TECHNICAL REPORT CR-RD-88-5

FEASIBILITY IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION

The University of Alabama in Huntsville
Huntsville, Alabama 35805

NOVEMBER 1986

Prepared for:
System Simulation and Development Directorate
Research, Development, and Engineering Center

Contract No. DAAH01-82-D-A008

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.
FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION


Final from 12/6/86 to 5/31/86 November 1986

Implementation of the FEASIL database management system under a VAX VMS operating system is described. Technical considerations are reviewed, and new features that resulted and code changes that were required are summarized. Improvements in the SORT function are also described and quantified. A significant amount of FEASIL design information is presented in the form of flow charts for the entire body of code.
This technical report was prepared by the Research Staff of the Electrical and Computer Engineering Department, School of Engineering, The University of Alabama in Huntsville. The purpose of the report is to provide documentation of the technical work performed and results obtained under delivery order 0043 of MICOM Contract No. DAAH01-82-D-A008; Terry N. Long, Principal Investigator. The report also provides an addendum to the documentation of the technical work performed and results obtained under delivery order 0027; Dr. James D. Marr, Principal Investigator. The project was performed by Dr. James D. Marr, Bruce Tucker, Tim Palmer, David Pool, Terry Long, and several other members of the UAH ECE research staff. Dr. M. M. Hallum III, Chief, System Evaluation Branch, was the technical monitor; and Mr. Mark Horton also from the Systems Evaluation Branch of the Research, Development, and Engineering Center, U. S. Army Missile Command, provided technical coordination.

The technical viewpoints, opinions, and conclusions expressed in this report are those of the authors and do not necessarily express or imply policies or positions of the U. S. Army Missile Command.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS</td>
<td>2</td>
</tr>
<tr>
<td>3.0 NEW FEATURES</td>
<td>4</td>
</tr>
<tr>
<td>4.0 CODE CHANGES</td>
<td>5</td>
</tr>
<tr>
<td>5.0 REVISED SORT PROGRAM</td>
<td>10</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>10</td>
</tr>
<tr>
<td>5.2 Revised SORT Technique</td>
<td>10</td>
</tr>
<tr>
<td>5.3 SORT Improvement Quantification</td>
<td>11</td>
</tr>
<tr>
<td>6.0 FEASIL Design Summary</td>
<td>12</td>
</tr>
<tr>
<td>7.0 CONCLUSIONS AND RECOMMENDATIONS</td>
<td>13</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>25</td>
</tr>
<tr>
<td>APPENDIX A - MAIN</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B - RETEVD</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C - DSPLAY</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D - RLFUNC</td>
<td>D-1</td>
</tr>
<tr>
<td>APPENDIX E - PRINTREL</td>
<td>E-1</td>
</tr>
<tr>
<td>APPENDIX F - SORT2</td>
<td>F-1</td>
</tr>
<tr>
<td>APPENDIX G - EDTREL</td>
<td>G-1</td>
</tr>
<tr>
<td>APPENDIX H - NEWRELAT</td>
<td>H-1</td>
</tr>
<tr>
<td>APPENDIX I - DELREL</td>
<td>I-1</td>
</tr>
<tr>
<td>APPENDIX J - MODCOL</td>
<td>J-1</td>
</tr>
<tr>
<td>APPENDIX K - MERGRL</td>
<td>K-1</td>
</tr>
<tr>
<td>APPENDIX L - RORGRL</td>
<td>L-1</td>
</tr>
<tr>
<td>APPENDIX M - BKUPRL</td>
<td>M-1</td>
</tr>
<tr>
<td>APPENDIX N - DBSTAT</td>
<td>N-1</td>
</tr>
<tr>
<td>APPENDIX O - HELP</td>
<td>O-1</td>
</tr>
<tr>
<td>APPENDIX P - TPL1B1</td>
<td>P-1</td>
</tr>
<tr>
<td>APPENDIX Q - TPL1B2</td>
<td>Q-1</td>
</tr>
<tr>
<td>APPENDIX R - TPL1B3</td>
<td>R-1</td>
</tr>
<tr>
<td>APPENDIX S - TPFILECR</td>
<td>S-1</td>
</tr>
<tr>
<td>APPENDIX T - F710</td>
<td>T-1</td>
</tr>
<tr>
<td>DISTRIBUTION</td>
<td>Dist-1</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The FEASIL Database Management System (DBMS) is a relational system; that is, it is designed to most effectively support tabular data. It was first developed about 1978 by M. M. Hallum and has evolved several new features since that time. Currently, it can accept new data from keyboard or file, alter existing data, select only certain parts of the data for further manipulation, sort data into ascending or descending order, perform limited mathematical functions, display or plot data, print tables of data, output data for use by other programs, and perform several other functions.

FEASIL was first developed under a Perkin-Elmer OS-32 operating system and has since received many refinements through several revisions. The principal task of the work documented by this report was to transfer FEASIL to a VAX-11/780 computer under a VMS operating system. Section 2.0 describes several of the technical considerations which affected the work. New features which resulted from the transfer are then described in Section 3.0. Code changes in addition to the new features were required for the transfer. They are documented in Section 4.0. Major performance improvements resulted during the transfer with the SORT function receiving the greatest improvements. These improvements are described and quantified in Section 5.0. As part of this and previous tasks, an effort was made to carefully examine the body of FEASIL code through development of flow charts. This design summary effort is described in Section 6.0, and the flow charts are presented in the appendices. And finally, an extensive list of recommendations and conclusions is presented in Section 7.0.
2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS

The principal goal of this project was to port FEASIL from a Perkin-Elmer (P-E) computer to a VAX computer. Differences between the P-E and VAX that made this conversion nontrivial included file structure differences, different file name limitations, Flex dialect differences, and operating system differences. Within the main goal was the requirement that the transported version be upward compatible and have substantially similar capabilities. A secondary goal was to improve feedback to the user.

From the programmer's viewpoint, the P-E files reside directly on a device, with pathnames and filenames such as

```
USER:FILE.EXT
```
typical. On the VAX, the device is subdivided into directories and possibly subdirectories, with a name such as

```
dual:[userdisk1.humanbeingname.activityname]filename.extension
```
typical. The device name and directory path for a first level user subdirectory is typically 28 characters. Further, files in the "current default directory" of the VAX can be accessed without either the device name or the directory path name. This forces creation of a large buffer for device name (including path) and the option to have no device name.

Filenames on the P-E are limited to 8 characters with a 3-character extension, as noted above. The VAX filenames are limited to 39 characters with a 39-character extension. The easiest solution is to just continue use of the 8+3 filenames. Since the directory name is also limited by the same 39-character limit, and operator tolerance is the principal limit on the number of levels of subdirectory, the device name buffer and any descendant buffers should be huge. As a practical limit, we assumed 44 characters as an upper bound for name and path.

The most annoying difference between the Flex dialects was that the construct

```
WHEN(I.LT.0)
  I=0
  GO TO 73
ELSE
  I=1
FIN
.....
```
converts without error on the P-E, while it produces code on the VAX of the form

```
IF(.NOT.(I.LT.0))GO TO 99999
I=0
GO TO 73
GO TO 99998
99999 I=1
99998 .......
```
This causes a FORTRAN compile error because the statement "go to 99998" is unreachable. A more significant problem is that some of the more powerful FORTRAN 77 (F77) constructs (such as the "if...then...else...endif") have keyword conflicts with Flex, making them incompatible. However, Flex can understand only capital letters; making all F77 constructs lower case lets them pass the flecs preprocessor unmodified and be compiled.

Operating system differences became most apparent in file operations. The two computers differ in their mechanisms for renaming files, but that problem can be overcome within the FILEREN subroutine itself. As expected, I/O error numbers were incompatible, so a separate routine called VAXERR was created and linked through the original error handler. Tape handling is sufficiently difficult that the backup function was forced into a stand-alone program.
3.0 NEW FEATURES

New features visible to the user fall into two categories, flexibility and feedback. The first group includes menu modifications. The second group consists of messages to inform the user of current activity and of the progress of long-duration activities.

The main menu originally had functions 0 to 11 (function 12 is limited to system programmers). The VAX version has the same choices, but will also accept commands by the first letter(s) of the command name; the user need enter only enough letters for a nonambiguous choice. This is of benefit to casual users and novice typists who can reach letters more easily than digits. Another system program function (13) has been added to perform a brute-force tuple dump to the printer; this is effective for archival dumps and for debugging. Help options are available at the main menu and at several of the secondary menus.

FEASIL now provides feedback approximately every twenty seconds or less. When the original .ADF and .TF are being copied at the start of "retrieve, manipulate, and plot" (8), the program declares that the copy is beginning and reports every hundred records copied. The SORT function now provides a crude advance estimate of the time needed and periodically reports how the bubble sort is progressing. The DELETE function in EDIT (4) now reports every twentieth record for large delete actions.

Additional feedback information was added to several of the error messages and to dangerous activities. When a FORTRAN error occurs, instead of giving machine-dependent error numbers, the system now generally gives a text string defining the error. Since the MOVE (M) command in RETRIEVE, MANIPULATE AND PLOT can delete tuples from the original relation, the user is now warned a second time that he has made a dangerous choice and asked to reconfirm the decision.
4.0 CODE CHANGES FROM PERKIN-ELMER VERSION

Changes to the program were made for two reasons, machine factors and program enhancement. Differences in the file structure were the main reason for the machine-dependent changes, and an attempt was made to confine these to the F710 library. Program enhancement changes included message changes, conversion from Flex to FORTRAN 77, and algorithm changes. Errors found in the flowcharting project were corrected on both the VAX and P-E versions; the errors were predominantly evolutionary in origin, frequently being related to the string-in-TF data compression method. The structure of the .TDF, .ADF, and .TF files was maintained for compatibility with the P-E FEASIL. The rest of this section will provide a brief description of the changes to each program file.

Runtime memory usage was considered first. Currently, the main menu driver and all five libraries are permanently resident in memory, with the ten main-menu options overlaid. A secondary overlay is used for some of the activities under "RETRIEVE, MANIPULATE, AND PLOT" (option 8). Specifically, only one of FUNCTION, PLOT, PRINT, or SORT will be in memory at a given time. Table 1 shows that while many subprograms are called by all ten activities, some are called only by one of the main menu choices. An analysis of this subprogram calling hierarchy indicates that a restructuring of the libraries would be both possible and beneficial. The subprograms and their locations are listed in Table 2, and Table 3 shows the same information from the opposite perspective, location and contents. A file-by-file analysis of changes follows.

P-E files have read/write keys, but these are not available on the VAX; FEASIL generally does not ask for keys on the VAX. A more general rewrite would add a function in the machine-dependent library F710 to request or not request keys.

The file BKUPSTUB (BACKUPRL on the P-E) contains the tape backup routine. Since operating system restrictions on the VAX make the traditional backup method extremely inconvenient, the backup routine, BACKUPRL, has been converted to a "stub" explaining how to call the current version of the stand-alone input program for tapes generated by FEASIL on the P-E. It is suggested that backup on the VAX be performed by standard VAX procedures. Efforts are in progress to write a program to prepare tapes for return to the P-E. One of the smaller compatibility problems is that the VAX and P-E disagree on the byte order for integer numbers on a tape; they agree on text order. Additionally, the VAX insists on specialized system calls for tape data transfer rather than standard FORTRAN READ and WRITE statements.

The file DABA08 contains both the main program and a BLOCK DATA initialization section. The main program is predominantly F77, which more effectively supports the text selection of desired activity. The main menu handler now accepts letters as well as digits; only upper case is supported to remain consistent with the following menus in the original P-E version. All subprogram calls have been modified so that the array length is also passed for any array sent to the subprogram. In the future, it will be possible to dimension all arrays in subroutines to the correct
length activate runtime subscript checking, and eliminate most references to the LCONST COMMON; this will aid in discovering runtime errors that can corrupt data. The COMMON's in the BLOCK DATA section have been adjusted to a minimum length of 19 words to accommodate the possibility of longer filenames on the VAX. The new COMMON named MAXLEN has been added to indicate the longest allowable name, which is set at 13 words at the present. The default device/directory name has been set to 0 characters with text of "Current System Default Directory"; the FILDES routine (elsewhere) was also tampered to be able to echo this default despite its zero length.

The file DABAEXIT contains the status reporting routine DBSTAT. It originally supported the normal status function (11 or S) and the system programmer function (12 or P). It has been modified slightly toward F77, but the principal change was the addition of a second system function TOTAL (13 or T), which does a total dump of the entire relation to the printer. Damaged or defective ADF records are reported, rather than system failure resulting. Suppression of this second system function could be accomplished most easily by modification of the main menu to reject the new commands. The function to be performed is now specified by a character variable passed as the final parameter. The modifications were also retrofitted to the P-E version.

The file DELETREL contains the relation deletion routine DELREL. It is still in Flex, but is only 46 lines long including comments. No changes were necessary.

The file DSPLAY contains the plotting routine DSPLAY, originally written by M. Castellano based on earlier code by J. Marr. It was changed only slightly and still relies heavily on Flex. The printer device number was changed from 13 to 6. The pen plotter function is still a nonfunctioning stub. It could stand some modification for compactness and efficiency since it reimplements some functions available in the libraries.

The file EDTREL7 contains the data entry and editing routine EDTREL. Few changes were made in the conversion process, but several logic errors were discovered during flowcharted and were corrected on both the VAX and P-E versions. The most visible changes were periodic feedback in the add-data-from-a-file function (A), the search/find function (F), and the delete function (D); all now comment on progress about every 20 seconds.

The file F710 is a machine-dependent library. All routines have built-in error checking and messages. Subroutine FILEOPN uses the standard OPEN function for 'OLD' and 'DIRECT'. Subroutine FILEDEL on the VAX opens a file and then closes it with 'DELETE' option; this may be portable for future use. Subroutine CREAFIL checks for file prior existence (OPEN as 'OLD') before creating the file, using an OPEN as 'NEW' and 'DIRECT' with specified record length and file length. The check for prior existence is because the VAX allows creation of a new version of an existing file without producing an error message. Subroutine FILEREN required extensive rewriting. The name of the file on the input logical unit is determined using an INQUIRE; the logical unit is closed; the file is renamed by a system library function; and the file (under either the new or old name) is reOPENed on the logical unit. Subroutine FILECLS consists of only the
FORTRAN CLOSE function. Subroutine MODAP is a stub with no executable statements. Finally, subroutine VAXERR accepts an error number, prints to the screen its meaning (if known), and reports back whether a meaning was printed.

The file FUNCTION contains the mathematical functions routine RLFUNC used in option 8. It was modified to run faster by simple rearrangement of some of the code. A logic error was also detected that caused an infinite loop for partially active data; this was corrected in both versions. It still uses Flex.

The file HELP contains the help routine HELP.

The file MERGERL contains the relation-merging routine MERGRL. It now provides a progress report every 20 lines during the add (A) function.

The file MODCOLUM contains the column-modification routine MODCOL. It was substantially unchanged, and still relies heavily on Flex.

The file NEWRELAT contains the relation-creation routine NEWREL. It was changed only slightly and still relies heavily on Flex.

The file PRINTREL contains the printing routine PRINTREL used in option 8. It was changed to account for text length differences. It still relies heavily on Flex.

The file REORGREL contains the relation-reorganization routine REORGRL.

The file RETRIEV7 contains the retrieve/manipulate/plot function RETEVD, which is option 8. Most modifications were implemented to improve readability of the code or feedback to the user, but some logic errors were also eliminated. A warning and request for verification were added to the delete-tuples-after-move option on the move command (M). It relies heavily on Flex.

The file SORT2 contains five routines in support of the sort function used in option 8. The CSORTF routine builds a file containing the tuple number and the tuple as text data from the designated column; it was not changed and still relies heavily on Flex. The SORTFL routine was substantially rewritten and details appear in the next section. The supporting routines SORTFA, ASCTYP, ASCCHR are new and include only two instances of Flex.

The file TPFILECR is a library of routines related to file creation. Unless noted, changes are for readability and feedback only. Subroutine FILDES inquires directory name (calling it "volume name"), verifies legality, updates the default volume name if appropriate, and builds an extended name of volume and filename and period. FILDES needed substantial modification to handle the changes in names on the VAX. The handling of the "default directory" message was ad hoc, and should be changed in later revisions. SECURE gets the read/write keys, but is not called on the VAX; it could be replaced by a stub. EMIARY just writes zeros into an array; it
now consists of a DO loop. ADFCRE, TDFCRE, and TFCRE add the appropriate extension to the base relation name and create the file via CREATFIL. RELFC just calls these three routines. ADFDEL, TDFDEL, and TFDEL add the appropriate extension to the base relation name and delete the file via FILEDEL. RELFD just calls these three routines. EQUIVI copies N words from one array into another; it now consists of a pretest and a DO loop. OPNADF, OPNTDF, and OPNTF add the appropriate extension to the base relation name and open the file on the specified logical unit. FILERR checks for recognized FORTRAN error numbers, but is somewhat machine dependent. NAME gets a relation name of not more than 42 characters, which is later "hashed" to make the file name. All of these routines were originally Flex, but some have been altered to remove some of the vestiges. The OPNTF routine originally used a FORTRAN COMMON area for building the extended name. However, the RETEVD function, which calls OPNTF, used the same COMMON for another purpose, and the conflict resulted in system malfunctions. As a result, OPNADF, OPNTDF, OPNTF, ADFCRE, TDFCRE, TFCRE, ADFDEL, TDFDEL, and TFDEL were modified to eliminate use of the COMMON. This change to local arrays caused no change in the size of the .obj file and reduced the size of the .exe (executable) file.

The file TPLIB1 is a library of text and number manipulation routines. Few changes were necessary in this file, and most routines rely heavily on Flex. STRLEN reports the length of a string and copies it to FEASIL format. FTIPI converts a floating point (real) number to pseudo-integer by saving it to memory as "real" and reading it back as an integer; an EQUIVALENCE installed in a rewrite about 1984 simplified the code and made it portable. PITFP converts pseudo-integer to real by the same approach. YESNO demands from the user an answer of YES or NO (Y or N); it still uses FEASIL string manipulation techniques. Subroutine GETNUM gets text in alpha-numeric form from the user and converts it via REANUM. REANUM converts text to the corresponding integer value. ELEB eliminates leading edge blanks by shifting the FEASIL format text left. ENFPTA converts real numbers to FEASIL string format. ENINTA does the same for integers. GETFP converts a text number from the terminal into floating point. REAFP does the actual conversion. All of the routines in this library use the GET routine to read characters from the terminal.

The file TPLIB2 is a library of routines for file access. Few changes were necessary; about half of the routines are pure F77. Routines PUTTDF and RTVTDF open the TDF file via OPNTDF, write or read the file, respectively, and close the file via FILECLS. ADDTUP adds a tuple to an existing relation. DELTUP removes a tuple. PUTIFO puts data into the .ADF or .TF file as appropriate for the data strategy. PTFRC and PADFRC write data to the .TF and .ADF files, respectively. If the string sent to PADFRC is shorter than 8 characters and an override has not been declared, then PADFRC will return a signal and two words to be stored in the TF. RTFRC and RADFRC read data from the .TF and .ADF files respectively. COLNAM gets the name of a specified column from the .ADF file. HASHIT computes the file name by hashing the relation name. The first four letters of the hash name are the same as for the relation name; the fifth is a length signal; the sixth and seventh are a hash of relation name characters 5 to 42; and the eighth letter is 'A'. It was modified recently (1985) to use an algorithm which
reduces hashing collisions; the P-E version checks to see if a file from the previous version exists and recommends conversion. UTFAR converts the two TF words into a 1 to 7 character FEASIL format string into the ADF buffer, inverting the TF storage mode of PADFRC.

The file TPLIB3 is a library of routines for file maintenance. RDOADF and RDOTF create a larger .ADF/.TF file, copy the old information into it, and delete the old version. CPYREL copies both the .ADF and .TF files into a new name. FSLIDE shifts an array downward one word and is used to convert FEASIL format text into FORTRAN format. Only slight changes have been made from the original Flex text.

Some additional tools have been built as part of the conversion process. NAMES examines an existing relation, reports the long name, and prints a macro to rename the relation to the new hash code from either the original "hash" or the original name. QUICK is a VAX program to convert a tape-transferred relation from P-E tape format (copied to the disk first using MOUNT/FOREIGN and COPY) to a standard relation. The command file which does the copy and calls QUICK is named FEASILTAPE. A relation repair function named PATCH is under development.
5.0 REVISED SORT PROGRAM AND TIMING TESTS

5.1 Introduction

Users have long complained of the slowness of the SORT function, and it has been speculated that this is due to the frequent disk accesses of the file-to-file algorithm. When an attempt was made to run the original algorithm on the VAX, the machine malfunctioned. It was then decided to modify the routine. The original sort program operated in two phases, (1) copying of the tuple number and the data from the designated column to a sort file SF in CSORTF and (2) sorting of the data in SORTFL. These two routines were called in sequence by the RETEVD routine. In order to preserve modularity, all changes were made internal to the SORTFL routine.

