Development of a Text-Editor based Relational Database Management System

Grant No. AFOSR-81-0131
June 1, 1981 - August 31, 1981

Thomas W. Mason
Florida A&M University
Tallahassee, FL 32307
Database management systems have historically been the domain of large mainframes. However, the popularity of mini and micro-computers has spurred the development of database systems appropriate for those devices. Concurrently, database systems design is turning away from traditional hierarchic and CODASYL models to embrace the conceptually simpler relational database approach (1,2,3,4).

The relational database approach views data as being in tables. The entries form the rows and are called tuples. The columns are called (CONTINUED)
ITEM #20, CONTINUED: Attributes. Simple selection commands are provided to search for entries with attributes of a given value. Other commands allow the extraction of a subset of entries (found by the selection commands) and the incorporation of that subset with others. This approach is not as efficient as the traditional model in its implementation but offers far greater flexibility in the incorporation of data and the ability to 'explore' the data base.

This report documents the attempt to develop a relational database management system for the Harris Minicomputer at Florida A&M University.
1.0. INTRODUCTION

Database management systems have historically been the domain of large mainframes. However, the popularity of mini and microcomputers has spurred the development of database systems appropriate for those devices. Concurrently, database systems design is turning away from traditional hierarchic and CODASYL models to embrace the conceptually simpler relational database approach (1,2,3,4).

The relational database approach views data as being in tables. The entries form the rows and are called tuples. The columns are called attributes. Simple selection commands are provided to search for entries with attributes of a given value. Other commands allow the extraction of a subset of entries (found by the selection commands) and the incorporation of that subset with others. This approach is not as efficient as the traditional model in its implementation but offers far greater flexibility in the incorporation of data and the ability to "explore" the data base.

This report documents the attempt to develop a relational database management system for the Harris Minicomputer at Florida A&M University.

2.0 SYSTEM DESIGN

It is unusual for the design of database management systems to be described in detail. Therefore, the description of RISS, a relational database management system for minicomputers (5), seemed a most fortuitous occurrence. The system development strategy was to translate RISS to the Harris computer and then augment its capabilities in accordance with the original project proposal.
2.1 RISS: RELATIONAL INQUIRY AND STORAGE SYSTEM

RISS was developed at the forest Hospital in Des Plaines, Illinois and implemented on a PDP 11/40 with the RSTS/E operating system. Data can be retrieved from RISS by human interaction from a terminal (called the naive-user interfaced level) or by requests from a computer program (called the applications' program interface level).

The conversion effort focussed solely on the properties of the naive user interface level. Three subsystems provide the pathway for a human user to access a RISS database a relational editor, a retrieval package and a database manipulation and maintenance package. They are described below:

A. Relational editor

The editor is needed to create, examine and update entries in the database. The editor design is based on dline-oriented text edition. Thus, there is a record pointer which identifies the entry to be entered, examined or changed. The editor commands implemented in RISS are:

1. move the record pointer forward in backward through the existing entries
2. search for a specified substring in the database and move the record pointer to the next occurrence of that substring.
3. delete one or more entries from the database.
4. enter a new entry in the database.
5. display or change the value of a field (attribute) of an existing entry.
6. provide descriptive information about a given set of entries (relation) in the database.

B. Retrieval Package

The retrieval commands allow the user to retrieve and analyze data in RISS relations:

1. selection of entries which satisfy a given attribute specification (i.e., Sex = "male", Age = 18) and the formation of a resulting relation
2. formation of the union or intersection of relations, including relations formed by the proceeding command
3. extraction of a subset of columns (attributes) of a relation
4. Printing tabular reports based on a retrieved relation
5. Printing simple statistical information
6. grouping specific attribute values into user-specified ranges
7. producing a frequency distribution for all unique values of a given attribute.

c. Database manipulation and maintenance package

The database manipulation and maintenance package provide the usual utility functions associated with database management:

1. creating a relation (a set of related entries in database)
2. deleting a relation
3. copying a relation
4. sorting a relation
5. merging two relations
6. combining two relations
7. redefining the structure of a relation by adding or deleting a column (attribute)

3.0 SYSTEM IMPLEMENTATION

The initial implementation strategy was to copy all routines from the RISS text into the Harris computer. It was thought that the differences in the BASIC language between the two systems would be minor. This proved not to be the case.

The implementation of BASIC on the two computers differs significantly. Furthermore, closer investigation revealed that the RISS data structure design incorporated fundamental features of the PDP-11/40. In other words, to use the RISS code directly would entail the emulation of PDP-11/40 features on the Harris minicomputer. To compound the problem to the breaking point, at this time the air conditioning system of the Harris minicomputer broke down and remained inoperative for several weeks. This exclude computer use during the time although program development was possible.
In light of the problems discussed above, it was decided to re-design the FAMU Relational Database system (FREDB) using RISS as a guide but exploiting the features of the Harris minicomputer. The central core of the re-design was to abandon the RISS data structure.

The RISS data structure allowed for storage of four types of data - single ASCII characters, integers, floating-point numbers and alphanumeric character strings. Three files or tables were used for storage and descriptions - a tuple (entry) descriptor table, a tuple file and an alpha data file. Without going into a detailed description of the process, suffice it to say that the storage structure was based very closely on the actual storage format of the PDP-11/40.

Upon reconsideration of the design, questions began to arise as to the need for a relational editor. The reasoning was that on edition already existed in the Harris operating system. It could be used for all of the functions of the RISS relational editor. However, the use of the Harris line editor implied the use of the Harris storage strategy. Hence, one simplification led to another. FREDB has no provision for relational editing. The system recognizes the equivalence of flat files and relations. Therefore, the user enters, modifies and deletes all data using existing Harris editing procedures. The relations are then described in FREDB and those field (attribute) descriptions form a pathway for FREDB routine to access user-created files. The adoption of this approach led to the development of an initial system with many of the features of RISS.
4.0. The FREDB System

The actual implementation consists of a method to define and create relations. Additional procedures to join and select relations were attempted but not completed. The computer programs for relation definition and definition (called CREREL) are given in Appendix 1.
Bibliography


$MO BS=1000
$MO NU
$SR,SRN
$SR.S *OUTFILE
$SR.S *NNN = 0
$SR.S *NNM NAME
$SR.F *NNM=10
$SR.S 20 = *NNM
$JE IGEN

ILAB2 $$$ RELATION
$SR.S *NNM 0
$SR.S *NNM = 0
$SR.F *AAA = 10
$$$ RELATION
$SR.N #NMM RE
$SR.N #DL=0
$SR.N #END=0
$SR.N #BEG=0
$SR.S #VAL=0
$SPR
$SPR
$SR.F *TPIN=0
$SR.I *TPIN
$JE IKEEP

