF SECT EQ "NEW" .GOTO 2@@
        .IF SECT EQ "OLD" .GOTO 2@@
        .IF SECT EQ "FRM" .GOTO 2@@
        .IF SECT EQ "TO"  .GOTO 2@@
        .IF SECT EQ "REM" .GOTO 2@@
        .IF SECT EQ "INS" .GOTO 2@@
        .IF SECT EQ "COM" .GOTO 2@@
        .DISABLE QUIET
        !Bad abbreviation; please try again.
        !Valid abbreviations are:
        !       GEN
        !       REQ
        !       WRK
        !       NEW
        !       OLD
        !       FRM
        !       TO
        !       REM
        !       INS
        !       COM
        .ENABLE QUIET
        .GOTO 1@@
.2@@:   .SETS UPDATE "UPD"+"'SECT'"
        .OPEN UPDATECWR.DTR
.ENABLE DATA
SET DICTIONARY DU1:[27@,3]PSR.DIC
!"UPDATE"
.DISABLE DATA
        .CLOSE
        DTR @UPDATECWR.DTR
        PIP UPDATECWR.DTR;*/DE/NM
        .ASK CONTIN Have you more CWR''s to process
        .IFT CONTIN .GOTO 1@@
        .IF CLI NE "MCR" SET /'CLI'=TI:
        .EXIT
```

Figure 6. Indirect command file for generalized data entry

```
!DU1:[27@,3]PRELOGRPT.CMD
DELETE PRELOGRPT;
DEFINE PROCEDURE PRELOGRPT
FIN!SH
DEFINE FILE FOR PRELOG SUPERSEDE
READY PRELOG WRITE
READY CWR SHARED READ
DECLARE CWR-TMP PIC IS 9(6).
DECLARE PORT-TMP  PIC IS X(6).
DECLARE ASS-TMP USAGE IS DATE.
DECLARE COMP-TMP USAGE IS DATE.
DECLARE BY-TMP  PIC IS X(15).
DECLARE STAT-TMP PIC IS X(9).
FOR AA IN CWR WITH REC-TYP=1 AND SITE NE "EP"
        STORE Z IN PRELOG USING BEGIN
                Z.CWR-NO          = AA.CWR-NO
                Z.REC-TYP         = AA.REC-TYP
                Z.INPUT-DATE      = AA.INPUT-DATE
                Z.LOG-DATE        = AA.LOG-DATE
                Z.SITE            = AA.SITE
                Z.PROJECT-NO      = AA.PROJECT-NO
                Z.MEMO-DATE       = AA.MEMO-DATE
                Z.ASSIGNED-DATE   = @
                Z.COMPLE-TION-DATE      = @
                Z.BY              = " "
                Z.STATUS          = " "
        END
FOR A IN CWR WITH REC-TYP=2 BEGIN
        CWR-TMP = CWR-NO
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.REQ-FIRST = A.FIRST-NAME
                        Z.REQ-LAST = A.LAST-NAME
                        Z.REQ-ORG = A.DIVISIA.GRP
                        Z.MAIL-STOP = A.MAIL-STOP
                        Z.PHONE = A.PHONE
                        END
        END
FOR B IN CWR WITH REC-TYP=4 BEGIN
        CWR-TMP = CWR-NO
        PORT-TMP = B.PRT-NO
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.NEW-PORT-NO = PORT-TMP
                        Z.NEW-PART = B.PAR
                        Z.NEW-STATUS = B.PRT-STA
                        Z.NEW-SPEED = B.PRT-SPD
                        Z.NEW-TYPE = B.PRT-TYP
                        Z.TO-DEV-TYPE   = B.DEV-TYP
                        Z.TO-DEV-NO     = B.DEV-NO
                        END
        END
FOR C IN CWR WITH REC-TYP=5 BEGIN
        CWR-TMP = CWR-NO
        PORT-TMP = C.PRT-NO
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.OLD-PORT-NO = PORT-TMP
                        Z.OLD-PART = C.PAR
                        Z.OLD-STATUS = C.PRT-STA
                        Z.OLD-SPEED = C.PRT-SPD
                        Z.OLD-TYPE = C.PRT-TYP
                        Z.FROM-DEV-TYPE = C.DEV-TYP
                        Z.FROM-DEV-NO   = C.DEV-NO
                        END
        END
FOR D IN CWR WITH REC-TYP=6 BEGIN
        CWR-TMP = CWR-NO
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.FROM-TA         = D.TA
                        Z.FROM-BLDG       = D.BLDG
                        Z.FROM-ROOM       = D.ROOM
                        Z.FROM-BOX-NO     = D.TRM-BOX
                END
        END
FOR E IN CWR WITH REC-TYP=7 BEGIN
        CWR-TMP = CWR-NO
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.TO-TA           = E.TA
                        Z.TO-BLDG         = E.BLDG
                        Z.TO-ROOM         = E.ROOM
                        Z.TO-BOX-NO       = E.TRM-BOX
                END
        END
FOR F IN CWR WITH REC-TYP=9 BEGIN
        CWR-TMP = CWR-NO
        ASS-TMP = F.ASSIGNED-DATE
        COMP-TMP = F.COMPLETION-DATE
        BY-TMP  = F.CUT-BY
        STAT-TMP = F.CUT-STATUS
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.ASSIGNED-DATE = ASS-TMP
                        Z.COMPLE-TION-DATE = COMP-TMP
                        Z.BY              = BY-TMP
                        Z.STATUS          = STAT-TMP
                END
        END
FOR G IN CWR WITH REC-TYP=11 BEGIN
        CWR-TMP = CWR-NO
        ASS-TMP = G.ASSIGNED-DATE
        COMP-TMP = G.COMPLETION-DATE
        BY-TMP  = G.CUT-BY
        STAT-TMP = G.CUT-STATUS
        FOR Z IN PRELOG WITH CWR-NO=CWR-TMP
                MODIFY USING BEGIN
                        Z.ASSIGNED-DATE = ASS-TMP
                        Z.COMPLE-TION-DATE = COMP-TMP
                        Z.BY              = BY-TMP
                        Z.STATUS          = STAT-TMP
                END
        END
END-PROCEDURE
```

Figure 7. Procedure to create flat work file for the Report writer