5.2 Revised SORT Technique

Three acceleration measures were instituted simultaneously. They were an in-memory routine for small data sets, a different buffering scheme for other sorts, and code tightening. The new SORTFL routine first builds file names and opens the SF to a logical unit. If the strategy is nonstring and the tuple count is less than 1000, it calls SORTFA to do the in-memory sort and then returns. Otherwise, it creates two scratch files 11 and 12. If the text is nonstring, the records of these files are only two words wide. The SF is then copied into 11, and the data is sorted using 11 and 12 as "ping-pong" buffers. When the sort is completed, the data is transferred back to the SF as text, the files are closed (and scratch files deleted), and the program returns.

The sort algorithm now used is a weaving bubble sort. Sorting starts at the bottom, and the largest value is carried to the top, remembering when the last swap occurred. Next, from the point of the last swap, the smallest value is carried to the bottom, again remembering the last swap point. The swap points are the new limits for later passes, and execution can stop when they meet. The sort code reports the number of data values swapped every 100 passes to maintain user confidence that the program is still running.

The fast in-memory sort SORTFA also uses the weaving bubble sort, but does not need scratch files. It was decided, for code compactness, to have only one sort sequence and make the order decision based on strategy. The use of logical variables and logical assignment statements and comparisons also aided compactness.

Two auxiliary functions were added to aid the sort. ASCTYP returns an integer value to indicate whether the character input is (1) a blank, (2) a capital letter, (3) a lower case letter, (4) a digit, or (5) something else. This appeared to be close to the original sequence, but there was some evidence that digits should precede letters instead. If some other sorting sequence is desired in the future, this routine could be modified easily. ASCCHR accepts two character values c1, and c2, and returns a logical value of TRUE if c1 is less than or equal to c2. ASCCHR uses ASCTYP to separate nonsimilar characters.
5.3 SORT Improvement Quantification

The new version appears to require less memory. The rewrite reduced the .FLX file from 953 to 884 lines, and the derived .FOR (FORTRAN) file from 648 to 500 lines. The .OBJ file which contains resulting assembly code and linkage information was reduced from 32 to 31 blocks, a negligible difference but still a reduction.

The major difference is in speed. The new algorithm requires about 10 seconds for the CSORTF conversion and un-conversion, and 23 seconds to sort 500 random floating point numbers (263 passes). The old algorithm took 10 + 600 seconds. This is a factor of 20 to 25 speed improvement. For larger data sets, the operating system handles buffering and the nonstring records are packed many to a disk block, instead of singly. This results in fewer disk accesses and greater speed of execution. String data appeared to sort slightly faster, but this is hard to measure on a multiuser system.
6.0 FEASIL DESIGN SUMMARY

A major effort was initiated under a previous task to complete examine the body of FEASIL code through flow charting it. This effort greatly assisted the transfer process through identification of dormant errors in FEASIL code. The flow charts are presented in Appendices A through T. They should prove to be a valuable tool in the analysis of errors still hidden in the VAX implementation. They should also assist in the refinement process that will inevitably follow this effort.
Limited testing reveals no errors, so the system appears ready for production work. As with any large software system, problems will appear from time to time and can be treated as they are revealed. Recommendations for future effort can be divided into two areas, enhancement and rewrite.

Several functions could be added to the FEASIL DBMS to enhance its power. The most powerful function would be a mathematical function to compute a linear or nonlinear function of two columns and store the result in a third column. Small changes that might aid the user would include modifying the HELP function to loop, offering its menu again, until the user selects "quit."

Small invisible code changes which would improve performance include (1) increasing the buffer size in SORTFA and (2) when deleting tuples in the edit function, delete the one with the highest number first. The current tuple delete function removes the lowest numbered one and shifts all tuples above it downward; it then deletes the next one. If a relation is 100 tuples long and tuples 11 to 100 are being deleted, then tuple 11 is removed, all 89 remaining are shifted, the new tuple 11 is removed, all 88 remaining are shifted, .... In the same case with the suggested change, no shifts would occur and thousands of disk accesses would be eliminated.

FEASIL has evolved into a powerful tool. It would be desirable to eventually port FEASIL to the HP 9000, the IBM PC, and several other computers. However, the flowcharting and VAX conversion efforts have revealed some problems that impact on portability.

FEASIL was originally written in Flex on a machine subject to frequent malfunctions. Several compromises were made as a result.

(1) Most file accesses (read or write) are preceded by an open and followed by a close; the overhead and disk accesses of these opens and closes can slow down a program significantly.

(2) The string handling capabilities of Fortran 77 were not available, and several activities were performed in a difficult fashion. For example, a 16-character string can be shifted left one character in F77 by

\[
\text{str} = \text{str}(2:16)
\]

or the eighth letter of a file name can be changed from 'A' to 'B' by

\[
\text{file}(8:8) = 'B'
\]

Other similar operations are as simple. HASHIT was written using the substring capability shown and is much more compact than the functionally equivalent Flex code would be.
(3) As in any evolving program, array lengths changed with time and arrays were declared as IDENT(1) in subprograms; this forced use of the LCONST COMMON and variables for the constants from 0 to 30 to bypass the automatic subscript checking of FORTRAN.

(4) Ad hoc changes have crept into the code, and nesting has resulted in redundant or contradictory testing for certain conditions. For example, two consecutive blocks of code are conditioned on the same test in DBSTAT rather than merging the blocks and only testing once.

(5) The Flex loop construct appears frequently. The code

\[
\begin{align*}
I &= 0 \\
&\text{REPEAT UNTIL (I.EQ.N-1)} \\
&\quad A(I+1) = 0. \\
&\quad I = I + 1 \\
&\text{FIN} \\
&\text{....}
\end{align*}
\]

results in FORTRAN code similar to

\[
\begin{align*}
I &= 0 \\
&\text{GO TO 99999} \\
&\text{99998 IF(I.EQ.N-1)GO to 99997} \\
&\quad A(I+1) = 0. \\
&\quad I = I + 1 \\
&\quad \text{GO TO 99998} \\
&\text{99997 ....}
\end{align*}
\]

rather than the F77 code of

\[
\begin{align*}
\text{DO} & \text{ 10 I = 1, N} \\
10 & \quad A(I) = 0.
\end{align*}
\]

which will probably compiles to more compact and faster assembly language code.

(6) Some of the routines, such as EDTREL, have evolved to over 1000 lines. Even with the use of Flex macrofunctions, this is a larger amount of code based on the same global variables than the average programmer can grasp. Reduction of program unit size and more use of subprograms would simplify the code.

(7) Several functions implemented as Flex macros occur in multiple top-level routines. For example, both EDTREL and RETEVU (main menu options 4 and 8) ask for a column number. This is just a request for a number within a certain range, and could be implemented trivially as an integer FORTRAN FUNCTION in a library.

(8) Several routines, such as TFDEL, ADFDEL, and TDFDEL differ only in one location. A generic routine would reduce code space by nearly a factor of 3; the routine could even be called by the existing routines, each of which could be reduced from 24 to 5 lines.
Flowcharting revealed that several routines frequently appear in sequence. A new routine implementing the entire function would shorten the calling program and improve readability. For example, a new routine EXTEND could call EQUIVI, CATSTR, and FSLIDE to create an extended name from the specified base name and specified extension. This would cost one 6-line routine but reduce dozens of triple calls to single highly readable calls.

Given one machine, it was not obvious which routines were really machine dependent, and such functions occur in many places. For example, the read/write keys should be requested in a machine-dependent subroutine. Similarly, all questions could be asked through a subroutine QUEST which prints the test without a carriage return (if the machine allows such an option). A logical function QUESYN could call QUEST to provide the text and then call YESNO to demand a yes or no answer.

A complete top-down restructuring of the code is recommended. Guidelines would include (1) pure portable Fortran 77 code consistent with the standard, (2) copious use of comments in the code, (3) routines rarely longer than 2 pages including the imbedded comments (header comments do not count), (4) addition of new functions/routines whenever appropriate, and (5) restructuring of the libraries to minimize runtime size and to place machine-dependent code in as few files as possible. Implementation of guideline 1 along would result in an estimated code reduction as shown in Table 4; the quantitative benefits of the other guidelines have been estimated.

Two additional recommendations are related to "user-friendliness". They are (6) any potentially dangerous or time-consuming activity should offer a 'punt' option, and (7) new algorithms should be considered for time-consuming activities. The code should also allow graceful exits from accidentally chosen paths, lest the user be stuck in an infinite question-answer loop that can only be exited by damaging the relation or by killing the program and losing all previous work. Similarly, any activity, such as sort, which could take hours to perform should offer a graceful exit after providing a time estimate; the user may prefer an unsorted list now rather than a sorted list in three hours (or even fifteen minutes). The rewrite of SORT was a start toward algorithm improvement, but room exists for more improvement.

Implementation of FEASIL on the VAX has greatly improved the usefulness of the system. Reduced execution time and more reliable data handling has greatly increased its acceptance by users. Continued refinement and elimination of any errors that may remain should guarantee user acceptance in the future.
### TABLE 1: SUBPROGRAM CALLS OF FEASIL ROUTINES

<table>
<thead>
<tr>
<th>name</th>
<th>called by (directly, digit=indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>newrel</td>
<td>!</td>
</tr>
<tr>
<td>delrel</td>
<td>.!</td>
</tr>
<tr>
<td>edtrel</td>
<td>...!</td>
</tr>
<tr>
<td>modcol</td>
<td>...!</td>
</tr>
<tr>
<td>mergrl</td>
<td>...!</td>
</tr>
<tr>
<td>rorgrl</td>
<td>..... !</td>
</tr>
<tr>
<td>retevd</td>
<td>..... .!</td>
</tr>
<tr>
<td>bkuprl</td>
<td>..... ..!</td>
</tr>
<tr>
<td>dbstat</td>
<td>..... ....!</td>
</tr>
<tr>
<td>help</td>
<td>..... ..... !</td>
</tr>
<tr>
<td>rdoadf</td>
<td>1 ..**. ..... !</td>
</tr>
<tr>
<td>rdotf</td>
<td>1 ..*.. ..... .!</td>
</tr>
<tr>
<td>putifo</td>
<td>1 ..*.. ..... ..!</td>
</tr>
<tr>
<td>cpystr</td>
<td>1 ..*.. ..... ..!</td>
</tr>
<tr>
<td>rlfunc</td>
<td>1 ..... ..*...... !</td>
</tr>
<tr>
<td>dsplay</td>
<td>1 ..... ..*...... .!</td>
</tr>
<tr>
<td>printrell</td>
<td>..... ..*...... ..!</td>
</tr>
<tr>
<td>csortf</td>
<td>1 ..... ..*...... ....!</td>
</tr>
<tr>
<td>sortfl</td>
<td>1 ..... ..*...... .....!</td>
</tr>
<tr>
<td>asctyp</td>
<td>2 ..... ..... ..... ....*t</td>
</tr>
<tr>
<td>ascchr</td>
<td>2 ..... ..... ..... ......*t</td>
</tr>
<tr>
<td>sortfa</td>
<td>2 ..... ..... ..... ......*t</td>
</tr>
<tr>
<td>cpyrel</td>
<td>1 ..... ..*...... .....!</td>
</tr>
<tr>
<td>ftipf</td>
<td>1 ..<em>.. ..</em>...... ...... t</td>
</tr>
<tr>
<td>pitfp</td>
<td>1 ..<em>.. ..</em>...... ..<em>..</em>.. t</td>
</tr>
<tr>
<td>enfpta</td>
<td>1 ..<em>.. ..</em>...... ......*..!</td>
</tr>
<tr>
<td>eninta</td>
<td>1 ..<em>.. ..</em>...... ......*..!</td>
</tr>
<tr>
<td>utfar</td>
<td>1 ..<em>.. ..</em>...... ......<em>..</em> ..t</td>
</tr>
<tr>
<td>eleb</td>
<td>2 ..<em>.. ..</em>...... .....*..... **!</td>
</tr>
<tr>
<td>name</td>
<td>1 ***** *****, ..... ....*...!</td>
</tr>
<tr>
<td>hashit</td>
<td>1 ***** *****, ..... ....*...!</td>
</tr>
<tr>
<td>fildes</td>
<td>1 <em>1</em>** ,***<em>, ..... ......</em>...!</td>
</tr>
<tr>
<td>yesno</td>
<td>1 <em>1</em>*** ,*****, ..... ......*...!</td>
</tr>
</tbody>
</table>

**MAIN**

<table>
<thead>
<tr>
<th>rorgrl</th>
<th>rlfunc</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>retevd</td>
<td>dsplay</td>
<td>hashit</td>
</tr>
<tr>
<td>newrel</td>
<td>bkuprl</td>
<td>printrel</td>
</tr>
<tr>
<td>delrel</td>
<td>dbstat</td>
<td>csortf</td>
</tr>
<tr>
<td>edtrel</td>
<td>help</td>
<td>sortfl</td>
</tr>
<tr>
<td>modcol</td>
<td>cpystr</td>
<td></td>
</tr>
<tr>
<td>mergrl</td>
<td>rdoadf</td>
<td></td>
</tr>
<tr>
<td>rdotf</td>
<td>enfpta</td>
<td></td>
</tr>
<tr>
<td>cpystr</td>
<td>eleb</td>
<td></td>
</tr>
</tbody>
</table>

(VP11 - MNA.)
<table>
<thead>
<tr>
<th>name called by</th>
<th>(*-directly, digit-indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>relfc</td>
<td>1 *... .... . .... .... !</td>
</tr>
<tr>
<td>relfd</td>
<td>1 .* .... . . .... !</td>
</tr>
<tr>
<td>addtup</td>
<td>1 .* .* *!</td>
</tr>
<tr>
<td>deltup</td>
<td>1 .* .* !</td>
</tr>
<tr>
<td>colnam</td>
<td>1 .* .* *!</td>
</tr>
<tr>
<td>adfcre</td>
<td>1 .1.1* .**. .* .... .... !</td>
</tr>
<tr>
<td>tdfcre</td>
<td>1 .1.* .**. .* .... .... !</td>
</tr>
<tr>
<td>tfcre</td>
<td>1 .1.1* .**. .* .... .... !</td>
</tr>
<tr>
<td>adfdel</td>
<td>1 <em>!</em> *!</td>
</tr>
<tr>
<td>tdfdel</td>
<td>1 **!1111 *1111 *!</td>
</tr>
<tr>
<td>tfdel</td>
<td>1 <em>!</em> .**. .* .... .... !</td>
</tr>
<tr>
<td>radfrc</td>
<td>1 .1.* .**. .* .... .... !</td>
</tr>
<tr>
<td>rtvtdf</td>
<td>1 .*** .*** .* .... .... !</td>
</tr>
<tr>
<td>rtfrc</td>
<td>1 .*** .*** .* .... .... !</td>
</tr>
<tr>
<td>padfrc</td>
<td>1 .*** .*** .* .... .... !</td>
</tr>
<tr>
<td>puttdf</td>
<td>1 .** .*** .* .... .... !</td>
</tr>
<tr>
<td>ptfrc</td>
<td>1 .<strong>! .</strong>* .* .... .... !</td>
</tr>
<tr>
<td>opnadf</td>
<td>2 111111 1112 . .!2**! . . ....!</td>
</tr>
<tr>
<td>opntdf</td>
<td>2 11111111 . ...!* . !</td>
</tr>
<tr>
<td>opnrf</td>
<td>2 .11111111 .!1 . **<em>!</em> . !</td>
</tr>
<tr>
<td>creafil</td>
<td>2 .2211 1<em>11 . 11 . .!111 . .111 ..</em></td>
</tr>
<tr>
<td>filecls</td>
<td>1 !1111 1*11 . !111111 . !11111111 .1</td>
</tr>
<tr>
<td>fileren</td>
<td>1 ..11 .!* . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>filererr</td>
<td>1 12113 11!1 . .1 . .!222 .11111111 .1</td>
</tr>
<tr>
<td>emairy</td>
<td>1 .111* .**!1 . .!111 . !11111111 .1</td>
</tr>
<tr>
<td>equivi</td>
<td>1 .11111111 . .**!1 . .!11111111 .1</td>
</tr>
<tr>
<td>filedel</td>
<td>1 111 .!**22 . .! . . !</td>
</tr>
<tr>
<td>fileopen</td>
<td>1 222221 1*33 .111222 .11111111 .1</td>
</tr>
</tbody>
</table>

**TABLE 1** SUBPROGRAM CALLS OF FEASIL ROUTINES (continued)
### TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES (continued)

<table>
<thead>
<tr>
<th>name</th>
<th>called by (directly, digit-indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>getfp</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>reafp</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>getnum</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>reanum</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>getch</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>get</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>streq</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>strlen</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>cpsub</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>catstr</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>catsub</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>fslide</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
<tr>
<td>chtyp</td>
<td>main, rorgrl, rfunc, name, -cre, opn-</td>
</tr>
</tbody>
</table>

Note 1: The symbol to the right of each line indicates the type of routine:

- t: terminal routine (calls no others)
- m: machine-dependent (only system calls or assembly language)