IKEEP $$$ FLAG DEVICE
$$$ THIS SECTION EDITS RELATION AND
$$$ INSERTS BLANK RECORDS FOR INSERTION.
$SED #NMM RE
$PE 0
AE 1000000
$JE IOUT

IOUT $BE;1
$EO 0
$SR.F LIN=ERM();
$SED *NMM AB
$SR.S *IN=0
ILAB $$$
IN LIN BLANK 1
$SR.S *IN = *IN + 1
IF (*IN=*TPIN) SJU IOUT2
$SJU ILAB

IOUT2 $$$ THE INSERTION OF ATTRIBUTES BEGINS
$$$ $UP
$UP
$SED *NMM AB
$SR.F *LIN = *LIN + 1
$EO *LIN
AGAIN $$$
$SR.F *VAL=10
$SR.F *BEG=10
$SR.F *END=10
$SPR
$SPR ATTRIBUTE *EE
IREN $$$
$SPR *VAL
$SR.S *IN=NEW
$SR.I *IN
$JE 178 NPO
INPO $$$ INSERT BLANK CHAR
$IF (*IN=SEND) SJU IOUT4
$SR.S *CHR = *END - *BEG + 1
$IF ( \#CLE > \#CHE ) \$JU IKEN
$SC \#BEGIN=\#END,\#IN
$IF ( \#EE = \#MA ) \$JU IOUT3
$SR,N \#EE = \#EE + 1
\$JU IAGAIN

IOUT3 $\$
FIRST ATTRIBUTE INSERTED
$SR,N \#EE = 1
$SR,N \#NN = \#NN + 1
$SR,N \#LIN = \#LIN + 1
$RM 10
$SR,F \#NMM=10
$IF ( \#NN = \#TPIN ) \$JU IOUT4
$E \#LIN

$PR * INSERT SEND ON NEXT ENTRY TO TERMINATE
$PR ---- DATA ENTRY...
$JU IAGAIN

IOUT4 $\$
* IF SEND WAS ENTRIED THIS SECTION WILL DELETE
THE REMAINING LINES THAT WAS ORGINALLY REQUESTED.
$IF ( \#NN = \#TPIN ) \$JU IOUT5
$IF ( \#NN < \#TPIN ) \$DE \#LIN
$SR,N \#DL = \#DL + 1
$SR,N \#NN = \#NN + 1
$SR,N \#LIN = \#LIN + 1
\$JU IOUT4

IOUT5 $\$
THIS SECTION TERMINATES THE DATA ENTRY PROCESS.
$UP
$SR,N \#LIN = \#LIN = \#DL
$SR,N \#LIN = \#LIN = 2
$UP
$PR
$PR
$PR RELATION NAME = \#NMM ; NUMBER OF RECORDS = \#LIN

$ME IGEN $\$
$PR RELATION \#NMM NOW BEING GENERATED
$GE \#NMM
CO BLANK \#NMM
\$JU ILAB2
$ME
IKEN SPR ERROR GENERATED STRING LENGTH OF \#CHE EXCEEDED
\$JU IREN
IDENTIFICATION DIVISION.

PROGRAM-ID. GET-A-RELATION.

AUTHOR. ARTHUR ROBERTS JR.

DATE-WRITTEN. MARCH 10, 1982.

DATE-COMPILED.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. HARRIS-123.

OBJECT-COMPUTER. HARRIS-123.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT UNIT-OUTPUT-FILE ASSIGN "OUTFILE".
SELECT UNIT-INDEX-FILE ASSIGN "INDEXLOG"
ORGANIZATION IS RELATIVE
ACCESS MODE IS SEQUENTIAL
RELATIVE KEY IS REC-POS.

DATA DIVISION.

FILE SECTION.

FD UNIT-OUTPUT-FILE
DATA RECORD IS OUT-REC.

01 OUT-REC.
   02 FILLER PIC X(80).

FD UNIT-INDEX-FILE
DATA RECORD IS INDEX-REC.

01 INDEX-REC.
   02 REL-NAME PIC X(8).
   02 INFOMAT PIC X(991).

WORKING-STORAGE SECTION.

77 REC-POS PIC 9(5) VALUE 1.
77 RES PIC X(3) VALUE SPACES.
77 TREL-NAME PIC X(8) VALUE SPACES.
77 CONDI PIC X(1) VALUE "N".
77 SUB PIC 999 VALUE 0.

01 OUTPUT-DATA.
   02 DATA-LINE.
      03 FILLER PIC X(991) VALUE SPACES.
   02 DATA-OUT REDEFINES DATA-LINE.
      03 FILLER PIC X.
      03 NO-OF-ATT PIC 9(3).
      03 OCC-OF-ATT OCCURS 34 TIMES.
      04 FILLER PIC X.
      04 ATT-NAME PIC X(20).
      04 FILLER PIC X.
      04 BEGIN PIC X(3).
      04 FILLER PIC X.
      04 END-C PIC X(3).
      03 FILLER PIC X.

PROCEDURE DIVISION.

PROCESS-CONTROL.
   OPEN OUTPUT UNIT-OUTPUT-FILE
CLOSE
UNIT-OUTPUT-FILE
UNIT-INDEX-FILE.

STOP RUN.

MAIN-Routine.
MOVE "N" TO COND1.
DISPLAY "INPUT RELATION NAME ? ",
DISPLAY " THIS NAME CAN ONLY BE 8 CHARACTERS LONG ".
DISPLAY " NO SPECIAL CHARACTERS OR BLANKS ".
ACCEPT TREL-NAME FROM TERMINAL.
DISPLAY ",
DISPLAY "RELATION NAME = ", TREL-NAME.
DISPLAY ",
DISPLAY " IS THIS INFORMATION CORRECT ? (Y OR N) ".
DISPLAY ",
ACCEPT RES FROM TERMINAL.
IF RES = "Y"
DISPLAY " OK ",
PERFORM CHECK-RELATION-NAME THRU CHECK-RELATION-NAME-EXIT
IF COND1 = "Y"
MOVE INFORMAT TO OUTPUT-DATA
WRITE OUT-REC FROM REL-NAM0
WRITE OUT-REC FROM NO-OF-ATT
PERFORM WRITE-ROUTINE
VARYING SUB FROM 1 BY 1
UNTIL SUB > NO-OF-ATT
ELSE
DISPLAY " ERROR RELATION TREL-NAM0 DOES NOT EXIST ",
DISPLAY ",
DISPLAY " DO YOU WISH TO TRY AGAIN? (Y/N) ",
ACCEPT RES FROM TERMINAL.
IF RES = "Y"
PERFORM CLOSE-OPEN-FILE
GO TO MAIN-Routine.
ELSE
NEXT SENTENCE.
ELSE
DISPLAY " **** ERROR *****
GO TO MAIN-Routine.

