1985

Data base documentation and editor (DB-ADE) /

Robert Allen Branch
Lehigh University

DATA BASE DOCUMENTOR AND EDITOR (DB-ADE)

by

Robert Allen Branch

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1985

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 8, 1985
(date)

John C. Wiginton
Professor in Charge

S. C. Kane
Chairman of Department

ACKNOWLEDGEMENTS

The author wishes to gratefully acknowledge the assistance and direction provided by his thesis advisor, Professor John C. Wiginton, Department of Industrial Engineering, Lehigh University.

The author also wishes to express a special thanks to his loving wife Janet, for her constant encouragement during his pursuit of this thesis work.

- TABLE OF CONTENTS -

- LIST OF FIGURES -

DATA BASE DOCUMENTOR AND EDITOR (DB-ADE)
I. Abstract

I. Abstract.

A normality in early development phases of data base application systems is the knowledge of the specific data base management system (DBMS) on which the application will be developed. Application data base accesses (logical transactions) are therefore often documented in a syntax form similar to the specific DBMS's data manipulation language (DML) commands.

Logical transaction definitions, if specified in DML-like syntax forms, serve as templates for application program design, since they represent requirements for sequencing of data base accesses that efficiently execute data updating and retrieval requirements. The analytical path taken from initial logical transaction definition to program DML specification is shown in Figure 1.

The goal of the process is to insure that application programs properly and efficiently execute against the physical data base structure while supporting the transactional requirements of the system. Quality assurance concerns should fuel updating of the transaction definition documentation when physical data base design changes impact trans-

action logic, and suggest maintaining transaction definition documentation in close-to program DML format to minimize errors when translating transaction requirements into program design.

This thesis research delivers a PC-based system that provides 1) on-line updating of data base structure and transaction definitions, and 2) proofing of the transaction definitions against a pre-defined DML syntax and the physical data base structure with which the transaction interacts. The system, known as the Data Base Documentor and Editor (DB-ADE), provides analytical tools that support the documentation and feedback needs of the previously-described design process. The general DB-ADE system structure and the manner in which DB-ADE supports design feedback are shown in Figure 2.

DB-ADE is programmed in Borland International's Turbo Pascal[1] and utilizes Borland International's Turbo-ISAM Access System[2] for data file management. DB-ADE runs under

[1] Turbo Pascal and [2] Turbo Access System are copyrighted by Borland International, Inc., 4113 Scotts Valley Drive, Scotts Valley, California, 95066.

DATA BASE DOCUMENTOR AND EDITOR (DB-ADE)
I. Abstract

MS-DOS and PC-DOS, and executes in 64K of main memory. DB-ADE contains utilities that assist its setup on a PC and assist in establishing and maintaining DB-ADE data files.

DB-ADE supports application development on Cullinet's IDMS data base management system. DB-ADE's Data Base Definition Subsystem defines IDMS-type network data base structures, and the Transaction Definition Subsystem uses a syntax that emulates the IDMS data manipulation language.

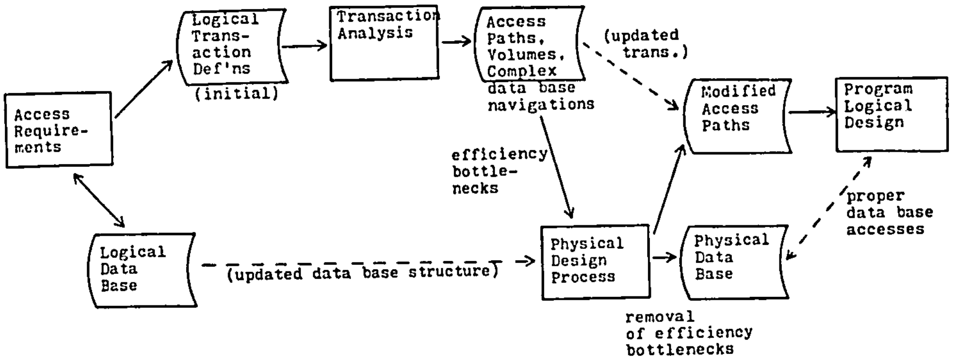


Figure 1

Initial Logical Transaction Definition to
Program DML Specification Analyses

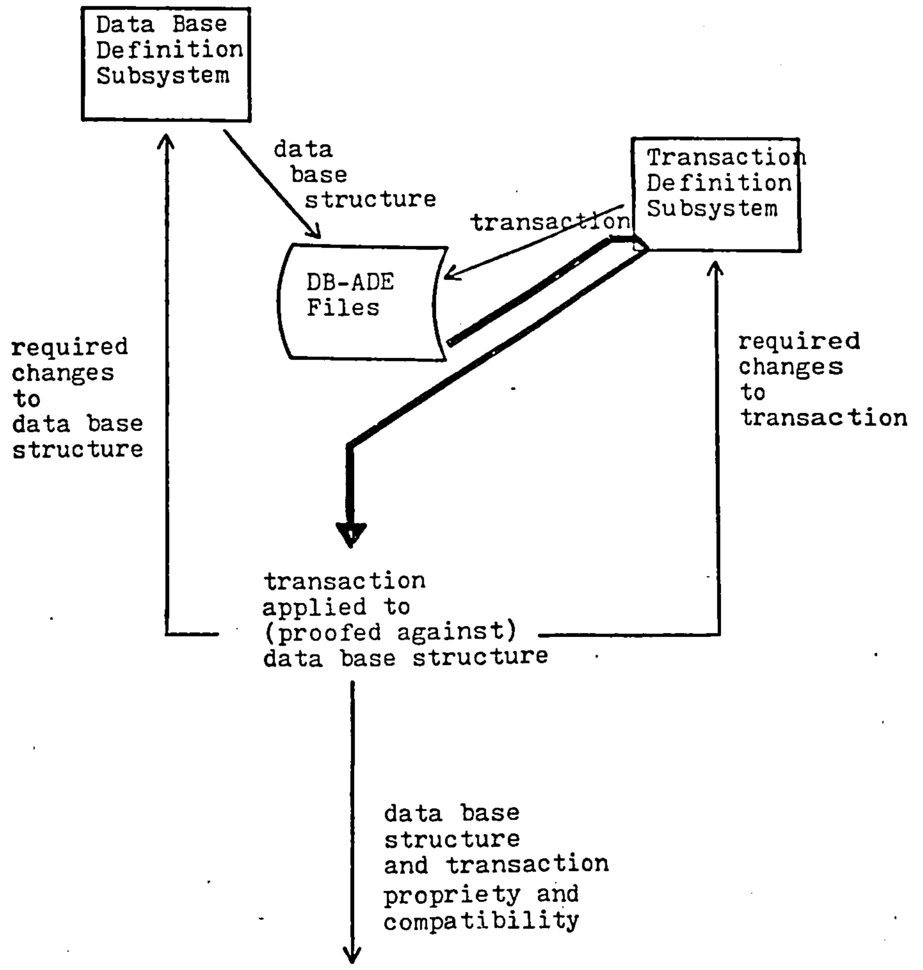

Data Base Definition Subsystem
data base structure
Transaction Definition Subsystem
transaction
DB-ADE Files
required changes to data base structure
required changes to transaction
transaction applied to (proofed against) data base structure
data base structure and transaction propriety and compatibility

Figure 2

General DB-ADE System Structure

II. Organization of the Thesis Document.

The remainder of this thesis document is organized as follows:

Functional Structure of the DB-ADE System, Section III, provides a functional description of the DB-ADE system, and details the functionality of the DB-ADE Transaction Definition Subsystem, which is the hub of the DB-ADE system. This description also explains how DB-ADE validates an inputted transaction.

DB-ADE Transaction Structure and Syntax, Section IV, provides knowledge of how to define a transaction to DB-ADE and how to interpret and react to the results provided by the Transaction Proof Subsystem.

The DB-ADE Program/PC Environment, Section V, identifies all DB-ADE program run modules, data files, and utilities, and explains how to install DB-ADE and perform DB-ADE file management.

Keyboard Control Input Characters, Section VI, documents keyboard characters used rather consistently within the DB-ADE system.

DB-ADE Screen Field Descriptions, Section VII, defines input field requirements for the DB-ADE input screens.

DB-ADE Error and Warning Messages Listing, Section VIII, lists alphabetically DB-ADE system error and warning messages and their meanings.

Location of DB-ADE Demonstration System, Section IX, identifies where program source code listings and DB-ADE program run module and demonstration diskettes have been filed in fulfillment of thesis requirements.

Directions for Further Research, Section X, identifies the author's current thoughts on possible future extensions of DB-ADE system functionality.

III. Functional Structure of the DB-ADE System.

The functional structure of the DB-ADE system is shown in Figure 3. DB-ADE is comprised of a main menu and three subsystems:

- the Data Base Definition Subsystem;
- the Transaction Definition Subsystem;
- the Entity Lists Subsystem.

These subsystems are in turn comprised of specific programs as follows:

Data Base Definition Subsystem --

- Data Base Identification Program;
- Area Definition Program;
- Record Definition Program;
- Set Definition Program.

Transaction Definition Subsystem --

- Transaction Maintenance Program;
- Transaction Proof Program.

Entity Lists Subsystem --

- List Data Base Identifiers Program;
- List Areas For A Data Base Program;
- List Records For A Data Base Program;
- List Sets For A Data Base Program;
- List Transactions For A Data Base Program.

The functionality of the main menu, each of these subsystems, and the subsystem program components are described in the sections that follow.

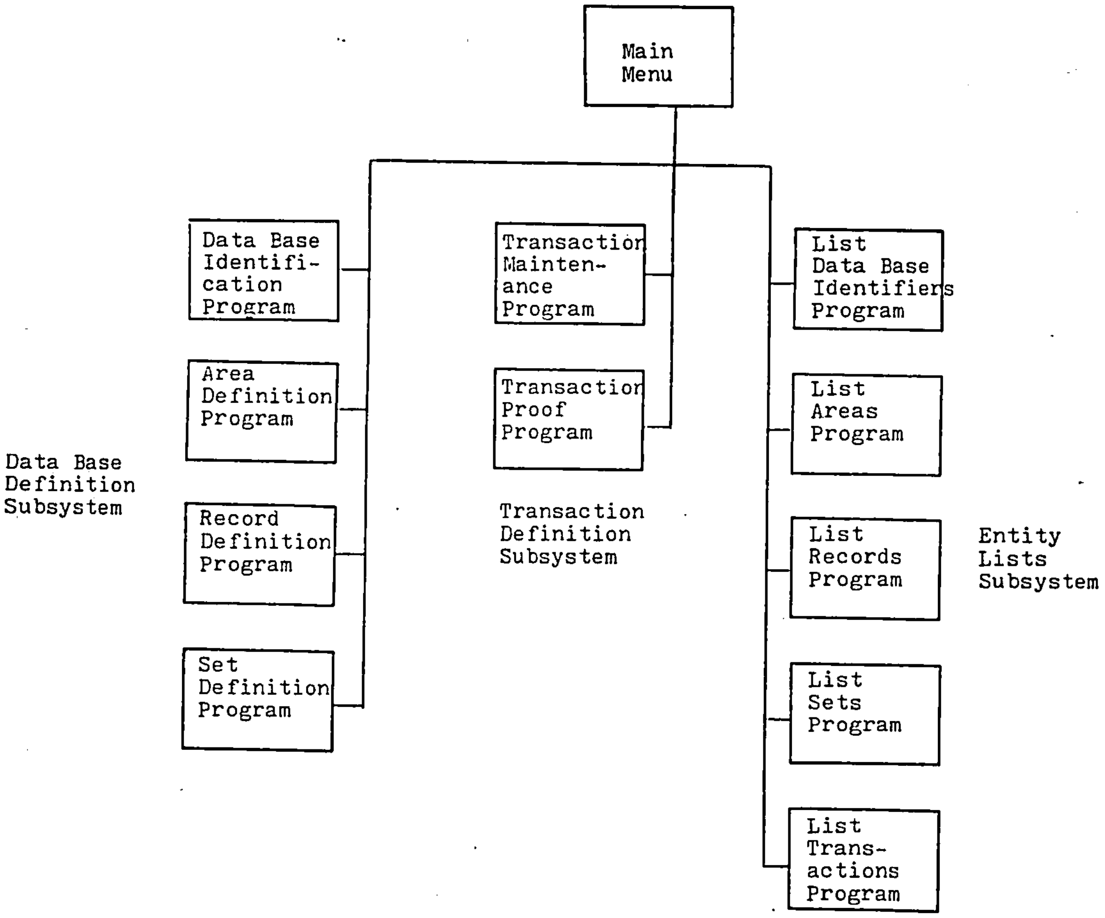

Main Menu
Data Base Identification Program
Area Definition Program
Record Definition Program
Set Definition Program
Data Base Definition Subsystem
Transaction Maintenance Program
Transaction Proof Program
Transaction Definition Subsystem
List Data Base Identifiers Program
List Areas Program
List Records Program
List Sets Program
List Transactions Program
Entity Lists Subsystem

Figure 3

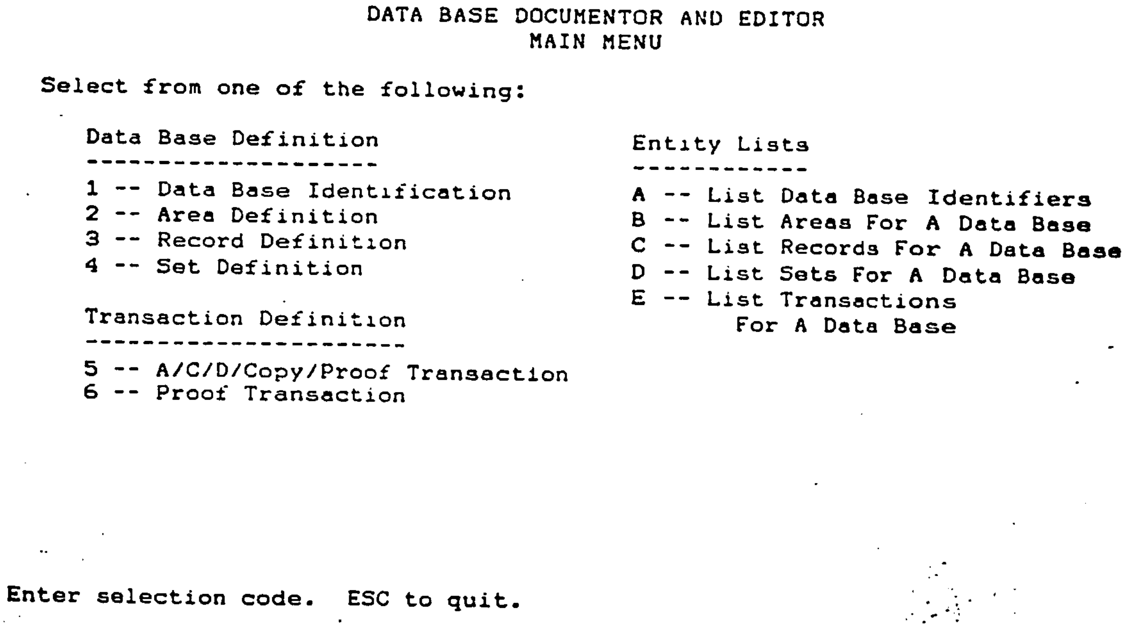
DB-ADE Functional Structure

A. The DB-ADE Main Menu.

The DB-ADE Main Menu Screen is shown in Figure 4. Each program within the DB-ADE system can be accessed by entering a keyboard character shown on the menu lists. If an Entity Lists Subsystem program is chosen (except for the List Data Base Identifiers Program), the Main Menu Program will request that a data base identifier be entered to denote the identifier of the data base structure for which the entity list is to be obtained (see Figure 5). If selection "6 - Proof Transaction" is selected, the Main Menu Program will request that a data base identifier and a transaction identifier be entered to denote the transaction that is to be proofed against its data base structure (see Figure 6).