Note 2: Some routines are included in the source, but not called:
- catnum: (calls catstr, putnum)
- closef
- filrenb
- hash
- modap
- newno
- opnrel
- put: (calls catsub, cpsub, cpsub, putch, putnum)
- putch
- putnum: (calls putch)
- secure: (calls getnum)(called by fildes on the P-E)
- strlt
<table>
<thead>
<tr>
<th>name</th>
<th>file (location)</th>
<th>name</th>
<th>file (location)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>dabaO8</td>
<td>MERGERL</td>
<td>MERGERL</td>
</tr>
<tr>
<td>blockdata</td>
<td>dabaO8</td>
<td>MODAP</td>
<td>F7IO (unused)</td>
</tr>
<tr>
<td>addtup</td>
<td>tplib2</td>
<td>MODCOL</td>
<td>MODCOL</td>
</tr>
<tr>
<td>adfcre</td>
<td>tpfilercr</td>
<td>NAME</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>adfdel</td>
<td>tpfilercr</td>
<td>OPNADF</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>ascchr</td>
<td>sort2</td>
<td>OPNDF</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>ascrypt</td>
<td>sort2</td>
<td>OPNTF</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>bkuprl</td>
<td>backuprl</td>
<td>PADFRC</td>
<td>tplib2</td>
</tr>
<tr>
<td>catstr</td>
<td>sc</td>
<td>PUTFPO</td>
<td>tplib2</td>
</tr>
<tr>
<td>catsub</td>
<td>sc</td>
<td>PUTFRC</td>
<td>tplib2</td>
</tr>
<tr>
<td>colnam</td>
<td>tplib2</td>
<td>PRINTREL</td>
<td>printrel</td>
</tr>
<tr>
<td>cpyrel</td>
<td>tplib3</td>
<td>PTFRC</td>
<td>tplib2</td>
</tr>
<tr>
<td>cpystr</td>
<td>sc</td>
<td>PUTIFO</td>
<td>tplib2</td>
</tr>
<tr>
<td>cpysub</td>
<td>sc</td>
<td>PUTTFRC</td>
<td>tplib2</td>
</tr>
<tr>
<td>creaafil</td>
<td>F7O</td>
<td>Radfrc</td>
<td>tplib2</td>
</tr>
<tr>
<td>csortf</td>
<td>sort2</td>
<td>RDADF</td>
<td>tplib2</td>
</tr>
<tr>
<td>dbstat</td>
<td>dabaexit</td>
<td>RDOTF</td>
<td>tplib3</td>
</tr>
<tr>
<td>delrel</td>
<td>deleterel</td>
<td>REAFP</td>
<td>tplib1</td>
</tr>
<tr>
<td>deltup</td>
<td>tplib2</td>
<td>REANUM</td>
<td>tplib1</td>
</tr>
<tr>
<td>display</td>
<td>display</td>
<td>RELFC</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>edtrel</td>
<td>editerel7</td>
<td>RELFD</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>eleb</td>
<td>tplib1</td>
<td>RETEVD</td>
<td>RETREIV7</td>
</tr>
<tr>
<td>emiary</td>
<td>tpfilercr</td>
<td>RLFUNC</td>
<td>function</td>
</tr>
<tr>
<td>enfpta</td>
<td>tplib1</td>
<td>RORGR1</td>
<td>reorgrel</td>
</tr>
<tr>
<td>eninta</td>
<td>tplib1</td>
<td>RTFRC</td>
<td>tplib2</td>
</tr>
<tr>
<td>equivi</td>
<td>tpfilercr</td>
<td>RTVTDF</td>
<td>tplib2</td>
</tr>
<tr>
<td>fildes</td>
<td>tpfilercr</td>
<td>F7IO</td>
<td>F7IO</td>
</tr>
<tr>
<td>filecls</td>
<td>F7IO</td>
<td>SECURE</td>
<td>tpfilercr (unused)</td>
</tr>
<tr>
<td>filedel</td>
<td>F7IO</td>
<td>SOFTFA</td>
<td>SOFT2</td>
</tr>
<tr>
<td>fileopn</td>
<td>F7IO</td>
<td>SOFTFL</td>
<td>SOFT2</td>
</tr>
<tr>
<td>fileren</td>
<td>F7IO</td>
<td>STRLEN</td>
<td>tplib1</td>
</tr>
<tr>
<td>filerr</td>
<td>tpfilercr</td>
<td>TDFCRE</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>fslide</td>
<td>tplib3</td>
<td>TDFDEL</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>ftip</td>
<td>tplib1</td>
<td>TFCRE</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>getfp</td>
<td>tplib1</td>
<td>TFDDEL</td>
<td>tpfilercr</td>
</tr>
<tr>
<td>getnum</td>
<td>tplib1</td>
<td>UTFAR</td>
<td>tplib2</td>
</tr>
<tr>
<td>hashit</td>
<td>tplib2</td>
<td>YESNO</td>
<td>tplib1</td>
</tr>
<tr>
<td>help</td>
<td>help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Library SC is part of Flex (alternate spelling Flecs); we cannot alter the code for these routines.
<table>
<thead>
<tr>
<th>file</th>
<th>contents (subprograms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>backuprl(P-E)</td>
<td>bkuprl</td>
</tr>
<tr>
<td>bkupstub(VAX)</td>
<td>bkuprl</td>
</tr>
<tr>
<td>daba08</td>
<td>MAIN block data</td>
</tr>
<tr>
<td>dabaexit</td>
<td>dbstat</td>
</tr>
<tr>
<td>deletrel</td>
<td>delrel</td>
</tr>
<tr>
<td>dsplay</td>
<td>dsplay</td>
</tr>
<tr>
<td>editrel7</td>
<td>edtre1</td>
</tr>
<tr>
<td>f7io</td>
<td>creafil filecls modap vaxerr(vax only)</td>
</tr>
<tr>
<td></td>
<td>filedel</td>
</tr>
<tr>
<td></td>
<td>fileopn</td>
</tr>
<tr>
<td></td>
<td>fileren</td>
</tr>
<tr>
<td>function</td>
<td>rifunc</td>
</tr>
<tr>
<td>help</td>
<td>help</td>
</tr>
<tr>
<td>mergerel</td>
<td>mergrel</td>
</tr>
<tr>
<td>modcol</td>
<td>modcol</td>
</tr>
<tr>
<td></td>
<td>newrela+ newrela+</td>
</tr>
<tr>
<td>printrel</td>
<td>printrel</td>
</tr>
<tr>
<td>reteiv7</td>
<td>retevd</td>
</tr>
<tr>
<td>reorgrel</td>
<td>rorgrl</td>
</tr>
<tr>
<td>sc</td>
<td>catstr catsub</td>
</tr>
<tr>
<td></td>
<td>cpysstr cpysub</td>
</tr>
<tr>
<td>sort2</td>
<td>csortf sortfl ascchr asctyp sortfa</td>
</tr>
<tr>
<td>file</td>
<td>contents (subprograms)</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>-----------------------</td>
</tr>
<tr>
<td>tpfilecr</td>
<td>adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equival name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf opntdf secure</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel filerr</td>
</tr>
<tr>
<td>tplib1</td>
<td>eleb ftipi getnum strlen yesno</td>
</tr>
<tr>
<td></td>
<td>enfpta pitfp getfp reanum reafp</td>
</tr>
<tr>
<td>tplib2</td>
<td>addtup padfrc radfrc colnam</td>
</tr>
<tr>
<td></td>
<td>deltup putifo rtfrc hashit</td>
</tr>
<tr>
<td></td>
<td>puttdf rtvtdf utfar</td>
</tr>
<tr>
<td>tplib3</td>
<td>cpyrel fslide rdoadf rdotf</td>
</tr>
<tr>
<td>file</td>
<td>subprogram</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>daba08</td>
<td>MAIN</td>
</tr>
<tr>
<td></td>
<td>block data</td>
</tr>
<tr>
<td>dabaexit</td>
<td>dbstat</td>
</tr>
<tr>
<td>deletrel</td>
<td>delrel</td>
</tr>
<tr>
<td>display</td>
<td>display</td>
</tr>
<tr>
<td>editrel7</td>
<td>edtrel</td>
</tr>
<tr>
<td>f7io</td>
<td>creafil</td>
</tr>
<tr>
<td></td>
<td>filecls</td>
</tr>
<tr>
<td></td>
<td>filedel</td>
</tr>
<tr>
<td></td>
<td>fileopn</td>
</tr>
<tr>
<td></td>
<td>fileren</td>
</tr>
<tr>
<td></td>
<td>modap</td>
</tr>
<tr>
<td></td>
<td>vaxerr</td>
</tr>
<tr>
<td>total</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>rlfunc</td>
</tr>
<tr>
<td>modcolum</td>
<td>modcol</td>
</tr>
<tr>
<td>newrelat</td>
<td>newrel</td>
</tr>
<tr>
<td>printrel</td>
<td>printrel</td>
</tr>
<tr>
<td>reteiv7</td>
<td>retevd</td>
</tr>
<tr>
<td>sort2</td>
<td>asctyp</td>
</tr>
<tr>
<td></td>
<td>aschr</td>
</tr>
<tr>
<td></td>
<td>csorf</td>
</tr>
<tr>
<td></td>
<td>sortfa</td>
</tr>
<tr>
<td></td>
<td>sortfl</td>
</tr>
<tr>
<td>tpf1ecr</td>
<td>adfcre</td>
</tr>
<tr>
<td></td>
<td>adfdel</td>
</tr>
<tr>
<td></td>
<td>emlary</td>
</tr>
<tr>
<td></td>
<td>equivi</td>
</tr>
<tr>
<td></td>
<td>fildes</td>
</tr>
<tr>
<td></td>
<td>filerr</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>opnadm</td>
</tr>
<tr>
<td></td>
<td>opntf</td>
</tr>
<tr>
<td></td>
<td>opntdf</td>
</tr>
<tr>
<td></td>
<td>relfc</td>
</tr>
<tr>
<td></td>
<td>relfd</td>
</tr>
</tbody>
</table>
## TABLE 4. CURRENT SIZES AND ESTIMATED REWRITE SIZE (cont.)

<table>
<thead>
<tr>
<th>file</th>
<th>subprogram</th>
<th>lines now</th>
<th>estimated lines in F77</th>
<th>%change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>secure</td>
<td></td>
<td>34</td>
<td>30</td>
<td>34-</td>
</tr>
<tr>
<td>tdfcre</td>
<td></td>
<td>32</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>tdfdel</td>
<td></td>
<td>40</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>tfcre</td>
<td></td>
<td>32</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>tfdel</td>
<td></td>
<td>40</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>802</td>
<td>389</td>
<td>268</td>
</tr>
<tr>
<td>tpplib1</td>
<td>eleb</td>
<td>46</td>
<td>28</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>enfpta</td>
<td>53</td>
<td>32</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>eninta</td>
<td>26</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>ftipi</td>
<td>15</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>getfp</td>
<td>35</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>getnum</td>
<td>29</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>pitfp</td>
<td>13</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>reafp</td>
<td>225</td>
<td>144</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>reanum</td>
<td>156</td>
<td>89</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>strlen</td>
<td>78</td>
<td>37</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>yesno</td>
<td>59</td>
<td>29</td>
<td>24</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>774</td>
<td>404</td>
<td>170</td>
</tr>
<tr>
<td>tpplib2</td>
<td>addtrup</td>
<td>59</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>colnam</td>
<td>24</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>deltup</td>
<td>47</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>hashit</td>
<td>61</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>padfrc</td>
<td>103</td>
<td>67</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>ptfrc</td>
<td>35</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>putifo</td>
<td>83</td>
<td>43</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>puttdf</td>
<td>40</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>radfrc</td>
<td>56</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>rtfrc</td>
<td>46</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>rtvtdf</td>
<td>38</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>utfar</td>
<td>47</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>668</td>
<td>311</td>
<td>279</td>
</tr>
<tr>
<td>tpplib3</td>
<td>cpyrel</td>
<td>108</td>
<td>68</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>fslide</td>
<td>26</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>rdoadf</td>
<td>135</td>
<td>87</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>rdotf</td>
<td>139</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>426</td>
<td>246</td>
<td>219</td>
</tr>
</tbody>
</table>

OVERALL TOTALS

- total lines including comments: 10,845 (38% comments)
- total lines code: 6,800
- estimated lines if FORTRAN 77: 5,758
- estimated code reduction: 1,042
- estimated code reduction percentage: 15%

23/(24 blank)
REFERENCES


APPENDIX A

MAIN
MAIN p.1
Files: PARADOX.FLC
calls the other functions.

```
start

DESCRIPTION
constants, parameters, defaults

set array sizes, init menu loop

IQUEIT

no

stop

print menu and prompt

SELECT

read classical IANS

IANS

1-q. set quitting IANS, failure

NEWREL

create new relation

2-CASE

2-DREL

DELETE delete a relation

3-D

2

A-1
```
Subroutine: RETRIEVE

This subroutine allows the user to output or manipulate relations. In outputting a relation the user may:
1. Print data in a relation
2. Retrieve data
3. Create a relation from data
4. Plot data to screen, plotter, or printer

In manipulating a relation the user may:
1. Sum columns
2. Average columns
3. Calculate the mean and standard deviation
START

ISORT = false
ACTIVE = false
AND = false
OR = false
COMP = false

LUTF = 11
LUADF = 12
LUSAV = 13
LUTEMP = 14
LUACT = 10

"Retrieve, manipulate
and plot relation.
Name of relation to be
manipulated"

NAME
HASHIT
FILES
Get name and process

EMARY, EQUIV,
CATSTR, FSLIDE,
Build names for
ACT, SPI, SP

KIVISF
Load TDF data

Were there any errors?
yes

RETURN
no

Is the relation empty?
yes
"Relation empty:
No manipulation
possible."

no

I = 0

RETURN

(page 2)
RETEVD

1 (page 1)

Increment I

Set Up Tempfile Name

COPYREL
Copy TF and ADF to tempfiles

Are there any errors?

Yes

Set Up Tempfile Name

ADFDREL
Delete temp ADF

Set Up Tempfile Name

TFDREL
Delete temp TF

Attempting recovery...

Set Up Tempfile Name

Are there no errors or have there been too tries?

2 (page 1)

B-3
RETEVD

PAGE 3

2 (page 2)

Were there any errors?

yes

"Recovery not possible."

no

RETURN

Initialize Relation Activity File

"Ready for manipulation."

Has the exit flag DONE been set to true?

no

yes

GET, STRLEN, CPUSUB
Get command letter CRCT

RETURN

A (page 4)

B (page 4)
What is CAMcr (commercial letter)?

List Column

```
A Relaton Reatio
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
```
RETEVD
SET UP
TEMPFILE
NAME

PAGE 5

BACK

CHECK IF
REATION
EMPTY

START

Is IDENT(L8) greater than 0?

OK = true

no

Is IDENT(L8) = 0?

OK = false

yes

no

"Relation empty."

BACK

START

BEGIN:
COPYSUB
Copy the first 7 letters of the filename

CATSTR
Append "B."

BACK

B-6
START

"Column on which to sort?"

Get Column Number (NUM)

ISCNUM

yes

Is the sort column number = 0?

no

CSORTIF
Create a new sort file.

yes

Has ISORT been set to true?

no

FILEDEL
Delete any old sort files.

yes

ISORT = false

no

ISORT = true

OK = true

"Ascending or descending sort (A or D)?"

GET STR,WN,
Get the choice (CRACK)

FILEDEL
Delete any old sort files.

OK = true

"Ascending or descending sort (A or D)?"

GET STR,WN,
Get the choice (CRACK)

FILEDEL
Delete any old sort files.

OK = true

"Ascending or descending sort (A or D)?"

GET STR,WN,
Get the choice (CRACK)
What was the action command? (COMACT) (conditional)

- YES = false
- YES = true
- OK = false

Is OK true?
- no
- yes

SORTFL
sort the column

"Sort on column complete."

BACK
RETEVD
I QUIT
PAGE 8

START

FILEDEL
Delete sort file.

FILEDEL
Delete .ACT

DONE = true

BACK

START

DONE = false

"Type 'H' for help"

BACK

IDLE AND
DO NOTHING
START

Are there any columns in this relation?

- No
  
  "No columns in this relation."

- Yes
  
  Have all the columns been examined?

  
  Yes
  
  no

  $K=\text{ITDF}(3,1)/6$

  
  COLUMN
  
  Get the name of column $i$

  
  WRITE COLUMN NAME

  
  BACK
RETVED
COPY RELATION
PAGE 10

START

'Copied
relation
name:'

NAME, HASHIT
Get the relation
name and hash
it.

CATSIGN
Append "," to
the end of the
relation name.

CPYREL
Copy the TF
and the ADF

Were there any errors?

yes

no

TDFCRE
Create TDF

Were there any
errors?

yes

no

TDF copy file
creation
error

PUTTDF
Save the TDF

Did both the TF ADF copy
and the TDF
operation
work?

no

'Nothing
done.'

Relation
reproduced.

BACK
START

OK = true

GET NUM
Get the number

Is the requested column number negative or greater than the maximum column number? yes

OK = false

Is OK = true? no

OK = true?

BACK
START

Has an activity file been created?

yes

no

J = 0

Increment J (counts # tries)

CREAFIL
Create a .ACT file

 Were there any errors?

no

yes

Is the error 314?

no

yes

FILEDEL
Delete the old activity file.

 Were there any errors in file deletion?

no

yes

"Error in recovery."

Have there been 2 tries or no errors?

no

Have been 2 tries or no errors?
RETEN
AND OR OR
TO ACTIVE
FILE
PAGE 13

Were there any
errors?

yes

no

ACTIVE = true

ACTIVE = false

Is ACTIVE = true?

no

yes

Initialize
Relation
Activity File

Is ACTIVE = true?

no

yes

Get Selection
Criteria

For TEXC Criteria
Into Proper
Type

Is AND = true?

yes

And Acceptable
Records To
Active File

no

Is OR = true?

yes

Or Acceptable
Records To
Active File

no

"Selection
process complete"

Active records"

"Nothing
Done."

BACK

B-14
RETEVD
CHECK
ACTIVITY
FILE
OPENING
PAGE 14

START

ACTERR = false

Were there any errors on opening? (ACTERR)

no

yes

"Error in opening activity file"

ACTERR = true

BACK
RETEVD
INITIALIZE RELATION
ACTIVITY FILE
PAGE 56

A

B

C

D

E

I

OK = false

FILEDEL
Delete the old activity file .ACT

Were there any errors deleting the file?

yes

no

Error deleting activity file:

ACTIVE = false
SKIP = true
OK = true

Has OK been set to true or have there been no errors?

no

yes

Has a new ACT been created?

yes

ACTIVE = true

no

Is SKIP true?

yes

no

FILEDMP
Open the .ACT file.

A

(page 15)

(page 15)

(page 15)

(page 15)

(page 15)

(page 17)

(page 17)

(page 17)

R-17
RE TEVD
INITIALIZE
RELATION
ACTIVITY
FILE
PAGE 17

Check Activity File Opening

Was the .ACT opened successfully?

no

"Nothing Done."

yes

NUMACT = 0

Have all records been examined?

yes

no

Increment Tuple count

Activate Record

FILECLS close .ACT

BACK
AND ACCEPTABLE RECORDS TO ACTIVITY FILE

START

Is the strategy real or integer or is the character data equal?

NR = 0
J = IDENT(13)+1
IDI = IDNF(4,40)

FILEOPEN
Open the .ACT

Check Activity File Opening

Were there any errors in file open? "Nothing done."

Have all the records been examined?

Read a record into ITUF, EDF

Is this the end of file?

RTFRC Get the tuple

Check Found Or Not Found

FILECLS Close the .ACT

A (page 19) B (page 19) I (page 19) (page 19) C
RETEVD
AND ACCEPTABLE RECORDS TO ACTIVITY FILE

Is FOUND = true?

- INREC -AR
- Deactivate Record

Sort On Column IOC

Find String Or Single Character GT LT Criteria

Is FOUND = true?

- FILEDYN Open .ADC
- Check Activity File Opening

Were there any file open errors?

- "Nothing Done"
- FILEDYN Open .SF2

B-20
REIEV AND ACCEPTABLE RECORDS TO ACTIVITY FILE

Have all the records been examined?

yes

Read sort record into IREC

Decrement IREC

Read .ACT record into ITUP, EOF

Is this the end of file?

no

Deselect Record

yes

FILECLS Close .ACT and .SF2

FILEDEL Delete the sort file .SF2

BACK
SORT ON COLUMN ICC

START

CSORTF
Set up second sort file .SF2

Was an ascending sort specified? (GT)

yes  
SRIDIR = false

no  
SRIDIR = true

SORTFL
Sort the column

BACK
START

Is the strategy real or integer or is the character data equal?

Yes

No

NR = 0
J = IDENT(13)-1
IDK = ITBF(A, ECF)
ID2 = ID1 + 1

FILEOPEN
Open the .ACT

Check Activity File Opening

Were there any errors in open? Yes

"Nothing done."

FILECLS Close .ACT

no

Have all the records been examined?

yes

Read a record into ITUP, EDF

Is this the end of file?

yes

no

RTFR Get the tuple

Check Found Or Not Found

No

Yes

(page 22)

A

B

C

(page 23)

I

D

(page 23)

B-23
RETRIEVE OR ACCEPTABLE RECORDS TO ACTIVITY FILE

PAGE 23

Is FOUND = true?

IRCD = NR

Activate Record

Sort on Column LOC

Find String or Single Character GT LT Criteria

Is FOUND = true?

FILEOPEN
Open the .ACT

Check Activity File Opening

Were there any errors in opening the file?

B-24
REIEVU
OR ACCEPTABLE
RECORDS TO
ACTIVITY
FILE

FILEOPEN
Open the sort
file .SF2

Have all the records been
processed?

yes

no

Read IREC
in from the
sort file(SF2)

Increment to
the next record
in IREC

Read IDUP, IDIF
in from the
.ACT file

Is this the end of file?

yes

no

Activate
Records

FILECLS
Close the sort
file .SF2

(page 23)

(page 24)

(page 25)
RETEVD
OR ACCEPTABLE
RECORDS TO
ACTIVITY FILE

PAGE 25

3 (p.24)

Is FOUND = true?

no

yes

FILET3
Close the sort file .SF2

FILEDEL
Delete the sort file .SF2

"Dring on column complete."

BACK
RETEVD

FIND STRING OR SINGLE CHARACTER GT LT CRITERIA

START
FILEOPEN
Open the sort file .SF2
Start Search

Is there single character strategy?

yes
no

CECACT(1)=1
CECACT(2)=
FROMAT(2)

GETCH
Get the character and put it into CH

CHTYP
ITHT = type of CH

OK = true

no

OK = false

ILK = 0

FOUND = false

Is FOUND = true, or are there no remaining records or, is OK true and ILK = 0?

yes

no

Read a record from .SF2 into ITUP, IDUMMY, ILK

A C D
(page 27) (page 27) (page 27)

B-27
RETEVD
FIND STRING
OR SINGLE
CHARACTER
GT LT
CRITERIA
PAGE 27

Get the
character and
put into CH

(conditional)

Increment
ILK

OK = false

Matched
Type

CHTYP(CH) = ITCH
OK = true

Miss Match

CHTYP = ITCH
OK = true

Decrement
ILK

ILK = ITEST
ILKUP less than or equal to ILXIN

Read a record
into ITUP, I1TUP, ICK

Is ICK = ITEST
or ILK = 0?

no

yes

FOUND = true

Is ILK less than or equal to
zero and equal to ITEST?

no

yes

FOUND = false

(p.26)

(p.26)
FIND STRING OR SINGLE CHARACTER GT LT CRITERIA

PAGE 28

A

Is there string strategy?

no

yes

IREC = 0

Get String From Sort File

Is YES = true ?

no

yes

Is LADFRC(1) greater than or equal to FORMAT(1) ?

no

yes

Is LADFRC(1) = FORMAT(1) ?

no

yes

STREQ Compare LADFRC with FORMAT

Are they the same?

no

yes

FOUND = true ILK = IREC-1

Is IREC = 0 ?

no

yes

FOUND = false ILK = 0

BP 29)

(page 29)

2

(page 29)

(p.29)

B-29
Increment INREC

Is FOUND = true or is INREC = IDENT(5)

FILECLS
Close the sort file .SF2

BACK
RETEVD

MATCHED
TYPE

START

Is ICK less than ITEST?

no

yes

Move Up

Is ICK greater than ITEST?

no

yes

Move Down

BACK

START

MIS MATCH

Is ICK less than ITEST?

no

yes

Move Down

Is ISK greater than ITEST?

no

yes

Move Up

BACK
START

ILK = IDENT(8) / 2
ILKUP = IDENT(8)
ILKON = 0
BACK

RETEVD
START SEARCH
PAGE 31

START
ILKON = ILK
I = ILKUP - ILK
ILK = ILK + 1/2

Is I an odd number?

no

yes
ILK = ILK + 1

Is ILK greater than IDENT(8)?