```
DELETE PRELOG;
DEFINE DOMAIN PRELOG
 USING PRELOGREC ON DU1:[27#,3]PRELOG.DAT;
DELETE PRELOGREC;
DEFINE RECORD PRELOGREC
 USING
1  PRELOG-REC.
        5  CWR-NO          PIC IS X(4).
        5  REC-TYP         PIC IS X(2).
        5  SITE            PIC IS X(2).
        5  PROJECT-NO      PIC IS X(6).
        5  REQ-ORG         PIC IS X(7).
        5  OLD-PORT-NO     PIC IS X(8).
        5  FROM-TA         PIC IS X(2).
        5  FROM-BLDG       PIC IS X(5).
        5  FROM-ROOM       PIC IS X(6).
        5  FROM-BOX-NO     PIC IS X(7).
        5  FROM-DEV-TYPE PIC IS X(3).
        5  FROM-DEV-NO     PIC IS X(8).
        5  NEW-PORT-NO     PIC IS X(8).
        5  TO-TA           PIC IS X(2).
        5  TO-BLDG         PIC IS X(5).
        5  TO-ROOM         PIC IS X(6).
        5  TO-BOX-NO       PIC IS X(7).
        5  TO-DEV-TYPE     PIC IS X(3).
        5  TO-DEV-NO       PIC IS X(8).
        5  BY              PIC IS X(12).
        5  STATUS          PIC IS X(9).
        5  REQ-FIRST       PIC IS X(6).
        5  REQ-LAST        PIC IS X(10).
        5  MAIL-STOP       PIC IS X(4).
        5  OLD-PART        PIC IS X(2).
        5  OLD-STATUS      PIC IS X(1).
        5  OLD-SPEED       PIC IS X(5).
        5  OLD-TYPE        PIC IS X(3).
        5  NEW-PART        PIC IS X(2).
        5  NEW-STATUS      PIC IS X(1).
        5  NEW-SPEED       PIC IS X(5).
        5  NEW-TYPE        PIC IS X(3).
        5  PHONE           PIC IS 9(4) EDIT-STRING IS XXXX.
        5  INPUT-DATE      USAGE IS DATE.
        5  LOG-DATE        USAGE IS DATE.
        5  ASSIGNED-DATE USAGE IS DATE.
        5  COMPLE-TION-DATE USAGE IS DATE.
        5  MEMO-DATE       USAGE IS DATE.
```

Figure 8. Domain and record definitions for the flat work file

MULTIDOMAIN DATA BASES

When the lengths of the children records are substantially different from one another or from the parent record, and the number of occurences of children varies markedly from parent to parent, a multidomain data base may be appropriate. Similarly, if relationships other than the single parent-child association are to be shown, a multidomain data base may be required.

As with the multirecord-type domain, a method of linking records is needed. One or more sets of pointers may be satisfactory for this purpose. Something as simple as the common-valued field, like CWR-NO in the multirecord-type domain above, may suffice. All the data entry and retrieval problems described above for multirecord-type domains apply here, but most are more pronounced. Using VIEWS helps to solve these problems, but the limitations of the Datatrieve-11 pool are very noticeable. Therefore using other software devices, including indirect command files and compiled programs, becomes highly desirable.

USE OF SUPERVISOR MODE RMS WITH DATATRIEVE-11

Another paper scheduled for presentation here at the symposium discusses the linking of Datatrieve-11 to an RMS supervisor mode library. With help from our local DEC office, we are now running DTRSUP on our PDP11/7# under RSX11M+. Figures 9, 1#, and 11 show the task builder command file and ODL files used to create the supervisor mode Datatrieve-11. We have experienced no noticeable improvement in execution time; however, the roughly 4#% increase in pool space has resulted in much greater convenience when creating procedures and in ad hoc queries. Of particular note is the new ability to READY five or six domains at once and still be able to perform useful searches, and so forth. We had found that with overlaid RMS two domains were about the maximum that could be readied at once and do much else.