B. Data Base Definition Subsystem.

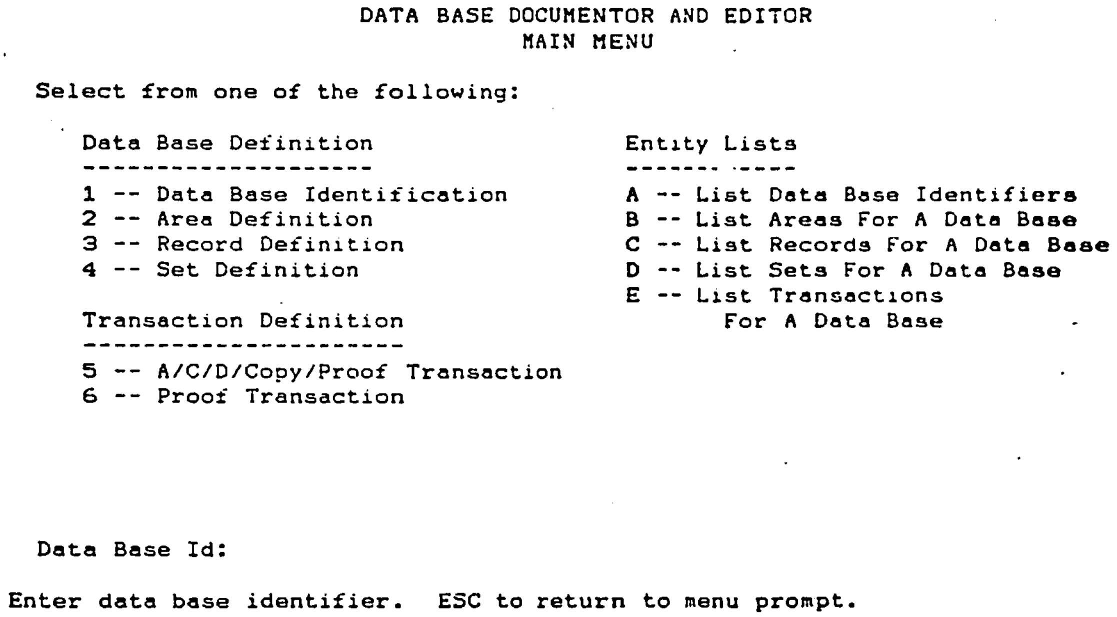
The Data Base Definition Subsystem defines the details of the physical IDMS data base structures within the DB-ADE system files. All entities within the DB-ADE system (i.e., all areas, records, sets, and transactions) are qualified by a twelve-character data base identifier (established in the Data Base Identification Program), thus allowing for storage of the physical characteristics of multiple IDMS



```
                        DATA BASE DOCUMENTOR AND EDITOR
                                  MAIN MENU

 Select from one of the following:

   Data Base Definition                  Entity Lists
   --------------------                  ------------
   1 -- Data Base Identification         A -- List Data Base Identifiers
   2 -- Area Definition                  B -- List Areas For A Data Base
   3 -- Record Definition                C -- List Records For A Data Base
   4 -- Set Definition                   D -- List Sets For A Data Base
                                         E -- List Transactions
   Transaction Definition                      For A Data Base
   ----------------------
   5 -- A/C/D/Copy/Proof Transaction
   6 -- Proof Transaction




Enter selection code.  ESC to quit.
```

Figure 4

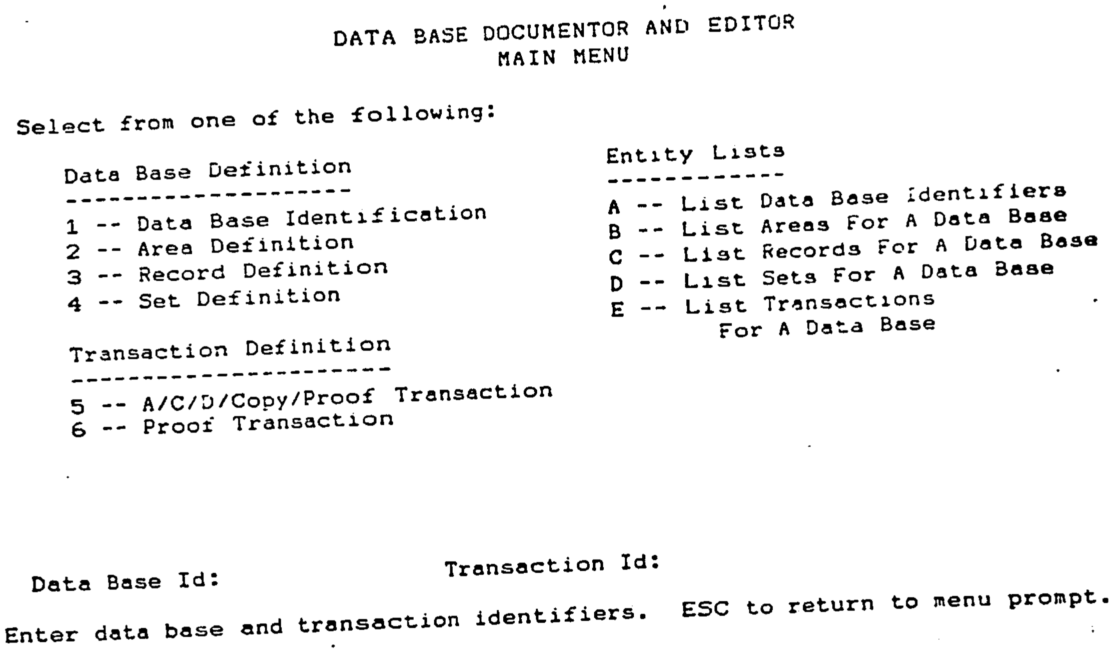
DB-ADE Main Menu Screen



```
                      DATA BASE DOCUMENTOR AND EDITOR
                                 MAIN MENU

 Select from one of the following:

    Data Base Definition                  Entity Lists
    --------------------                  ------- ----
    1 -- Data Base Identification         A -- List Data Base Identifiers
    2 -- Area Definition                  B -- List Areas For A Data Base
    3 -- Record Definition                C -- List Records For A Data Base
    4 -- Set Definition                   D -- List Sets For A Data Base
                                          E -- List Transactions
    Transaction Definition                       For A Data Base
    ----------------------
    5 -- A/C/D/Copy/Proof Transaction
    6 -- Proof Transaction


 Data Base Id:

Enter data base identifier.  ESC to return to menu prompt.
```

Figure 5

DB-ADE Main Menu Requests Data Base Identifier



```
                  DATA BASE DOCUMENTOR AND EDITOR
                             MAIN MENU

Select from one of the following:
                                           Entity Lists
  Data Base Definition                     ------------
  --------------------                     A -- List Data Base Identifiers
  1 -- Data Base Identification            B -- List Areas For A Data Base
  2 -- Area Definition                     C -- List Records For A Data Base
  3 -- Record Definition                   D -- List Sets For A Data Base
  4 -- Set Definition                      E -- List Transactions
                                                  For A Data Base
  Transaction Definition
  ----------------------
  5 -- A/C/D/Copy/Proof Transaction
  6 -- Proof Transaction


 Data Base Id:                Transaction Id:
Enter data base and transaction identifiers.  ESC to return to menu prompt.
```

Figure 6

DB-ADE Main Menu Requests Transaction Identifier data base structures within the DB-ADE files, and the identification of each transaction to a specific data base structure.

DATA BASE IDENTIFICATION PROGRAM

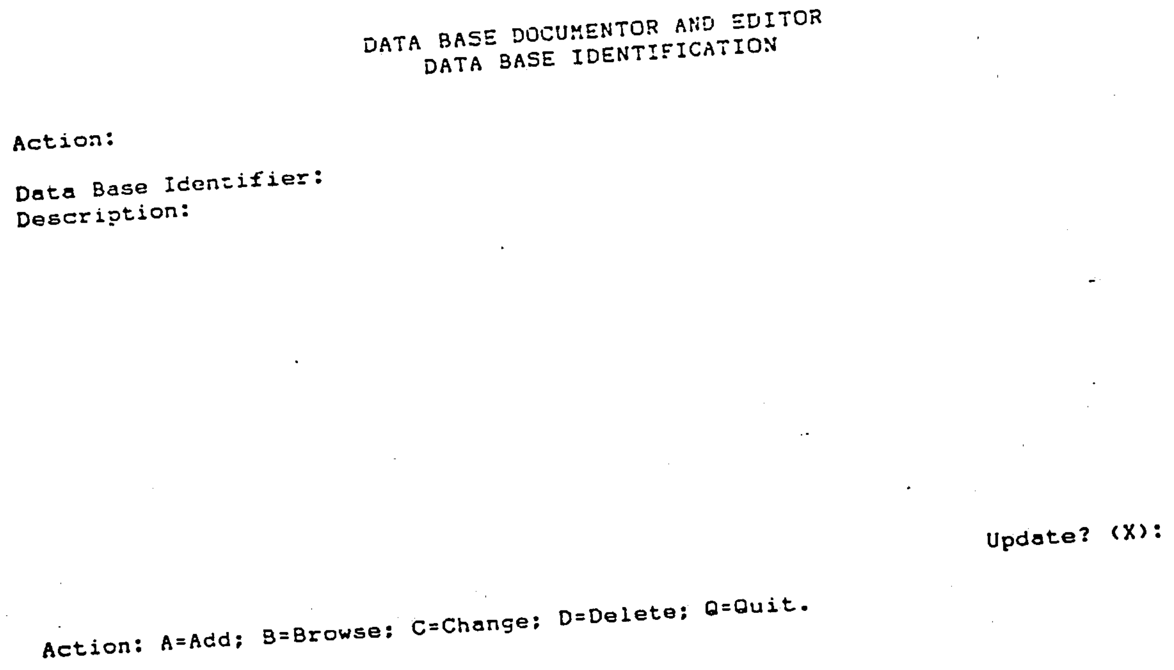
The Data Base Identification Program establishes the aforementioned data base identifier and its description. Input to this program is provided via the Data Base Identification Screen (see Figure 7).

AREA DEFINITION PROGRAM

The Area Definition Program establishes the identities of the valid IDMS areas within the data base structure that is identified by the data base identifier. Input to this program is provided via the Area Maintenance Screen (see Figure 8). The key to a DB-ADE IDMS area is the data base identifier plus a twelve-character area identifier.

RECORD DEFINITION PROGRAM

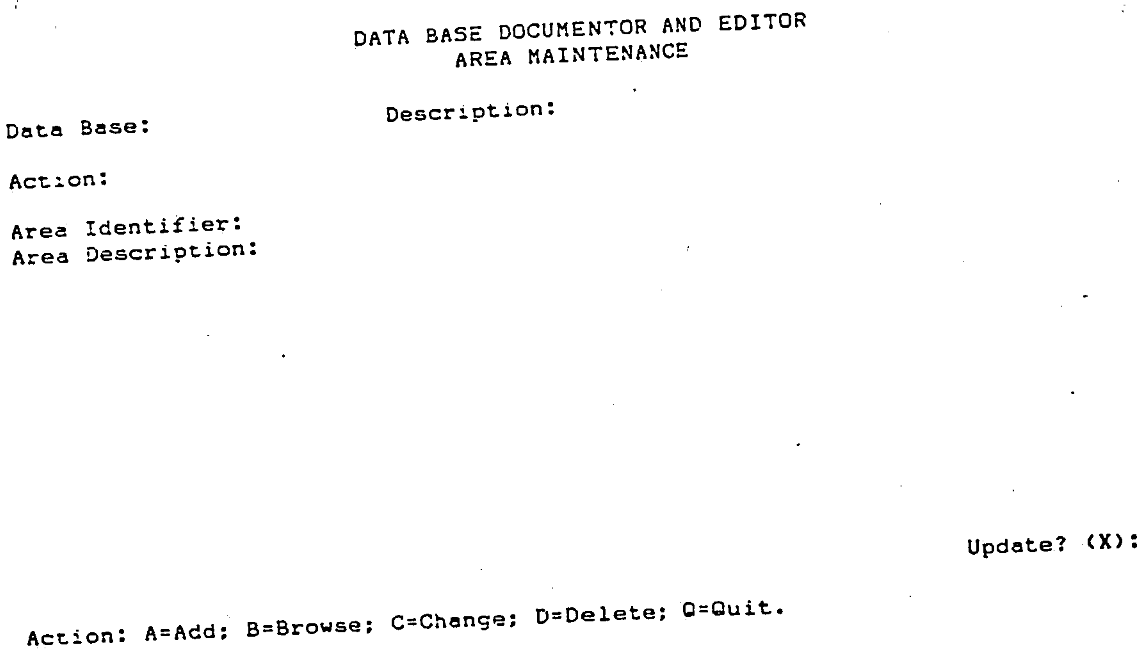
The Record Definition Program establishes the identifiers for and attributes of the IDMS records within the data base structure that is identified by the data base identifier. Input to this program is provided via the



```
                  DATA BASE DOCUMENTOR AND EDITOR
                     DATA BASE IDENTIFICATION

Action:

Data Base Identifier:
Description:




                                                        Update? (X):

 Action: A=Add; B=Browse; C=Change; D=Delete; Q=Quit.
```

Figure 7

Data Base Identification Screen



```
                    DATA BASE DOCUMENTOR AND EDITOR
                           AREA MAINTENANCE

Data Base:              Description:

Action:

Area Identifier:
Area Description:




                                                            Update? (X):

Action: A=Add; B=Browse; C=Change; D=Delete; Q=Quit.
```

Figure 8

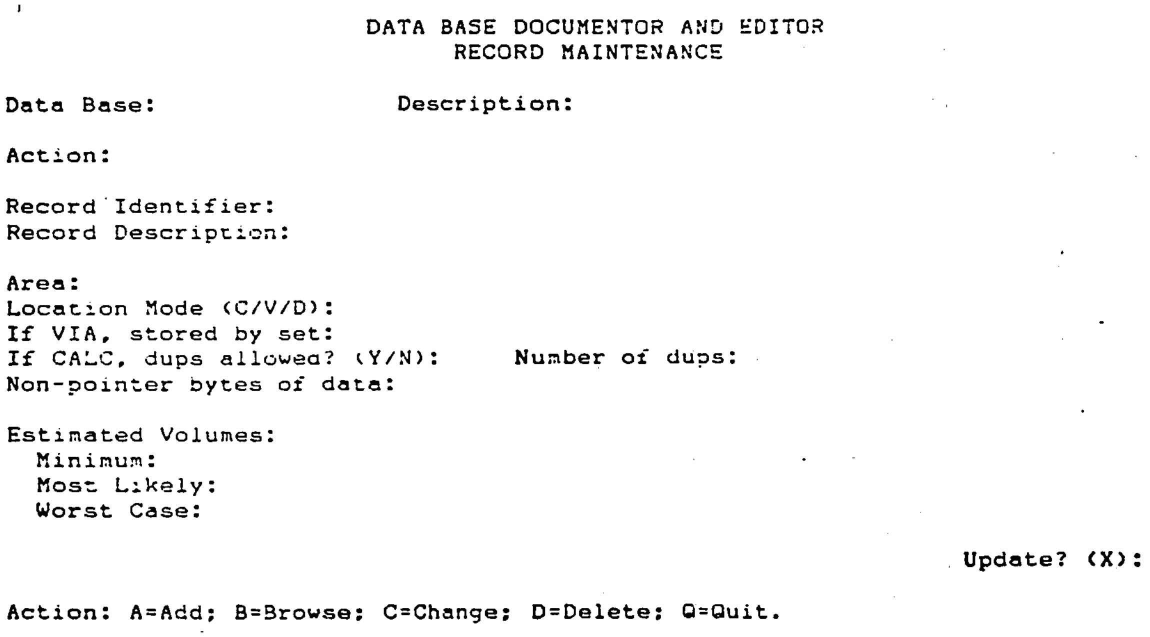
Area Maintenance Screen

Record Maintenance Screen (see Figure 9). The key to the DB-ADE IDMS record is the data base identifier plus a twelve-character record identifier. The attributes that can be established for a record identifier are:

- record description;
- the IDMS area in which the record resides in the data base structure;
- the location mode (i.e., how IDMS will store the record, e.g., CALC, VIA, DIRECT);
- the set by which the record is stored VIA (if the location mode is VIA);
- a duplicates allowed indicator (Y = yes, N = no, used with CALC location mode);
- the number of expected duplicate CALC records;
- the non-pointer bytes of data to be stored in a single occurrence of the record;
- the expected minimum number of record occurrences;
- the most likely number of record occurrences;
- the expected maximum number of record occurrences.

SET DEFINITION PROGRAM

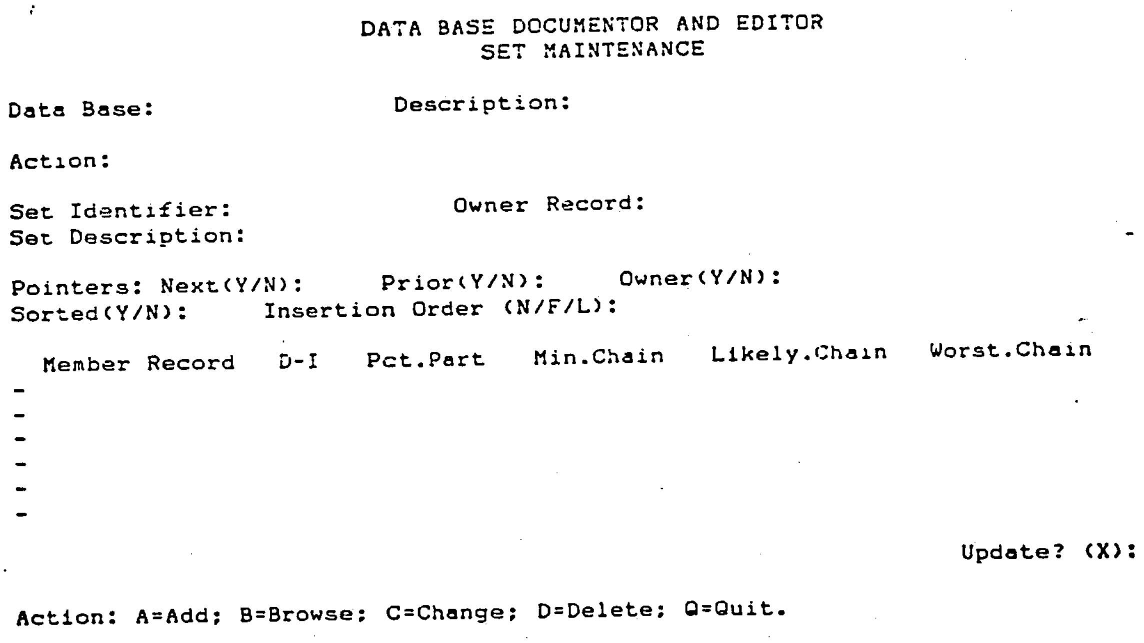
The Set Definition Program establishes the identifiers for and attributes of the IDMS sets within the data base structure that is identified by the data base identifier. Input to this program is provided via the Set Maintenance Screen (see Figure 10). The key to the DB-ADE IDMS set is the data base identifier plus a twelve-character set identifier. The attributes that can be established for a set identifier are:



```
                        DATA BASE DOCUMENTOR AND EDITOR
                              RECORD MAINTENANCE

Data Base:                  Description:

Action:

Record Identifier:
Record Description:

Area:
Location Mode (C/V/D):
If VIA, stored by set:
If CALC, dups allowed? (Y/N):     Number of dups:
Non-pointer bytes of data:

Estimated Volumes:
  Minimum:
  Most Likely:
  Worst Case:
                                                                Update? (X):

Action: A=Add; B=Browse; C=Change; D=Delete; Q=Quit.
```

Figure 9

Record Maintenance Screen



```
                     DATA BASE DOCUMENTOR AND EDITOR
                             SET MAINTENANCE

Data Base:                 Description:

Action:

Set Identifier:                 Owner Record:
Set Description:

Pointers: Next(Y/N):     Prior(Y/N):     Owner(Y/N):
Sorted(Y/N):     Insertion Order (N/F/L):

  Member Record   D-I   Pct.Part   Min.Chain   Likely.Chain   Worst.Chain
-
-
-
-
-
-
                                                            Update? (X):

Action: A=Add; B=Browse; C=Change; D=Delete; Q=Quit.
```

Figure 10

Set Maintenance Screen

- set description;
- the record that owns the set;
- the set member record insertion order (e.g., LAST, FIRST, SORTED);
- a set sorted indicator (Y = set is sorted, N = set is not sorted);
- a set next pointers indicator (Y = set has next pointers, N = set does not have next pointers);
- a set prior pointers indicator (Y = set has prior pointers, N = set does not have prior pointers);
- a set owner pointers indicator (Y = set has owner pointers, N = set does not have owner pointers);
- set member record identifiers (up to six member records can be specified);
- deletion-insertion rules for each member record (i.e., MA, OA, MM, OM);
- the percentage of each member record's occurrences that typically participate in the set;
- the expected minimum number of member record occurrences that will participate in a typical set chain;
- the most likely number of member record occurrences that will participate in a typical set chain;
- the expected maximum number of member record occurrences that will participate in a typical set chain;

C. Transaction Definition Subsystem.

The Transaction Definition Subsystem allows the DB-ADE user to build and edit IDMS transactions, and allows him to proof these transactions for proper syntax, structure, and compliance with IDMS currency laws when executing against the IDMS structure defined in the Data Base Subsystem.