MAIN-Routine-EXIT.
EXIT.

CHECK-RELATION-NAME.
READ UNIT-INDEX-FILE
AT END
GO TO CHECK-RELATION-NAME-EXIT.

IF REL-NAM0 = TREL-NAM0
MOVE "Y" TO COND1
ELSE
GO TO CHECK-RELATION-NAME.

CHECK-RELATION-NAME-EXIT.
EXIT.

CLOSE-OPEN-FILE.
CLOSE UNIT-INDEX-FILE.
WRITE OUT-REC FROM ATT-NAME (SUB).
WRITE OUT-REC FROM BEG-C (SUB).
WRITE OUT-REC FROM END-C (SUB).
EOF.
BEGIN
$RR, SNR
$PR
$PR INPUT THE NAME OF THE RELATION IN WHICH YOU WOULD LIKE COPIED?
$SR, I #NAME
$PR
$PR #NAME IS THE RELATION IN WHICH YOU WOULD LIKE COPIED? (Y/N)
$SR, I #NAME
$IF, (#NAME=N) $JU BEGIN
$JE IERR
$IF, (#NAME=Y) $JU ICONT
$PR
IERR
$PR PLEASE TYPE IN 'Y' OR 'N'...
$JU IRESTA
ICONT
$GE #NAME
$JE K, 318, IERR44
$JE INEXT
$JU IERR1
INEXT
$PR
$PR WHAT IS THE NAME OF THE NEW RELATION?
INEXT1
$SR, I #NAME
$PR
$PR #NAME IS THE NEW RELATION? (Y/N)
IRESTA
$SR, I #NAME
$IF, (#NAME=N) $JU INEXT
$JE IERR2
$IF, (#NAME=Y) $JU ICONT2
IERR2
$PR
$PR PLEASE TYPE IN 'Y' OR 'N'...
$JU IRESTA
ICONT2
$GE #NAME
$JE K, 2150, IERR3
$JE IERR33
$CO #NAME #NAME
$PR
$JU IINDEX
$PR
$PR #NAME IS AN NON-EXISTING RELATION
$PR
$PR WOULD YOU LIKE TO COPY ANOTHER RELATION? (Y/N)
$JU IREST
$IJE INEXT3
INEXT3
$PR
$PR #NAME IS AN EXISTING RELATION (OR HARRIS FILE)
$PR
ICONT33
$PR CHOOSE ANOTHER NAME FOR THE NEW RELATION...
$JU INEXT1
INDEX
$CO INDEXLOG W7
$SA 12=W
$SPR, F, 12, #NAME
$SA 13=#W
SOURCE1
$SR, F #LNE=13
$ED INDEXLOG
$IN #LNE INDEXLOG #LNE
$UP
$SR, N #LNE=#LNE+1
$TAB
$TAB 10
$ED INDEXLOG
SPR #NM HAS BEEN COPIED INTO #NM

SPR

SPR WOULD YOU LIKE TO COPY ANOTHER RELATION? (Y/N)

!REST

SSR, I #ANG

$IF, (#ANG = N) $SUU !END1

$JE IERR4

$IF, (#ANG = Y) $SUU !BEGIN

IERR4

$PR

SPR PLEASE TYPE IN 'Y' OR 'N'...

$SUU REST

IERR33 $JE INEXT33

INEXT33 $PR

SPR #NM IS AN INVALID RELATION NAME:

$PR

1) NAME HAS MORE THAN 8 CHARACTERS

$PR

2) FIRST CHARACTER IN THE NAME IS A NUMBER

$PR

3) NAME HAS AN INVALID CHARACTER.

$PR

$SUU ICONT33

IERR44 $JE INEXT44

INEXT44 $PR

SPR #NM IS AN INVALID RELATION NAME:

$PR

1) NAME HAS MORE THAN 8 CHARACTERS

$PR

2) FIRST CHARACTER IN THE NAME IS A NUMBER

$PR

3) NAME HAS AN INVALID CHARACTER.

$PR

SPR TRY AGAIN...

$PR

$SUU !BEGIN

!END1

B2

EOF..
100 AS 10 = "#2"
200 AS 20 = "OUTFILE"
400 AS 40 = "#3"
500 DIM INDX$(50), REL$(50), LABLS(99), BEGS(99), ENDS(99)
600 DIM BEG(99), END(99), FILES(3)
700 OPEN #20
1000 GOSUB 8100
5400 FOR FI = 1 TO 3
5500 START = 1
5600 FOR P = 1 TO ATT
5700 GOSUB 8900
5800 SIZE = (ENDD(P) - BEG(P)) + 1
5900 PRINT #10 TAB(11); "#5; TAB(15); FILES(FI); TAB(18); LABLS(P); ->
                   TAB(50); "PIC X("
5910 IF SIZE < 10 DO
5920  PRINT #10 USING "#5; TAB(56); SIZE;
5930 ELSE
5940   PRINT #10 USING "#5; TAB(56); SIZE;
5950 DOEND
5960 PRINT #10 ""
5970 START = ENDD(P) + 1
6100 NEXT P
6200 IF FI = 1 DO
6300 PRINT #10
6400 PRINT #10 TAB(7); "FD OUT-FILE"
6500 PRINT #10 TAB(11); "DATA RECORD IS OT-REC."
6600 PRINT #10
6700 PRINT #10 TAB(7); "01 OT-REC."
6800 ELSE
6900 IF FI = 2 DO
7000 PRINT #10
7100 PRINT #10 TAB(7); "SD SORT-FILE"
7200 PRINT #10 TAB(11); "DATA RECORD IS SORT-REC."
7300 PRINT #10
7400 PRINT #10 TAB(7); "01 SORT-REC."
7500 ELSE
7600 DOEND
7700 DOEND
7800 NEXT FI
7900 PRINT #40 TAB(19); FILES(3); TAB(22); KEYS
8000 STOP
8100 INPUT #20 INRELS
8150 INPUT #20 ATT
8160 FOR P = 1 TO ATT
8170 INPUT #20 LABLS(P)
8180 INPUT #20 BEG(P)
8190 INPUT #20 END(P)
8200 NEXT P
8300 FILES(1) = "IN-"
8400 FILES(2) = "OT-"
8500 FILES(3) = "ST-"
8510 PRINT
8520 PRINT " WHICH KEY(S) IN " INRELS; " DO YOU WANT THE FILE TO" ->
                   " BE SORTED ON"*
8700 INPUT " TYPE IN THE FIELD NAME(S) " KEYS
8800 RETURN
8900 FIL = BEG(P) = START
9000 IF FIL > 0 DC
9100 PRINT #10 TAB(11); "05 FILE"; TAB(50); "PIC X("
9105 IF FIL < 10 DO
   **
   ** INPUT #10 USING "#5; TAB(56); FILE;
PRINT #10 USING "*7TAB(56)";FIL
doend