```
DTRSUP/FP/CP,DTRMP/-SP=DTR[1MP/MP
; LINK TO SUPERVISOR MODE RMSRES
RESSUP=[3,54]RMSRES/SV:#

; FOLLOWING LINE APPLIES ONLY TO NON-DBMS-11 DATATRIEVE
PAR=GEN[1,1774##

ASG=TI:5
UNITS=15
TASK=...DTR
//
```

Figure 9. Task builder command file for building Datatrieve 11 with Supervisor mode RMS*

```
; THE BUILD ROOTS ARE:
;
;       DTRDBM -- DBMS SUPPORT
;       DTRRMS -- NO DBMS SUPPORT (RMS ONLY)
;       DTRRDB -- RMS DEBUG VERSION
;       DTRFPE -- FLOATING POINT EMULATION INCLUDED
;                 (RMS ONLY)
;       DTRDBG -- DBMS DEBUG VERSION
;

        .ROOT   DTRRMS
@DTRMP.ODL
        .END
```

Figure 10. First ODL file for building Datatrieve-11 with Supervisor-mode RMS*

```
; DTRMP.ODL - LINKS TO SUPERVISOR MODE RMSRES

; FACTORS ENDING IN DOUBLE DIGITS INDICATE CONDITIONAL BUILD FOR
; THE VARIOUS DATATRIEVE VERSIONS.  THE CODES ARE:

;       XXXX#1 -- FPP PRESENT OR NOT NEEDED
;       XXXX#2 -- FLOATING EMULATION CODE NEEDED *

        .PSECT  IOOV,GBL
        .PSECT  INIT,GBL
;
; FOR RMS V2.# AS SUPER MODE, REMOVE REFERENCES TO "TREE#2"

DTRRMS: .FCTR   RT#1-IO1-*(DD1,COMPLR,PARS#1,EXEC#1)
DTRFMS: .FCTR   RT#1-IO1-*(DD1,COMPLR,PARS#1,EXEC#2)
DTRMY#: .FCTR   RT#1-FP1-IO3-*(DD1,COMPLR,PARS#1,EXEC#1)