NO

yes
ILK = IDENT(8)

BACK

B-32
START

ILKUP = ILK
I = ILK-ILKON
ILK= ILK - 1/2

Is I an odd number?

no

yes

ILK = ILK -1

Is ILK greater than IDENT(8)?

no

yes

ILK = IDENT(8)

BACK

---

START

Read record from sort file into ITUP, IADFNC

KR= IADFNC(1)/8

Read the record into ITUP, IADFNC

BACK

GEI STRING FROM SORT FILE
RETEVD
CHECK FOR
STRING
MATCH

START
Is IADFRC(1) = FORMAT(1)
  yes
  STREQ
  Compare IADFRC and FORMAT
  Are they equal?
  no
  yes
  Found = true

no
Is IADFRC(1) greater than FORMAT(1)?
  no
  K = 0
  Increment K
  COPY
  Copy first FORMAT(1) characters from IADFRC to CKACT
  STREQ
  Compare FORMAT and CKACT
  Is FORMAT = CKACT?
  no
  yes
  Found = true

Is FOUND true or K-FORMAT(1) greater than IADFRC?
  yes
  BACK
  no

B-34
Is there string or character strategy?

if yes:
  Copy K words of IADPRC into FORMAT

Is the strategy real and EQ not true?

if yes:
  ITEST=ITPRC(IREC)

Convert ITPRC(IREC) into XTEST

BACK

---

START

NUMACT=NUMACT+1

BDF = true

Write ITPF,BDF to the record IREC = 1

BACK

---

REIEVD

PUT TEST CRITERIA INTO PROPER TYPE

PAGE 34
RETEVU
DEACTIVATE
RECORD

PAGE 35

START

NUMACT=NUMACT-1
EOF = False

Write ITUP_EOF
 to record INREC+1

BACK
START

GET SELECTION CRITERIA

Is NUM = 0

no

yes

"Column number out of range."

OK = false

Is OK = true?

no

yes

ICC = NUM

Is there string strategy?

no

yes

"Is substring match desired?"

YES NO
Get the yes or no answer (YES)

1 (page 37)
Is OK equal to true?

What is in the ACTION array?

Go out or LT OK is set to true if any of the conditions are true.

EQ = true

LT = true

GT = true

Selection criteria
START
IREC=ITDF(4, ICC)
Location in the TF record

Is the strategy real?

no

yes

GETFP
Get real value X, LEN characters long

Is the length = 0?

no

yes

FTIP
Convert X into ITFR(Irec)

ITFR(IREC)=
NULL(2)

Is there integer strategy?

no

yes

GETNUM
Get integer value ITFR()
GET
VALUE
OR
STRING

1

Is there character strategy?

no

yes

Is the LEN greater than 1?

no

K = 2

yes

"Single character only."

OK = false

Is there string strategy?

no

yes

LEN = IADFRC(i)

K = LEN/6

"Tuple editing error! See programmer."

OK = false

Is OK = true?

no

BACK

yes
Is there string strategy ITFRC?  

Is FORMAT(1) less than 8?  

Yes:  

No:  

UTPAR convert ITFRC into IADFRC  

Is YES = true?  

Yes:  

No:  

STREQ compare IADFRC to FORMAT  

Is IADFRC = FORMAT?  

Yes:  

No:  

Is there string strategy and  

ITFRC(ID2) greater than or equal to  

FORMAT(1)?  

Yes:  

No:  

READ text into IADFRC  

STREQ compare IADFRC to FORMAT  

Is IADFRC = FORMAT?  

Yes:  

No:  

Is there integer strategy and  

ITFRC(ID2) greater than ITEST?  

Yes:  

No:  

FOUND = false

START

CHECK FOUND OR NOT FOUND

PAGE 41
START

**RETLVD**
MOVE ACTIVE RECORDS TO NEW RELATION

NAME, HASHIT
Get the relation name.

Is ACTIVE = true and NAMACT greater than 0?

**If no**

**If yes**

NAME-INDEX

**If no**

**If yes**

FILDES
Get volume and protection keys

NAME-INDEX

**To**

NAME-INDEX

NAME-INDEX

**To**

NAME-INDEX

ENARY
empty the IDENT array

Fill IDENT with zeroes

IDENT(3) = IDENT(3)
IDENT(6) = IDENT(6)
IDENT(9) = IDENT(9)

n = 12-45
IDENT = IDENT

A (page 44)
RETEVU
MOVE ACTIVE RECORDS TO NEW RELATION

C

(page 44)

2

(p.44)

MOVE
ACTIVE
RECORDS
TO
NEW
RELATION

3

(p.46)

COLUM
Get the column name

INDEX = INDEX+
INDEX = INDEX+
OVR = true

PADFRC
Save the ADF

ITDP(2, ICC) = IDENTI(1)
IDENTI(1) = IDENTI(1) - 1
IDENTI(2) = IDENTI(2) - 1ADFR(1) = 4

FILEDPN
Open the ACT

Is ISORT = true?

FILEDPN
Open the scratch file SFI

FILEDPN
Open the scratch file SFI

READV0 = 0
NR = 0
IXP = IDENT(3) - 1

Read ACT record into ITUP EDF

Is SORT true 2?

Read SFI record into IREC

IREC = IREC - 1

Read ACT record into ITUP EDF

(p.46)

A

(p.46)

B-46
Is EDF true?

IFMI

IWKEY

ICC-I

Is N-IDENTI(15) or ICC=IDENTI(9)?

False

ICG=ICG+1

Is there a string strategy V?

yes

no

ITFRIC(k) or ITFRIC(k+1) = null?

Is ITFRIC(k) greater than 0?

yes

no

ITFRIC(k) = 0

Is N=IDENTI(15) or NMOVED=FALSE?

no

yes

NR = NR + 1

ADDITUP
Save the tuple to the new TF

ITFRIC(k-1) = 0

Is N=IDENTI(15) or NMOVED=FALSE?

no

yes

RADREC
Get the ADF record

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE

ITFRIC(k-1) = 0

INKEY = IRSAVE
MOVE ACTIVE RECORDS TO NEW RELATION

1. FILECLS Close the sort file SFI
2. FILECLS Close the ACT
3. Is ISORT true?
   - Yes
   - No
4. ISKEY=INMOVE ISKEY=INMOVE
5. PUTFDF Save the new TDF
6. ISKEY=ISAVE ISKEY=ISAVE
7. Is YES true?
   - Yes
   - No
8. KL=LONGNUM(1)/6
9. "Records transferred. Begin deleted at high end."
10. NUMACT records moved to LONGNUM.
11. FILEDFM Open ACT
12. Have all the records been examined?
   - Yes
   - No
13. Checking record number__
14. Read ACT record into ITUP EDF

B-48
MOVE ACTIVE RECORDS TO NEW RELATION

Is EOF true? (p.47)
- yes
  - DELETE
  - Delete corresponding tuple of original
- no
  - PUTDF
  - Save the IDF
  - CPYREL
  - Copy relation to TEMPFL
  - FILECLS
  - close ACT
  - FILEDEL
  - Delete ACT and SFL
  - ISORT = false
  - ACTIVE = false
  - NUMACT deleted
  - K1 = LONCHAR(1) / 0
  - "NUMACT records moved to LONCHAR"

BACK

B-49/(B-50 blank)
GET-COLUMN-NUMBER

GET-COLUMN-NUMBER

IS SELECTION VALID?

"SELECTION OUTSIDE RANGE!!"

ARE THERE NO ERRORS?

FIN
TEST FOR NUMERIC STRATEGY

IS THE COLUMN NUMERIC STRATEGY UDEFINED?

TRUE

GET YES-OR-NO RESPONSE

NO

IS RESPONSE TRUE?

TRUE

LEAVE SUBROUTINE

FALSE

SUBROUTINE

FIN

THE COLUMN THAT YOU HAVE SELECTED DOES NOT HAVE A NUMERIC STRATEGY. DO YOU WISH TO MAKE ANOTHER SELECTION <YES OR NO>?
GET-COLUMN-NAME

GET-COLUMN-NAME

FIN
GET YES OR NO

READ RESPONSE

IS RESPONSE VALID

TRUE

FALSE

HAVEN'T GOT A VALID RESPONSE

TRUE

FALSE

'ANSWER YES OR NO'

FIN
GET AXIS LIMITS

IS A COLUMN PRESENT IN MINIMUM

TRUE

SET MINIMUM AND MAXIMUM AXIS LIMITS

SEE IF READ WAS INPUT CORRECT WITHOUT ERRORS

IS TRUE

TRUE

IS EVERYTHING OK?

TRUE

FALSE

FALSE

FALSE

FALSE

FIN

ENTER THE MINIMUM VALUE FOR THE AXIS

READ MINIMUM VALUE

SET IF READ WAS READ CORRECT WITHOUT ERRORS

IS OK

TRUE

IS EVERYTHING OK?

IS EVERYTHING OK?

IS EVERYTHING OK?

IS EVERYTHING OK?

IS EVERYTHING OK?
SEE IF READ IS OK

IS THERE A STATUS ERROR?

TRUE

WHAT?

FALSE

FIN
BUILD-CPAGE

DETERMINE BOUNDARIES, AXES AND SCALE

OPEN PLOT FILE

READ POINTS IN RELATION

APPLY SCALE FACTORS AND DETERMINE POINTS IN BOUNDS

CLOSE PLOT FILE

FIN
THE LEGAL COMMANDS ARE:

- Q (QUIT) TERMINATE PLOTTING PROGRAM
- C (CLEAR) RESET PLOTTING PROGRAM
- S (STATUS) PRINT STATUS REPORT
- X (Y-RANGE) SET X-AXIS RANGE
- Y (Y-RANGE) SET Y-AXIS RANGE
- P (PLOT) DISPLAY THE PLOT
- L (LINES) DRAW THE DATA POINTS WITH LINES
- T (TEXT) CREATE X AND Y AXIS LABELS
LEAVE - SUBLROUTINE

OPEN PLOT FILE

CLOSE PLOT FILE WITH STATUS SET TO DELETE

RETURN

FIN
APPENDIX D

RLFUNC
Subroutine : FUNCTION (Rlfunc)

This subroutine provides the user with data analysis functions to allow analysis of sum of column, averaging, mean and standard deviation, etc. This subroutine is constructed that more functions may be easily added as the need arises. The functions available are:

1. Total Column Data
   This function algebraically adds the active records of a relation for the column specified. The strategy must be integer or floating point.

2. Mean and Variance
   This function computes the mean and variance of the active records in the specified column. Output is the mean and standard deviation.
defn units
LUTF=11
LUACT=12

set up activity file name

EMIARY
CPYSUR
CATSTR
FSLIDE

set up TEMPFILE.TF
name

NOFUNC+2
(number of functions)

offer menu

RLFUNC+1
get function selection number

IAMS

0

return

1

"RLFUNCPS" total column data

RLFUNCPS write column total

return

2

RLFUNCPS total column data

RLFUNCPS write mean and variance

return

otherwise "Response not understood. Returning to menu" return

D-2
get function selection number

RLFNC1

repeat

OK = true

(prompt)RMP?

GETNUM

get answer
IANS, error
flag IERR

IANS<0

T

F

OR

IANS>0

NOFUNC

"selection outside range"

OK = false

IERR=0

and

OK=true

F

T

BACK

D-3
2 get column number

repeat

OK = true

GETNUM

get column
ICC, error
flag IERR

ICC<0

OR

ICC>number of columns

"selection outside range"

OK = false

IERR=0

and

OK=true

T

BACK
write column mean and variance

mean and variance of column
for: ______
Records = ______

Is the absolute value of xmean

> 10^9 or < 10^-9
write the mean using scientific notation
format E16.9

≥ 1
write the mean using floating point format
F12.2

otherwise
write the mean using floating point format
F10.7

<

Is the absolute value of standard deviation (STDOY)

> 10^9 or < 10^-9
write the standard deviation using scientific notation
format E16.9

≥ 1
write the standard deviation using floating point format
F12.2

otherwise
write the standard deviation using floating point format
F10.7

back
write column total

is there integer column strategy

is the absolute value of total

>10^9 or <10^-9

"Column Totals" written in scientific notation in F16.9 format

>1

"Column Totals" written in floating point F12.2 format

otherwise

"Column Totals" written in floating point F10.7 format

"with active records"

back
(repeat)

active T → read OK from ACT

F → read value

read value

OK T → read value

F → increment ICT

increment ICT

OK or ICT = ISTOP+1 F

empty F → IANS = 1

F → IANS = 2

by strategy, add to ITOTAL or ITOTAL

x = value

inc ITOTAL by x

inc ITOTAL by 1

JMEAN = ITOTAL/JTOTAL

SS = SS + x²

ITOTAL>1 T

VAR = 0

VAR = (SS-JMEAN²JMEAN/JTOTAL) / JTOTAL

STDOEV = SQRT(VAR)
E-3
Subroutine: SORT

This subroutine allows the user to organize relational data in ascending or descending order. If the relation to be sorted is small (1000 rows or less) then a "fast" sort is done in core. For larger relations, a general file to file weaving bubble sort is implemented.
SORT
ASCTYP

START

Determine the ASCII number of the character

Set ASCTYP = 5

Is the character blank?

Yes ASCTYP = 1

No

Is the character a capital letter?

Yes ASCTYP = 2

No

Is the character a lower case letter?

Yes ASCTYP = 3

No

Is the character a digit?

Yes ASCTYP = 4

No

RETURN
START
Pass in two characters.

ASCCHR
Determine the character type.

Are the character types in ascending order?

no

ASCCHR = true

RETURN

Are the character types in descending order?

no

ASCCHR = false

RETURN

ASCCHR = true (characters are of the same type.)

Are the characters in ascending order?

yes

RETURN

no

ASCCHR = false

RETURN
START

Is the strategy = 4? (character string)

yes

'Should not be in SORTFA-strategy 1.'

RETURN

no

Have all the records been examined?

yes

no

Read in an ADF record

Decode by strategy into DATA or DAIK arrays.

Sort These

Output in Ascending Or Descending Order

FILECLS
Close the sort file (UFSF)

RETURN
Set up to start from the beginning and sort forward to the end of file.

Has the last record been examined?

Is the strategy == 1?
(integers) Ascend = DATI(1) less than or equal to DATI(1:1)

Is the strategy == 2?
(decimal) Ascend = DATI(1) less than or equal to DATI(1:1)

Is the strategy == 3?
(characters) Use ASCOMP function to determine order. Then assign to DATI

Is the data in the right order?

Swapping the data.

Has all the data been successfully sorted?

RETURN
SORT THEM

PAGE 5

Set up to start from the end and sort backwards to the beginning.

Has the last record been examined?

Has all the data been successfully sorted?

Is the strategy = 1?
   Ascend = DAT(I)
   less than or [equal to DAT(I+1)

RETURN

Is the strategy = 2?
   Ascend = DATX(I)
   less than or [equal to DATX(I+1)

Is the strategy = 3?
   Use ASCORDER function to
determine order.
   Then assign to DATI

Is the data in the right order?

yes

no

Samp the data.

yes

no

Has all the data been successfully sorted?

RETURN

yes

no
SORT
OUTPUT IN
ASCENDING OR
DECENDING
ORDER

PAGE 6

START

Have all the ADF records been examined?

yes

BACK

no

ENCODE data according to strategy into INDRAC.

Write the sorted data back to LUSF.

P-7
START

TXTLEN(4)-STRMAX
MAXADOF-STRMAX/6
WLEN(4)+MAXADOF

CATSTR
FSLIDE
Build file names
for .SF,.11,.12

FILEOPEN
Open data file

Is the strategy not 4 and the number
of records less than 100? yes

no

ESTIMATE
and report the time
needed to do the
sort.

IWIDE = 0

Is the strategy equal
or not equal to 4?

yes

IWIDE = 8

no

CREAFIL
Create files
.11 and .12
(IWIDE x NUMREC)

Were there any errors during
file creation?

yes

"Error in creation of file"

no

RETURN

(page 8)

F-8
Is the strategy = 4?

no

open .11

yes

FILEOPEN
open .11

Were there any errors in opening the .11 file?

yes

"Error in file open"

no

RETURN

Is the strategy = 4?

no

open .12

yes

FILEOPEN
open .12 file

Were there any errors in opening the .12 file?

yes

"Error in file open"

no

RETURN

KR = 1 + WLEN(strategy)

Is the strategy = 4?

yes

no

A (page 9)  B (page 9)
Is the strategy = 3?

- yes: Sort Data As Character

- no: Is the strategy = 4?

- yes: Sort Data As String

- no: FILECLS
  Close SF, .11..12

FILEDEL
Delete .11..12

RETURN
SORT
SORT DATA
AS INTEGER

START

Read a record into ROMBUF, and IBUF

Have all records been examined?
yes

Read a record pair into ROM, DATI

Is DATA greater than IBUF?
no

Swap order and write ROM, DATI back to file UART

Write ROMBUF and IBUF back to file UART

Advance through.
ROMBUF=ROW
IBUF=DATI

Write out the last record to UART

Was any data swapped?
no

BACK

yes

Switch Units

Is this pass a multiple of 100?
no

"_ on pass"

yes

F-12
SORT
SORT DATA
AS REAL

START

Read a record into ROMBUF, and XBUF

Have all records been examined?
yes

Read a record pair into ROM, DATX

Swap order and write ROM,DATX back to file LLMT.

Is DATX greater than XBUF?
no

Write ROMBUF and XBUF back to file LLMT

Write out the last record to LLMT

Was any data swapped?
no

BACK

yes

Switch Units

Is this pass a multiple of 100?
yes

on pass

no
SORT
SORT DATA
AS
-CHARACTER

START

Read a record into ROMBUF, and IBUF.

Have all the records been examined?

yes

no

Read a record pair into ROM DAT. Then, using the ASCIND function determine if they are in the correct order.

Are they in the right order? (ASCIND)

yes

Write the data back to UART

no

Swap the order and write the data back to UART

Advance position

Write out the last record to UART

Was any data switched?

no

BACK

yes

Switch Units

Is this pass a multiple of 100?

yes

no

PASS
SORT DATA AS STRING

START

read a record into ROBRUP ADFBUF

Have all records been examined?

yes

no

Read a record pair into ROM IADFRC

Set ASCORD to true.

Is the length of IADFRC = 0?

yes

no

Set ASCORD to false

Is the length of ADFBUF = 0?

yes

no

Have the first two characters in ADFBUF been read in?

yes

no

GETCH Get the second character of ADFBUF

GETCH Get the second character of IADFRC

Using the ASCORD function, compare the second characters of ADFBUF with IADFRC to check for the correct order.

A (page 15) B (page 15) C (page 15) D (page 15)

F-15
Are the characters in the correct order? (ASC/ORD)

Are they the same character?

Write ROWBUF, ADFBUF back to the file UWART

ADFBUF = LADFRC

Are the comparison characters in the correct order?

Write ROW, LADFRC back to UWART

Write the last ROWBUF, ADFBUF back to UWART

Were there any switches made?

Switch Units

Is this pass a multiple of 100?

On pass ""
START

Has SWITCH been set to true?

yes

LURED = .12
LIMRT = .11
SWITCH = true

no

LURED = .11
LIMRT = .12
SWITCH = false

BACK
START

CATTAX
FILEIDE Build .SF file

IFFF='empty' symbol.
STATMAX = 0

Has ISORT been set to true?

yes

no

J=0

Increment J
New number of records

CREATFIL
Create a sort file

Is there an error that is not code 314?

yes

no

"SortFile creation error"

RETURN

Is there an error code 314?

yes

no

FILEDEL
Delete the old sort file

Was there any error?

no

yes

"SortFile creation error"

RETURN

Have there been 2 tries or a successful creation?

yes

no

1 (page 18)

F-18
FILEDPN
open .SF

Were there any errors in file open?
yes
Error opening sort file in CSORTF

no

OPMFT
Open the TF

RETURN

Is the strategy = 4?
yes

OPMADF
Open the ADF

no

KR = IDENT(13)-1

Have all the tuples been examined?