9115   DOEND
9120   PRINT #10 ");
9200   ELSE
9300   DOEND
9400   RETURN
9800   END
THE CREREL RELATION WAS CREATED TO SERVE AS A DATA ENTRY SYSTEM FOR FREDDB. THE VERSATILITY OF THIS SYSTEM WILL ALLOW THE USER TO HAVE A MAXIMUM RECORD SIZE OF 999 CHARACTERS, THE DATA FOR THIS SYSTEM MAYBE ENTERED IN ANY OF THE THREE (3) WAYS LISTED BELOW.

1). IF THE FILE ALREADY EXIST IT CAN BE ADDED TO THE INDEXLOG BY DEFINING THE RELATION NAME AND ATTRIBUTES USING CREREL.

2). DATA CAN BE ADDED AT THE SAME TIME THE RELATION NAME AND THE ATTRIBUTES ARE DEFINED.

3). DATA CAN ALSO BE ADDED TO THE FILE AT A LATER DATE.

THIS IS A SERIES OF ENTER ACTIVE COBOL PROGRAMS WHICH WILL PROMPT THE USER FOR THE FOLLOWING INFORMATION, TO CREATE A RELATION.

1). NAME OF RELATION TO BE CREATED.
   THIS NAME CAN NOT BE OVER 8 CHARACTERS LONG AND IT SHOULD NOT CONTAIN ANY SPECIAL CHARACTERS (%$&=-?1234).

2). NUMBER OF ATTRIBUTES THE RELATION IS TO CONTAIN.
   THIS NUMBER CAN NOT EXCEED 34.

***** THE FOLLOWING INFORMATION WILL APPEAR *****
***** FOR EVERY ATTRIBUTE THAT IS REQUESTED *****

3). NAME OF ATTRIBUTE.
   THIS NAME CAN NOT EXCEED 20 CHARACTERS.

3.1). WHEN THE LETTER 'B' IS TYPED IN FOR THE ATTRIBUTE NAME IN THE ATTRIBUTE LENGTH FIELD TYPE THE NUMBER OF SPACES THAT IS TO BE PLACED BETWEEN THE ATTRIBUTES.
   THIS WILL NOT COUNT AS A ATTRIBUTE.

4). LENGTH OF ATTRIBUTE.
   THE MAXIMUM LENGTH OF THE ATTRIBUTE.

THE ENTIRE SYSTEM IS IED TOGETHER BY JCL.

FILENAME PURPOSE
---------- ----------
CREATE
CREATE1
CREREL
CRE2

CREATE
CREATES A NEW RELATION
CREATE1
RETREIVES THE ATTRIBUTES
FOR A GIVEN RELATION
CREREL
JCL FOR CREREL SYSTEM
CRE2
JCL FOR INSERTING OF ATTRIBUTES

LINK MODULE PURPOSE FILENAME
NEWRL CREATE A NEW RELATION CREATE
GETRL GET A RELATION CREATE1
EOF.
WELCOME TO SPR

PR FFFFFFFFFF RRRRRRRRR DDDDDDDDD
PR F R R D D D B
PR F R R EEEEEEE D D B
PR F R R EEEEEEE D D BBBBBB
PR F R R E E D B B
PR F R R EEEEEEE D D B B
PR F R R EEEEEEE DDDDDDDDD BBBBBB

FAMU RELATIONAL DATABASE

NOTE: TO GET A LIST OF THE AVAILABLE MODULES IN THE 'FRED6' SYSTEM, TYPE IN:

FRED6.H

NOTE: THESE ARE THE ONLY AVAILABLE MODULES IN THE 'FRED6' SYSTEM.

CREREL EDREL
COPREL COLREL
SORREL MODREL
DELREL RETREL

NOTE: IF YOU WISH TO GET A DESCRIPTION OF EACH MODULE, TYPE:

FRED6.O


EDREL (RELATION EDITOR): THE RELATION EDITOR ALLOWS THE USER TO CHANGE, DELETE, AND INSERT TUPLES (ROWS) IN THE RELATION.

COPREL (RELATION COPIER): THE RELATION-COPIER MODULE ALLOWS A USER TO MAKE AN EXACT COPY OF AN EXISTING RELATION INTO A NON-EXISTING RELATION.

COLREL (COLUMN SPECIFICATION MODIFIER): THIS MODULE ALLOWS A USER TO DELETE ATTRIBUTES FROM A RELATION.