DTR%D#: .FCTR   RT#1-IO1-DEBUG-*(DD1,COMPLR,PARS#1,EXEC#1),BDTTRE
DTRMYD: .FCTR   RT#1-FP1-IO3-DEBUG-*(DD1,COMPLR,PARS#1,EXEC#1),BDTTRE

; ROOT SECTIONS

        .NAME   DTR
RT#1:   .FCTR   DTR-RT2-RT3-DTRLIB/LB:MODB

RT2:    .FCTR   DTRLIB/LB:TEST:AL:BL:CK:JTYD:FS:MS:DA
RT3:    .FCTR   DTRLIB/LB:QD:SAVREG
;
; ADD BASE RMS SUPPORT, REQUIRED ROOT SEGMENTS, AND FP STUFF
; FOR RMS V2.# SUPPORT.
;
RMSROT: .FCTR   LB:[1,1]RMSLIB/LB:R#AUTS:R#IMPA:R#EXSY:RMSSYM-FP2

; I/O GUYS

IO1:    .FCTR   DTRLIB/LB:RX-IO2,RMSROT
IO2:    .FCTR   DTRLIB/LB:IO:RM:RA
IO3:    .FCTR   IO2-DTRLIB/LB:RMS2I

; INITIALIZATION STUFF.

IN#1:   .FCTR   INIT-IN1
IN1:    .FCTR   DTRLIB/LB:IN

; DEBUGGING STUFF
DEBUG:  .FCTR   DTRLIB/LB:DBPRX:AUTO:PATCH-DTRLIB/LB:BDT/DA
        .NAME   BDTX
BDTTRE: .FCTR   BDTX-*(BD2-BD3-BD4,BD5)
BD2:    .FCTR   DTPLIB/LB:BDT2
BD3:    .FCTR   DTRLIB/LB:BDT3
BD4:    .FCTR   DTRLIB/LB:BDT4
BD5:    .FCTR   DTRLIB/LB:DBPR

; DODO-TRIEVE

DD1:    .FCTR   DTRLIB/LB:DD

; PARSER
        .NAME   PARSE
PARS#1: .FCTR   PARSE-IOOV-P1-*(P2,ADT1,EDIT,CMND#1)
P1:     .FCTR   DTRLIB/LB:DI:HT:LN:CD:PA:PU
P2:     .FCTR   DTRLIB/LB:HE:SH
ADT1:   .FCTR   DTRLIB/LB:AD

; COMMANDS
CMND#1: .FCTR   C1-*(C2,IND1-C3,RDF1#1)
C1:     .FCTR   DTRLIB/LB:RP:PR
C2:     .FCTR   DTRLIB/LB:PA
C3:     .FCTR   DTRLIB/LB:LM:TL

; READY/FINISH
RDF1#1: .FCTR   RF1-RF2

RF1:    .FCTR   DTRLIB/LB:RI
RF2:    .FCTR   DTRLIB/LB:UP:ML:DF

; EDITOR
EDIT:   .FCTR   DTRLIB/LB:TE:,TPARS

; COMPILER-ISH STUFF
        .NAME   CMPLR
COMPLR: .FCTR   CMPLR-DTRLIB/LB:CR:RC:AS:PF:SC:TC

; EXECUTION SYSTEM.
        .NAME   RUNTIM
EXEC#1: .FCTR   RUNTIM-EX1-EX2-EX3-EX6-*(EX4,EX5)
EX1:    .FCTR   DTRLIB/LB:AR:CM:EA:RS:RU:US
EX2:    .FCTR   DTRLIB/LB:RUNTIM:GETTIM
EX3:    .FCTR   DTRLIB/LB:BV:EV:CO:MOPL-SORT
EX4:    .FCTR   DTRLIB/LB:ME
EX5:    .FCTR   DTRLIB/LB:BE
EX6:    .FCTR   DTRLIB/LB:MOFM
EX7:    .FCTR   DTRLIB/LB:FN:FDV:FDVPRT:FDVERR:FDVDAT:FDVTIO:FMSSAV