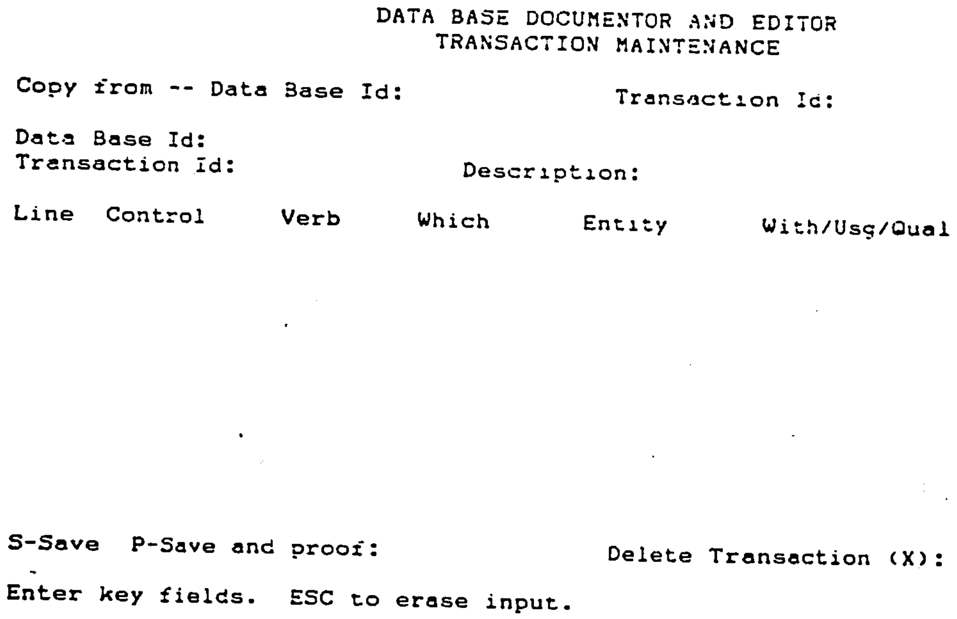
TRANSACTION MAINTENANCE PROGRAM

The primary purpose of the Transaction Maintenance Program is to serve as a tool in building and editing the transactions that the DB-ADE user wishes to proof against data base structures defined in the Data Base Definition Subsystem. Input to this program is provided via the Transaction Maintenance Screen (see Figure 11).

The key to a DB-ADE transaction is the data base identifier plus a twelve-character transaction identifier. The qualifying data base identifier always denotes the data base structure that is stored in DB-ADE against which the transaction is to be proofed.

Up to two hundred (200) transaction statements (lines) can be supplied within a DB-ADE transaction. The screen at any time displays only twelve (12) lines for view and editing. The program provides paging and cursor line movement within the transaction to allow access to all two hundred line entries.

A line may contain a comment, may be left blank, or may contain an IDMS DML-like statement (e.g., a FIND, STORE, or MODIFY statement). By using paging and cursor movement,



```
                      DATA BASE DOCUMENTOR AND EDITOR
                         TRANSACTION MAINTENANCE

Copy from -- Data Base Id:                   Transaction Id:

Data Base Id:
Transaction Id:                  Description:

Line   Control     Verb     Which       Entity        With/Usg/Qual




S-Save   P-Save and proof:                   Delete Transaction (X):

Enter key fields.  ESC to erase input.
```

Figure 11

Transaction Maintenance Screen

the DML-like statements can be ordered to define the IDMS transaction. DB-ADE requires a pre-defined syntax be used to enter these transactions.

A new transaction can be established by using the COPY FROM program feature. This feature calls up an existing transaction into the program work area and allows revisions to the transaction and the eventual storage of the revised transaction under a new identifier name, while keeping the "copied-from" transaction intact.

Transactions can be deleted from the DB-ADE files via this screen. The Transaction Proof Program can be invoked from this program to proof the transaction that is currently in the program's work area.

TRANSACTION PROOF PROGRAM

The Transaction Proof Program validates a DB-ADE transaction in a three-step process:

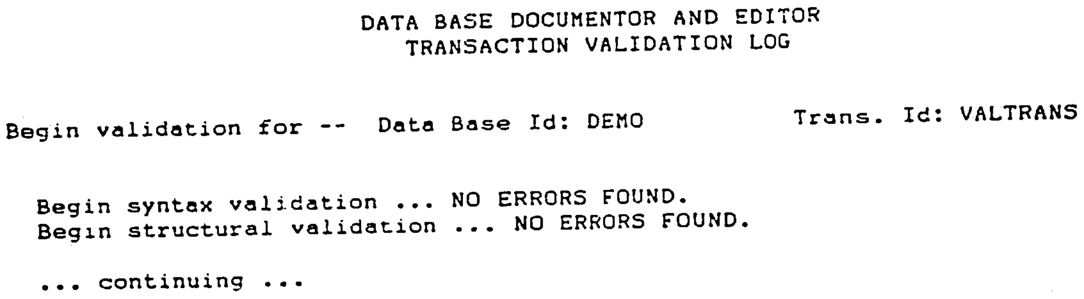
Step 1: Syntax Validation -- Each non-blank, non-comment statement is assumed to be an IDMS DML-like statement. Each statement of this type is validated against a pre-loaded table of valid syntax forms.

Step 2: Structural Validation -- If the transaction passes syntax validation, it is passed along for structural validation. The DB-ADE syntax allows for entering of control statements within the transaction that define the conditional execution of one or more DML statements. Through use of these control statements (e.g., IF, ELSE), forms of logical IF ... ELSE constructs that are typically supported by programming languages syntaxes can be built. The structural validation program module verifies the propriety of these constructs much like a programming language's compiler would (e.g., all ELSE statements must be assignable to a preceding IF statement). While proofing these conditional constructs, the program is also building tables internally that define all of the possible transaction "paths" that could be executed from the beginning statement to the ending statement of the transaction, given the conditional control statements that it encounters.

Step 3: Logical Path Validation -- Once the transaction has passed structural validation, it undergoes logical path validation. The logical path validation program module "executes" each individual transaction path that was identified by the structural validation module. This module insures that the statement order within each path is proper in that the statement order does not violate IDMS currency laws. The module is therefore simulating the interrogation and the updating of IDMS run-unit currency tables as it sequentially executes each statement within a transaction path. The module performs this simulation independently for each identified transaction path.

While the Transaction Proof Program modules are executing, a transaction validation log is written to the CRT screen (see Figures 12 and 13). Figure 12 shows the log as it is written upon successful completion of the syntax validation and the structural validation modules. Figure 13 shows the log as it is written upon successful completion of the logical path validation module. The writing of these two separate screens to the CRT results from the syntax-structural validations and the logical path validation being programmed in and therefore executed by two separate Turbo Pascal program run modules.

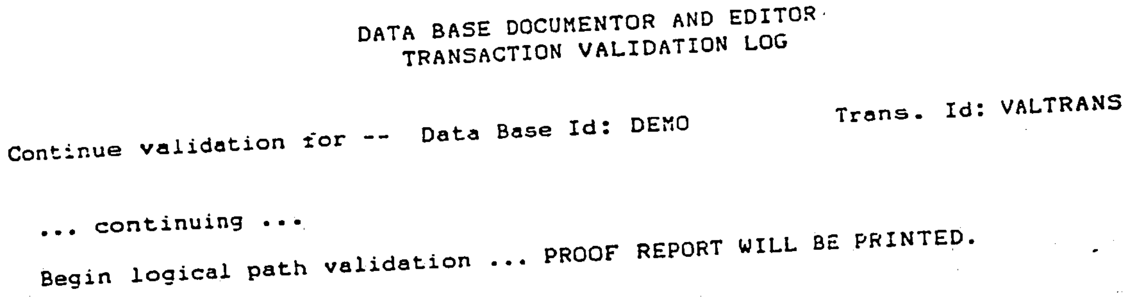
The three validation program modules are executed serially. The execution of any one of the modules is contingent upon the successful completion of the prior module.



```
                         DATA BASE DOCUMENTOR AND EDITOR
                            TRANSACTION VALIDATION LOG


Begin validation for --  Data Base Id: DEMO          Trans. Id: VALTRANS


  Begin syntax validation ... NO ERRORS FOUND.
  Begin structural validation ... NO ERRORS FOUND.

  ... continuing ...
```

Figure 12

Transaction Validation Log (1st Screen)



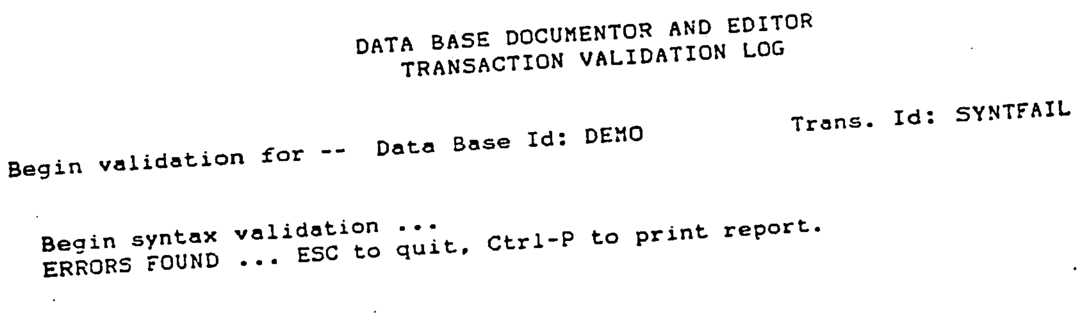
```
                 DATA BASE DOCUMENTOR AND EDITOR
                    TRANSACTION VALIDATION LOG

Continue validation for --  Data Base Id: DEMO          Trans. Id: VALTRANS

  ... continuing ...
  Begin logical path validation ... PROOF REPORT WILL BE PRINTED.
```

Figure 13

Transaction Validation Log (2nd Screen)

Thus, for example, errors found in the syntax validation module will prohibit the execution of the structural validation and logical path validation modules. When errors are detected, a message is displayed on the CRT asking the user whether he wishes to print an error report on his workstation printer or whether he wishes to leave the validation modules (see Figure 14). The hard-copy validation error report layout is shown in Figure 15. Error messages, which begin "++", always follow the transaction statement that is in error.

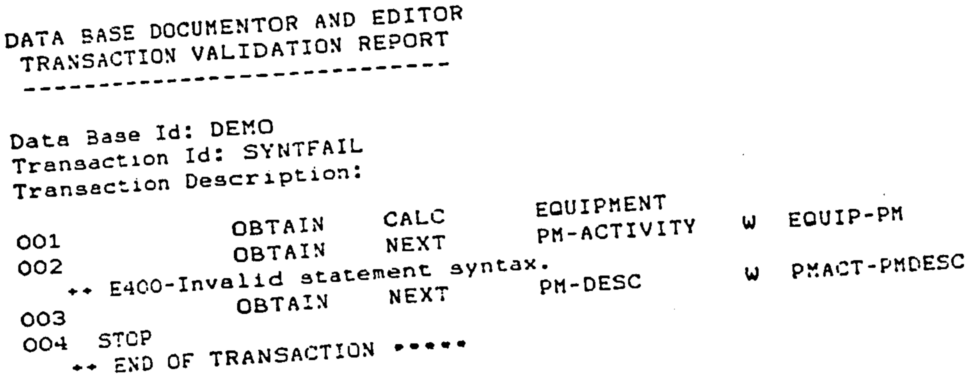
A hard-copy report detailing logical path validation results always prints upon completion of the logical path validation module. An example is shown in Figure 16. The transaction, as it was inputted, is printed first. The transaction DML statements executable under the first possible transaction path follow. Encountered currency errors and warning messages are printed using the same "++" denotation convention. Results of the logical path validation of each possible transaction path are successively printed until all paths have been exhausted.



```
                    DATA BASE DOCUMENTOR AND EDITOR
                      TRANSACTION VALIDATION LOG

Begin validation for --  Data Base Id: DEMO          Trans. Id: SYNTFAIL

  Begin syntax validation ...
  ERRORS FOUND ... ESC to quit, Ctrl-P to print report.
```

Figure 14

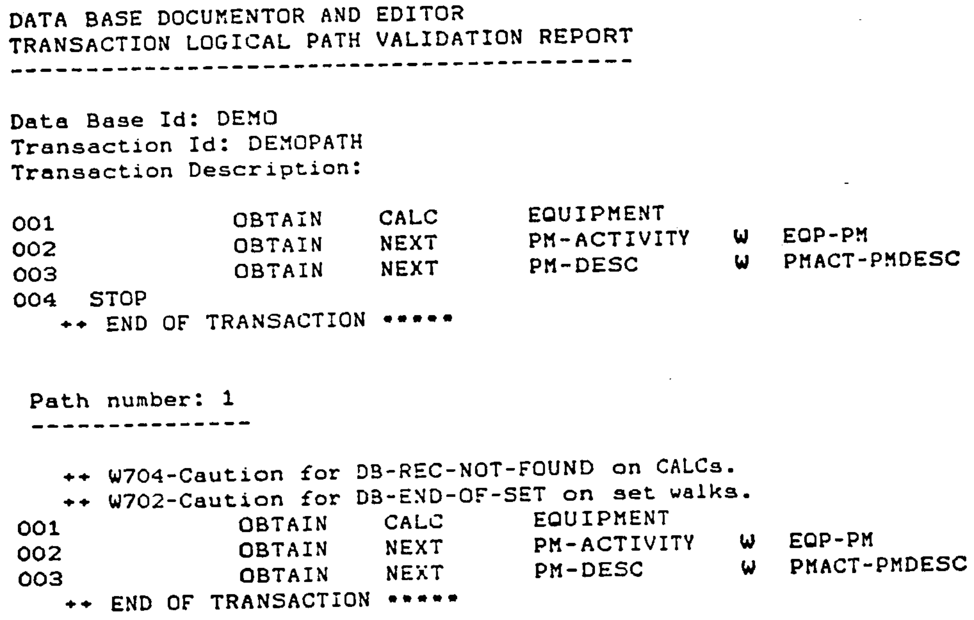
Transaction Validation Error Noted on
Transaction Validation Log



```
DATA BASE DOCUMENTOR AND EDITOR
 TRANSACTION VALIDATION REPORT
 ----------------------------

Data Base Id: DEMO
Transaction Id: SYNTFAIL
Transaction Description:

001            OBTAIN   CALC      EQUIPMENT
002            OBTAIN   NEXT      PM-ACTIVITY    W  EQUIP-PM
   ** E400-Invalid statement syntax.
003            OBTAIN   NEXT      PM-DESC        W  PMACT-PMDESC
004  STOP
   ** END OF TRANSACTION *****
```

Figure 15

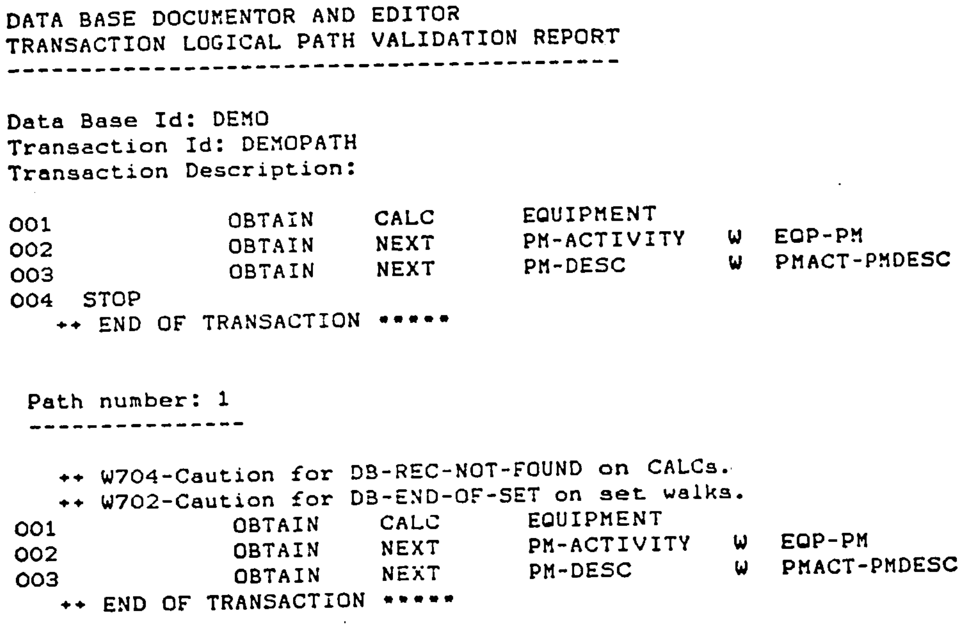
Validation Error Report



```
DATA BASE DOCUMENTOR AND EDITOR
TRANSACTION LOGICAL PATH VALIDATION REPORT
----------------------------------------

Data Base Id: DEMO
Transaction Id: DEMOPATH
Transaction Description:

001            OBTAIN    CALC      EQUIPMENT
002            OBTAIN    NEXT      PM-ACTIVITY   W   EQP-PM
003            OBTAIN    NEXT      PM-DESC       W   PMACT-PMDESC
004  STOP
   ** END OF TRANSACTION *****

 Path number: 1
 ---------------

   ** W704-Caution for DB-REC-NOT-FOUND on CALCs.
   ** W702-Caution for DB-END-OF-SET on set walks.
001            OBTAIN    CALC      EQUIPMENT
002            OBTAIN    NEXT      PM-ACTIVITY   W   EQP-PM
003            OBTAIN    NEXT      PM-DESC       W   PMACT-PMDESC
   ** END OF TRANSACTION *****
```