SORREL (RELATION SORTER): THIS MODULE ALLOWS A USER TO SORT THE TUPLES OF A RELATION BASED ON THE CONTENTS OF SOME SPECIFIED ATTRIBUTE IN THE RELATION.
SPR DELREL (RELATION DELETER): THIS MODULE ALLOWS A USER TO SELECTIVELY RETRIEVE DATA FROM A 'FREDB' RELATION.
Ims
SFRE ALL
BBLK
BEGIN $RR, $NR
SIM BS=999
$PR
$PR INPUT THE NAME OF THE RELATION WHICH YOU WOULD LIKE_DELETED?
$PR, $#NME
$PR
$PR #NME IS THE RELATION IN WHICH YOU WOULD LIKE_DELETED? (Y/N)
IRESTA $RR, $#ANG
$IF, ($#ANG=N) $JU BEGIN
$JE 1ERR
$SIF, ($#ANG=Y) $JU 1CONT
1ERR $PR
$PR PLEASE TYPE IN 'Y' OR 'N'...
JU 1RESTA 1CONT
SEL $#NME
$JE, $#, 312, 1ERR3
$JE 1ERR1
$PR
$JU 1INDEX
1ERR1 $PR
$PR #NME IS AN NON-EXISTING RELATION
$PR
$PR WOULD YOU LIKE TO DELETE ANOTHER RELATION? (Y/N)
JU 1RESTA
1INDEX $MO RE
$EO INDEXLOG
$SAE, 1-8, $#NME
$DE 0
$UP
$MO AB
$PR
$FREE ALL
$PR #NME HAS BEEN_DELETED
$PR
$PR WOULD YOU LIKE TO DELETE ANOTHER RELATION? (Y/N)
IRESTA $RR, $#ANG1
$IF, ($#ANG1=N) $JU END
$JE 1ERR2
$SIF, ($#ANG1=Y) $JU BEGIN
1ERR2 $PR
$PR PLEASE TYPE IN 'Y' OR 'N'...
JU 1RESTA
1ERR3 $JE 1NEXT3
1NEXT3 $PR
$PR #NME IS AN INVALID RELATION NAME:
$PR 1) NAME HAS MORE THAN 8 CHARACTERS
$PR 2) FIRST CHARACTER IN THE NAME IS A NUMBER
$PR 3) NAME HAS AN INVALID CHARACTER.
$PR
$PR TRY AGAIN...
$PR
$JU BEGIN
1END 92
EOF..
$MS
$******************************************************************************
$ ** THIS MACRO RETRIEVES A RELATION FROM THE **
$ ** FAMU RELATIONAL DATABASE < FRED > **
$******************************************************************************
$BEGIN $$
BBLK
B2
B2
$PR THIS IS THE RELATION RETREIVING MODULE
$PR
$PR
$FR ALL
$******************************************************************************
$ * GETRL++ IS THE LINK MODULE FROM A COBOL **
$ * PROGRAM THAT SEARCHES FOR A RELATION
$ * ASSIGNMENTS ARE MADE INTERACTIVE TO THE *
$ * SAME FILE IN THE COBOL PROGRAM < TREL > **
$ * GOMIT** IS THE EXECUTABLE LINK MODULE OF A *
$ * COBOL PROGRAM < TREL > THAT RETRIEVE*
$ * A WANTED ATTRIBUTE.
$******************************************************************************
GETRL
$AS -20 = OUTFILE
$SR,F *RRR = 20
$JE INOREC
$FR ALL
$AS 20 = *RRR
GOMIT
$JU INOREC
INOREC $$
$******************************************************************************
BBLK
$PR WOULD YOU LIKE TO TRY AGAIN
$SR, I #RES
$IF (*RES = Y ) JU !BEGIN
BBLK
B2
B2
$STOP $$
$PR **** THIS IS THE END OF THE RETREL RELATION ****
$PR
$PR
$PR
$PR
$ME
EOF..
** WELCOME TO COLREL **

**

THIS COLREL MODULE ALLOWS YOU THE USER TO DELETE ATTRIBUTES (COLUMNS) FROM AN EXISTING RELATION.

TO DELETE A COLUMN FROM A RELATION THE RELATION NAME AND ATTRIBUTE NAME MUST BE TYPED IN WHERE SPECIFIED IN THE MODULE.

PLEASE INPUT THE RELATION NAME>

IS THIS THE CORRECT RELATION NAME ?

PLEASE ENTER (Y) FOR YES OR (N) FOR NO.

IS THIS THE CORRECT RELATION NAME ?

IF (#R=Y) JU ICON

ICON AS 20=#N

AS 10=INPUT

F,10,#N

COBOL I "COL2"

EL #N

OUTREL #N

OUTREL

ME
WHICH FUNCTION WOULD YOU LIKE TO PERFORM

1) CREATE A NEW RELATION
2) WRITE INTO A EXISTING RELATION
3) NO OPERATION

ENTER

 WHICH YOU LIKE TO ENTER DATA INTO THE NEW RELATION?

ENTER ( Y FOR YES OR N FOR NO )

GREEN

 WOULD YOU LIKE TO ENTER DATA INTO THE NEW RELATION?

ENTER ( Y FOR YES OR N FOR NO )

CREATE

CREATING A NEW RELATION.

WRITING INTO A EXISTING RELATION.

GETRL

AS 21=OUTFILE
SR,F #RES=21
JE ISTP
CRE2

SIF ( #N = 1 ) $JU ICRLN
SIF ( #N = 2 ) $JU ICRO
SIF ( #N = 3 ) $JU ISTP

ISTP 02

************** FINISHED ***************
IDENTIFICATION DIVISION.
PROGRAM-ID.   SORT-ALL.
AUTHOR. ROBERT SAWYER.
DATE-WRITTEN. 4-1-82.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. HARRIS-123.
OBJECT-COMPUTER. HARRIS-123.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT IN-FILE ASSIGN TO "W8".
SELECT OUT-FILE ASSIGN TO "W9".
SELECT SORT-FILE ASSIGN TO "SORTING".

DATA DIVISION.
FILE SECTION.
FD IN-FILE
   DATA RECORDS IS IN-REC.
01 IN-REC.

WORKING-STORAGE SECTION.

PROCEDURE DIVISION.
PAR-SORT.
   SORT SORT-FILE ON ASCENDING KEY
       USING IN-FILE
       GIVING OUT-FILE.
       STOP RUN.

EOF...
$MS
B2
EL-LO
SJE.P IERROR1
!ERROR1 EL LR
SJE.P IERROR2
!ERROR2 $PR *** COBCLING &1 ***
COBOL I &1
B2
$PR ** EXECUTING &1 **
VX
B2
$PR *** COMPILATION COMPLETE ***
$ME
EOF...
$MS

BBLK

$SR, N #A = 1

!LOOPY SIF (#A = 13) SJU IRUN

$SR, N #A = #A + 1

$PR

SJU !LOOPY

IRUN $PR

$PR

*** FILE SORTED ***

$SEL LO

$SJE, P 1STOP

$PR

1STOP $ PR

$ME

EOF...
$MS
BBLK
$PR
$PR
GETRL
FR ALL
$AS 12 = OUTFILE
$SR.F #FN = 12
$JE !NREC
$CO #FN W8
BA.C I SORR=8
VX
BBLK
$PR
$PR
$PR
$PR
$PR
$PR
$PR
$CO SORR=C W7
ED W7
$IN 28 W2
$IN 35 W3
UP
CB2 W7
CO W9 #FN
$REND
!NREC $ME
EOF..
BEGIN $R,R,S,R

SPR

SPR INPUT THE NAME OF THE RELATION WHICH YOU WOULD LIKE TO
SPR EDIT?