EXEC#2: .FCTR   RUNTIM-EX1-EX2-EX6-*(EX3-EX4,EX7)

; SORT
SORT:   .FCTR   DTRLIB/LB:SORTS:FIORMS:ST

; RMS CO-TREE PLUS OUR STUFF.  (ELIMINATED FOR SUPER MODE RMS, FPP MOVED
;                               TO ROOT)

;TREE#0:        .FCTR   RMS11-*(FP2,RMSFIL,RMSREC,DBS*RE)
;TREE#2:        .FCTR   RMS11-*(FP2,RMSFIL,RMSREC)

F2:     .FCTR   DTRLIB/LB:ROFP:MISC
FP1:    .FCTR   DTRLIB/LB:FP:MISC
        .PSECT  DTRVSE
        .PSECT  DBSFOV,GBL,RW
```

Figure 11. Second ODL file for building Datatrieve 11 with Supervisor mode RMS

One would like to take advantage of the added pool space to speed up procedures by reducing disk I/O. Figure 12 illustrates how one might approach this to speed up the writing of the flat work file created by the procedure of Figure 7. The idea is to DEFINE variables for temporary storage of field contents from the hierarchical domain(s), then use a single STORE statement to write to the flat file instead of the initial STORE followed by several MODIFYs. Alas, there are too many variables DEFINED by the procedure of Figure 12, even with the increased pool space. Nevertheless, the principle is valid and should be used where possible.

COMPILED PROGRAMS

When Datatrieve's performance proves inadequate, it may be appropriate to turn to compiled programs. Using these can result in very large savings in execution time and can overcome Datatrieve-11 pool limitations associated with readying more than two or three domains simultaneously. As an example of the improvement in execution time that can be expected, refer again to the procedure shown in Figure 7. As mentioned above, this procedure required about an hour of execution time to produce a flat, keyed, work file for the Datatrieve Report Writer. We created a FORTRAN 77 program that produced the same work file (but without keys) in 2 minutes, 15 seconds, using the indexed sequential access method on the CWR domain. This corresponded to about a factor of 25 improvement in execution time. This particular program was linked to an PMS supervisor mode library under RSX11M+, V2.1. We have experienced similar improvements with overlaid PMS libraries under RSX11M, V4.0.

A major disadvantage of writing compiled programs to access data files is that file organization must be included in each program's source code, rather than be generally available, as in a Datatrieve dictionary. One way to minimize the labor required to maintain several programs in the face of file organization changes is to create a single source code file that contains a description of the data file's organization. For the FORTRAN language, this might include a set of TYPE declaration statements, DIMENSION statements, and FORMAT statements for use with READ and WRITE statements. This source code file can then be appended to each of the compiled programs that access the data file. If the source code is written in FORTRAN, the INCLUDE statement is a convenient tool for appending the data file's organization description to the main program and its subroutines.

SORT-11

The Sort-11 software tool has proved especially useful in preparing large reports from Datatrieve-11 data bases. One method of creating large, sorted reports is to prepare a flat work file either with Datatrieve or, if necessary, a compiled program. Then use Sort 11 to order the records appropriately before starting Datatrieve's Report Writer. In this way one can create sorted reports that would exhaust the Datatrieve pool. Furthermore, Sort-11 is extremely fast, performing sorts in 1-2 min that would take Datatrieve nearly an hour, if it could do the job at all.

CONCLUSION

The data base definitions discussed in this paper push the capabilities of Datatrieve-11 close to their practical limits of performance. For more complex linkage among records and fields, a more complete data base management system should be considered.

ACKNOWLEDGEMENTS

The author wishes to thank Chuck Milich, Digital Equipment Corporation, Los Alamos, New Mexico, and Dave Carroll CSC/CS, Colorado Springs, Colorado, for their making it possible for us to have DTPSUP on our system.