Figure 16

Transaction Logical Path Validation Report



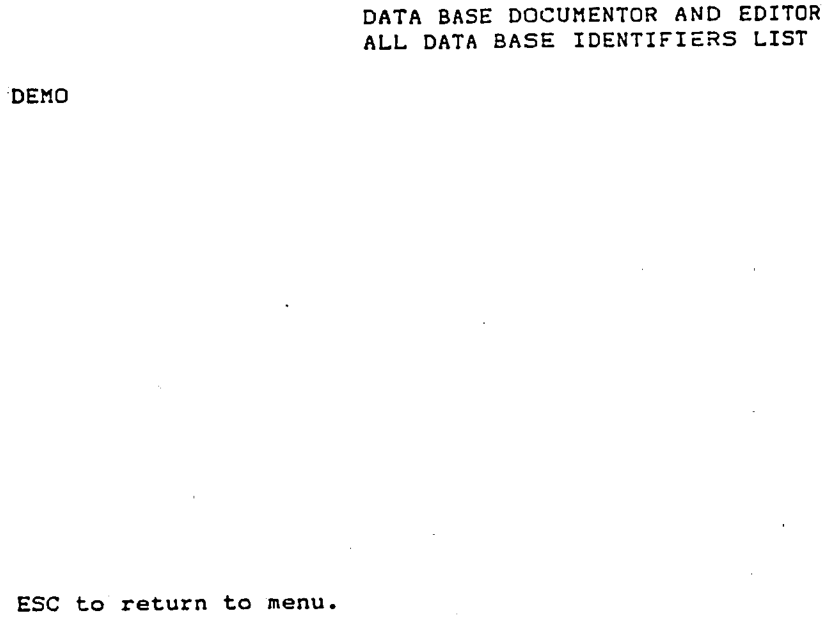
```
DATA BASE DOCUMENTOR AND EDITOR
TRANSACTION LOGICAL PATH VALIDATION REPORT
----------------------------------------

Data Base Id: DEMO
Transaction Id: DEMOPATH
Transaction Description:

001             OBTAIN    CALC      EQUIPMENT
002             OBTAIN    NEXT      PM-ACTIVITY    W   EQP-PM
003             OBTAIN    NEXT      PM-DESC        W   PMACT-PMDESC
004   STOP
   ++ END OF TRANSACTION *****

 Path number: 1
 ---------------

   ++ W704-Caution for DB-REC-NOT-FOUND on CALCs.
   ++ W702-Caution for DB-END-OF-SET on set walks.
001             OBTAIN    CALC      EQUIPMENT
002             OBTAIN    NEXT      PM-ACTIVITY    W   EQP-PM
003             OBTAIN    NEXT      PM-DESC        W   PMACT-PMDESC
   ++ END OF TRANSACTION *****
```

Figure 16

Transaction Logical Path Validation Report

D. Entity Lists Subsystem.

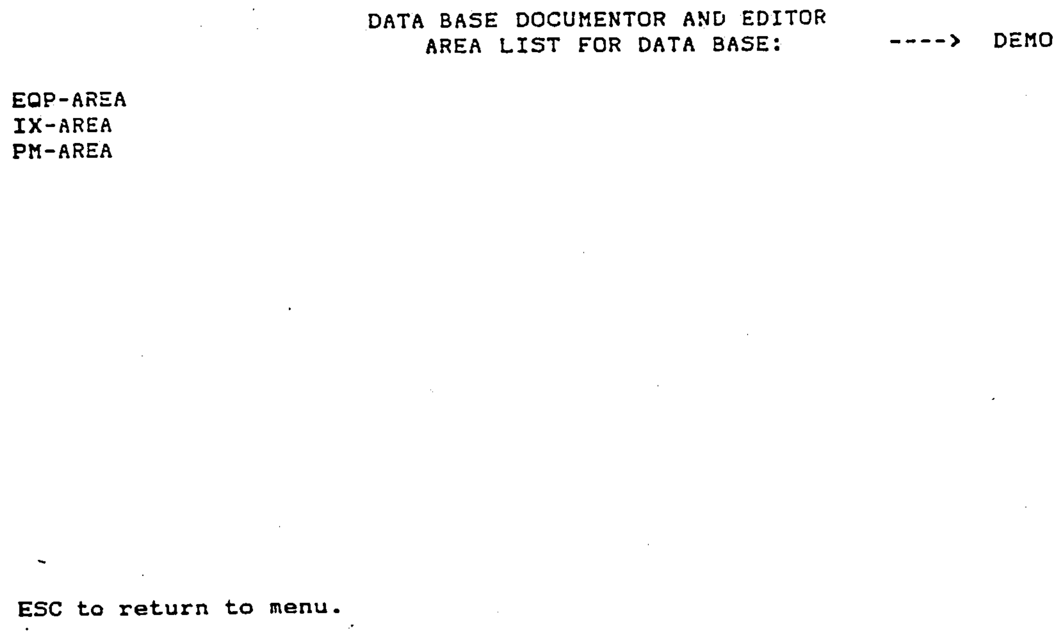
The Entity Lists Subsystem provides a "tickler" file mechanism to the DB-ADE user. The program invoked, as appropriate, lists out on the CRT screen all of the data base identifiers on the DB-ADE files or the specific area, record, set, or transaction identifiers on file for a specific data base identifier. The layout of each displayed screen is relatively the same. Examples of the screen layouts are shown in Figures 17 through 21. These examples report the components of the demonstration data base of Figure 22.



```
                              DATA BASE DOCUMENTOR AND EDITOR
                              ALL DATA BASE IDENTIFIERS LIST
DEMO









ESC to return to menu.
```

Figure 17

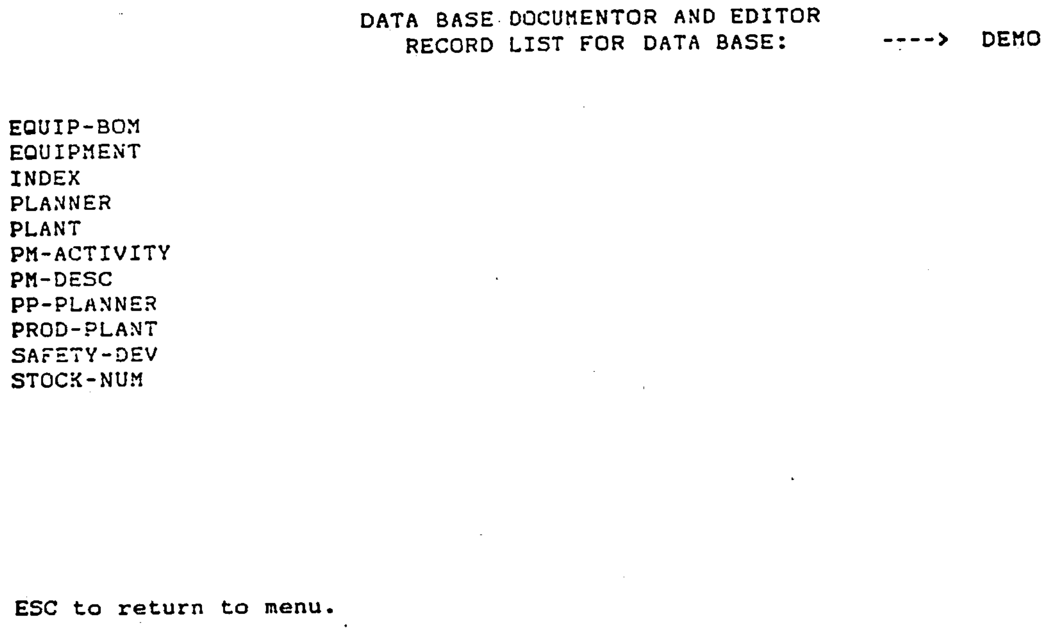
All Data Base Identifiers List Screen



```
                         DATA BASE DOCUMENTOR AND EDITOR
                            AREA LIST FOR DATA BASE:        ---->  DEMO

EQP-AREA
IX-AREA
PM-AREA




ESC to return to menu.
```

Figure 18

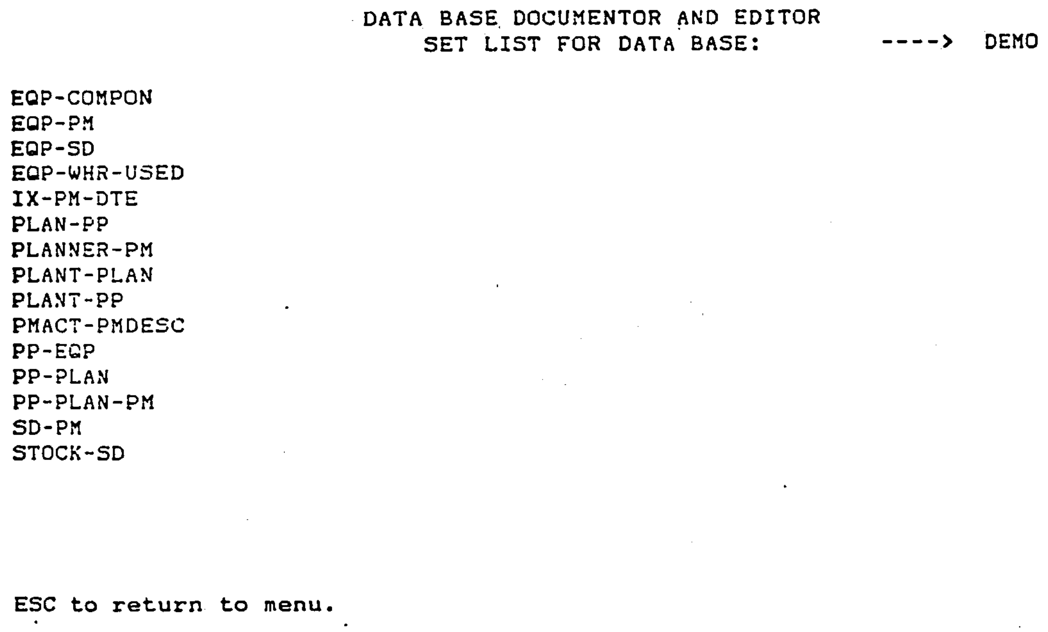
Area List for Data Base Screen



```
                    DATA BASE DOCUMENTOR AND EDITOR
                      RECORD LIST FOR DATA BASE:        ---->  DEMO

EQUIP-BOM
EQUIPMENT
INDEX
PLANNER
PLANT
PM-ACTIVITY
PM-DESC
PP-PLANNER
PROD-PLANT
SAFETY-DEV
STOCK-NUM




ESC to return to menu.
```

Figure 19

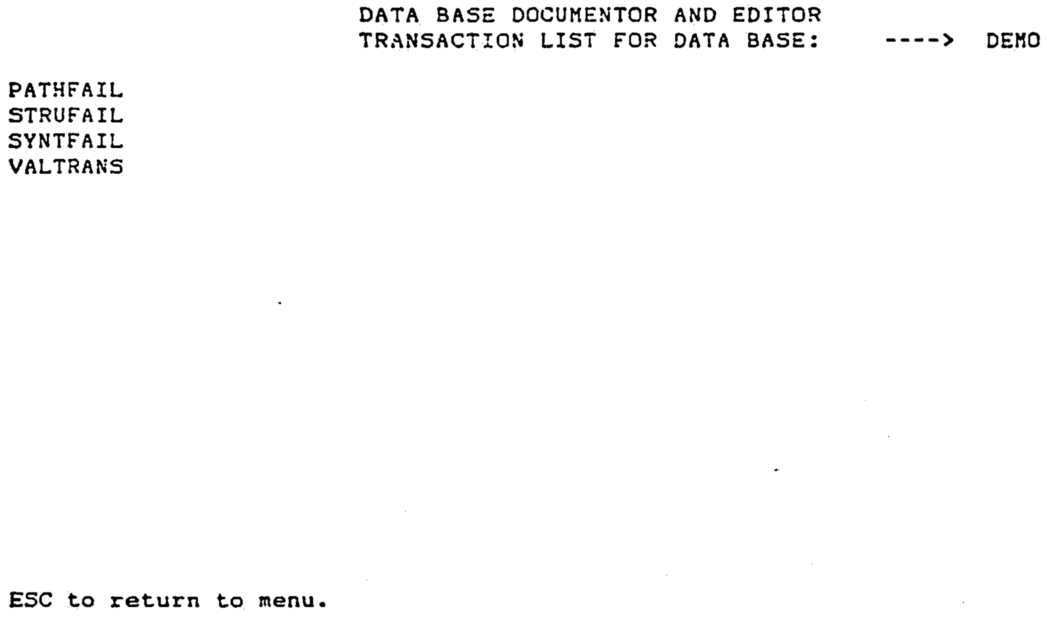
Record List for Data Base Screen



```
                    DATA BASE DOCUMENTOR AND EDITOR
                      SET LIST FOR DATA BASE:          ---->  DEMO

EQP-COMPON
EQP-PM
EQP-SD
EQP-WHR-USED
IX-PM-DTE
PLAN-PP
PLANNER-PM
PLANT-PLAN
PLANT-PP
PMACT-PMDESC
PP-EQP
PP-PLAN
PP-PLAN-PM
SD-PM
STOCK-SD



ESC to return to menu.
```

Figure 20

Set List for Data Base Screen



```
                    DATA BASE DOCUMENTOR AND EDITOR
                    TRANSACTION LIST FOR DATA BASE:     ---->  DEMO

PATHFAIL
STRUFAIL
SYNTFAIL
VALTRANS









ESC to return to menu.
```

Figure 21

Transaction List for Data Base Screen

IV. DB-ADE Transaction Structure and Syntax.

A. Defining the Transaction Structure.

The key functionality of the DB-ADE system is its ability to validate IDMS transactions against an IDMS physical data base structure. To use this functionality properly, the DB-ADE user must thoroughly understand the following:

- how to define the transaction to the DB-ADE system via the Transaction Maintenance Program;
- how to interpret and react to the results provided by DB-ADE's Transaction Proof Program.

The most important aspect of defining the transaction to DB-ADE is to understand the form a transaction is allowed to take, so that DB-ADE software interprets the transaction as the user wishes the transaction to be interpreted. To examine this form, it is necessary to digress momentarily from DB-ADE system specifics and to think more generally about programming logic.

Premise 1: Programs are typically iterative processes, that is, the same general logic is executed a multiple number of times, on each iteration processing a single transaction occurrence. The exact logic, or specific executable instructions, that are executed within the logic may differ from iteration to iteration, due to conditional execution logic (e.g., IF ... ELSE constructs, REPEAT WHILE constructs where the controlling variable is indeed variable and changes across iterations). Therefore, across all iterations, when conditional execution logic is dependent upon variables that change across iterations, there exist different transaction paths within program logic. Any single transaction occurrence is destined to take one-and-only-one one transaction path from program beginning to program end.

Premise 2: All control constructs in programs, including iterative constructs (such as REPEAT WHILEs, PERFORM UNTILs, et. al.) can be broken down into the simpler form of an IF ... ELSE construct to denote what program instructions execute for a given transaction occurrence. By definition, zero or more transaction occurrences will execute the iterative construct, and exactly one transaction occurrence will not execute the iterative construct.

For simplicity of transaction input and transaction maintenance programming, the available control statements within the DB-ADE system are limited to IF, ELSE, END, and STOP. The uses of END and STOP will be discussed shortly.

DB-ADE transactions are structured into blocks and within block structures into block paths. A block is defined as a group of DB-ADE control and verb (i.e., DML verbs, e.g., FIND, OBTAIN) statements that have as a start point a control statement that is common to all transaction paths and have as an end point a control statement that is also common to all transaction paths. A block path is one of the segments of a transaction path that leads from this start point to this end point.

As an example, consider the demonstration IDMS structure shown in Figure 22. Suppose a transaction called for the reporting of all preventive maintenance activities (PM-ACTIVITY) by planner (PLANNER) within a plant (PLANT). For each PM-ACTIVITY accessed, the piece of equipment (EQUIPMENT) or the safety device (SAFETY-DEV) on which the activity is performed is to be reported, as is all of the activity's descriptive detail text (PM-DESC). Assume a PM-ACTIVITY can be directly associated with (owned by) an EQUIPMENT or a SAFETY-DEV, but not both.