Read a record into ITFRG

FILECLS
close .SF, .TF, .ADF

Is the strategy = 1 ?

no

RETURN

yes

Is the data in the desired column empty?

no

yes

Set the data to zero.

A (page 19)

B (page 19)
ENCODE the data in the particular column into an integer format (II2)

Is the strategy = 2?

no

Is the data in the desired column empty?

yes

Set the data to zero (0.0)

no

Set to floating point format.

ENCODE the data in the particular column into a scientific notation (E16.9)

Is the strategy = 3?

no

Put the value in the particular column into LADFRC(II2)

Is the strategy = 4?

no

Is the string empty?

yes

no

C (page 20)

D (page 20) (p.20)
SORT
OUTPUT IN
ASCENDING
OR DECENDING
ORDER

START

FILEOPEN
Open the .SF

LEN=LENGTH(strategy)

Have all the records been examined?
yes

no

NUMIN= the
number of records
+ 1 - counter

Is STRDIR true?
yes

NUMIN = counter

no

Read record NUMIN into ROM and DATI or IADFRC by strategy

ENCODE by strategy
data into IADFRC

Write ROM, IADFRC

PAGE 21
APPENDIX G

EDTREL
Subroutine: EDITREL

This subroutine allows the user to change, add, delete, or examine tuple data.

The following edit commands are available to the user:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>?(STATUS)</td>
<td>GIVES CURRENT VALUES OF ROW AND COLUMN POINTERS, RELATION NAME, NUMBER OF RECORDS AND NUMBER OF COLUMNS</td>
</tr>
<tr>
<td>Q(QUIT)</td>
<td>TERMINATES EDITING</td>
</tr>
<tr>
<td>C(COLUMNS)</td>
<td>LIST NAMES OF EACH COLUMN BY NUMBER IN THE RELATION</td>
</tr>
<tr>
<td>+[] (PLUS)</td>
<td>MOVES CURRENT ROW POINTER FORWARD [X] ROWS(LAST GIVEN IF TOTAL EXCEEDED)1 IS UNDERSTOOD IF [0] OR [ ] USED.</td>
</tr>
<tr>
<td>-[] (MINUS)</td>
<td>SAME AS + EXCEPT IN OTHER DIRECTION</td>
</tr>
<tr>
<td>F(FIND)</td>
<td>FIND THE FIRST OCCURANCE OF DATA ITEM IN SPECIFIED COLUMN.</td>
</tr>
<tr>
<td>S(SUBSTITUTE)</td>
<td>USER SUBSTITUTES NEW DATA INTO LOCATION POINTED TO BY CURRENT ROW AND CURRENT COLUMN pointers.</td>
</tr>
<tr>
<td>E(EXAMINE)</td>
<td>DISPLAYS TO USER CONTENTS OF CURRENT ROW AND COLUMN POINTER LOCATION.</td>
</tr>
<tr>
<td>D[] (DELETE)</td>
<td>DELETES [X] NUMBER OF TUPLES STARTING WITH CURRENT TUPLE. NONE DELETED IF NONE SPECIFIED.</td>
</tr>
<tr>
<td>I(INSERT)</td>
<td>INSERTS TUPLE AFTER CURRENT TUPLE POINTER. PROMPTS USER FOR COLUMN VALUES.</td>
</tr>
<tr>
<td>B(BOTTOM)</td>
<td>SAME AS INSERT ACCEPT PLACES TUPLE AT BOTTOM.</td>
</tr>
</tbody>
</table>
P([PRINT) PRINTS [X] TUPLES OF RELATION
STARTING WITH OUTPUT DATA IN A
FORMAT THAT WILL ALLOW RE'ADD'ING
BY THAT OR ANOTHER RELATION WITH
EQUAL STRATEGIES WITH CURRENT
TUPLE. 1 ASSUMED IF 0 OR BLANK.

R(RE-START) USER MAY RESTART EDIT PROCESS TO
CHANGE DISPLAY FORMAT.

H(HELP) LIST THE POSSIBLE COMMANDS

A(ADD TUPLES) READS DATA TUPLES INTO RELATION
IN FREE FORMAT, CARD IMAGE
SEPARATED BY USER DELIMITER.
# FEASIL

## EDITREL

Subroutines and Procedures

(Listed in the order in which they appear in the code.)

<table>
<thead>
<tr>
<th>Subroutine or Procedure Name</th>
<th>Flow Chart Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Quit</td>
<td>Page 7</td>
</tr>
<tr>
<td>Display Help</td>
<td>Page 7</td>
</tr>
<tr>
<td>Check If Relation Empty</td>
<td>Page 8</td>
</tr>
<tr>
<td>Present Relation Status</td>
<td>Page 8</td>
</tr>
<tr>
<td>Find Column Display Description</td>
<td>Page 9</td>
</tr>
<tr>
<td>Insert A Tuple</td>
<td>Page 11</td>
</tr>
<tr>
<td>Delete Tuples</td>
<td>Page 12</td>
</tr>
<tr>
<td>Add Tuples At Bottom</td>
<td>Page 13</td>
</tr>
<tr>
<td>Print Data</td>
<td>Page 14</td>
</tr>
<tr>
<td>Print Variable Length Output</td>
<td>Page 15</td>
</tr>
<tr>
<td>Generate Write statement Maybe Column Names</td>
<td>Page 16</td>
</tr>
<tr>
<td>Add Tuple Data To Block</td>
<td>Page 17</td>
</tr>
<tr>
<td>Restart Description</td>
<td>Page 18</td>
</tr>
<tr>
<td>Move Tuple Pointer Forward</td>
<td>Page 19</td>
</tr>
<tr>
<td>Get Next Tuple</td>
<td>Page 20</td>
</tr>
<tr>
<td>Move Tuple Pointer Backwards</td>
<td>Page 20</td>
</tr>
<tr>
<td>Find Value</td>
<td>Page 21</td>
</tr>
<tr>
<td>Substitute New Value</td>
<td>Page 25</td>
</tr>
<tr>
<td>Examine Current Values</td>
<td>Page 26</td>
</tr>
<tr>
<td>Idle and Do Nothing</td>
<td>Page 26</td>
</tr>
<tr>
<td>Present Current Value</td>
<td>Page 27</td>
</tr>
<tr>
<td>Get Column Number</td>
<td>Page 27</td>
</tr>
<tr>
<td>Get Value or String</td>
<td>Page 28</td>
</tr>
<tr>
<td>Insert Tuple Data</td>
<td>Page 32</td>
</tr>
<tr>
<td>Display Selected Tuple Values</td>
<td>Page 34</td>
</tr>
<tr>
<td>Check TF Allocation</td>
<td>Page 34</td>
</tr>
<tr>
<td>Add Data Tuples To Relation</td>
<td>Page 35</td>
</tr>
<tr>
<td>Process All Cards</td>
<td>Page 40</td>
</tr>
<tr>
<td>Input As Non string</td>
<td>Page 43</td>
</tr>
<tr>
<td>Copy String Between Delimiter</td>
<td>Page 46</td>
</tr>
</tbody>
</table>
START

Clear the screen.

NAME

Get the relation name.

EQUIV

HASH

Hash the relation name

FILDES

setup for disk I/O

XTVIDF

Get the IDENT and ITDF arrays for modification.

Is there an I/O error?

yes

RETURN

no

Initialize pointers and bring it into area (XITRPC)

Present Relation Status

1 (page 2)
1. Get Column Description
2. Get Column Names into Code
3. Display Prompt
4. GET Get the command
5. STRLEN Interpret the command as a string
6. CPYSUR Use the first letter of the command to determine the action:
   - Is the answer 'P'?
     - Yes: Present Relation Status
     - No: Is the answer 'Q' (QUIT)?
       - Yes: QUIT
       - No: Is the answer 'C' (list column names)?
         - Yes: List Column Names
         - No: Continue
EDITREL

2 (page 2)

Is the answer "y" (move cursor forward)?

- Yes: Check if Relation Empty
  - Yes: Is the relation empty?
    - Yes: Move Tuple Pointer Forward
  - No: Is the answer "y" (move cursor backwards)
    - Yes: Check if Relation Empty
      - Yes: Is the relation empty?
        - Yes: Move Tuple Pointer Backwards
        - No: Display Selected Tuple Values
      - No: Is the answer "y" (restart)?
        - Yes: Restart Description
        - No: (page 4)

A (page 2)

3 (page 4)

4 (page 4)
I. 

Is the answer 'D' (output)?

- **Yes**
  - Check If Relation Empty
- **No**
  - Is the relation empty?
    - **Yes**
      - Save Like Add Command Expects
    - **No**
      - Is the answer 'F' (find)?
        - **Yes**
          - Check If Relation Empty
        - **No**
          - Is the relation empty?
            - **Yes**
              - Find Value
            - **No**
              - Is the answer 'S' (substitute)?
                - **Yes**
                  - Check If Relation Empty
                - **No**
                  - Is the relation empty?
                    - **Yes**
                      - Substitute New Value
                    - **No**
EDITREL

QUIT

START

PUTIDF put new IDENT and ITDF arrays into .IDF file

Is there an I/O error?

yes

RETURN

no

BACK

---

DISPLAY_HELP

START

'The valid commands are:"
(list of valid commands)

BACK
EDITREL

START
CHECK IF RELATION EMPTY

Is the relation empty?
Is IDENT (LB) greater than zero?

no

Relation empty

yes
Set the empty empty relation flag to true.

BACK

PRESENT RELATION STATUS

START

"" Relation"

"" number of records. Current record"

"" number of columns. Current column"

BACK

G-12
FILL COLUMN DISPLAY DESCRIPTION

START

Display column names (yes or no)?

YES NO Get yes/no answer. Set DISCOL to true if yes.

Has the last column been displayed?

yes

Display how many columns?

no

Get Column Number

Is the response greater than the actual number of columns?

yes

no

There are not that many columns.

Is the response negative?

yes

Value must be zero or positive.

no
Are there more than zero columns to be displayed?

Yes

Default order (yes or no)?

YES/NO

Get the yes or no answer.

Is the answer YES?

No

Display in column number

Get Column Number

Mark the columns to be displayed.

ACK
START

Check TF Allocation

Was the allocation successful?

no

yes

Insert Tuple Data

PUTTDF Update the tuple descriptor.

BACK
EDITHEL
DELETE TUPLES

PAGE 12

(START)

COPYSUB
Strip the letter 'Y' off the action string.

RENAME
Get the number of tuples to be deleted from the action string.

Is there an I/O error?

yes

no

Mark zero tuples to be deleted.

Mark the number of tuples to be deleted.

Are there more tuples to be deleted?

no

yes

DELTUP
Delete a tuple.

Are there any remaining tuples in the relation?

no

yes

Move the pointers to reflect deleted elements.

RTFRC
Retrieve current TF into QREZ.

(page 13)

G-16
EDITHEL

DELET TUPLES

PAGE 13

PUTTFD
Update the
Tuple Descriptor
File (TDF)

Are there tuples remaining
in the relation?

no

ye

Display
Selected Tuple
Values

BACK

---

ADD TUPLE
AT BOTTOM

START

Move the
pointer to the
bottom

Insert A
Tuple

BACK

G-17
START

Given the 'P' off the action

REMEMBER
Get the number
of tuples to be
printed

Check to see
if there are
tuples available
for printing

Are there tuples
remaining?

no

yes

Generate
Write
Statement
Maybe Column
Names

Print
Variable
Length
Output

Print
Next
Tuple

Is the pointer at
the end of the
relation?

no

yes

Move Tuple
Pointer
Backwards

BACK
START

Adjust the string length until it is as long as the variables.

Write out the FORMAT array.

BACK
EDITRAL

START

DATABASE
Initialize and
eempty the
FORMAT array

Are all the column names
and tuple data loaded
into FORMAT
BACK

yes

no

Has DISCOL been set to true
in Find Column Display
Description

yes

no

Has the column name been
loaded into CORE?

yes

no

COLNAME
Get column
names into
CORE.

CATSTR
Move EADFPRC
into FORMAT

CATSTR
Move FMTAR
into FORMAT

Add Tuple Data
To Block

CATSTR
Move ISLUR
into FORMAT

CATSTR
Move REPORT
into FORMAT

G-20
START

Find Column
Display Description

Align to Longest Displayed Column Name

Are there 1 or more elements in the relation?

no

yes

Display Selected Tuple Values

BACK
EDITREL

START

PUTF
Put the TP back

READMM
Get the number of spaces to move

Move the pointer the specified number of spaces forward

GETNXTTUP
Get Next Tuple

DISPSELTUPLE
Display Selected Tuple Values

BACK

MOVE TUPLE POINTER FORWARD

PAGE 19
Will the increment to be moved to the end of the relation?

Yes:
Set the increment to the end of the relation.

No:
Get the tuple at the specified increment.

MOVE TUPLE POINTER BACKWARD:

START
GET NEXT TUPLE

REANUM
Get the number of spaces to move

MOVE THE POINTER THE SPECIFIED NUMBER OF SPACES BACKWARD

GET THE NEW TUPLE FILE VALUES INTO CORE

BACK
START

Has the desired column been found?

no

"Find in which column?"

Get Column Number

Get Value or String

Is the response an empty string?

yes

Set IADFRC to empty set identifiers

no

"Find what string?"

Is the string to be found into a working array (array)?

yes

no

Has the string been found?

no

Has the search reached the end of the tuple file?

yes

"Searching..."

(page 22)

A (pages 22, 23)

B
Is the data non-string? 

Is this the element being searched for? 

Set a flag that the element has been found. 

Is there string data? 

Is the string less than or equal to zero? 

UPFAR 

Is the ACTION string the same length as the data string? 

Is the first element of each string empty? 

Yes

No

Yes

No

Yes

No
Clear the screen.

"Current record
Current column"

Display selected tuple values

START
EDITREL

SUBSTITUTE
NEW VALUE

PAGE 25

START

"Column
Record"

Present
Current
Values

"Enter new
string."

Is there a new string?

no

yes

Check ADF
Allocation

Has the allocation OK?

no

yes

"Change not
made"

Get Value or
String

PUTFD
Substitute the
new data

Display
Selected
Tuple Values

BACK

G-29
EXAMINE CURRENT VALUE.

START

"Examine which column?"

Get Column Number

Present Current Value

BACK

---

IDLE AND DO NOTHING

START

"Command not understood. Type 'N' for help."

BACK
START

Adjust current pointers

IFREC
Get the requested data from tuple file

Output Tuple Data

BACK

---

GET COLUMN NUMBER

START

Has the column number been found with no problems?

yes

no

GETNUM
Get the column number

Is the number larger than the TF?

yes

Input error: Value must be in range.

no

BACK

---

START

PRESENT CURRENT VALUE

PAGE 27
Is there integer strategy?

Yes

"Integer"

GETINT
Get the integer data.

Is the data zero?

No

Make a zero length assignment to EIPR.

Yes

Is there floating point strategy?

No

Is the data zero?

Yes

"Decimal"

GETFP
Get the floating point data.

Is the data zero?

No
Is the strategy a character string or a single character?

Has OK been set to true?

Is the strategy single character?

GET the new character data.

Make zero length assignments to DATA.

Has a valid input been provided?

 Character string. 

Single character.
STREND determine string length

Is the string of zero length?

Yes
Set the value to zero.

No

Adjust length qualifiers and make assignments

CPYSUB Copy the adjusted data in.

CATSTR

Is there a single character strategy?

No

Is the data of length one?

No

Yes

CPYSUB "Single character only"

Yes

4 (page 31)
Is the strategy not character string or single character?

- No
- Yes

"Tuple editing error... See programmer!"
START

Check IF Allocation

Was the allocation successful?

no

yes BACK

yes

no

Was the column name been loaded into core?

no

yes COLNAM Get column name into core

Write out the column name.

Get Value Or String

IS there string strategy?

no

yes

Is the string length zero?

no

yes Assign proper IIIFC element to empty.

PADIFC Return TFAR

(page 33) (page 33) A
Is TFAR true?

- **No**
  - Make assignments to IDENT

- **Yes**
  - Make assignments to ITYRC

- ADDTUP
  - Add the tuples where they belong.

- Display
  - Selected Tuple Values

- BACK
EDITREL
DISPLAY
SELECTED
TUPLE
VALUES  PAGE 34
BACK

START

yes

Have all the marked tuples been displayed?

no

Calculate which tuples are to be displayed and which are remaining.

Mark the remaining tuples (selectively) for display.

Output Tuple Data

CHECK TF ALLOCATION

START

Is there enough space allocated in the tuple file?

yes

no

REDOTF reallocate a new TF

Was the allocation successful?

no

yes

PUTTF Move the data into the new tuple file.

BACK

G-38
START

Has a valid answer been received?

yes

"Enter number of rows to be added."

Display Prompt

GETNUM
Get the number of rows to be added.

Has the answer valid?

yes

no

Has the answer less than zero?

yes

"Must be positive."

no

Has the data been found or has the error limit (3) been reached?

yes

no

"Enter device or file where data is to come from."

Display Prompt

A (page 36)  I (page 36)  B (page 36)

G-39
Interpret the given string. Write the file to an array. Determine if the device or file is now open.

No

Yes

Add data to the array. 1

Get the device or file name.

Process the array. 2

Binary by the device or file name.

Split the string into the array and a string.

Get the device or file name.

Are there any errors? 3

No

Yes

Enter line to edit:

Too many errors.

Line to be edited:

Data to be edited:

Enter the line you want to edit.
EDITREL

ADD DATA ELEMENTS TO RELATION

Was the error limit reached?

yes → PAGE 37 → BACK

no →

Has a valid answer been received?

yes →

Enter single character delimiter between data elements.

Display Prompt

GET
Get the delimiter

STLEN
Use the delimiter as a string.

Determine the validity of the delimiter.

EQVIVI
Copy the delimiter into EBLK

Is the tuple file large enough to contain the new data?

yes →

no →

C (page 38)

(page 38)

C (page 18)

G-41
ADD DATA TUPLES TO RELATION

PAGE 38

C(page 37)

2(page 37)

REDUF
Reallocate a new tuple file.

Was the allocation successful?

no

yes

Set the number of tuples to zero.

PUTDF
Put the data into the new tuple file.

Is the ADF large enough to contain the new data?

no

yes

REDADF
Reallocate a new ADF.

Was the allocation successful?

no

yes

Set the number of tuples to zero.

PUTDF
Put in the new data.

Could either of the reallocations not be successfully accomplished?

yes

no

"Not enough rows to add specified data. Attempt reinvented."

3(page 39)
PROCESS ALL CARS
Get the data from the data file.
Process the data.
Remove a card from the relation.
Data has been removed from relation.
BACK
Add data to relation.

DATA

TUPLES
RELATION

Ftocas All data on the cards.

DATA

TUPLES
RELATION

Ftocas All data on the cards.

DATA

TUPLES
RELATION

Ftocas All data on the cards.

DATA

TUPLES
RELATION

Ftocas All data on the cards.
EDITHEL

PROCESS ALL CARDS

START

Read the next data card

Have all the tuples to be added been processed? Y/N

Have all the columns been examined? Y/N

ADDUP
Add the tuple data.

PUTIF
Put the new IDF away.

Working

Is there non-string strategy? Y/N

Input as Non String

Is there string strategy? Y/N

Is there non-string strategy?

Have all the data cards been read? Y/N

A (page 41)

B (page 41)

C (page 41)

G-44
A (page 41) \rightarrow D (page 41) \rightarrow \text{Increment through the string} \rightarrow \text{CPYSUB: Copy all the non-blank data} \rightarrow \text{PADFRG: Put the data into the ADF} \rightarrow \text{Update IIFRC and IDENT arrays.} \rightarrow \text{E (page 41)} \rightarrow \text{PROCESS ALL CARDS} \rightarrow \text{PAGE 42}
INPUT AS
NON STRING

START

Have all the data cards been read?

yes

no

Is the counter greater than or equal to 80?

yes

Read The Next Data Card

Find The Next Delimiter

Increment the counter

Copy String Between Delimiter

CATSTR
Append the delimiter to the string.

Is the counter greater than 80?

yes

no

CPYSTR

CPYSUB

(page 44)
I1(page 43)

Has a non-blank character been found or has the end of string been found?

yes

no

GETCH
Get another character

Increment through the string

CPYSUB
Copy all the data

Remove trailing edge blanks.