$R,R,I $NAME

SPR NAME IS THE RELATION IN WHICH YOU WOULD LIKE TO EDIT? (Y/N)

IREST $R,R,I $ANS

$IF,($ANS=Y) $JU IBEGIN
$JE IERR

$IF,($ANS=N) $JU ICONT

IERR $SPR PLEASE TYPE 'Y' OR 'N'...

$JU IRESTAR

ICONT $SPR $NAME

$JE IERR1 $SPR

SPR WOULD YOU LIKE TO CHANGE, DELETE, OR INSERT A TUPLE?

IREST $R,R,I $ANS

$IF,($ANS=CHANGE) $JU ICHLN
$IF,($ANS=DELETE) $JU IDELN
$IF,($ANS=INSERT) $JU IINLN

$JU IERR2

IERR1 $SPR

SPR $NAME IS A NON-EXISTING RELATION

SPR

SPR WOULD YOU LIKE TO EDIT ANOTHER RELATION? (Y/N)

IREST $R,R,I $ANS

$IF,($ANS=Y) $JU IBEGIN
$JE IERR1

$IF,($ANS=N) $JU IEND1

IERR1 $SPR

SPR PLEASE TYPE 'Y' OR 'N'...

$JU IREST

SPR

IERR2 $SPR

SPR PLEASE TYPE IN CHANGE, DELETE, OR INSERT...

$JU IRESTA

SPR

ICHLN $SPR

SPR

SPR INDICATE WHICH TUPLE YOU WOULD LIKE TO CHANGE BY
SPR GIVING CORRESPONDING LINE NUMBER?

$R,R,I $LNC
$DI $NAME $LNC 1
$JE ICHLN

$SPR

$SPR

SPR IS THIS THE TUPLE THAT YOU WOULD LIKE TO CHANGE? (Y/N)

IRESTA1 $R,R,I $ANS

$IF,($ANS=N) $JU ICHLN
$JE IERR3

$IF,($ANS=Y) $JU ICONT1

IERR3 $SPR

SPR PLEASE TYPE IN 'Y' OR 'N'...

$JU IRESTA1

SPR

SPR

ICONT1 $SPR

SPR THIS IS THE TUPLE THAT YOU WOULD LIKE TO CHANGE.

PLEASE TYPE IN THE TUPLE TO YOUR DECIDED TUPLE.
\textbf{IRESTA2 \#ANS}

\textbf{SIF, (#ANS=N) $JU$ IEND}
\textbf{SJE IERR4}
\textbf{SIF, (#ANS=Y) $JU$ ICHLN}

\textbf{IERR4 \#ANS}

\textbf{SPR PLEASE TYPE IN 'Y' OR 'N'... $JU$ IRESTA2}

\textbf{IDELN \#ANS}

\textbf{SPR INDICATE WHICH TUPLE YOU WOULD LIKE TO DELETE}
\textbf{SPR BY GIVING CORRESPONDING LINE NUMBER.}
\textbf{SPR $DI$ #NME #LN 1}
\textbf{SJE IDELN}
\textbf{SPR}

\textbf{PR IS THIS THE TUPLE THAT YOU WOULD LIKE TO DELETE? (Y/N)}

\textbf{IRESTA3 \#ANS}

\textbf{SIF, (#ANS=N) $JU$ IDELN}
\textbf{SJE IERR5}
\textbf{SIF, (#ANS=Y) $JU$ ICONT2}

\textbf{IERR5 \#ANS}

\textbf{SPR PLEASE TYPE IN 'Y' OR 'N'... $JU$ IRESTA3}

\textbf{ICONT2 \#ANS}

\textbf{SJE #LN 1}
\textbf{SJE IERR6}

\textbf{SPR}

\textbf{PR TUPLE #LN 1 HAS BEEN DELETED}
\textbf{PR}

\textbf{PR WOULD YOU LIKE TO DELETE ANOTHER TUPLE? (Y/N)}

\textbf{IRESTA4 \#ANS}

\textbf{SIF, (#ANS=N) $JU$ IEND}
\textbf{SJE IERR7}
\textbf{SIF, (#ANS=Y) $JU$ IDELN}

\textbf{IERR6 \#ANS}

\textbf{PR #LN 1 IS A NON-EXISTING TUPLE}

\textbf{PR}

\textbf{PR WOULD YOU LIKE TO DELETE ANOTHER TUPLE? (Y/N)}

\textbf{SJE IDELN}

\textbf{SPR}

\textbf{PR}

\textbf{PR PLEASE TYPE 'Y' OR 'N'... $JU$ IRESTA4}

\textbf{SPR}

\textbf{IINLN \#ANS}

\textbf{SEL w9}

\textbf{SIF, (#ANS=N) $JU$ IINLN}
\textbf{SJE IERR7}
\textbf{SIF, (#ANS=Y) $JU$ ICONT3}

\textbf{IERR7 \#ANS}

\textbf{PR PLEASE TYPE 'Y' OR 'N'... $JU$ IINLN}

\textbf{SPR}

\textbf{PR INDICATE THE TUPLE THAT YOU WOULD LIKE TO INSERT A NEW}
\textbf{PR TUPLE AFTER BY GIVING THE CORRESPONDING LINE NUMBER.}
\textbf{SPR $DI$ #NME #LN 1}
\textbf{SJE IINLN}

\textbf{SPR}

\textbf{PR}

\textbf{PR IS THIS THE TUPLE THAT YOU WANT TO INSERT AFTER? (Y/N)}

\textbf{IRESTA5 \#ANS}

\textbf{SIF, (#ANS=N) $JU$ IINLN}
\textbf{SJE IERR8}
\textbf{SIF, (#ANS=Y) $JU$ ICONT3}

\textbf{IERR8 \#ANS}

\textbf{PR PLEASE TYPE 'Y' OR 'N'... $JU$ IRESTA5}
CONTEMP

INPUT THE TUPLE(S) IN WHICH YOU WOULD LIKE INSERTED...
NOTE:
FOLLOW THE FORMAT OF THE TUPLE DISPLAYED TO THE TERMINAL.
IF YOU DESIRE TO STOP INSERTING, TYPE 'Q' IN THE FIRST COLUMN.