In documenting the IDMS DML commands that would be executed to obtain this data, an analyst might generate the following:

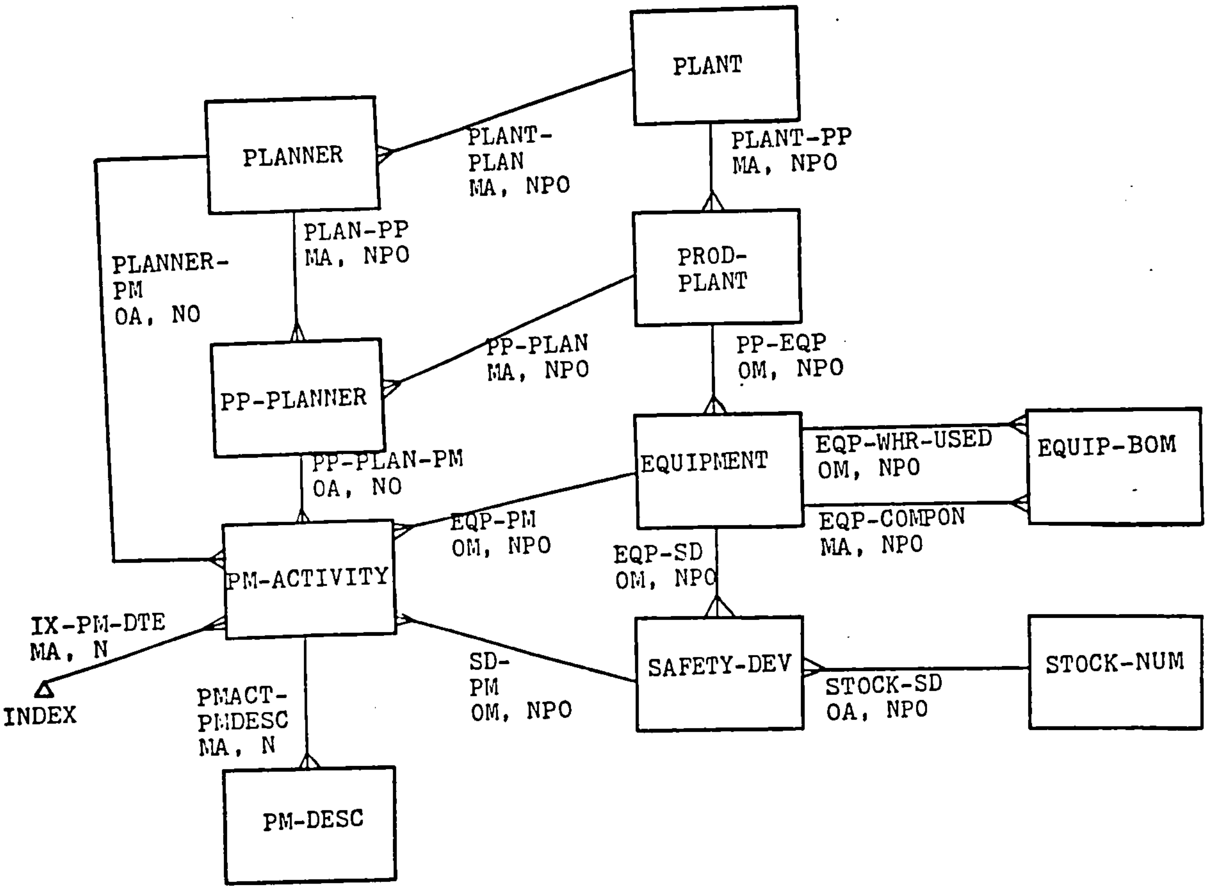


Figure 22

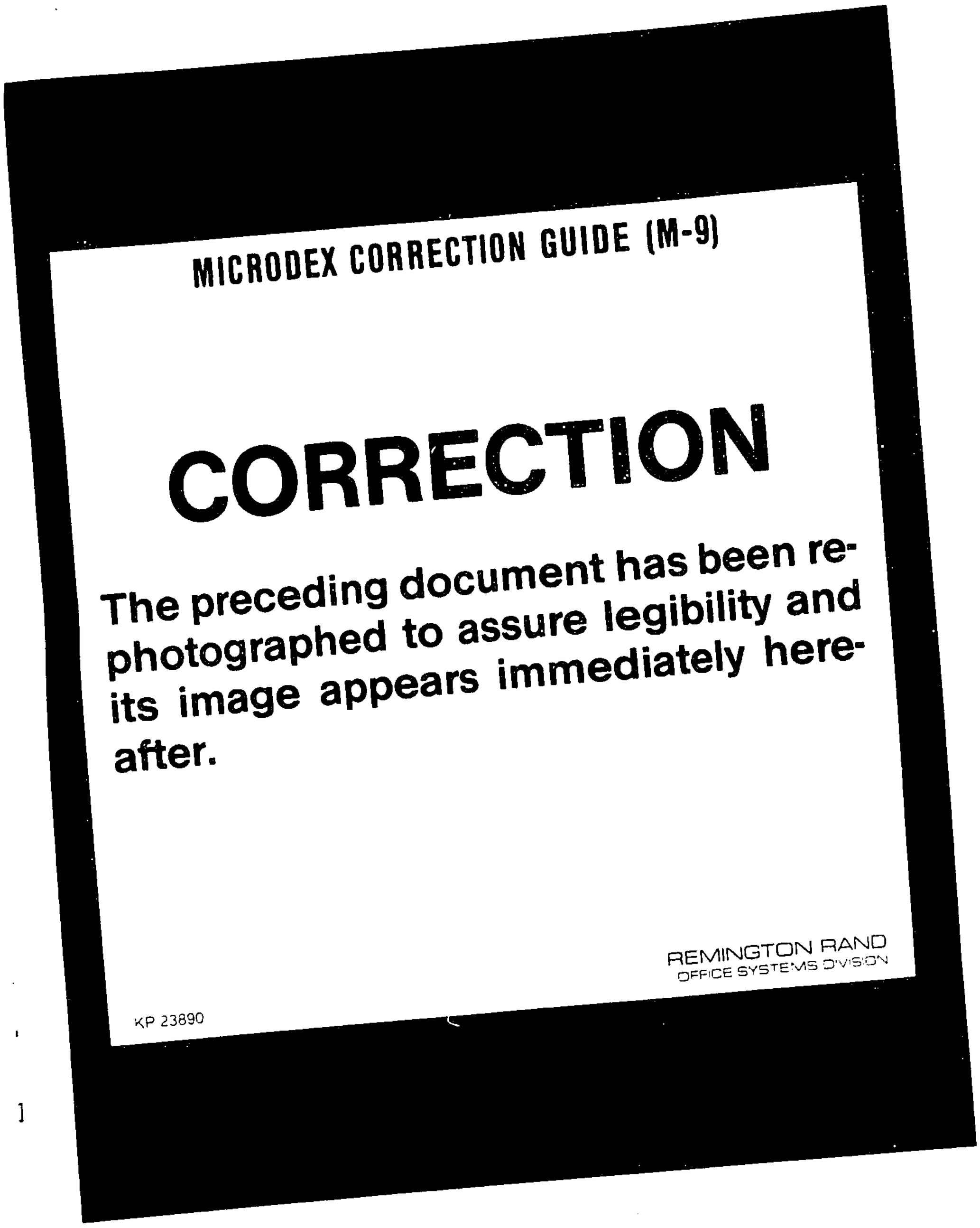
Demonstration Data Base Structure

```
OBTAIN NEXT PLANT WITHIN EQP-AREA (until end of area).
OBTAIN NEXT PLANNER WITHIN PLANT-PLAN (until end
      of set).
OBTAIN NEXT PM-ACTIVITY WITHIN PLANNER-PM (until end
      of set).
IF MEMBER EQP-PM
      then OBTAIN OWNER WITHIN EQP-PM
      else IF MEMBER SD-PM
                  then OBTAIN OWNER WITHIN SD-PM
                  else error condition (perhaps abend).
OBTAIN NEXT PM-DESC WITHIN PMACT-PMDESC (until end
      of set).
FIND CURRENT PLANT (to reset EQP-AREA area currency).
```

The logical flow of this transaction can be depicted as in Figure 23. Looking at the above code and Figure 24, three transaction paths, or exact sequence of statements that can possibly execute, exist for this transaction.

In Figure 24, Ⓐ is the endpoint of the first block of the transaction. All three block paths (i.e., the segments of each of the three transaction paths from the beginning of the transaction to Ⓐ), end (converge) on Ⓐ . The second block of the transaction contains only one block path between Ⓐ and the transaction endpoint Ⓑ .

In the DB-ADE system, this transaction would be written as shown in Figures 25 and 26. The transaction statements follow the DB-ADE syntax rules that will be discussed shortly. The END control statement is used to mark the end



MICRODEX CORRECTION GUIDE (M-9)

CORRECTION

The preceding document has been rephotographed to assure legibility and its image appears immediately hereafter.

REMINGTON RAND
OFFICE SYSTEMS DIVISION

KP 23890

]

1

n

s ----- statement is used to mark the end

```
OBTAIN NEXT PLANT WITHIN EQP-AREA (until end of area).
OBTAIN NEXT PLANNER WITHIN PLANT-PLAN (until end
      of set).
OBTAIN NEXT PM-ACTIVITY WITHIN PLANNER-PM (until end
      of set).
IF MEMBER EQP-PM
      then OBTAIN OWNER WITHIN EQP-PM
      else IF MEMBER SD-PM
                  then OBTAIN OWNER WITHIN SD-PM
                  else error condition (perhaps abend).
OBTAIN NEXT PM-DESC WITHIN PMACT-PMDESC (until end
      of set).
FIND CURRENT PLANT (to reset EQP-AREA area currency).
```

The logical flow of this transaction can be depicted as in Figure 23. Looking at the above code and Figure 24, three transaction paths, or exact sequence of statements that can possibly execute, exist for this transaction.

In Figure 24, Ⓐ is the endpoint of the first block of the transaction. All three block paths (i.e., the segments of each of the three transaction paths from the beginning of the transaction to Ⓐ), end (converge) on Ⓐ . The second block of the transaction contains only one block path between Ⓐ and the transaction endpoint Ⓑ .

In the DB-ADE system, this transaction would be written as shown in Figures 25 and 26. The transaction statements follow the DB-ADE syntax rules that will be discussed shortly. The END control statement is used to mark the end

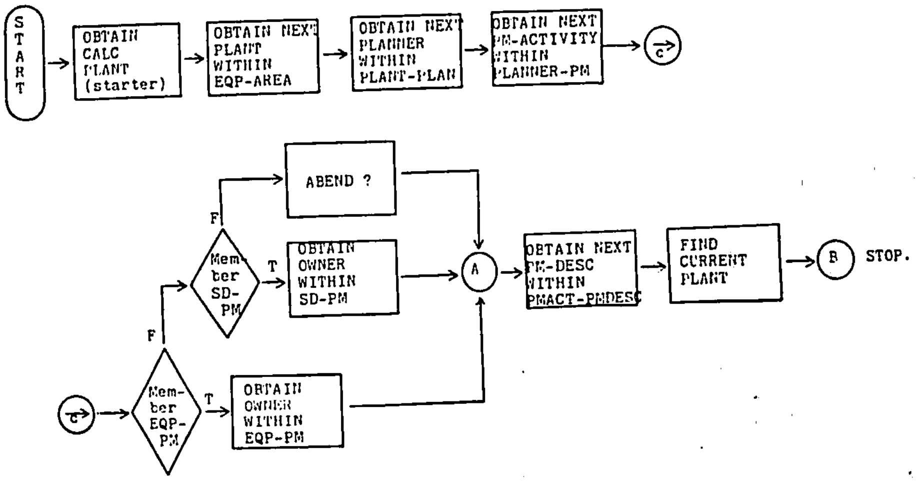

START
OBTAIN CALC PLANT (starter)
OBTAIN NEXT PLANT WITHIN EQP-AREA
OBTAIN NEXT PLANNER WITHIN PLANT-PLAN
OBTAIN NEXT PM-ACTIVITY WITHIN PLANNER-PM
C
C
Member EQP-PM
T
OBTAIN OWNER WITHIN EQP-PM
F
Member SD-PM
T
OBTAIN OWNER WITHIN SD-PM
F
ABEND ?
A
OBTAIN NEXT PM-DESC WITHIN PMACT-PMDESC
FIND CURRENT PLANT
B
STOP.

Figure 23

Demonstration Transaction Logical Flow

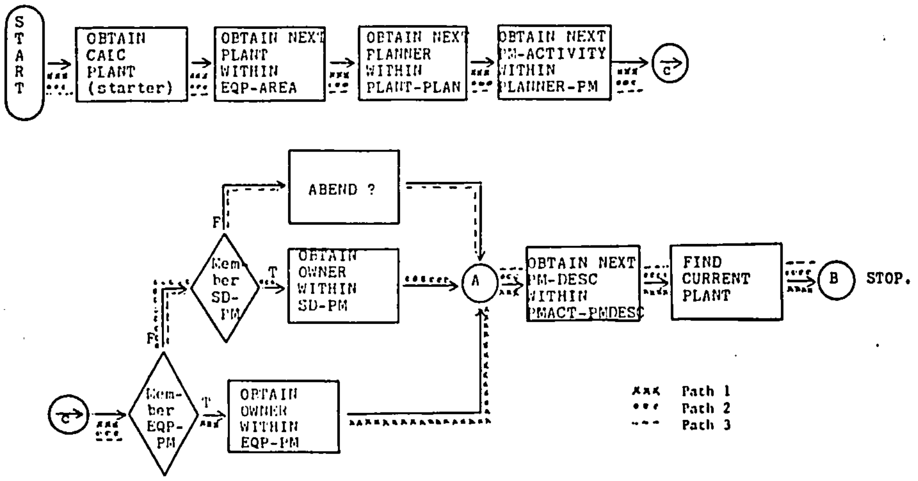
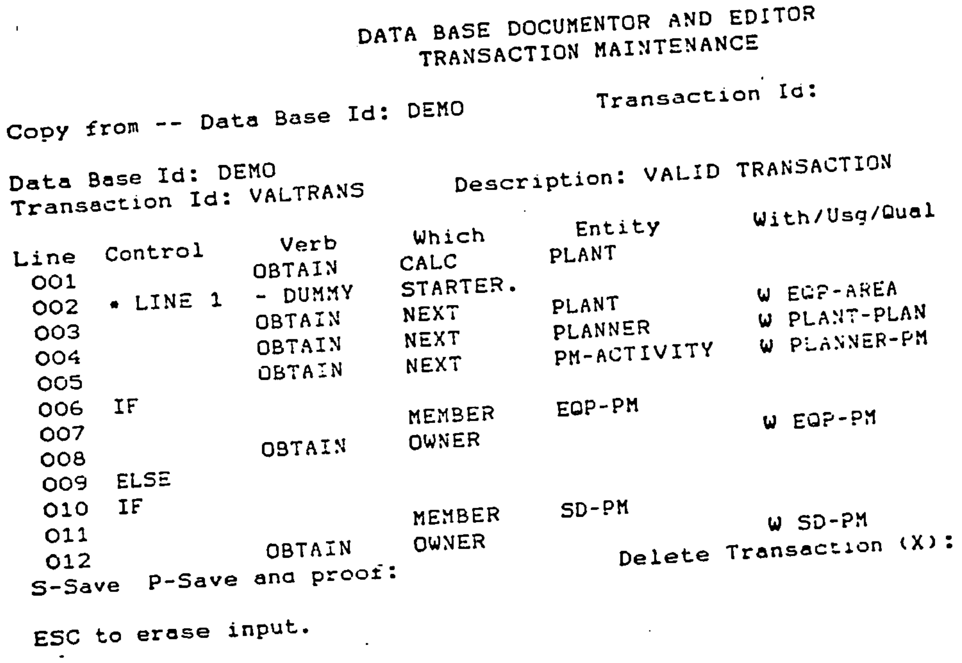

START
OBTAIN CALC PLANT (starter)
OBTAIN NEXT PLANT WITHIN EQP-AREA
OBTAIN NEXT PLANNER WITHIN PLANT-PLAN
OBTAIN NEXT PM-ACTIVITY WITHIN PLANNER-PM
C
ABEND ?
F
Member SD-PM
T
OBTAIN OWNER WITHIN SD-PM
A
OBTAIN NEXT PM-DESC WITHIN PMACT-PMDESC
FIND CURRENT PLANT
B
STOP.
F
C
Member EQP-PM
T
OBTAIN OWNER WITHIN EQP-PM
Path 1
Path 2
Path 3

Figure 24

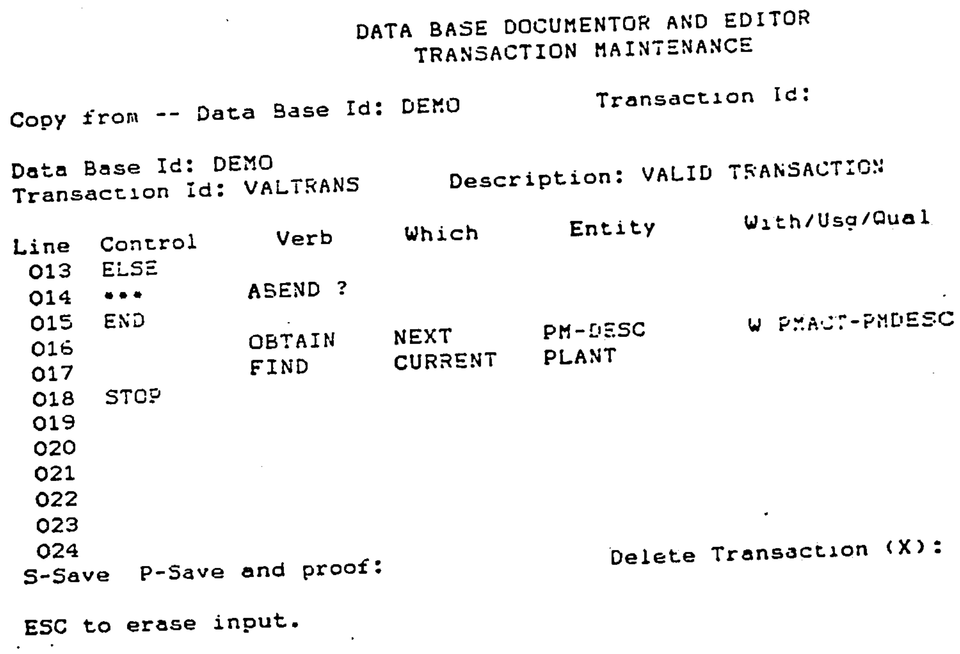
Three Transaction Paths within
Demonstration Transaction



```
                       DATA BASE DOCUMENTOR AND EDITOR
                          TRANSACTION MAINTENANCE

                                                    Transaction Id:
Copy from -- Data Base Id: DEMO

Data Base Id: DEMO
Transaction Id: VALTRANS        Description: VALID TRANSACTION
                                                       With/Usg/Qual
                          Which         Entity
                Verb
Line  Control                           PLANT
                OBTAIN    CALC
 001
 002  * LINE 1  - DUMMY   STARTER.
                OBTAIN    NEXT          PLANT          W EQP-AREA
 003            OBTAIN    NEXT          PLANNER        W PLANT-PLAN
 004            OBTAIN    NEXT          PM-ACTIVITY    W PLANNER-PM
 005
 006  IF                  MEMBER        EQP-PM
 007                      OWNER                        W EQP-PM
 008            OBTAIN
 009  ELSE
 010  IF                  MEMBER        SD-PM
 011                      OWNER                        W SD-PM
 012            OBTAIN                     Delete Transaction (X):
S-Save  P-Save and proof:

ESC to erase input.
```