Has a non-blank character been found or has the end of string been found?

yes

no

GETCH
Get another character

Increment through string

CPYSUB
Copy all the non-blank data.

I2(page 44)
Yes is the string blank?

Make empty assignments

Is there floating point strategy?

No

Make empty assignments

REAPP

Is the string blank?

Yes

REANM

Is there integer strategy?

No

Is the string blank?

Make empty assignments

Yes

Is there floating point strategy?

Is there single character strategy?

No

Is the string length greater than one?

Yes

Copy one character into ITFC.

"Single character only. Only first character used."

No

PAGE 43

INPUT AS NON STRING
EDITIE

START

Mark the string

COPYSUB
Copy the marked string

BACK

COPY STRING BETWEEN DELIMITER

PAGE 46

FIND THE NEXT DELIMITER

START

Has the delimiter or the end of string been found?

yes

no

COPYSUB
Copy in another character.

BACK

G-50
START

Read a block of data into FORMAT array.

Were there any I/O errors?

Yes

Write Data Card Read Error

IO STATUS CODE

Has an end of file detected?

No

Yes

Write Data File End of File

BACK
START

"Data card read error."

Set the number of tuples to zero.

BACK

WRITE DATA CARD READ ERROR

PAGE 48

START

End of file. Tuples were added.

Set an end of file marker.

BACK

WRITE DATA FILE END OF FILE

START

Find the longest string.

Move all strings for right justification.

BACK

ASSIGN TO LONGEST DISPLAYED COLUMN NAME
START

Blank Formac Array

COPYSTR Copy the data into FORMAT

Does the user want columns to be displayed (DISCOL) ?

yes

Has the maximum column name length been reached?

no

Align To Longest Displayed Column Name

Are the column names loaded into core ?

Move names into NEAMD

yes

no

COLNAM Get the column names from ADP into NEAMD.

CATSTR Append NEAMD (column names) into FORMAT.

CATSTR Fill up the rest of FORMAT with spaces.

CATSTR Append ';' to the end of each string in FORMAT

1 (page 50)
EDITREL
GET COLUMN NAMES INTO CORE
PAGE 53

START

Is the column name longer than 64 characters?

yes

Have all the columns been examined?

no

COLUM
Get the column name

Load the column name into CWARES

Set COLUM to true to indicate that column names are in core.

Write out the column names.

"Column names"

BACK

G-57
Are the column names loaded into core?

Yes: Write out the column names (CNAMES)

No: COLUMN Get the column names into core from ADF

Back

Has the counter (I) reached 63?

Yes: BACK

No: Set FORMAT(I) equal to zero.
START

Blank Format Array

COPYSTR
Copy FORMAT array into MEMDATA array

Has the file (DOUT) been successfully opened?

yes

Display Prompt

Read from DOUT

Open the DOUT file.

Could the device or file be successfully opened?

no

no

Set flags to indicate proper opening.

yes

(page 56)
EDITREL
SAVE LIKE
ADD COMMAND
EXPECTS PAGE 56
"Too many errors. Check output device and try again."

Were there more than 2 attempts to open an output file or device?

Has the output file or device successfully opened?

Has a valid number of records been given?

"Output how many records?"

Display Prompt

GETNAM
Get the number of records

Has the answer verified?

"Invalid input. Try again."

Close the file (WRITE)

Was the number of records to be outputted greater than zero?

"Nothing done."

A (page 57)  2 (page 57)
EDI
I H EL
A
(pae
5
6
) SAVE IF I I I' F
. AoL
M
, r I D.
FXPFC
Hs
a
valid
PII57
dolimiter
lspsciM
d
? deli
'uter
be
ftap
L-=
ISRL
Iuse
the
Idelite
l
Test
the
answer for
validity

EDIIHEL
ADD CLIM II
EXPECTS
PAGE 57

A (page 56)

2 (page 56)

Has a valid
delimiter
been specified?

no

yes

Enter single
character
delimiter between
data elements

Display
Prompt

GET
Get the
delimiter

STRIEN
Use the
delimiter
as a string

Test the
answer for
validity

EQUIV
Copy the
delimiter into
IBLK and ACTION
array

Write out
the output
array

yes

no

Have all the marked
records been copied
into the output array.

records
saved to

Close the
file (LISAV)

RTFRC

Have all the designated
columns been examined?

yes

A (page 58)

B (page 58)

G-61
Blank Format Array

Copy the column data into FORMAT depending on position. (PITF, UTFAR, RADFRC)

Write the FORMAT array out into USAV

RTFRC
Put away RELDES

BACK
OVERLAY: NEWRELAT.FLC

SUBROUTINE: NEWREL.FLC

SYNTAX:
CALL NEWREL(IDENT,ISAFTF,ITDF,INC,INR,IDEM,IDADF,ITDF,ITFRC,IADFRC)

NEWREL---CREATE A NEW RELATION.

THIS ROUTINE IS USED TO INITIALLY ESTABLISH A RELATION. THE THREE FILES .TF, .TDF AND .ADF ARE ALLOCATED, AND THE COLUMN HEADINGS AND STRATEGIES ARE REQUESTED AND SET UP. ALSO THE INITIAL VALUES FOR THE RELATION (IDENT AND ITDF) ARE ASSIGNED TO THE FILES.

THE SUBROUTINE ARGUMENTS ARE:

1. IDENT  - RELATION STATUS: CORE RESIDENT,
2. IDEM    - DIMENSION OF IDENT,
3. ITDF    - CORE RESIDENT TDF,
4. INC     - NUMBER OF COLUMNS IN TDF, AND
5. INR     - NUMBER OF RECORDS IN TDF.
NEWREL

DEFINE
IF, IFES, ...
DEFINE, ETC

ENTIRE
NAME FOR
NEW RELATION

NAME
LONGNAME
Get the
relation
name.

CONFIRM
LOCATION OF
SAME NAME
YES or NO

YES NO

REPEAT
OK

REPEAT
OK

YES NO

WHEN

NUMBER OF
COLUMNS (WIDTH)
OF RELATION

NUMBER
LONG/SHORT

IF

IEXC = 0

1

(Due to 2)

IEXC = 1
APPENDIX I

DELREL
OVERLAY: DELTREL.FLC

SUBROUTINE: DELREL.FLC

SYNTAX:
CALL DELREL

THIS ROUTINE DELETES AN UNWANTED RELATION. ALL FILES .TF, .TDF, AND .ADF ARE DELETED. THIS ROUTINE REQUIRES CONFIRMATION THAT THE RELATION NAMED IS THE CORRECT RELATION TO BE DELETED BEFORE IT IS DELETED.
DELETE

DEFINE VARIABLES, TYPES, CONSTANT FOR COMMON BLOCKS

ENTER NAME OF RELATION TO BE DELETED?

NAME (LONGNAME) Get the relation name.

HASH (LONGNAME, FILNAME) Hash the relation name.

'CONFIRM DELETION OF FILENAME 'YES OR NO'

YES NO

IF (DELETE) FALSE

RETURN
APPENDIX J

MODCOL
Subroutine: MODCOLUM

This subroutine provides for the manipulation of columns in an existing relation. No changes in strategy are allowed, since this would assume a change in data.

The basic functions provided are:

1. Delete a column
2. Add a column
3. Alter a column name
4. List current column names
START

Clear the screen

"Modify column specifications"
"Relation to be modified."

NAME
Get the relation name.

HASHT
Hash the relation name.

FILDES
Set-up for disk operation.

RTVIDF
Get IDENT and ITDF arrays for modification.

Is there an I/O Error?...

yes
RETURN

no

Has an exit flag been set by the QUIT option?

yes

no

"Modcol command:"

GET
Get the command answer.

A (page 2)  B (page 2)
MODCOLUMN

PAGE 2

A (page 1) B (page 1)

STREN
Interpret the answer as a string.

Is the answer C? (list column names)
yes
no

Is the answer R (rename a column)?
yes
no

Is the answer D (delete a column)?
yes
no

Is the answer A (add a column)?
yes
no

Is the answer Q (quit)?
yes
no
Set an exit flag.

Write the valid commands

J-3
MODCOLUM

Are there more column names in this relation?

- yes: COLUMN
  - Get the column name
  - "column", column name
  - Increment to next column

- no: A (page 1,2)
MODCOLUMN

Has OK been set to true?

Yes

Column to rename

GET COLUMN NUMBER
find the column, return OK

No

New column name

GET
Get the column name.

STILEN
Interpret the column name as string

Update ADF byte count and add new bytes to the count.

PADFRC
Put column name into ADF

Column: is now named

(page 2)

A (page 1,2)
MODCOLUM

(page 5)

Is the number of
tries greater than 3?

no

yes

"User response error. Nothing done."

RETURN

Adjust the number
of deleted columns,
the number of columns
in the relation, and the number of bytes
in the file.

Is the strategy an integer,
single character, or decimal number?

no

Are there more tuples
in this relation?

no

yes

Update the strategy
count.

KTRC
retrieve TF
record into
cone.

Shift each
element in
KTRC by one.

KTRC
write the
shifted TF
back to tuple file.

Adjust the
TF length
count.

Increment
to next record.

G (page 5)

H (page 5)

J (page 5)

K (page 7)
Is the strategy string data?

Are there more tuples in this relation?

Update string strategy count

KIFRC Retrieve the TF inode

Reduce the ADF byte count.

Shift each element in KIFRC by two.

KIFRC Resize the shifted TF block.

Adjust the TF length count.

Increment to next TF record.

Shift the column indicators for those columns past the deleted one.
Has OK been set to true?

- yes: Write the list of possible strategies.
  - GETINM: Get the answer to strategy.
  - Determine when to stop moving elements.

- no: Is the strategy an integer number, a decimal number, or a single character?
  - no: Are there more tuple file records?
    - yes: Adjust strategy count.
    - no: Move all the elements out one slot until a hole is made for the new column data to go in.

- yes: Are there more tuple file records?
  - yes: KITRC: retrieve the TF record into core.
  - no: Put blanks into the new TF location.
  - no: PTRFC: Put the new TF record back.

- yes: Update TF length count.

M (page 10) (page 10) N (page 10) P R (page 10)

J-10
MODCOLUMN
GET COLUMN NUMBER

START

Set OK to true.

GETNUM
Get the column number

Is the column number valid?

yes

no

"request outside column range"

Set OK to FALSE.

BACK
APPENDIX K

MERGRL
Subroutine: MERGEREL

This subroutine allows the user to merge portions of two similar relations into a third composite relation.

The options are:
(1.) Merge two relations into a third relation containing only those records which meet the matching of like column pairs.

(2.) Add two relations by copying first into a third relation and then appending the second relation. The column headings must be identical to use this option.
START

'Primary Merge Relation'

NAME
Get primary relation name.

HASHIT
Hash Relation Name

FILES
Get storage vol., protection keys, etc.

ROUTE
Get IDENT and TDF for this relation.

Is there an I/O Err?
YES
RETURN

No

'Secondary Merge Relation'

NAME
Get secondary merge relation.

HASHIT
Hash secondary merge relation.

FILES
Get secondary storage vol., protection key, etc.

1 (page 2)
MERGEREL

PAGE 2

1 (page 1)

FROM

Get IDENT and TDF for second relation.

Is there an I/O Error?

Yes → RETURN

No → (page 3)

Has an EXIT flag been set?

Yes → RETURN

No → "Merge"

GET

Get the character string.

STLEN

Return the length of the character string.

CPYSUB

Do a merge?

Yes → Perform the merge.

No → Do an ADD?

Yes → Perform the add.

No → Quit?

Yes → Set an EXIT flag.

No → No

2 (page 3)

K-3
Do you need help?

Yes

No

Start

Yes

Reset?

No

Write the menu

"The valid commands are:"

2 (page 2)

3 (page 2)
START

Is either relation empty?

no

BINARY
Get the IDENTM and IDSN arrays

Empty the ITDFM array

"Name of Merged Relation."

NAME
Get the name of the relation

HASHIT
Hash the relation name

FILDES
Set up for disk operation

Find Column Pairs For Merge

Do the columns match?

"No column domain matches to perform merge."

BACK

"X Records resulted from this merge."

"One relation empty, Merge cannot be performed."

yes

MERGEREL
DO_A_MERGE
PAGE 4

K-5
START

Have all columns been checked?

Yes

No

Has an exit flag been set?

Yes

No

Compare column names.

Are the column names the same?

Yes

Move to next column.

No

Compare the lengths and strategies.

Are the lengths and strategies the same?

Yes

COLNAME Get column name for relation 1.

COLNAME Get the column name for relation 2.

Set an exit flag.

No

BACK

PAGE 5

MERGEREL

FIND_COLUMN_PAIRS_FOR_MERGE

K-6
START

COMPARE COLUMNS PAIRS

PAGE 6

MERGEREL

RTFRC
Get the tuple for Rel. 1

Are there any columns remaining?

No

Yes

Is match true?

No

BACK

Yes

RTFRC
Get the tuple for Rel. 2

Are there pairs remaining?

No

Do the number of matches equal the number of pairs?

No

MATCH = true

Increment counter

Yes

1 (page 7)

A (page 7)
Does IINFL(L1, LCC1) = 4?

- yes
  - Do the elements match?
    - no
    - yes
      - Matches = Matches + 1

- no
  - yes
    - Matches = Matches + 1

Does ITFRC(K1+1) = ITFRC(K2+1)?

- no
  - yes
    - Are both elements of ITFRC empty?
      - yes
        - Make them unequal.
      - no
        - Make them unequal.

- yes
  - Are both elements of ITFRC2 empty?
    - yes
      - Make them unequal.
    - no
      - Make them unequal.
MGEREL
COMPARE
COLUMN
PAIRS

Are both elements of ITFRC and ITFRC2 empty?

yes

no

Do they match?

yes

no

Matches =
Matches + 1

Is ITFRC2(K2) less than zero?

yes

no

UTFAR
RADFRC

Is ITFRC(K1) less than zero?

yes

no

UTFAR
RADFRC

K-9
START

Move Headings and strategies 1 to 3

Increment Column

Move nonzero data from relation 2 to merged relation

Increment Next TP Location

COLUMN
Get column name for relation 2

Put Column Names in New Relation

Is there another column?

Yes
No

BACK

PAGE 9
MERGEREL
PUT.COLUMN_NAME
IN.NEW.RELATION

START

PADPRC
Get column name information.

Is the data in a tuple file?

No

Yes

Make assignments from IDENTITY

Make assignments from LA1 and LA2

BACK

PAGE 10
START

Is there another column?

No → BACK

Yes → Increment Column Number

Move headings and strategies from Relation 1 to merged relation.

Increment Next TV Location

COLUMN

Get column name for Relation 1

Put column name in New Relation

MERGEREL
MOVE.HEADINGS.AND
STRATEGIES.1.TO.3
PAGE 12

K-13
MERGEREL
MOVE TUPLE DATA TO NEW RELATION
PAGE 13

START

MOVE TUPLE DATA FROM FIRST.

Is IDENT2(L9) = 1st counter?

Y: BACK

N: Is IDENT4(L9) = 2nd counter?

Y: Increment 1st counter

N: Set SKIP to false.

Is IPAIR = 3rd counter?

No: Is SKIP true?

Yes: Increment 3rd counter.

No: Is SKIP false?

Yes: Increment 2nd counter.

(page 14) (page 14)
MOVE TUPLE DATA TO NEW RELATION

Is INTF2(L1,1st count) > 4?
Yes: Move integer Data

No

Is the 1st string empty?
Yes: Is the 2nd string empty?
Yes: Make assignments to ITFRC
No

Are the strings less than zero?
No

Is the first string greater than zero?
No: Is the first string empty?
No: Make assignments to IKEY, DKEY
Yes: Is ITFRC less than zero?
Yes: RADFRC

No: UTFAR

(page 13)
MOVE TUPLE DATA TO NEW RELATION

MERGEREL

(page 15)

Make assignments to INKEY, INKEY

PADFRC

Is TPAR true?

No

Make assignment to ITFRC

Yes

Make assignments to ITFRC and IDENTIM

A (page 13)
MERGEREL
MOVE TUPLE DATA FROM FIRST
PAGE 17

1 (page 16)

PADFRC

Is TFAR true?

No

Yes

MAKE assignments to ITFRAC, IDENT

MAKE assignments to ITFRAC.
MERGEREL
MOVE TUPLE DATA FROM SECOND

START

Does IDENT2(L1, counter) = counter? Yes → BACK
No

Increment counter

Does ITDF2(L1, counter) = 4? Yes → Is the first string empty?
No → Make assignments to IRKEY, IMKEY

Is the first string empty? Yes → Is the 2nd string empty?
No → Make assignments to IRKEY, IMKEY

Is ITFC less than zero? No → RADFRC
Yes → UTFAR

Make assignments to IRKEY, IMKEY

1 (page 19)

A (page 19)

K-19
MODEL
MOVE TUPLE DATA FROM SECOND PAGE 19

PADFRC

Is TFAR true?

No

Make assignments to ITFRC, IDENTM

Yes

Make assignments to ITFRC

K-20
MERGEREL
MERGE TWO RELATIONS
PAGE 20

START

TDFRE
Create a tuple data file.

Put name into IDENT

TDFRE
Create a tuple file.

Is IDENT(i,j) a counter

Yes

No

Increment counter

Does TDFRE(i1,counter) = 4?

No

Yes

Have all the pairs in the column been examined?

Yes

No

Is this a comparison column?

No

Yes

Zero the counter.

1 (page 21)
MERGEREL
MERGE TWO RELATIONS
PAGE 22

1. ADFORCE
   Create an AD

2. Move Column Names and Strategies

3. Fill the IDENT array

4. Zero the counter.

   Is IDENT(L9) = counter?

   Yes

   No

   Increment counter.

   Is [IDPM(L1, counter) = 1?

   Yes

   IDENT(L9) = IDENT(L9 + 1)

   No

   Is [IDPM(L1, counter) = 2?

   Yes

   IDENT(L9) = IDENT(L9 + 1)

   No

   Is [IDPM(L1, counter) = 3?

   Yes

   IDENT(L9) = IDENT(L9 + 1)

   No

   Is [IDPM(L1, counter) = 4?

   Yes

   IDENT(L9) = IDENT(L9 + 1)

   No

   Is [IDPM(L1, counter) = 5?

   Yes

   IDENT(L9) = IDENT(L9 + 1)

   No
MERGEREL
MERGE TWO RELATIONS

3 (page 22)

Make assignments to INDEX, INDEX

PUTTDF Put new file IDENTM and IDPM away.

Are there columns left to examine?

yes

INCREMENT COUNTER

Compare Column Pairs

Do the column pairs match?

no

yes

Move Tuple Data to New Relation

ADDTDF

PUTTDF

BACK
START

Check If Columns Match

Do the columns match?

Yes

'Name of Added Relation'

NAME
Get the relation name

HASH
Hash the relation name

FILES
Set up for disk operation

Copy the names and strategies

Copy the data

BACK

MERGEREL
DO AN ADD

PAGE 2A
START

Set OK = true

Does IDENT1(L1, counter) = counter?

Back

no

Increment Counter

Does TDIF1(L1, counter) = TDIF2(L1, counter)?

OK = false

Yes

OK = true

no

Does TDIF1(L3, counter) = TDIF2(L3, counter)?

OK = false

Yes

OK = false

Are the names the same?

OK = False

no

yes
START

Is IDENT1(L9) = counter?

No

READRNC Read first TPR

Move Tuple Data from First

ADD/TUP Add tuples from first

PUT/TFUF Put tuples from list rel. into new rel.

Increment counter

"Tuples copied from primary selection..."

Zero the counter.

1

(page 28)
MERGEREL
COPY THE DATA
PAGE 28

(page 27)

Is IDENTIFY(L4) = counter?

Yes

no

RTFAC
Read 2nd TP

Move Tuple
Data from Second

ADDTP
Add tuples from second relation

PUTTPR
Put tuples from 2nd rel. into new rel.

Increment counter

"Tuples added from second relation, resulting in a total of X tuples.

BACK
Subroutine: REORGREL

This subroutine is used to reorganize a relation. The purpose is to remove unwanted "dead space" from the .ADF. The deleted records are removed from the .ADF file then the revised .ADF is written back for storage.
REOR-11

START

"Name of relation to be reorganized."

NAME
Get the relation name.

HASHIT
Hash the relation name.

FILES
Set up for disk operation.

KVIDF
Bring tuple data into core memory.

Is there an I/O Error?
Yes
RETURN

no
"Beginning reorganization."

ENLIST
Empty the IDENT and new tuple descriptor array.

Fill the IDPT array with errors.

(page 2)
Equate things in TDF that do not change.

Transfer the relation name into LDNT array.

RESET TEMPFILE
Clear the temporary file.