```
!DUI1:[270,3]PRELOGRAM.CMD
DELETE PRELOGRAM;
DEFINE PROCEDURE PRELOGRAM
FINISH
DEFINE FILE FOR PRELOG SUPERSEDE, KEY = CWR-NO;
READY PRELOG WRITE
READY CWR SHARED READ
DECLARE CWR-TMP          PIC IS 9(6) USAGE IS INTEGER.
DECLARE REC-TMP          PIC IS 9(6) USAGE IS INTEGER.
DECLARE INPUT-TMP        USAGE IS DATE.
DECLARE LOG-TMP          USAGE IS DATE.
DECLARE SITE-TMP         PIC IS X(2).
DECLARE PROJECT-TMP      PIC IS X(6).
DECLARE MEMO-TMP         USAGE IS DATE.
DECLARE ASSIGNED-TMP     USAGE IS DATE.
DECLARE COMP-TMP         USAGE IS DATE.
DECLARE BY-TMP           PIC IS X(12).
DECLARE CUT-STATUS-TMP   PIC IS X(9).
DECLARE FIRST-TMP        PIC IS X(6).
DECLARE LAST-TMP         PIC IS X(10).
DECLARE ORG-TMP          PIC IS X(7).
DECLARE MAIL-TMP         PIC IS X(4).
DECLARE PHONE-TMP        PIC IS 9(4).
DECLARE NEW-PORT-TMP     PIC IS X(8).
DECLARE NEW-PART-TMP     PIC IS X(2).
DECLARE NEW-STATUS-TMP   PIC IS X(1).
DECLARE NEW-SPEED-TMP    PIC IS X(5).
DECLARE NEW-TYPE-TMP     PIC IS X(3).
DECLARE TO-DEV-TYP-TMP   PIC IS X(3).
DECLARE TO-DEV-NO-TMP    PIC IS X(8).
DECLARE OLD-PORT-TMP     PIC IS X(8).
DECLARE OLD-PART-TMP     PIC IS X(2).
DECLARE OLD-STATUS-TMP   PIC IS X(1).
DECLARE OLD-SPEED-TMP    PIC IS X(5).
DECLARE OLD-TYPE-TMP     PIC IS X(3).
DECLARE FROM-DEV-TYP-TMP PIC IS X(3).
DECLARE FROM-DEV-NO-TMP PIC IS X(8).
DECLARE TO-TA-TMP        PIC IS X(2).
DECLARE TO-BLDG-TMP      PIC IS X(6).
DECLARE TO-ROOM-TMP      PIC IS X(6).
DECLARE TO-BOX-TMP       PIC IS X(7).
DECLARE FROM-TA-TMP      PIC IS X(2).
DECLARE FROM BLDG-TMP    PIC IS X(6).
DECLARE FROM-ROOM-TMP    PIC IS X(6).
DECLARE FROM-BOX-TMP     PIC IS X(7).
FOR AA IN CWR WITH REC-TYP=1 AND SITE NE "EP" BEGIN
                         CWR-TMP          = AA.CWR-NO
                         REC-TMP          = AA.REC-TYP
                         INPUT-TMP        = AA.INPUT-DATE
                         LOG-TMP          = AA.LOG-DATE
                         SITE-TMP         = AA.SITE
                         PROJECT-TMP      = AA.PROJECT-NO
                         MEMO-TMP         = AA.MEMO-DATE
     FOR A IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=2 BEGIN
                         FIRST-TMP        = A.FIRST-NAME
                         LAST-TMP         = A.LAST-NAME
                         ORG-TMP          = A.DIVIA.GRP
                         MAIL-TMP         = A.MAIL-STOP
                         PHONE-TMP        = A.PHONE
                         END
     FOR B IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=4 BEGIN
                         NEW-PORT-TMP     = B.PRT-NO
                         NEW-PART-TMP     = B.PAR
                         NEW-STATUS-TMP   = B.PRT-STA
```

Figure 12. Procedure to transfer data fr n CWR to flat work file