Figure 25

Demonstration Transaction Defined to DB-ADE
(1st Screen)



```
                      DATA BASE DOCUMENTOR AND EDITOR
                          TRANSACTION MAINTENANCE

Copy from -- Data Base Id: DEMO          Transaction Id:

Data Base Id: DEMO
Transaction Id: VALTRANS      Description: VALID TRANSACTION

Line  Control     Verb      Which       Entity       With/Usg/Qual
 013  ELSE
 014  ***        ASEND ?
 015  END                                             W PMACT-PMDESC
 016             OBTAIN    NEXT        PM-DESC
 017             FIND      CURRENT     PLANT
 018  STOP
 019
 020
 021
 022
 023
 024
S-Save  P-Save and proof:                    Delete Transaction (X):

ESC to erase input.
```

Figure 26

Demonstration Transaction Defined to DB-ADE
(2nd Screen)

of a transaction block. The STOP control statement is used to mark the end of the last transaction block (and therefore the end of the entire transaction). DB-ADE recognizes the IF .. ELSE constructs that are permissible in many programming languages:

- the simple IF ... ELSE:

```
IF
     ( optional imperative statements )
ELSE
     ( optional imperative statements ).
```

- the "rolling" IF ... ELSE:

```
IF
     ( optional imperative statements )
ELSE
     ( optional imperative statements )
     IF
          ( optional imperative statements )
     ELSE
          ( optional imperative statements )
          IF
                         ●
                         ●
                         ●
```

- the "nested" IF ... ELSE:

```
IF
     ( optional imperative statements )
     IF
          ( optional imperative statements )
          IF
               ( optional imperative statements )
                         ●
                         ●
                         ●
                         ●
                         ●
                         ●
          ELSE
               ( optional imperative statements )
       ELSE
            ( optional imperative statements )
    ELSE
         ( optional imperative statements ).
```

B. <u>DB-ADE Transaction Statement Syntax</u>.

As seen in Figure 25, the components of a DB-ADE transaction statement are:

- a line number (pre-loaded by the system);
- a control form;
- a verb form;
- a "which" form;
- an entity form;
- two "within/using/qualifier" forms.

DB-ADE allows for up to two hundred (200) transaction statements per transaction. Valid DB-ADE control forms and verb forms are shown in Figure 27. All valid DB-ADE transaction statement syntaxes are shown in Figure 28. The DB-ADE format of an IDMS DML form should be apparent to the knowledgeable IDMS user from referencing Figure 28.

Comment lines can be entered on the Transaction Maintenance Screen by keying an asterisk ("*") as the first character of the statement's control form. DB-ADE also al-

Control Forms

IF
ELSE
END
STOP

Verb Forms

FIND
OBTAIN
GET
ERASE
CONNECT
DISCONN
STORE
MODIFY
ACCEPT

Figure 27

Valid DB-ADE Control and Verb Forms

DATA BASE DOCUMENTOR AND EDITOR (DB-ADE)
IV. DB-ADE Transaction Structure and Syntax

Control	Verb	Which	Entity	With/Usg/Qual
	FIND OBTAIN	CALC DUP CURRENT	rec-name	
	FIND OBTAIN	NEXT PRIOR FIRST LAST NTH SORTKEY	rec-name	W set-name W area-name
	FIND OBTAIN	CURRENT		W set-name W area-name
	FIND OBTAIN		rec-name	U DBKEY
	FIND OBTAIN	OWNER		W set-name
	CONNECT DISCONN		rec-name	set-name
	GET		rec-name	
	STORE		rec-name	
	MODIFY		rec-name	
	ERASE		rec-name	
	ACCEPT	NEXT PRIOR OWNER	rec-name set-name	CURRENCY
	ACCEPT		area-name	CURRENCY

IF
ELSE
END
STOP				

		MEMBER NOTEMPTY EMPTY	set-name	

Figure 28

Valid DB-ADE Transaction Statement Syntaxes lows the user to input a completely blank transaction statement.

V. The DB-ADE Program/PC Environment.

A. DB-ADE DOS Files.

Three categories of DOS files are used within the DB-ADE system:

- DB-ADE program run modules;
- DB-ADE data files;
- DB-ADE utilities.

DB-ADE program run modules include the following:

- DBADE.BAT --> Invokes DB-ADE when DBADE is executed from >C. Prior to executing DBADRIVE.COM, takes a backup of all .DAT and .IXF files resident on the diskette on drive A.
- DBADRIVE.COM --> Invokes the DBADE main menu program.
- DBAMENU.CHN --> The main menu program.
- DBADBID.CHN --> Data Base Identification Program.
- DBAAREA.CHN --> Area Definition Program.
- DBAREC.CHN --> Record Definition Program.
- DBASET.CHN
  DBASET.000 --> Set Definition Program modules.
  DBASET.001
  DBASET.002
- DBATINPT.CHN --> Transaction Maintenance Program.
- DBATVALD.CHN --> Transaction Proof Program
  DBATPATH.CHN modules.
- DBALSDBI.CHN --> List Data Base Identifiers Program.

- DBALSARE.CHN --> List Areas For A Data Base Program.
- DBALSREC.CHN --> List Records For A Data Base Program.
- DBALSSET.CHN --> List Sets For A Data Base Program.
- DBALSTRN.CHN --> List Transactions For A Data Base Program.

The DB-ADE program structure is shown in Figure 29.

DB-ADE utilities are used to manage the DB-ADE environment.

NOTE: DB-ADE configuration requirements call for the diskette containing DB-ADE .DAT and .IXF data files to be resident in drive A when operating the system. DB-ADE program run modules should be resident on drive C or drive B, depending on whether the PC has a fixed disk or a second diskette drive. The DB-ADE utilities upon release of DB-ADE perform DOS file management between the A and C drives, thus being compatible with fixed disk configurations such as the IBM PC/XT or the COMPAQ terminal. Execution of DB-ADE on two-diskette drive systems, such as the IBM PC and the IBM PC/AT, require alteration of the DB-ADE utilities so that they perform DOS file management between the A and B drives.

The DB-ADE utilities are the following:

- DBASYS.BAT --> Resident on DB-ADE object code diskette 1, execution of this file from >A when diskette 1 is in drive A will load all DBADE .BAT files onto drive C.

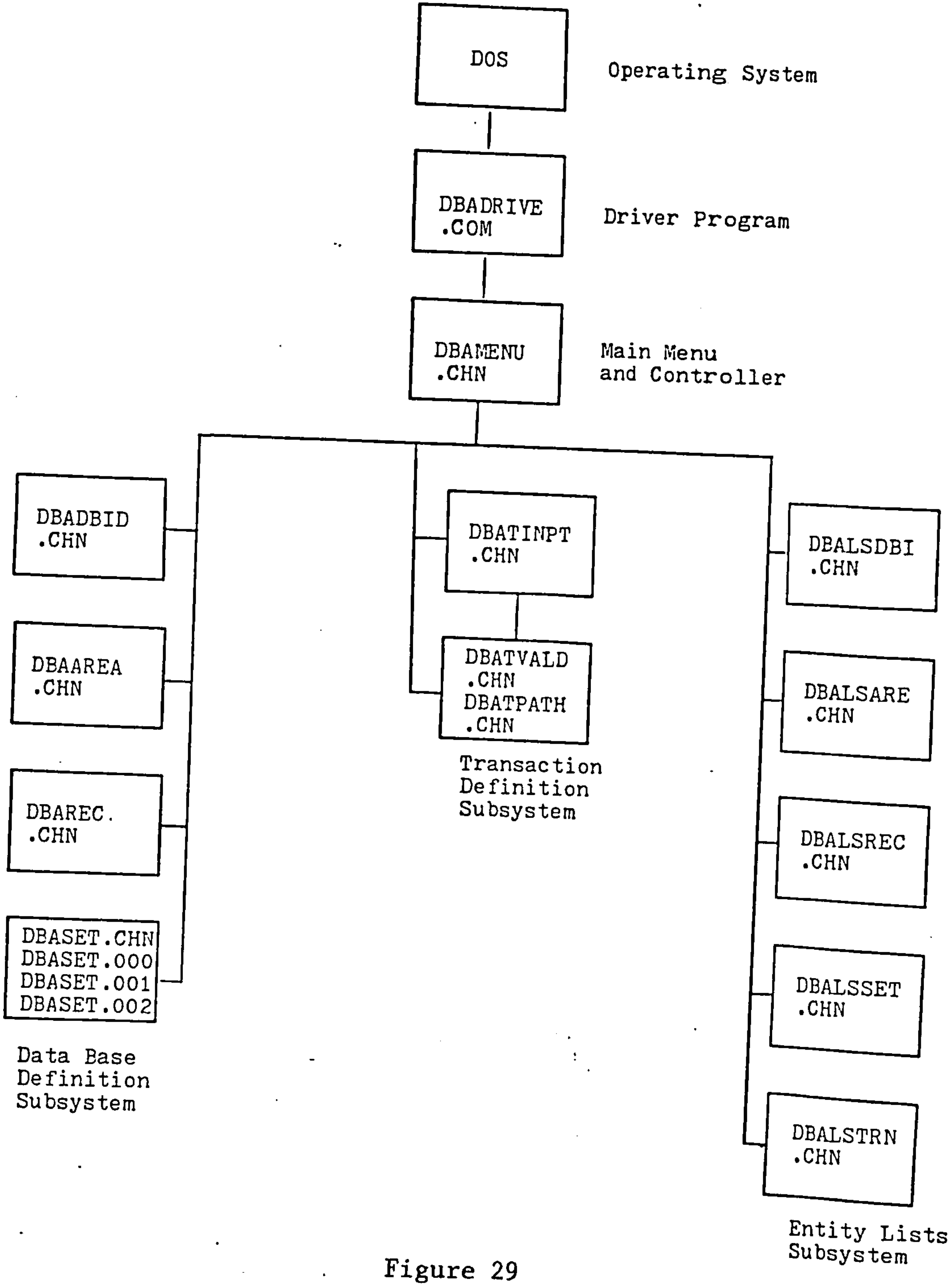


Figure 29

DB-ADE Program Structure

- DBAPROG1.BAT --> Resident on DB-ADE object code diskette 1, execution of this file from >A when diskette 1 is in drive A will load all DBADE program modules resident on diskette 1 onto drive C.

- DBAPROG2.BAT --> Resident on DB-ADE object code diskette 2, execution of this file from >A when diskette 2 is in drive A will load all DBADE program modules resident on diskette 2 onto drive C.

- DBABACK.BAT --> After being loaded onto drive C by execution of DBASYS.BAT, execution of this file from >C when DB-ADE data files diskette is in drive A will back-up (copy) .DAT and .IXF files from drive A onto drive C as .DAB and .IXB files, respectively.

- DBAREST.BAT --> After being loaded onto drive C by execution of DBASYS.BAT, execution of this file from >C when a diskette is in drive A will restore (copy) .DAB and .IXB back-up files from drive C onto drive A as .DAT and .IXF files, respectively.

- DBAMAKE.COM --> When executed from >C, will create empty .DAT and .IXF data files on the diskette resident in drive A.

- DBAMKTMP.COM --> When executed from >C, will restore DB-ADE temporary files DBAVAL.DAT, DBAVAL.IXF, DBAPAT.DAT, and DBAPAT.IXF to empty states.

DB-ADE data files include the following:

- DBADBI.DAT --> Data base identifier data and
  DBADBI.IXF index files.

- DBAARE.DAT --> Area identifier data and index files.
  DBAARE.IXF
- DBAREC.DAT --> Record identifier data and index files.
  DBAREC.IXF
- DBASET.DAT --> Set identifier data and index files.
  DBASET.IXF
- DBATRN.DAT --> Transaction data and index files.
  DBATRN.IXF
- DBAVAL.DAT --> Temporary files used to store validation report data; file contents are erased after the report is printed.
  DBAVAL.IXF
- DBAPAT.DAT --> Temporary files used to pass transaction data between Transaction Proof Program modules DBATVALD.CHN and DBATPATH.CHN.
  DBAPAT.IXF

B. How-To's.

To install the DB-ADE system, do the following:

1st -- place object code diskette 1 on drive A, and execute DBASYS.BAT from A>;
2nd -- keep object code diskette 1 on drive A, and execute DBAPROG1.BAT from A>;
3rd -- place object code diskette 2 on drive A, and execute DBAPROG2.BAT from A>;
4th -- place empty diskette that will contain DB-ADE .DAT and .IXF data files on drive A, and execute DBAMAKE.COM from >C.

To execute DB-ADE, do the following:

1st -- place current DB-ADE data files diskette on drive A;
2nd -- execute DBADE.COM from >C -- this will always back-up the data files onto drive C as .DAB and .IXB files, prior to bringing up the DB-ADE Main Menu Screen.

To back-up the DB-ADE .DAT and .IXF files, do the following:

1st -- place DB-ADE data files diskette on drive A;
2nd -- execute DBABACK.BAT from >C.

To restore the DB-ADE .DAT and .IXF files, do the following:

1st -- place diskette to which the data files are to be restored onto drive A;
2nd -- execute DBAREST.BAT from >C.

C. Routing of Validation Reports to the Printer.

DB-ADE transaction syntax validation, structural validation, and transaction path analysis reports are routed to a PC's printer output device. Therefore, the initial release of DB-ADE demands a PC configuration that includes a printer.

Changing of the DB-ADE system to print these reports on the CRT requires modifying the program source code in DBATVALD.PAS and DBATPATH.PAS to direct the writing of the reports to output device "Con" rather than "Lst", and a recompiling of these programs to DBATVALD.CHN and DBATPATH. CHN, respectively. However, the report lines will only list until they are exhausted on the CRT screen, since no sophisticated programming logic that controls page-by-page report viewing is coded within these programs.

VI. Keyboard Control Input Characters.

Within the remainder of this document, keyboard input to the DB-ADE system will be denoted by enclosing the keyboard responses within the "<" ("less than") and the ">" ("greater than") symbols. Furthermore, "Enter" denotes the keyboards ENTER (↵) key, "Esc" the keyboard's ESC key, and "Ctrl" the keyboard's CTRL key. Keyboard input responses are made within the DB-ADE system to navigate between and within program modules, to respond to prompts, to move between screen fields, and to manipulate character strings within data fields.

EXAMPLES:

<Enter> denotes the depressing of the keyboard's ENTER key;

<Esc> denotes the depressing of the keyboard's ESC key;

<A> denotes the depressing of the character "A" key;

<2> denotes the depressing of the number "2" key;

<Ctrl-E> denotes the depressing of the character "E" key while the keyboard's CTRL key is depressed.

The functionality of certain keyboard input responses are absolutely or relatively constant across all DB-ADE system programs. Identification of these "reserved" responses follows.

- The <Enter> response.

In general, <Enter> is used to move the cursor to the first character position of the next input data field on a screen. If the last key data field is <Enter>ed within a program that must perform data retrieval, validation of the key fields and the data retrieval process for screen display are invoked. In update program modules, the <Enter>-ing of proper responses to UPDATE? or SAVE? fields will result in input validation and DB-ADE file update.

- The <Ctrl-P> response.

On DB-ADE input screens, <Ctrl-P> is used to move the cursor to the first character of the prior input data field. <Ctrl-P> is also used to initiate the printing of the validation error report from the Transaction Validation Log screen.

- The <Esc> response.

<Esc> is used within a program module to erase the currently displayed input data field values, in order to restart the data input process, and in some instances is

used to exit the program module completely. DB-ADE displays the current use of <Esc> on the bottom line of the screen as long as error messages are not present on that line. A displayed message of "Esc to erase input." denotes the first use of <Esc>, while a displayed message of "Esc to quit." denotes the second use of <Esc>.

- The <Ctrl-F>, <Ctrl-B>, <Ctrl-D>, and <Ctrl-E> responses.

These keyboard inputs are used to manipulate the character string within a specific input data field on the screen.

<Ctrl-F> moves the cursor one position to the right (forward) within the character string, while not affecting the content of the character string.

<Ctrl-B> moves the cursor one position to the left (backward) within the character string, while not affecting the content of the character string.

<Ctrl-D> erases (deletes) the character on which the cursor is positioned, and shifts all characters that had been to the deleted character's right one position to the left. (EXAMPLE: "ERASE " will become "EASE ").

<Ctrl-E> erases the character on which the cursor is positioned and all characters to its right. (EXAMPLE: "CHARACTER" will become "CHAR_   ").