Open the tuple file, change access, prepare CATSTR RELDESA for rename FSLIDE.

DELETE TEMPFILE
IF IT EXISTS

Rename RELDESA.TDF to RELDESA.TF

DELETE TEMPFILE
IF IT EXISTS

Rename RELDESA.TDF to RELDESA.TF

SET LDFL
RESET TEMPFILE
Clear the temporary file.

Open tuple data file, change access, prepare RELDES, CATSTR for rename FSLIDE.

DELETE TEMPFILE
IF IT EXISTS

Rename RELDESA.TDF to RELDESA.TF

L-3
RESET TEMPFILE
Clear the temporary file.

Open the ADF, change access, prepare RELDESA for rename.

DELETE TEMPFILE
IF IT EXISTS

Rename RELDESA ADF to RELDESA ADF

FRE
Create a tuple file

FRE
Creates the tuple data file

Calculate the size of the ADF.

FRE
Create the ADF

Were there any errors in FRE?
RESET TEMPLFIE
Clear the temporary file

Move the headings and strategies.

COLUMN
Get the column name for the original relation.

PADFRC
Put column name into ADF Pointer to TDF

Are there more columns?

no

yes

Are there more tuples?

no

yes

PREPARE
Open the TDF, OPNTDF,
change access, MONADF,
rename RELDES.TDF back to RELDES.TDF

Are there more columns with STR strategy?

yes

no

NUMDF
Put column number in relation

Are there more columns?

yes

no

PUTDF
Print the relation

Are there more strategies?

yes

no

RETURN

"Warning
Relation not reorganized."

PREPARE
Open the ADF, OPNADF,
change access, MONADF,
rename "NEWLIB.ADF" FILESEL back to RELDES.ADF FILESEL

RESET TEMPLFIE
Delete the newly created files

RESET TEMPLFIE

PREPARE
Open the tuple file, OPNTDF,
change access, FILESEL,
rename RELDES back to RELDES.TDF

PREPARE
Open the tuple file, OPNTDF,
change access, FILESEL,
rename RELDES back to RELDES.TDF

PREPARE
Prepare the LADFR to be RELDES.TDF FILESEL
file descriptor

PREPARE
Prepare the LADFR to be RELDES.TDF FILESEL
file descriptor.
Calculate if it exists, either $T(K)$ or $\text{IF TM}(-1P,1)$. If $T(K)$ is less than $\text{TF}(-1P,1)$, return $\text{TF}(-1P,1)$. If $\text{TF}(-1P,1)$ is true, make assignment to $\text{IT}(-1P,1)$ and update $K$. If not, increase $K$.

Yes

No

Is $T(K)$ true?

Is $\text{TF}(-1P,1)$ true?
Subroutine : BACKUPREL

This subroutine allows the user to backup or restore relational data. In backing up a user will copy the data from the primary storage media (usually system disk) to a secondary storage media (magnetic tape.) This protects the user's data and allows reconstruction in the event of a system failure.

In restoring a relation, data is transferred from magnetic tape back to the system. This utility is used primarily after a system failure. This subroutine is setup to transfer the .TF, .ADF, and .TDF files from the user storage volume to the nine track tape drive. The user must insure mounting of the proper tape and placing the drive in the proper configuration.

The following information is stored on magnetic tape during a backup:

1. The relation name
2. The date of backup
3. Read/write security keys
4. The Tuple Descriptor File (TDF)
5. The Tuple File (TF)
6. The Alpha Data File (ADF)
Write the ADF element to tape

Rewind the tape

FILE interpretation

Close all tape files

Has been stored with protection type?

RETURN

Do you wish to restore a relation?

Yes or no?

YES/NO

Get the yes or no answer.

Was the answer "yes"?

RETURN

Yes

Place the tape on drive on on-line state. Press RETURN to continue.

No

Wait for any key to be pressed.

3 (page 4)
BACKUP-II

Relation present? Yes/No. Is this data current? Yes/No.

YES

Get yes or no answer.

no

Write Tape 10 Error

Create the new TDF.

Create the new TDF.

Read the file from tape

RETURN

PAGE 5

4 (page 4)

5 (page 6)
ADFORE
Create the new ADF

Were there any errors during TV, TDF, or ADF creation?

yes

Were there any errors in creating the TV?

no

Were there any errors in creating the TDF?

no

Were there any errors in creating the ADF?

yes

Display Create Error Code

Were there any errors in creating the ADF?

no

yes

Display Create Error Code

Was the error code 314?

no

"Nothing done."

yes

"A Relation already exists with that name. Do you wish to backup to a new name? (yes or no)"

YES

Get the yes or no answer.

RETURN

NO

Close the tape file.

M-7
Was the answer 'YES'? 

no > Close the tape file

yes

'Enter the new filename'

RETURN

NAME
Get the file name.

Hash
Hash the relation name

FILDES
Setup for proper disk IO

TFORE
Create a new TF under the new filename.

ADFORE
Create a new ADF under the new filename

Were there any errors during file creation?

'RETURN

PUTDF
Put the TDF away.

RETURN

PAGE 7

page 8

H-8
START

Setup using system "EXEOP" function.

Has the tape file been successfully opened or have there been 3 tries?

yes

no

Open the tape file.

Were there any file open errors?

no

yes

"Tape file not opened properly."

"Place tape on drive in on-line state, Press RETURN to continue."

Wait for any character to be pressed.

Were there any errors?

no

yes

"Must be having tape problems, Start over."

RETURN
This tape does not contain the relational data you seek.

Rewind the tape.

Close the tape file.

START

GET

RELA TION

DESCRIPTION

EMPTY empty the NAME array

"Name of relation"

NAME Get the relation name

HASHT Hash the relation name

FILES Setup for disk I/O

BACK
APPENDIX N

DBSTAT
Subroutine: DBSTAT

This routine gives the user the physical status of a relation. This information includes the size of the relation, the number of columns, the number of rows, the dead to active space ratio, as well as the column data, if desired. DBSTAT allows for a printer copy of the relation status, if desired.
DSTAT

File: DSTAT

Given a relation physical...

Notes: 1. return prints.

Significant imports:

BASES ("or ")
FOR C.

RESTART

SAME

MOUNT

FILES

GET TOP

TOP

OUT

ERROR

YES

RETURN

line

change the relation

and print the

new TOP

error

print

DVD

YES

read

YES

STOP

after current

information

YES

read choic

DSTART

FACOPY

YES

print header

with long name

and file name

NO

len

FACOPY

YES

print length

NO

display length

1

2
APPENDIX O

HELP
OVERLAY: HELP.FLC

SUBROUTINE: HELP.FLC

SYNTAX:
CALL HELP

THIS IS A SUBROUTINE ADDED TO FEASIL 77 TO AID THE USER IN MANIPULATION OF DATA RELATIONS UNDER FEASIL 77. THE VALID PARAMETERS (PASSED IN THE INTEGER (HOLLERITH) ARRAY "PARAM") ARE:

CREATE - EXPLAINS THE RELATION CREATION UTILITY.
EDIT - EXPLAINS EDIT COMMAND STRUCTURES.
DELETE - EXPLAINS THE DELETE UTILITY.
MERGE - EXPLAINS THE MERGE RELATIONS PROCEDURE.
MODIFY - EXPLAINS COLUMN MODIFICATION PROCEDURE.
RETRIEVE - EXPLAINS RETRIEVE AND MANIPULATION UTILITY.
REORGANIZE - EXPLAINS NEED FOR REORGANIZATION.
STATUS - EXPLAINS THE STATUS RELATION COMMAND.
BACKUP - EXPLAINS THE BACKUP UTILITY.
COLUMN - EXPLAINS RELATION COLUMN STRUCTURES.
RECORD - EXPLAINS RELATION RECORD STRUCTURES.
VOLUME - EXPLAINS THE SYSTEM VOLUME AND IT'S USE.
PROTECTION - EXPLAINS USE AND NEED FOR PROTECTION KEYS.
DEVICE - EXPLAINS SYSTEM DEVICE STRUCTURE.
STRATEGY - EXPLAINS FOUR TYPES OF DATA STRATEGIES.
HELP

DEFINE VARIABLES AND SET MENU=True.

ENTER CATEGORY OR (RETURN) FOR MENU?

GET (0,IWORK, ENDFIL)

STRLEN (IWORK, PARAM, LEN,MNDA)

WHEN (LEN.EQ.0)

MENU = .FALSE.

$x 
eq 45$

FALSE UNTIL (NOT MENU)

I = φ, FOUND = .FALSE.

I = I+1, J = φ

J = J+1, IWORK(J) = VAL(I, J)

UNTIL (J.EQ.VAL (I+1)+1)

IF (STMARK(IWORK, IWORK))

UNTIL (I.EQ.15), OR (FOUND)

FOUND = .TRUE.
**45** THE VALID HELP PARAMETERS ARE:

- BACKUP
- CREATE
- DELETE
- EDIT
- MERGE
- MODIFY
- REORGANIZE
- RETRIEVE
- STATUS
- COLUMN
- RECORD
- VOLUME
- PROTECTION
- DEVICE
- STRATEGY

**1001** "CREATE" IS USED TO CREATE THE FILES ASSOCIATED WITH EVERY FEASIL RELATION. TO CREATE A FILE, YOU MUST KNOW HOW MANY COLUMNS THE RELATION REQUIRE. THE DESIRED PROTECTION KEYS ASSOCIATED WITH THE RELATION (IF DESIRED), AND THE COLUMN STRATEGIES FOR EACH OF THE COLUMNS CREATED. THE USER MUST ALSO KNOW THE NAME OF THE VOLUME ON WHICH THE RELATION IS TO BE CREATED. SEE HELP ON: PROTECTION, STRATEGY, COLUMN, VOLUME.

**1002** "EDIT" IS USED TO ENTER DATA INITIALLY OR TO MODIFY ALREADY EXISTING DATA. TO USE THE "EDIT" FEATURE, THE USER MUST KNOW THE NAME OF THE DESIRED RELATION, THE PROTECTION KEYS ASSOCIATED WITH THE RELATION, AND THE NAME OF THE VOLUME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.

**1003** "DELETE" IS USED TO PERMANENTLY REMOVE A RELATION FROM THE SPECIFIED VOLUME. TO USE THE "DELETE" FUNCTION, THE USER MUST KNOW THE RELATION NAME, THE PROTECTION KEYS, AND THE VOLUME NAME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.

**1004** "MODIFY" IS USED TO PERMANENTLY CHANGE THE:

- A> COLUMN NAME, OR
- B> NUMBER OF COLUMNS (DELETE OR ADD ONE).

THE USER MAY ALSO LIST THE COLUMN NAMES USING THIS FEATURE.
SEE HELP ON: COLUMN.

**1005** "REORGANIZE" IS USED TO INCREASE THE EFFICIENCY OF A FEASIL RELATION. IT IS PRIMARILY USED AFTER COLUMN SPECIFICATIONS HAVE BEEN MODIFIED (i.e., USING MODIFY) SEE HELP ON: COLUMN, MODIFY.
"RETRIEVE AND MANIPULATE" IS USED TO OPERATE ON AN EXISTING RELATION BY THE RELATIONAL DATA BASE OPERATORS (i.e., "AND", OR "OR"). IN ADDITION, R & M IS USED TO SORT RELATIONAL DATA BY COLUMNS, PRINT RELATIONAL DATA TO A DEVICE OR FILE, REPRODUCE A RELATION UNDER A DIFFERENT NAME, OR TO MOVE A SUBSET OF THE PRIMARY RELATION TO A NEW RELATION. SEE HELP ON: COLUMN, DEVICE.

"BACKUP" IS USED TO SAVE A BACKUP (COPY) OF THE RELATION TO (FROM) MAGNETIC TAPE. IN ADDITION TO STORING THE RELATIONAL DATA TO THE TAPE, THE DATE OF BACKUP AS WELL AS AN OPTIONAL PROTECTION KEY SEQUENCE ARE ALSO STORED.

"STATUS" IS USED TO INQUIRE ABOUT THE SIZE OF A RELATION AS WELL AS TO INQUIRE ABOUT THE "DEAD" SPACE USED BY THE FILE. THE "DEAD" SPACE MAY BE MINIMIZED, IF DESIRED FOR EFFICIENCY USING THE REORGANIZE FEATURE. SEE HELP ON: REORGANIZE.

"MERGE" SERVES TWO PURPOSES. IT'S FIRST PURPOSE IS TO PRODUCE A THIRD RELATION FROM TWO INPUT RELATIONS BY PERFORMING A "MERGE". THE MERGE FUNCTION IS A "UNION" OPERATOR. THEREFORE, THE OUTPUT RELATION CONSISTS OF THE TOTAL NUMBER OF UNIQUE COLUMNS IN THE TWO INPUT RELATIONS. THE ONLY RECORDS WHICH ARE MAINTAINED THROUGH THE "MERGE" ARE THE ONES WHICH INTERSECT ONE-TO ONE WITHIN THOSE COLUMNS WHOSE NAMES INTERSECT ONE-TO ONE.

THE SECOND PURPOSE OF "MERGE" IS TO PRODUCE A THIRD RELATION FROM TWO INPUT RELATIONS WHICH HAVE THE IDENTICAL COLUMN NAMES AND STRATEGIES. THE RESULTING RELATION HAS THE SAME COLUMN NAMES AS THE TWO INPUTS AND THE RECORD ARE A "UNION" OF THE TWO INPUT RELATIONS. SEE HELP ON: COLUMN, RECORD, STRATEGY.

"COLUMN" IS THE NAME GIVEN TO EACH CATEGORY OF RELATION. EACH COLUMN HAS A UNIQUE COLUMN NAME AND MAY BE REFERENCED ALONE OR WITH OTHER COLUMNS IN THE SAME RELATION (IF THERE ARE ANY). AN EXAMPLE OF COLUMNS IN A RELATION CALLED "MYCLASS" MAY BE:

1 - NAME
2 - AGE
3 - SEX
4 - SCORE
EACH RECORD OF THE RELATION HAS AN ENTRY FOR EACH COLUMN IN THE RELATION. THE RELATION, THUS, FORMS A DATABASE OF ENTRIES UNDER THE COLUMN HEADINGS. SEE HELP ON: RECORD.

*1011 "RECORD" IS A COLLECTION OF COLUMNS WHICH TOGETHER DESCRIBE THE DESIRED ASPECTS OF THE DATABASE. EACH RECORD IN A RELATION CONTAINS DATA FOR ALL OF THE COLUMNS OF THE RELATION. RECORDS MAY BE ACCESSED INDEPENDANTLY OR WITH OTHER RECORDS USING "RETRIEVE AND MANIPULATE".
SEE HELP ON: COLUMN, RETRIEVE.

*1012 "VOLUME" IS THE NAME GIVEN TO BULK MAGNETIC MATERIAL USED IN COMPUTER INFORMATION STORAGE. FOR EASY LOCATION OF DATA, THESE VOLUMES ARE UNIQUE NAMES. TYPICAL VOLUME NAMES ARE:
MT32
MT6A
PRO1
TEMP
THE USER SHOULD INQUIRE ABOUT THE PROPER VOLUME TO USE FOR EACH APPLICATION VIA THE SYSTEM OPERATOR.

*1013 "PROTECTION". FEASIL -77 POSSESSES THE ABILITY TO OFFER A CERTAIN DEGREE OF PROTECTION AGAINST THE ACCIDENTAL DELETION OF RELATIONAL FILES. THIS SYSTEM OF PROTECTION INVOLVES THE USAGE OF PROTECTION "KEYS" ASSOCIATED WITH EACH RELATION. ON THE PERKIN-ELMER OS/32 SYSTEMS, THE READ/WRITE PROTECTION KEYS ARE CODED AS FOLLOWS:

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>UNPROTECTED. ANY KEYS WILL WORK WHEN ACCESSING THIS RELATION.</td>
</tr>
<tr>
<td>10-254</td>
<td>CONDITIONALLY PROTECTED. THE USER MUST KNOW THE CORRECT CODES TO ACCESS THE RELATION.</td>
</tr>
<tr>
<td>255</td>
<td>UNCONDITIONALLY PROTECTED. NO KEYS WILL ALLOW ACCESS TO THE RELATION.</td>
</tr>
</tbody>
</table>
"DEVICE" is the term used to identify a non-file peripheral attached to the computer to/from which data may be sent/received. Example devices are:

- PR: (system printer)
- CON: (user console)
- PLOT: (system plotter)
- MAGO: (9 track tape drive)

Contact the system operator for more information about the system’s peripheral devices.

"STRATEGY" is the term used to refer to the type of data that is contained in a particular column. The valid strategies are:

- "1" - integer number,
- "2" - decimal (real) number,
- "3" - single character, and
- "4" - character string.

See help on: column.
APPENDIX P

TPLIB1
Library: TPLIB1

This library is a collection of subroutines used throughout FEASIL. A list of these subroutines and a brief description of their purpose is provided below.

**STRLEN** - Returns the length of a string.

**FTIPI** - Converts floating point data to integer format.

**PITFP** - Converts integer format data back to floating point.

**YESNO** - Logical. Gets a "yes" or "no" answer from the console and returns true or false.

**GETNUM** - Get integer data.

**REANUM** - Converts a string into integer data.

**ELEB** - Eliminates leading edge blanks.

**ENFPTA** - Converts floating point data to ASCII format.

**ENINTA** - Converts integer data into ASCII format.

**GETFP** - Get floating point data.

**REAFFP** - Converts string data into floating point.
STRELEN

TRUNCATE TRAILING BLANKS FROM STRING

CALCULATE TOTAL LENGTH OF STRING

PLACE LENGTH OF STRING INTO ARRAY ELEMENT #1 OF OUTPUT ARRAY

PUT CHARACTERS 4 PER ELEMENT INTO ARRAY ELEMENT 2 THROUGH N OF OUTPUT ARRAY

RETURN
GETNUM

GET

ACCEPT INPUT

STRLEN

RESTRUCTURE LINE

READNUM

CONVERT TO INTEGER WITH ERROR CHECK INT

RETURN
REALNUM

SCAN ARRAY FOR '－' SIGN

DO CONVERSION FROM ASCII TO DECIMAL

ERRORS?

Y

WRITE-FORMAT-
ERROR MESSAGE
ERROR-AND-RETURN

N

RETURN EQUIVALENCE
DECIMAL OF ASCII
AS A SIGNED INTEGER

RETURN
BEGIN
GETCH
EXAMINE NEXT CHARACTER
SPACE?
YES
ADD ONE TO COUNTER:
NB = NB + 1
CPYSUB
DELETE FIRST NB CHARACTERS FROM STRING
RETURN
NO
ENINTA

ENCODE INTEGER INTO I 12 FORMAT

ELEB

STRIP LEADING BLANKS

RETURN

ENFPTA

ENCODE DATA INTO ASCII STRING OF CHARACTERS, USES DIFFERENT FORMATS BASED ON MAGNITUDE OF DATA

ELEB

STRIP LEADING BLANKS

RETURN
GETFP

GET

ALLOW INPUT

STRLEN

RECONSTRUCT INPUT

REAFP

CONVERT FROM ASCII TO DECIMAL

RETURN
SCAN INPUT FOR A '-' SIGN

LOOK FOR A (.)

'.' FOUND?

NO

TAKE OUTPUT AS WHOLE NUMBER

YES

TREAT REMAINDER AS A FRACTION

OUTPUT IS A WHOLE NUMBER + FRACTION

RETURN
APPENDIX Q

TPLIB2
Library : TPLIB2

This library is a collection of subroutines used throughout FEASIL. A list of each of these subroutines and a brief description of their purposes is provided below:

PUTTDF - Puts the IDENT and ITDF arrays on file.
RTVTDF - Retrieves the IDENT and ITDF arrays from disk.
ADDTUP - Adds tuple data to a relation.
DELTUP - Deletes tuple data from a relation.
PUTIFO - Put info into a relation.
PTFRC - Puts data record into the tuple file.
PADFRC - Puts data record into the Alpha Data File.
RTFRC - Retrieves a record from the tuple file.
RADFRC - Retrieves a record from the ADF.
PUTIFO - Puts data into TF and ADF.
COLNAM - Get the column names from the ADF.
HASHIT - Hashes a file name into 8 characters.
UTFAR - Retrieves TF alpha record.
PUT TDF

OPNTDF

OPEN TDF FILE

ERROR?

WRITE IDENT AND ITDF ARRAYS

FILECLS

CLOSE TDF FILE

RETURN
RTVTDF

QPNTDF
OPEN TF FILE

ERROR?

READ IDENT AND ITDF ARRAYS

FILECLS
CLOSE TF FILE

RETURN
ADDTUP

MOVE COMPLETE?