SPR

DI #NME #LNI 1
READ $SR, I - TUP
$JE !CONTUP
CONTUP $IF, (#TUP=0) $SU IEXIT
$JE !CONTP
CONT P $SPR,F,9,#TUP
$JE ICONTUP
CONTUP $SUJ - READ
EXIT $IN #LNI #9
SPR

SPR WOULD YOU LIKE TO INSERT ANOTHER TUPLE INTO THE RELATION?
$SR (Y/N)
READ $SR, I - AANS
$IF, (#AANS=N) JU IEND
$JE IERR8
$IF, (#AANS=Y) JU IINLN
ERR8 $SPR
$SPR PLEASE TYPE 'Y' OR 'N'...
$SUJ !RESTA6
$SPR
END $SUP
$SPR

SPR WOULD YOU LIKE TO EDIT ANOTHER RELATION? (Y/N)
READ $SR, I - AANS
$IF, (#AANS=N) JU IEND1
$JE IERRR4
$IF, (#AANS=Y) JU IBEGIN
$SPR
$SPR PLEASE TYPE IN 'Y' OR 'N'...
$SUJ !RETT
END1 $2
$MO SI
$ME
EOF...
IDENTIFICATION DIVISION.
PROGRAM-ID. CREATE-A-NEW-RELATION.
AUTHOR. ARTHUR ROBERTS JR.
DATE-WRITTEN. MARCH 10, 1982.
DATE-COMPiled.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. HARRIS-123.
OBJECT-COMPUTER. HARRIS-123.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT UNIT-OUTPUT-FILE ASSIGN "OUTFILE".
SELECT UNIT-INDEX-FILE ASSIGN "INDEXLOG"
ORGANIZATION IS RELATIVE
ACCESS MODE IS SEQUENTIAL
RELATIVE KEY IS REC-POS.

DATA DIVISION.
FILE SECTION.
FD UNIT-OUTPUT-FILE
DATA RECORD IS OUT-REC.
01 OUT-REC.
  02 FILLER PIC X(80).
FD UNIT-INDEX-FILE
DATA RECORD IS INDEX-REC.
01 INDEX-REC.
  02 REL-NAME PIC X(8).
  02 INFORMAT PIC X(991).
WORKING-STORAGE SECTION.
77 REC-POS PIC 9(5) VALUE 1.
77 TREL-NAME PIC X(8) VALUE SPACE.
77 CONDI PIC X(1) VALUE "N".
01 OUTPUT-DATA.
  02 DATA-LINE.
    03 FILLER PIC X(991) VALUE SPACES.
  02 DATA-OUT REDEFINES DATA-LINE.
    03 FILLER PIC X.
    03 NO-CF-ATT PIC X(3).
    03 CCC-CF-ATT OCCURS 34 TIMES.
      04 FILLER PIC X.
      04 ATT-NAME PIC X(20).
      04 FILLER PIC X.
      04 BEG-C PIC X(3).
      04 FILLER PIC X.
      04 END-C PIC X(3).
    03 FILLER PIC X.
  01 WORK-AREA.
    02 BEG-COLUMN PIC 999 VALUE 0.
    02 END-COLUMN PIC 999 VALUE 0.
    02 COLUMN-L.
      04 C=1 PIC X.
      04 C=2 PIC X.
PROCEDURE DIVISION.
PROCESS-CONTROL.
OPEN
OUTPUT-UNIT-OUTPUT-FILE
I-O UNIT-INDEX-FILE.
PERFORM MAIN-Routine THRU MAIN-Routine-EXIT.
CLOSE
UNIT-OUTPUT-FILE
UNIT-INDEX-FILE.
STOP RUN.

MAIN-Routine.
MOVE "N" TO CONDI.
DISPLAY "INPUT RELATION NAME ? ".
DISPLAY "THIS NAME CAN ONLY BE 8 CHARACTERS LONG."
DISPLAY "NO SPECIAL CHARACTERS OR BLANKS."
ACCEPT TREL-NAME FROM TERMINAL.

DISPLAY " ".
DISPLAY "RELATION NAME =>> ", TREL-NAME.
DISPLAY " ".
DISPLAY "IS THIS INFORMATION CORRECT ? (Y OR N) ".
DISPLAY " ".
ACCEPT RES FROM TERMINAL.
IF RES = "Y"
DISPLAY "OK 
PERFORM CHECK-RELATION-NAME THRU CHECK-RELATION-NAME-EXIT
PERFORM CLOS-OPEN-FILE
IF CONDI = "Y"
DISPLAY "ERROR RELATION " TREL-NAME " ALREADY EXISTS"
ELSE
GO TO MAIN-Routine
ELSE
DISPLAY "***** ERROR *****"
GO TO MAIN-Routine.

MAIN.
PERFORM BLANK-DISPLAY 5 TIMES.
DISPLAY "INPUT NUMBER OF ATTRIBUTES ? ".
DISPLAY "THERE CAN NOT BE MORE THAN 34 ATTRIBUTES"
ACCEPT COLUMN-L FROM TERMINAL.