- The <Ctrl-W>, <Ctrl-X>, <Ctrl-V>, <Ctrl-R>, <Ctrl-Z>, and <Ctrl-Q> responses.

These keyboard inputs are reserved for use within the Transaction Maintenance Program only. The responses allow flexibility in viewing and modifying the transaction being processed by the DB-ADE user.

<Ctrl-W> inserts a new line within the transaction after the line on which the cursor is positioned.

<Ctrl-X> deletes the transaction line on which the cursor is positioned.

<Ctrl-V> scrolls a page forward in the transaction.

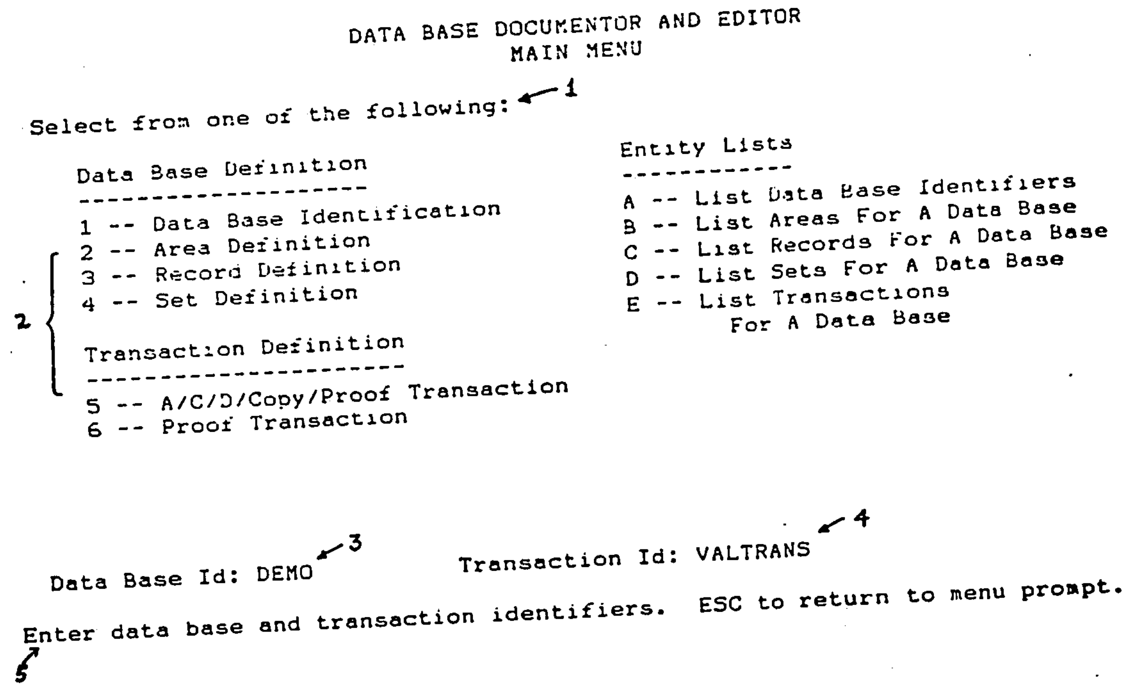
<Ctrl-R> scrolls a page backward in the transaction.

<Ctrl-Z> moves the cursor down one transaction line.

<Ctrl-Q> moves the cursor up one transaction line.

VII. DB-ADE Screen Field Descriptions.

The following sections provide specifications for input fields on the DB-ADE system input screens. The format of this section is a Figure displaying the screen followed by the screen's input requirements. Circled numbers on the screen Figures correspond to the field numbering scheme within the section's text.



```
                    DATA BASE DOCUMENTOR AND EDITOR
                              MAIN MENU

Select from one of the following:

    Data Base Definition                     Entity Lists
    --------------------                     ------------
    1 -- Data Base Identification            A -- List Data Base Identifiers
    2 -- Area Definition                     B -- List Areas For A Data Base
    3 -- Record Definition                   C -- List Records For A Data Base
    4 -- Set Definition                      D -- List Sets For A Data Base
                                             E -- List Transactions
    Transaction Definition                         For A Data Base
    ----------------------
    5 -- A/C/D/Copy/Proof Transaction
    6 -- Proof Transaction


  Data Base Id: DEMO           Transaction Id: VALTRANS

Enter data base and transaction identifiers.  ESC to return to menu prompt.
```

Figure 30

Main Menu Screen Fields

A. Main Menu Screen Field Descriptions (see Figure 30).

1. Program selection code. Key in the code associated with the program to which you wish to transfer control.

2. Valid program selection codes and the programs they cause to be invoked are displayed in the submenu lists.

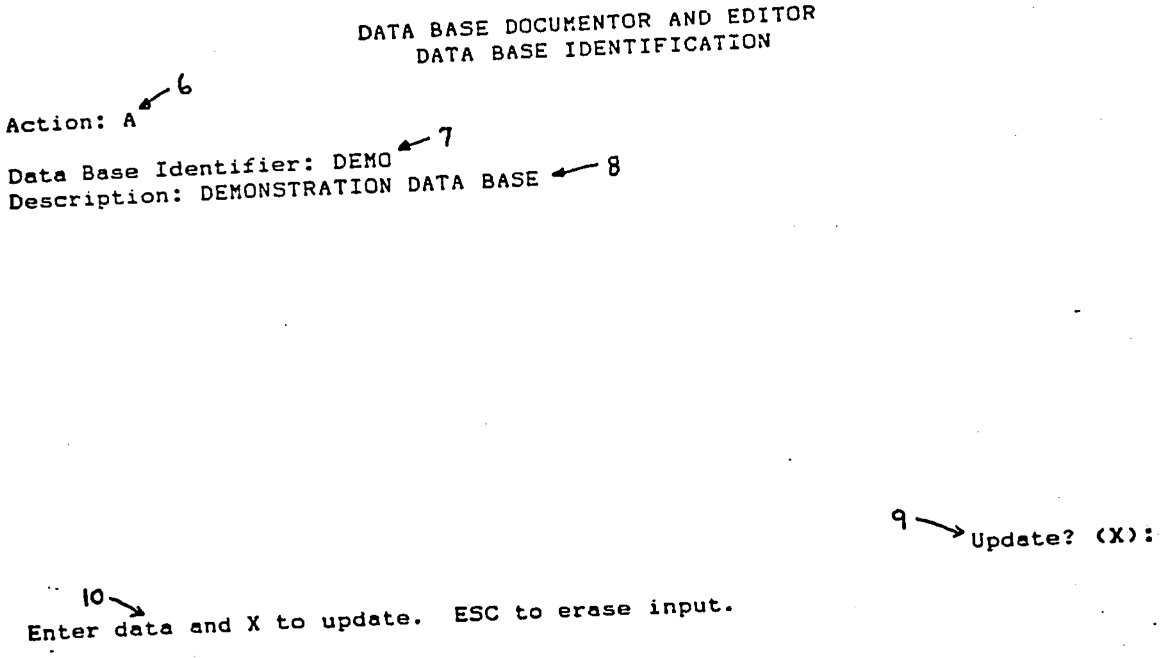
3. Data Base Id. Program selection codes 6, B, C, D, and E, when keyed, will prompt for the DB-ADE data base identifier to be <Enter>ed. <Enter>ing of the data base identifier (except when program selection code 6 is chosen) will cause the entered data base identifier to be validated and the proper program invoked. (12 characters).

4. Transaction Id. Program selection code 6 will also request that the identifier of the transaction to be proofed be entered in addition to the data base identifier. Only upon <Enter>ing of this transaction identifier will the transaction identifier be validated and the Transaction Proof Program invoked. (12 characters).

5. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Main Menu Screen include the following:

   - E001-Db id required.
   - E003-Db id not found.
   - E054-Trans id required.
   - E055-Trans id not found.

Consult Section VIII for more detailed descriptions of these error messages.



```
                    DATA BASE DOCUMENTOR AND EDITOR
                       DATA BASE IDENTIFICATION

Action: A

Data Base Identifier: DEMO
Description: DEMONSTRATION DATA BASE




                                                        Update? (X):

Enter data and X to update.  ESC to erase input.
```

Figure 31

Data Base Identification Screen Fields

B. Data Base Identification Screen Field Descriptions (see Figure 31).

6. Action code. A = add a data base identifier, B = browse (view) a data base identifier, C = change a data base identifier's attributes, D = delete a data base identifier, Q = return to main menu.

7. Data Base Identifier. (12 characters).

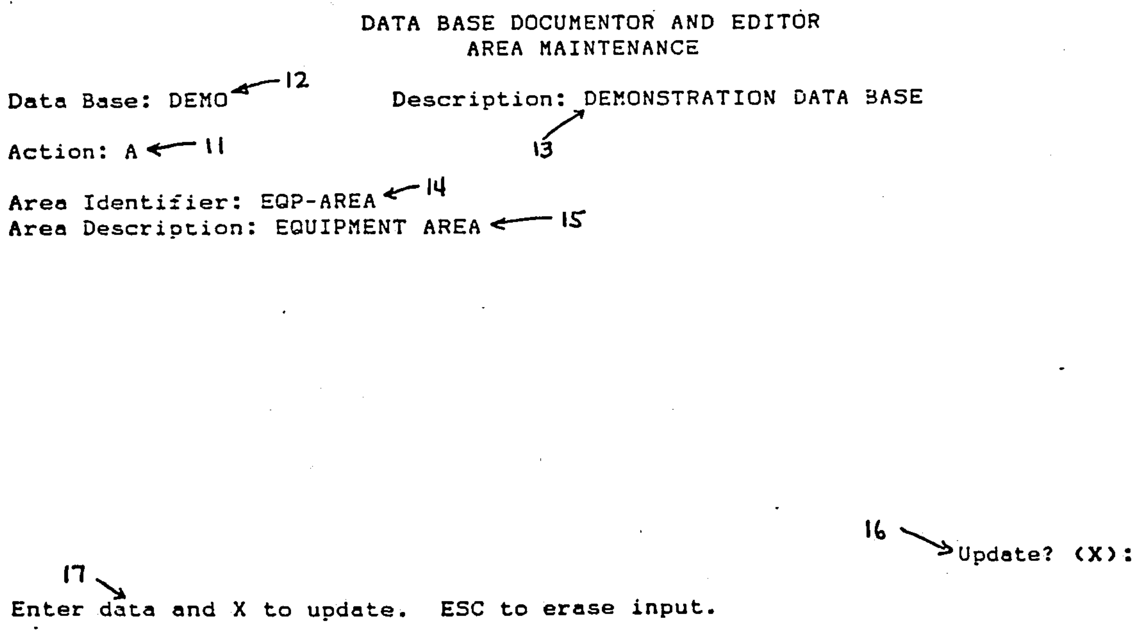
8. Data Base Description. A short description of the data base identifier. (30 characters).

9. Update? (X). <Enter>ing an "X" in this field causes the add, change, or delete update process to begin. (1 character).

10. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Data Base Identification Screen include the following:

    - E001-Db id required.
    - E002-Db id exists.
    - E003-Db id not found.

Consult Section VIII for more detailed descriptions of these error messages.



```
                         DATA BASE DOCUMENTOR AND EDITOR
                                AREA MAINTENANCE
Data Base: DEMO               Description: DEMONSTRATION DATA BASE

Action: A

Area Identifier: EQP-AREA
Area Description: EQUIPMENT AREA




                                                              Update? (X):
Enter data and X to update.  ESC to erase input.
```

Figure 32

Area Maintenance Screen Fields

C. Area Maintenance Screen Field Descriptions (see Figure 32).

11. Action code. A = add an area, B = browse (view) an area, C = change an area's attributes, D = delete an area, Q = return to main menu.

12. Data Base. The data base identifier that qualifies the area identifier. This must be entered unless its value was carried forward by the system from another program. (12 characters).

13. Description. The description assigned to the data base identifier. This will be displayed upon the <Enter>ing of a valid data base identifier in field 12. (30 characters).

14. Area Identifier. (12 characters).

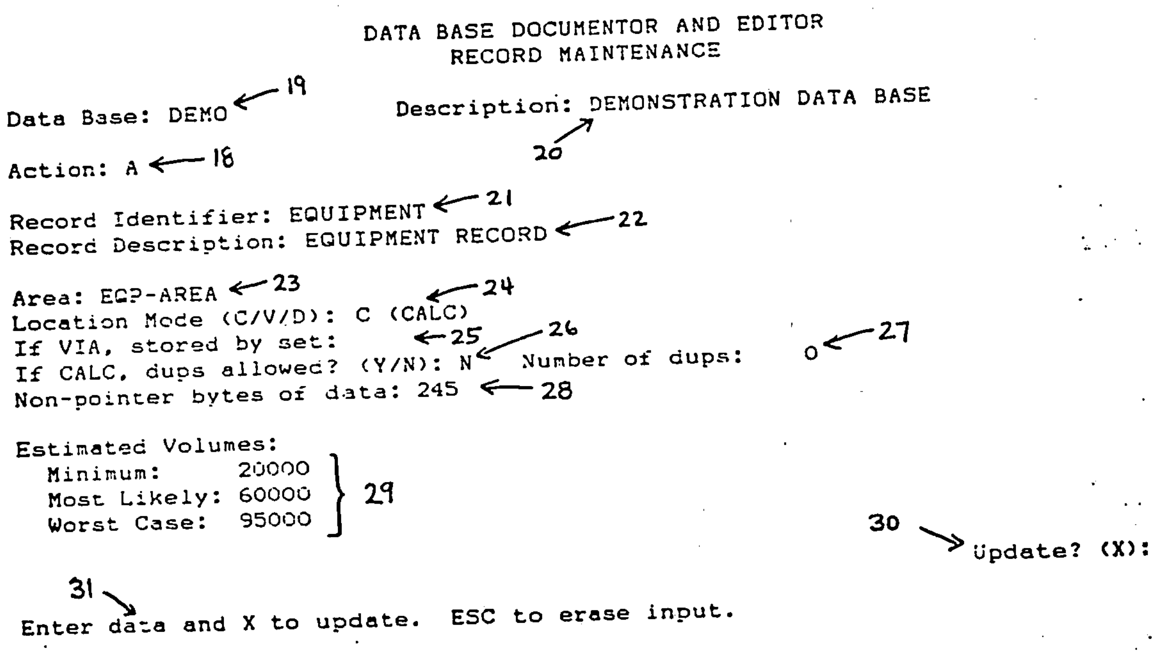
15. Area Description. A short description of the area identifier. (30 characters).

16. Update? (X). <Enter>ing an "X" in this field causes the add, change, or delete update process to begin. (1 character).

17. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Area Maintenance Screen include the following:

- E001-Db id required.
- E003-Db id not found.
- E004-Area id required.
- E005-Area exists for DB.
- E006-Area not in DB.
- E063-Areaid used for rec.
- E064-Areaid used for set.

Consult Section VIII for more detailed descriptions of these error messages.



```
                    DATA BASE DOCUMENTOR AND EDITOR
                          RECORD MAINTENANCE

Data Base: DEMO            Description: DEMONSTRATION DATA BASE

Action: A

Record Identifier: EQUIPMENT
Record Description: EQUIPMENT RECORD

Area: EQP-AREA
Location Mode (C/V/D): C (CALC)
If VIA, stored by set:
If CALC, dups allowed? (Y/N): N   Number of dups:      0
Non-pointer bytes of data: 245

Estimated Volumes:
  Minimum:      20000
  Most Likely:  60000
  Worst Case:   95000
                                                         Update? (X):

Enter data and X to update.  ESC to erase input.
```

Figure 33

Record Maintenance Screen Fields

D. Record Maintenance Screen Field Descriptions (see Figure 33).

18. Action code. A = add a record, B = browse (view) a record, C = change an record's attributes, D = delete a record, Q = return to main menu.

19. Data Base. The data base identifier that qualifies the record identifier. This must be entered unless its value was carried forward by the system from another program. (12 characters).

20. Description. The description assigned to the data base identifier. This will be displayed upon the <Enter>ing of a valid data base identifier in field 19. (30 characters).

21. Record Identifier. (12 characters).

22. Record Description. A short description of the record identifier. (30 characters).

23. Area. The data base area in which the record is stored. This must be a valid DB-ADE area identifier that was created through use of the Area Maintenance Screen. (12 characters).

24. Location Mode. Indicates how the record is stored. C = CALC, V = VIA, D = DIRECT. (1 character).

25. If VIA, stored by set. If the location mode (field 24) was entered as C or D, the system assigns spaces to this field and skips over it. If the location mode was entered as V, this field should be encoded with the DB-ADE set identifier by which this record will be stored VIA. NOTE: Record identifiers must be stored in DB-ADE prior to storing set identifiers that reference the records. Therefore, this field will not cause a validation error that precludes the record identifier from being accepted by DB-ADE. When entered, this field's presence will generate a W001 warning message reminder preceding the subsequent "Files updated." message. (12 characters).

26. If CALC, dups allowed? If location mode (field 24) was entered as V or D, the system assigns a value of N to this field and skips over it. If the location mode was entered as C, a Y or N must be entered in this field. (1 character).

27. Number of dups. If location mode (field 24) was entered as V or D, or if location mode was entered as C and "If CALC, dups allowed?" (field 26) was entered as N, the system assigns a value of 0 (zero) to this field. Otherwise, this field must be entered with a value of 2 or greater. (4 characters, numeric).

28. Non-pointer bytes of data. The number of bytes of application-oriented data that is to be stored on any one occurrence of this record (4 characters, numeric).

29. Estimated volumes. The minimum, most likely, and worst case number of occurrences of the record expected in the data base. Minimum ≤ most likely ≤ worst case. (8 characters, numeric).