Y

RTFRC
GET NEXT RECORD

PTFRC
PUT INTO NEXT HIGHER RECORD

PTFRC
PUT NEW TUPLE IN THE NEWLY CREATED SPACE

UPDATE THE IDENT ARRAY

RETURN
DELTUP

Y

DONE?

REDUCE THE "NUMBER OF TUPLES" COUNT IN IDENT ARRAY

RETURN

RTFRC

READ THE NEXT TUPLE FILE (I+1)

RENUMBER AS I

PTFRC

WRITE THE TUPLE BACK OUT AS THE TUPLE I
PUTIFD

RTFRC
READ IN THE TUPLE AS IT ON THE DISK

IS NEW DATA A STRING?

Y

IS NEW STRING EMPTY?

INDICATE IN TF

DOES STRING HAVE >7 CHARACTERS

Y

PADFRC
ADD STRING TO END OF ADF FILE

UPDATE IDENT

ERRORS?

Y

PTFRC
REPLACE UPDATED TUPLE

RETURN

Q-6
OPEN THE TUPLE FILE (TF)

ERRORS?

WRITE DATA TO APPROPRIATE TF RECORD #

CLOSE TF FILE

RETURN
RTFRC

UPNTF

OPEN THE TUPLE FILE

ERROR?

Y

READ TUPLE INFORMATION INTO ITFRC ARRAY

N

FILECLS

CLOSE TF FILE

RETURN
MAHIT

PREPARE-LONG-FI NENAME
MAKE ALL SPACES AND DASHES INTO A'S.
MAKE DIGITS 0-9 INTO A-J

PREPARE-HASHED -
EMPTY THE ARRAY
FILENAME

FORM A NEW CHAR.
FROM EVERY GROUP OF 6 AS FOLLOWS
CHAR* (CHAR+CHAR+CHAR...CHAR)/6

DONE?
Y
RETURN

COLNAM

READ ADF RECORD WHICH CONTAINS COLUMN NAME FOR COLUMN 'ICC'
REC = ITDF(2,I(CC)

RETURN

RADFRC

Q-11
UTFAR

NEIGATE THE 1st
ELEMENT TO RETURN
THE DATA TO ITS
PROPER FORMAT

EXTRACT THE
STRING LENGTH
FROM THE UPPER BYTE
OF THE FIRST WORD

UNPACK THE REST
OF THE STRING.
BYTES 2, 3, 4 of WORD
ONE AND BYTES
2, 2, 3, 4 of WORD
TWO

PLACE DATA
INTO
STRING ARRAY

RETURN

Q-12
APPENDIX R

TPLIB3
This is a collection of subroutines used for file reallocation and testing throughout FEASIL. A list of each subroutine and a brief description of their purpose is provided below.

- **RDOADF** - Reallocates the size of the ADF.
- **RDOTF** - Reallocates the size of the TF.
- **CPYREL** - Copies the TF and ADF files.
- **FSLIDE** - Slides the first six elements of the argument one slot forward.
RDOADF

FILEREN
RENAME ADF FILE TO A TEMPORARY NAME

CREATE A LARGER ADF FILE UNDER ORIGINAL RELATION NAME

CREATE ERROR?

1

T

COPY COMPLETE?

READ NEXT RECORD FROM TEMP ADF

WRITE NEXT RECORD TO NEW ADF

FILECLS
CLOSE NEW AND TEMPORARY ADF

FILEDEL
DELETE TEMP ADF

OPNADF
OPEN THE TEMPORARY ADF

OPNADF
OPEN THE NEW ADF

2

RETURN

R-2
1. FILEREN
   RENAME TEMPORARY FILE BACK TO ORIGINAL RELATION NAME

2. FILERR
   DIAGNOSE ERROR

   ISSUE WARNING TO THE USER
**RDOTF**

**FILEREN**
- RENAME TF FILE TO A TEMPORARY NAME
- CREATE A LARGER TF FILE UNDER ORIGINAL RELATION NAME

**CREATE ERROR ?**
- IF YES, THEN...
- IF NO, THEN...
- OPNTF
  - OPEN THE TEMPORARY TF
- OPEN THE NEW TF

**COPY COMPLETE ?**
- IF YES, THEN...
- READ NEXT RECORD FROM TEMP. TF
- WRITE NEXT RECORD TO NEW TF
- FILECLS
  - CLOSE NEW AND TEMPORARY TF
- FILEDEL
  - DELETE TEMP. TF

RETURNS
RENAMETEMPORARY
FILEBACKTO
ORIGINALRELATION
NAME

DIAGNOSEEERROR

ISSUE
WARNING
TO THE
USER

FILEERR

FILEREN

1

2
COPYREL

TFCRE
CREATE NEW TF FILE FOR NEW RELATION

CREATE ERROR?

Y

ADFCRE
CREATE NEW ADF FILE FOR NEW RELATION

CREATE ERROR?

Y

COPY RECORD FOR RECORD, THE ORIGINAL TF FILE TO THE NEW TF FILE

COPY RECORD FOR RECORD, THE ORIGINAL ADF FILE TO THE NEW ADF FILE

RETURN
FSLIDE

Move (EL+1) to (EL)

DONE

6 ELEMENTS ?

RETURN

EL = EL + 1

R-7/(R-8 blank)
APPENDIX S

TPFILECR
OVERLAY: TPFILECR.FLC

The following is a complete list of subroutines included in the overlay TPFILECR.FLC with appropriate syntax for their use and a brief description of each routine. These routines will be described in detail on the following pages along with their flowcharts.

1. FILDES(TOTFD,NAMFIL)---creates file descriptor.
2. SECURE(IWKEY,IRKEY)---assigns security keys.
3. EMIARY(IARRAY,N)---empties the array "IARRAY" of size N.
4. TFCRE(TOTFD,IWKEY,IRKEY,NSZTR,ISTTF)---creates .TF file.
5. ADFCRE(TOTFD,IWKEY,IRKEY,NSZADF,ISTADF)---creates .ADF file.
6. TDFCRE(TOTFD,IWKEY,IRKEY,NSZTDF,ISTTDF)---creates .TDF file.
7. RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)---creates relation.
8. TFDEL(TOTFD,IWKEY,IRKEY,ISTTF)---deletes .TF file.
9. TDFDEL(TOTFD,IWKEY,IRKEY,ISTTDF)---deletes .TDF file.
10. ADFDEL(TOTFD,IWKEY,IRKEY,ISTADF)---deletes .ADF file.
11. RELFD(IDELST,NAMFIL)---deletes relation
12. EQUIVI(ARRAY1,ARRAY2,N)---equates first N parts of 2 arrays.

FILES BY LOGICAL UNITS.

13. OPNTF(TOTFD,IWKEY,IRKEY,ISOTF,LUTF)---opens .TF file.
14. OPNADF(TOTFD,IWKEY,IRKEY,ISOADF,LUADF)---opens .ADF file.
15. OPNTDF(TOTFD,IWKEY,IRKEY,ISOTDF,LUTDF)---opens .TDF file.
16. FILERR(ISATUS,NOMATCH)---Diagnoses file I/O error codes.
17. NAME(I)---gets name "I".

S-1
SUBROUTINE: FILDES.FLC

SYNTAX: CALL FILDES(TOTFD,NAMFIL)

THIS ROUTINE INTERACTIVELY COMPLETES THE CONSTRUCTION OF THE FILE DESCRIPTOR, READ AND WRITE KEYS, AND STORAGE VOLUME.

NAMFIL IS AN INPUT NAME ARRAY DEFINED IN SUBROUTINE "NAME" AND TOTFD IS THE TOTAL FILE DESCRIPTOR FOR USE WITH STRING Routines SUCH THAT: TOTFD = <VOLUME NAME>:<FILE NAME>.<EXT>. 
SUBROUTINE: SECURE.FLC

SYNTAX: CALL SECURE(IWKEY, IRKEY)

THIS ROUTINE OBTAINS THE SECURITY READ/WRITE KEYS FOR THE RELATION, IRKEY AND IWKEY, RESPECTFULLY.
SUBROUTINE: EMIARY.FLC

SYNTAX: CALL EMIARY(IARRAY,N)

THIS ROUTINE INITIALIZES AN INTEGER ARRAY NAMED "IARRAY" AND OF DIMENSION "N" TO EMPTY.
SUBROUTINES ADFCRE.FLC, TFCRE.FLC, AND TDFCRE.FLC

SYNTAX: CALL ***CRE(TOTFD,IWKEY,IRKEY,NSZ***,IST***)
WHERE *** IS EITHER TF, TDF, OR ADF.

THIS ROUTINE ADDS THE EXTENSION *** TO THE TOTFD FILE TO
CREATE THE TUPLE FILE (TF) DESCRIPTOR. THE ALPHA DATA FILE (ADF),
OR THE TUPLE DESCRIPTOR FILE (TDF). THE APPROPRIATE FILE: EITHER
TF, ADF, OR TDF. IS THEN CREATED AS A CONTIGUOUS FILE OF SIZE
NSZ*** 256 BYTE SECTORS.
EMAPIR (IARRAY, N)

START

DIMENSION IARRAY. SET J = 0

INCORRC J. IARRAY(J) = 0

UNTIL (J.EQ.N)

RETURN

TFCRE; ADFCRE; TDFCRE

XXXCRE (TFD, JUKEY, IRKEY, N52BAR, ISTAR)

START

DEFINE VARIABLES AND COMMON BLOCKS

F D, T002F, -A

EQUIVALENT FD TO T00FD

UNTIL (J.EQ.LS)

RETURN

KMK IS EITHER TF, ADF, OR TDF. THESE ARE THREE IDENTICAL SUBROUTINES USED TO CREATE THE APPROPRIATE FILES.

S-8
SUBROUTINE RELFC.FLC

SYNTAX: CALL RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)

THIS ROUTINE IS TITLE THE "RELATIONAL FILE CREATOR" AND IT'S PURPOSE IS TO CREATE ALL DATABASE AND TEST MATRIX STORAGE FILES THE INITIAL TIME. THE VARIABLE "IFATAL" IS THE FILE CREATION RETURN STATUS INDICATOR. IF "IFATAL"=0 THEN THERE IS NO ERROR. IF IT EQUALS 1 THEN THERE IS NOT ENOUGH ROOM ON THE STORAGE VOLUME.
SUBROUTINES TFDEL.FLC, ADFDEL.FLC, AND TDFDEL.FLC

SYNTAX: CALL ***DEL(TOTFD,IWKEY,IRKEY,IST***)
WHERE *** EQUALS EITHER TF, ADF, OR TDF.

THESE ROUTINES ADD THE EXTENSION *** TO THE TOTFD FILE TO CREATE THE TUPLE FILE (TF), THE ALPHA DATA FILE (ADF), OR THE TUPLE DESCRIPTOR FILE (TDF) DESCRIPTOR. THE APPROPRIATE FILE IS THEN DELETED.
TFDEL (TOTFD, IWKEY, ERKEY, ISTTFF)
ADFDEL (TOTFD, IWKEY, ERKEY, ISTTFF)
TDFDEL (TOTFD, IWKEY, ERKEY, ISTTDF)

START

EQUIVL
(FD, TOTFD, LW)
EQUIVALENCE FD TO TOTFD

INITLILI
(FD, KNL, LW)
INITIALIZE FILE DESCRIPTOR TO ZEROS

CIRIIIR
(FD, EXT)
ADD EXTENSION

J = 0

J = J + 1
K = J + 1
FD XXX (J) = FD (K)

UNTIL (J .EQ. 15)

FILEDEL
(FD XXXX, IWKEY, ERKEY, ISTTXXX)

RETURN
SUBROUTINE RELFD.FLC

SYNTAX: CALL RELFD(IDELST,NAMFIL)

THIS ROUTINES TITLE IS "RELATIONAL FILES DELETER. IT DELETES ALL RELATIONAL DATABASE AND TEST MATRIX FILES ALL AT ONCE. "IDELST" IS A RETURN STATUS INDICATOR. IF IDELST EQUALS 0 THEN THERE IS NO ERROR. IF IDELST EQUALS 1 THEN THERE IS AN UNKNOWN ERROR. THE VARIABLE "NAMFIL" IS THE NAME OF THE RELATION TO BE DELETED. THE FORMAT FOR THIS VARIABLE IS DEFINED IN THE SUBROUTINE " NAME ".

THIS ROUTINE IS PRESET TO ALLOW THE USER THREE TRIES TO DELETE THE FILE. IF THE USER IS NOT SUCCESSFUL IT IS ASSUMED THAT HE EITHER DOES NOT KNOW WHAT HE IS DOING OR HE IS TRYING TO DELETE A FILE HE SHOULD NOT BE DELETING. AFTER THREE TRIES THE JOB IS TERMINATED.
RELD (FILE setStatus) = 1, 2, 3
**START**

**DEFINE VARIABLES AND COMMON BLOCKS**

**INITIALIZE FILE DELETE indicators**
- IFITF = 20
- ISTDTP = 20

**FILE DESC**
- IFITF, IWKEY, IKEY, ISTDTP

**IF**
- (ISTITF.EQ.0) THEN
  - TDFDEL: (TOITFD, IWKEY, IKEY, ISTDTP)

**IF**
- (ISTITDF.EQ.0) THEN
  - ADFDEL: (TOITFD, IWKEY, IKEY, ISTDTP)

**WRITE VOLUME**
- T
  - ISTDTP.EQ.52
    - ERROR

**WRITE FILENAME**
- T
  - ISTDTP.EQ.314
    - ERROR

**WRITE PROGRAM**
- T
  - ISTDTP.EQ.56
    - ERROR

**WRITE FILE**
- T
  - ISTDTP.EQ.61
    - ERROR

**WRITE UNK**
- T
  - ISTDTP.EQ.50
    - ERROR

**FILE DELETE**
- IF: IFITF.EQ.1: ERROR: ADF

**WRITE FILE**
- T

**RETURN**
SUBROUTINE EQUIVI.FLC

SYNTAX: CALL EQUIVI(ARRAY1, ARRAY2, N)

THIS ROUTINE EQUIVALENCES THE FIRST N ELEMENT OF INTEGER ARRAY1 TO INTEGER ARRAY2
SUBROUTINES OPNTF.FLC, OPNADF.FLC, AND OPNTDF.FLC.

SYNTAX: CALL OPN*** (TOTFD, IWKEY, IRKEY, ISO***, LU***)
WHERE *** EQUALS TF, ADF, OR TDF.

THESE ROUTINES OPEN THE TUPLE FILE (TF), THE ALPHA DATA FILE (ADF), OR THE TUPLE DESCRIPTOR FILE (TDF) FOR THE RELATION NAMED IN THE TOTFD.

********** WARNING ****** WARNING **********

ALL FILES ARE OPENED AS SHAREABLE READ/WRITE (i.e., IAP-4)

TF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTF.
ADF FILES ARE LOCATED ON LOGICAL UNIT LU-LUADF.
TDF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTDF.
IST IS AN ERROR INDICATOR. FOR MORE INFORMATION REFER TO THE FTN LIB REAL TIME EXTENSION MANUAL.
**EQUIV (AF1, A2, ..., A,N)**

```
START
DEFINE VARIABLES AND COMMON BLOCKS
(PRINT, TOFFD, E0)
EQUIVALENCE (A) TO TOFFD
```

**SETUP FILES AS SIMILAR (FILE WRITE (SPN))**

```
(INP1, LIN, (FD0, LI0)
INITIALIZE FILE DESCRIPTOR TO ZERO.
```

**INDEX (FD0, EXT)**

```
ADD AND EXTENSION
```

**COUNT**

```
IF (INDEX.EQ.0)
```

**FILEK**

```
(INP1, NOMATCH)
```

**IF (INDEX.EQ.1)**

```
IF (NOMATCH.EQ.1)
```

**IS OPENED PROPERLY, ERROR AT**

```
RETURN
```

**OPEN (TDFD, INKEY, INKEY, IS, IXL, E0)**

```
J = J + 1
K = J + 1
FD0(J) = FD(K)
```

**UNPL (J.EQ.15)**

```
RETURN
```

**RETURN**

**FILENAME (TDFD, FD0, FD0, FD0, FD0, FD0)**

**NAME (TDFD, FD0, FD0, FD0)**

**NAME (TDFD, FD0, FD0)**

**NAME (TDFD, FD0)**

**NAME (TDFD)**

```
**XXX = TDF, TDF, OR ADF.**
```

---

**PAGE 7**
SUBROUTINE FILERR.FLC

SYNTAX: CALL FILERR(ISTATUS,NOMATCH)

THIS ROUTINE IS USED TO DIAGNOSE BASIC FILE ERROR CODES. "ISTATUS" IS THE INPUT VARIABLE CONTAINING THE ERROR STATUS CODE. "NOMATCH" IS A RETURN VARIABLE THAT, IF EQUAL TO 1 INDICATES THAT THE ERROR IS UNKNOWN AND THE USER SHOULD CONSULT THE SYSTEM OPERATOR. IF "NOMATCH" DOES NOT EQUAL 1 THEN THE APPROPRIATE ERROR MESSAGE IS PRINTED ON THE USERS CONSOLE.
FILEREC (STAT, NOMATCH)

START

NOMATCH = Ø

(T (I:STAT .EQ. 50)
  'READ/WRITE KEY PROTECTION CODE ERROR'

(T (I:STAT .EQ. 315)
  'FILE DOES NOT EXIST'

(T (I:STAT .EQ. 59)
  'SPECIFIED VOLUME NOT MOUNTED'

(T (I:STAT .EQ. 62)
  'INVALID FILENAME / DESCRIPTOR'

NOMATCH = 1

RETURN
SUBROUTINE NAME.FLC

SYNTAX: CALL NAME(I)

THIS ROUTINE REQUESTS THE USERS FILE NAME AND THEN RECONSTRUCTS IT INTO A SUITABLE FORM FOR A FEASIL FILE NAME. THE FILE NAME CAN BE A MAXIMUM OF FORTY-TWO (42) CHARACTERS LONG AND THE FIRST CHARACTER MUST BE ALPHABETIC. THE NAME(I) IS RETURNED IN AN INTEGER ARRAY FORMAT WITH A MAXIMUM OF THIRTEEN ELEMENTS. THE ELEMENTS ARE:

I(1) = NUMBER OF CHARACTERS IN THE NAME,
I(2) = THE FIRST 4 CHARACTERS OF THE FILE NAME,
I(3) = THE SECOND 4 CHARACTERS OF THE FILE NAME,
          ⋮
I(13) = THE TWELFTH 4 CHARACTERS OF THE FILE NAME.
NAME (I)

START

DEFINE VARIABLES AND COMMON BLOCKS

GET @, NENW0, ENDLIF

CALL (VENVW0, NAMFIL, LEN, FDIM)

IF (LEN .GE. 42) F

T

*201 'NAME TO LONG - 42 CHARACTERS MAX - MUST NOT BE LONGER THAN THIS.'

*201

GETCH (NAMFIL(L2), I, CH)

ITYPE = CNTYPE(CH)

IF (ITYPE .NE. 2) F

T

FIRST LETTER MUST BE ALPHABETIC - TRY AGAIN!

EOQUI (I, NAMFIL, 15)

RETURN
TPFILECR
PROCEDURES ASSOCIATED
WITH SUBROUTINE RELFC
PAGE 11

1. SPECIFIED
VOLUME IS NOT
MOUNTED - CORRECT
AND START OVER;

IFATAL = 2

RETURN

2. NOT ENOUGH
VOLUME SPACE
TO STORE
RELATION.

IFATAL = 1

RETURN

3. UNKNOWN
ERROR HAS
OCCURRED - SEE
SYSTEM
PROGRAMMER.

IFATAL = 1

RETURN

TO WRITE:
VOLUME-ERROR

TO WRITE:
SIZE-ERROR

TO WRITE:
UNK-ERROR

TO WRITE:
FILENAME-ERROR

TO WRITE:
DEVICE-ERR

TO WRITE:
FILE NOT
EXISTS ON
SPECIFIED
VOLUME. RENAME
OR CHANGE VOLUME.

IFATAL = 2

RETURN

TO WRITE:
VOLUME IS NOT
A DIRECT-ACCESS
DEVICE!!

IFATAL = 2

RETURN

8-23/(8-24 blank)
Subroutine: F710

This routine is a collection of machine dependant subroutines that are concerned with file manipulation (opening, closing, copying, etc.). Those shown here are used on the VAX (FAST). The subroutines are:

- FILEOPN - opens a file
- FILEDEL - deletes a file
- CREATFIL - creates a file
- FILEREN - renames a file
- FILECLS - closes a file
- VAXERR - gives the meaning of VAX error codes.
What is the error code?

Start

NFOUND = 0

NFOUND = 1

undefined

unknown

else

T-5/(T-6 blank)
END 5 87 DTIC