MOVE 0 TO NUM.
INSPECT COLUMN-L TALLING NUM FOR ALL " 
IF NUM = 0
MOVE COLUMN-L TO NO-OF-ATT
ELSE
IF NUM = 1
MOVE C=2 TO M=3
ELSE
    IF NUM = 2
        MOVE C-1 TO M-3
        MOVE "0" TO M-2, M-1
        MOVE HOLD TO NO-OF-ATT
    ELSE
        MOVE COLUMN TO HOLD
        MOVE HOLD TO NO-OF-ATT.
        DISPLAY "*
        DISPLAY "*.
        DISPLAY ".
        IF NO-OF-ATT GREATER THAN '034'
        DISPLAY "** TO MANY ATTRIBUTES => ) NO-OF-ATT
        DISPLAY "**** ERROR ****
        GO TO MAIN.
    IF NO-OF-ATT NOT NUMERIC
        DISPLAY "NUMBER OF ATTRIBUTES MUST BE NUMERIC => "
        NO-OF-ATT
        DISPLAY "**** ERROR ****
        GO TO MAIN.
    ELSE
        DISPLAY "NUMBER OF ATTRIBUTES => " NO-OF-ATT.
        DISPLAY "*
        DISPLAY "*
        DISPLAY ".
        DISPLAY "IS THIS INFORMATION CORRECT? (Y OR N) "
        ACCEPT RES FROM TERMINAL.
        IF RES = "Y"
            DISPLAY " OK
        ELSE
            IF RES = "N"
                DISPLAY "**** ERROR ****
                GO TO MAIN
            ELSE
                DISPLAY "EXPECTING (Y OR N) "
                DISPLAY "**** ERROR ****
                GO TO MAIN.
        PERFORM MAIN-1 THRU MAIN-EXIT
        VARYING SUB FROM 1 BY 1
        UNTIL SUB GREATER THAN 34 OR
        SUB GREATER THAN NO-OF-ATT.
        MOVE TREL-NAME TO REL-NAME.
        WRITE OUT-REC FROM REL-NAME.
        WRITE OUT-REC FROM NO-OF-ATT.
        PERFORM WRITE-Routine
        VARYING SUB FROM 1 BY 1 UNTIL
        SUB GREATER THAN 34 OR
        SUB GREATER THAN NO-OF-ATT.
        MOVE DATA-LINE TO INFOMAT.
        WRITE INDEX-REC.
        WRITE INDEX-REC INVALID KEY GO TO COMPUTE-RECORD-POSITION.
        GO TO MAIN-Routine-EXIT.
        COMPUTE-RECORD-POSITION.
        COMPUTE REC-POS = REC-POS + 1.
        GO TO WRITE-INDEX-RECORD.
        MAIN-Routine-EXIT.
AT END
GO TO CHECK-RELATION-NAME-EXIT.

IF REL-NAME = TREL-NAME
MOVE "Y" TO CONDI
ELSE
GO TO CHECK-RELATION-NAME.

CHECK-RELATION-NAME-EXIT.
EXIT.

CLOSE-OPEN-FILE.
CLOSE UNIT-INDEX-FILE.
OPEN I-O UNIT-INDEX-FILE.

MAIN-1.
PERFORM BLANK-DISPLAY 20 TIMES.
DISPLAY "ATTRIBUTE #", SUB-, "-REQUESTED ", NO-OF-ATT.
DISPLAY "."
DISPLAY
"INPUT THE FOLLOWING FOR SPACES BETWEEN ATTRIBUTES ".
DISPLAY 'FOR ATTRIBUTE NAME ===> B'.
DISPLAY ' '.
DISPLAY 'FOR ATTRIBUTE LENGTH ===> NUMBER OF SPACES'.
DISPLAY ' '.
DISPLAY ' '.
DISPLAY ' '.
DISPLAY 'INPUT ATTRIBUTE NAME ? '.
DISPLAY "THIS NAME CAN BE A MAX OF 20 CHARACTERS".
ACCEPT AT-NANE (SUB) FROM TERMINAL.

DISPLAY "."
DISPLAY "ATTRIBUTE NAME ===> ", AT-TNAME (SUB).
DISPLAY " ".
DISPLAY " IS THIS INFORMATION CORRECT ? (Y OR N)".
DISPLAY "."
ACCEPT RES FROM TERMINAL.
IF RES = "Y"
DISPLAY "OK ".
ELSE
IF RES = "N"
DISPLAY "***** ERROR *****
GO TO MAIN-1
ELSE
DISPLAY "EXPECTING (Y OR N)"
DISPLAY "***** ERROR *****
GO TO MAIN-1.

MAIN-2.
PERFORM BLANK-DISPLAY 5 TIMES.
IF END-COLUMN GREATER THAN 999
MOVE 99 TO SUB
DISPLAY
"*** RECORD HAS REACHED MAXIMUM LENGTH ***"
GO TO MAIN-EXIT.

DISPLAY "ATTRIBUTE #", SUB-, "-REQUESTED ", NO-OF-ATT.
DISPLAY "."
DISPLAY 'INPUT ATTRIBUTE LENGTH ? '
ACCEPT COLUMN-L FROM TERMINAL.
ELSE
    IF NUM = 1
        MOVE C-2 TO H-3
        MOVE C-1 TO H-2
        MOVE "0" TO H-1
        MOVE HOLD TO COLUMN-L
    ELSE
        IF NUM = 2
            MOVE C-1 TO H-3
            MOVE "0" TO H-2, H-1
            MOVE HOLD TO COLUMN-L.
    DISPLAY "*
    DISPLAY "*
    DISPLAY "*
    IF COLUMN-L NOT NUMERIC
        DISPLAY "COLUMN LENGTH HAS TO BE 3 NUMERIC CHARACTERS ==== > "
        COLUMN-L
        DISPLAY "***** ERROR *****"
        GO TO MAIN-2
    ELSE
        MOVE NEXT-COLUMN TO BEG-COLUMN, HOLD=C
        MOVE COLUMN-L TO COLUMN-LEN
        ADD COLUMN-LEN, BEG-COLUMN GIVING NEXT-COLUMN
        SUBTRACT 1 FROM NEXT-COLUMN GIVING END-COLUMN.
        DISPLAY "ATTRIBUTE NAME ==== > ", ATT-NAME (SUB),
        " COLUMN LENGTH ==== > ", COLUMN-LEN.
        DISPLAY "*
        DISPLAY "STARTING POSITION ==== > ", BEG-COLUMN,
        " ENDING POSITION ==== > ", END-COLUMN.
        DISPLAY "*
        DISPLAY "NEXT AVAILABLE POSITION ==== > ", NEXT-COLUMN.
        DISPLAY "*
        DISPLAY "IS THIS INFORMATION CORRECT? (Y OR N)."
        ACCEPT RES FROM TERMINAL.
        IF RES = "Y"
            DISPLAY " OK "
        ELSE
            IF RES = "N"
                MOVE HOLD=C TO NEXT-COLUMN
                DISPLAY "***** ERROR *****"
                GO TO MAIN-2
            ELSE
                MOVE HOLD=C TO NEXT-COLUMN
                DISPLAY "EXPECTING (Y OR N) "
                DISPLAY "***** ERROR *****"
                GO TO MAIN-2.
        IF ATT-NAME (SUB) = "B"
            GO TO MAIN-1
        ELSE
            MOVE BEG-COLUMN TO BEG-C (SUB)
            MOVE END-COLUMN TC END-C (SUB).
    MAIN-EXIT.
    EXIT.
WRITE ROUTINE.
WRITE OUT-REC FROM ATT-NAME (SUB).
WRITE OUT-REC FROM BEG-C (SUB).
WRITE OUT-REC FROM END-C (SUB).

EOF..
END
DATE FILMED
9 - 83
DTIC