```
                         NEW-SPEED-TMP    = B.PRT-SPD
                         NEW-TYPE-TMP     = B.PRT-TYP
                         TO-DEV-TYP-TMP   = B.DEV-TYP
                         TO-DEV-NO-TMP    = B.DEV-NO
                         END
     FOR C IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=5 BEGIN
                         OLD-PORT-TMP     = C.PRT-NO
                         OLD-PART-TMP     = C.PAR
                         OLD-STATUS-TMP   = C.PRT-STA
                         OLD-SPEED-TMP    = C.PRT-SPD
                         OLD-TYPE-TMP     = C.PRT-TYP
                         FROM-DEV-TYP-TMP          = C.DEV-TYP
                         FROM-DEV-NO-TMP  = C.DEV-NO
                         END
     FOR D IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=6 BEGIN
                         FROM-TA-TMP      = D.TA
                         FROM-BLDG-TMP    = D.BLDG
                         FROM-ROOM-TMP    = D.ROOM
                         FROM-BOX-TMP     = D.TRM-RCP
                         END
     FOR E IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=7 BEGIN
                         TO-TA-TMP        = E.TA
                         TO-BLDG-TMP      = E.BLDG
                         TO-ROOM-TMP      = E.ROOM
                         TO-BOX-TMP       = E.TRM-RCP
                         END
     FOR F IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=9 BEGIN
                         ASSIGNED-TMP     = F.ASSIGNED-DATE
                         COMP-TMP         = F.COMPLETION-DATE
                         BY-TMP           = F.CUT-BY
                         CUT-STATUS-TMP   = F.CUT-STATUS
                         END
     FOR G IN CWR WITH CWR-NO=CWR-TMP AND REC-TYP=11 BEGIN
                         ASSIGNED-TMP     = G.ASSIGNED-DATE
                         COMP-TMP         = G.COMPLETION-DATE
                         BY-TMP           = G.CUT-BY
                         CUT-STATUS-TMP   = G.CUT-STATUS
                         END
     STORE Z IN PRELOG USING BEGIN
                         Z.CWR-NO         = CWR-TMP
                         Z.REC-TYP        = REC-TMP
                         Z.INPUT-DATE     = INPUT-TMP
                         Z.LOG-DATE       = LOG-TMP
                         Z.SITE           = SITE-TMP
                         Z.PROJECT-NO     = PROJECT-TMP
                         Z.MEMO-DATE      = MEMO-TMP
                         Z.REQ-FIRST      = FIRST-TMP
                         Z.REQ-LAST       = LAST-TMP
                         Z.REQ-ORG        = ORG-TMP
                         Z.MAIL-STOP      = MAIL-TMP
                         Z.PHONE          = PHONE-TMP
                         Z.NEW-PORT-NO    = NEW-PORT-TMP
                         Z.NEW-PART       = NEW-PART-TMP
                         Z.NEW-STATUS     = NEW-STATUS-TMP
                         Z.NEW-SPEED      = NEW-SPEED-TMP
                         Z.NEW-TYPE       = NEW-TYPE-TMP
                         Z.TO-DEV-TYPE    = TO-DEV-TYP-TMP
                         Z.TO-DEV-NO      = TO-DEV-NO-TMP
                         Z.OLD-PORT-NO    = OLD-PORT-TMP
                         Z.OLD-PART       = OLD-PART-TMP
                         Z.OLD-STATUS     = OLD-STATUS-TMP
                         Z.OLD-SPEED      = OLD-SPEED-TMP
                         Z.OLD-TYPE       = OLD-TYPE-TMP
                         Z.FROM-DEV-TYPE  = FROM-DEV-TYP-TMP
                         Z.FROM-DEV-NO    = FROM-DEV-NO-TMP
                         Z.TO-TA          = TO-TA-TMP
                         Z.TO-BLDG        = TO-BLDG-TMP
                         Z.TO-ROOM        = TO-ROOM-TMP
                         Z.TO-BOX-NO      = TO-BOX-TMP
                         Z.FROM-TA        = FROM-TA-TMP
                         Z.FROM-BLDG      = FROM-BLDG-TMP
                         Z.FROM ROOM      = FROM-ROOM-TMP
                         Z.ASSIGNED DATE  = ASSIGNED-TMP
                         Z.COMPLE-TION-DATE       = COMP-TMP
                         Z.BY             = BY-TMP
                         Z.STATUS         = CUT-STATUS-TMP
                         END
```

Figure 12. cont.