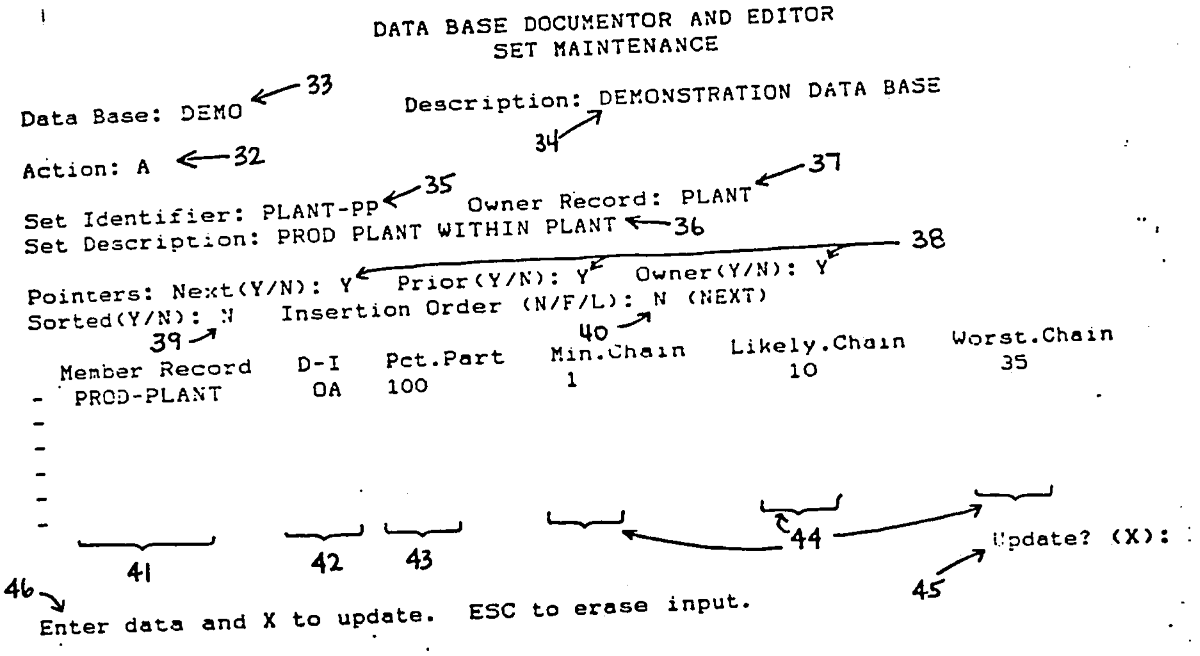
30. Update? (X). <Enter>ing an "X" in this field causes the add, change, or delete update process to begin. (1 character).

31. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Record Maintenance Screen include the following:

- E001-Db id required.
- E003-Db id not found.
- E004-Area id required.
- E006-Area not in DB.
- E007-Rec id required.
- E008-Rec id exists.
- E009-Locmode not C,V,D.
- E010-Set required when VIA.
- E011-Dupsind required.
- E012-Numdups required.
- E013-Dupsind not Y,N.
- E014-Numdups not > 1.

- E015-Numdup not numeric.
- E016-Bytes not numeric.
- E017-Min vol reqd.
- E018-Likely vol reqd.
- E019-Worst vol reqd.
- E020-Min vol not num.
- E021-Likely vol not num.
- E022-Worst vol not num.
- E023-Not min<like<worst.
- E024-Bytes required.
- E025-Rec id not found.
- E059-Recid used for set.
- E060-Recid used for area.

Consult Section VIII for more detailed descriptions of these error messages.



```
                      DATA BASE DOCUMENTOR AND EDITOR
                              SET MAINTENANCE

Data Base: DEMO              Description: DEMONSTRATION DATA BASE

Action: A

Set Identifier: PLANT-PP        Owner Record: PLANT
Set Description: PROD PLANT WITHIN PLANT

Pointers: Next(Y/N): Y   Prior(Y/N): Y   Owner(Y/N): Y
Sorted(Y/N): N   Insertion Order (N/F/L): N (NEXT)

  Member Record    D-I   Pct.Part   Min.Chain   Likely.Chain   Worst.Chain
-  PROD-PLANT      OA    100        1              10             35
-
-
-
-
-                                                             Update? (X):
Enter data and X to update.  ESC to erase input.
```

Figure 34

Set Maintenance Screen Fields

E. Set Maintenance Screen Field Descriptions (see Figure 34).

32. Action code. A = add a set, B = browse (view) a set, C = change a set's attributes, D = delete a set, Q = return to main menu.

33. Data Base. The data base identifier that qualifies the set identifier. This must be entered unless its value was carried forward by the system from another program. (12 characters).

34. Description. The description assigned to the data base identifier. This will be displayed upon the <Enter>ing of a valid data base identifier in field 33. (30 characters).

35. Set Identifier. (12 characters).

36. Set Description. A short description of the set identifier. (30 characters).

37. Owner Record. The valid DB-ADE record identifier, as established through the Record Maintenance Screen, that owns the set. (12 characters).

38. Pointers (Next, Prior, Owner). Enter Y or N in each field to indicate the existence of the pointers within the set. (each 1 character).

39. Sorted (Y/N). Enter Y or N to indicate if the set is sorted. (1 character).

40. Insertion Order. If field 39 was entered as Y, the system assigns a value of S (Sorted) to this field and skips over it. Otherwise, enter N (Next), F (First), or L (Last), to indicate where the stored member record is to be inserted into the set. (1 character).

41. Member Record (up to six occurrences). The valid DB-ADE record identifier(s), as established through the Record Maintenance Screen, that is (are) member(s) in the set. (12 characters).

42. D-I (up to six occurrences). The deletion-insertion rule for the member record. Enter MA (mandatory-automatic), OA (optional-automatic), MM (mandatory-manual), or OM (optional-manual). (2 characters).

43. Pct.Part. (up to six occurrences). Of all the occurrences of the set member record to be stored in the data base, the percentage that is expected to participate as members in this set at a typical point in time. (4 characters, numeric).

44. Min.Chain, Likely.Chain, Worst.Chain (up to six occurrences). The minimum, most likely, and worst case number of occurrences of the member record that will participate in one specific chain of the set at a typical point in time. (8 characters, numeric).

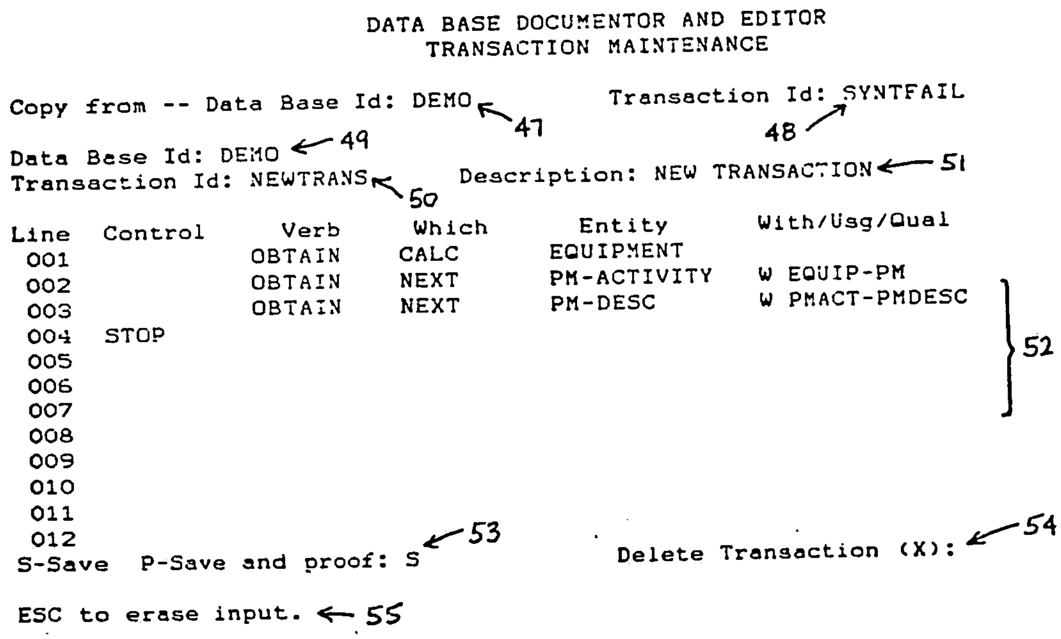
45. Update? (X). <Enter>ing an "X" in this field causes the add, change, or delete update process to begin. (1 character).

46. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Set Maintenance Screen include the following:

- E001-Db id required.
- E003-Db id not found.
- E026-Set id exists.
- E027-Next ptr not Y,N.
- E028-Prior ptr not Y,N.
- E029-Owner ptr not Y,N.
- E030-Sorted ind not Y,N.
- E031-Inst ord not N,F,L.
- E032-Own rec not found.
- E033-Mem rec not found.
- E034-D-I rule invalid.
- E035-Pct not <= 100.
- E036-Member = owner.
- E037-Min chain reqd.
- E038-Min chain not num.
- E039-Lik chain reqd.
- E040-Lik chain not num.

- E041-Worst chain reqd.
- E042-Worst chain not num.
- E043-Not min<like<worst.
- E044-Set id not found.
- E045-Set id required.
- E046-Owner rec required.
- E047-No member data.
- E048-Member rec reqd.
- E049-Pct. part. not num.
- E050-Pct. part. reqd.
- E051-Mand. must be 100.
- E052-No S if sorted set.
- E053-Next and prior = N.
- E061-Setid used for rec.
- E062-Setid used for area.

Consult Section VIII for more detailed descriptions of these error messages.



```
                     DATA BASE DOCUMENTOR AND EDITOR
                         TRANSACTION MAINTENANCE

Copy from -- Data Base Id: DEMO         Transaction Id: SYNTFAIL

Data Base Id: DEMO
Transaction Id: NEWTRANS      Description: NEW TRANSACTION

Line  Control     Verb    Which     Entity        With/Usg/Qual
 001              OBTAIN  CALC      EQUIPMENT
 002              OBTAIN  NEXT      PM-ACTIVITY   W EQUIP-PM
 003              OBTAIN  NEXT      PM-DESC       W PMACT-PMDESC
 004  STOP
 005
 006
 007
 008
 009
 010
 011
 012
S-Save  P-Save and proof: S            Delete Transaction (X):

ESC to erase input.
```

Figure 35

Transaction Maintenance Screen Fields

F. Transaction Maintenance Screen Field Descriptions
(see Figure 35).

47. Copy from Data Base Id. The data base identifier that qualifies the Copy from Transaction Id should the Copy from Transaction Id be entered. When doing a transaction copy, this field must be entered unless its value was carried forward by the system from another program. (12 characters).

48. Copy from Transaction Id. When entered, the transaction identifier of the transaction, qualified by the Copy from Data Base Id, that is to be retrieved onto the Transaction Maintenance Screen. (12 characters).

49. Data Base Id. The data base identifier that qualifies the transaction being defined on the Transaction Maintenance Screen. This field must be entered unless its value was carried forward by the system from another program. (12 characters).

50. Transaction Id. The identifier of the transaction, qualified by the Data Base Id, that is being defined on the Transaction Maintenance Screen. (12 characters).

51. Description. A short description of the transaction being defined. (30 characters).

52. Transaction definition area. See Section IV for description of use and Figure 28 for valid statement syntaxes.

53. Save/Save and proof indicator. An <Enter>ing of value S saves/resaves the transaction in the DB-ADE files. An <Enter>ing of value P saves/resaves the transaction and automatically invokes the Transaction Proof Program. (1 character).

54. Delete Transaction (X). An <Enter>ing of an "X" in the field deletes the transaction from the DB-ADE files. (1 character).

55. Message line. The message line will contain operator instruction messages and error messages as appropriate. Possible errors that can occur during the use of the Transaction Maintenance Screen include the following:

- E001-Db id required.
- E003-Db id not found.
- E054-Trans id required.
- E055-Copy db not found.
- E057-Copy trn not found.
- E058-Copy db spaces.

Consult Section VIII for more detailed descriptions of these error messages.

VIII. DB-ADE Error and Warning Messages Listing.

Message	Meaning
E001-DB id required.	The data base identifier has been left blank.
E002-DB id exists.	An attempt is being made to add a data base identifier that is already on file.
E003-DB id not found.	The data base identifier does not exist on the DB-ADE files.
E004-Area id required.	The area identifier has been left blank.
E005-Area exists for DB.	The area identifier already exists on file within the entered data base identifier.
E006-Area not in DB.	The area identifier does not exist within the entered data base identifier on the DB-ADE files.
E007-Rec id required.	The record identifier has been left blank.
E008-Rec id exists.	The record identifier already exists on file within the entered data base identifier.
E009-Locmode not C,V,D.	Location mode must be entered and must be C (calc), V (via), or D (direct).
E010-Set required when VIA.	The location mode is indicated as V (via), but the stored-by set field has been left blank.
E011-Dupsind required.	Duplicates-allowed indicator has been left blank.

E012-Numdups required.
Number of duplicate records expected field has been left blank.

E013-Dupsind not Y,N.
The duplicates-allowed indicator is non-blank, but does not contain allowable values Y or N.

E014-Numdups not > 1.
If duplicates-allowed indicator is Y, the expected number of duplicate records must be at least 2.

E015-Numdup not numeric.
The number of expected duplicate records field is non-blank but contains a non-numeric value.

E016-Bytes not numeric.
The non-pointer bytes field is non-blank but contains a non-numeric value.

E017-Min vol reqd.
The expected minimum number of record occurrences field has been left blank.

E018-Likely vol reqd.
The most likely number of record occurrences field has been left blank.

E019-Worst vol reqd.
The worst case number of record occurrences field has been left blank.

E020-Min vol not num.
The expected minimum number of record occurrences field is non-blank but contains a non-numeric value.

E021-Likely vol not num.
The most likely number of record occurrences field is non-blank but contains a non-numeric value.

E022-Worst vol not num.

The worst case number of record occurrences field is non-blank but contains a non-numeric value.

E023-Not min<like<worst.

The numerical values inputted do not satisfy the requirement that expected minimum number of record occurrences ≤ most likely number of record occurrences ≤ worst case number of record occurrences.

E024-Bytes required.

The non-pointer bytes field has been left blank.

E025-Rec id not found.

The record identifier does not exist within the entered data base identifier on the DB-ADE files.

E026-Set id exists.

The set identifier already exists on file within the entered data base identifier.

E027-Next ptr not Y,N.

The next pointers indicator must be entered and must be Y or N.

E028-Prior ptr not Y,N.

The prior pointers indicator must be entered and must be Y or N.

E029-Owner ptr not Y,N.

The owner pointers indicator must be entered and must be Y or N.

E030-Sorted ind not Y,N.

The sorted set indicator must be entered and must be Y or N.

E031-Inst ord not N,F,L.

When keyed in by the DB-ADE user, the insertion order field must be N (next), F (first), or L (last).

E032-Own rec not found.

The owner record field contents is not a valid DB-ADE record identifier within the entered data base identifier.

E033-Mem rec not found.

One of the specified member record field contents is not a valid DB-ADE record identifier within the entered data base identifier.

E034-D-I rule invalid.

One of the entered deletion-insertion rules is not MA, OA, MM, or OM.

E035-Pct not <= 100.

One of the entered percentage participation values is greater than 100 (percent).

E036-Member = owner.

A valid DB-ADE record identifier is specified as both the owner record and a member record in the set.

E037-Min chain reqd.

The expected minimum number of records in the typical set chain has been left blank for a specified member record.

E038-Min chain not num.

The expected minimum number of records in a typical set chain is non-blank but contains a non-numeric value.

E039-Lik chain reqd.

The most likely number of records in the typical set chain has been left blank for a specified member record.

E040-Lik chain not num.

The most likely number of records in a typical set chain is non-blank but contains a non-numeric value.

E041-Worst chain reqd.

The worst case number of records in the typical set chain has been left blank for a specified member record.

E042-Worst chain not num.

The worst case number of records in a typical set chain is non-blank but contains a non-numeric value.

E043-Not min<like<worst.

For a specified member record, the numerical values inputted do not satisfy the requirement that the expected minimum number of records in the chain $\leq$ most likely number of recordes in the chain $\leq$ worst case number of records in the chain.

E044-Set id not found.

The set identifier does not exist within the entered data base identifier on the DB-ADE files.

E045-Set id required.

The set identifier has been left blank.

E046-Owner rec required.

The set woner record filed has been left blank.

E047-No member data.

All six allowable member record line items are completely blank.

E048-Member rec required.

At least one field on a member record line item is non-blank, yet the member record identifier field is blank.

E049-Pct. part. not num.

One of the entered percentage paticipation values is non-blank and contains a non-numeric value.

E050-Pct. part. reqd. — The percentage participation field for an entered set member record identifier has been left blank.

E051-Mand. must be 100. — The percentage participation value for a mandatory member (i.e., MA or MM) must be 100 (percent).

E052-No S if sorted set. — The insertion order of S generated by the system (from user entry of a sorted set indicator of Y) has been overlayed with a non-S value, yet the sorted set indicator is still Y.

E053-Next and prior = N. — Next and prior pointer indicators are both specified as N. This means the set could never be traversed.

E054-Trans id required. — The transaction identifier has been left blank.

E055-Trans id not found. — *** CURRENTLY NOT INVOKED ***

E056-Copy db not found. — The data base identifier entered on the COPY FROM line does not exist on the DB-ADE files.

E057-Copy trn not found. — The transaction identifier entered on the COPY FROM line does not exist within the entered data base identifier on the DB-ADE files.

E058-Copy db spaces. — The data base identifier entered on the COPY FROM line has been left blank, yet a COPY FROM transaction identifier has been entered.

E059-Recid used for set.

The record identifier is already in use as a set identifier within the entered data base identifier.

E060-Recid used for area.

The record identifier is already in use as an area identifier within the entered data base identifier.

E061-Setid used for rec.

The set identifier is already in use as a record identifier within the entered data base identifier.

E062-Setid used for area.

The set identifier is already in use as an area identifier within the entered data base identifier.

E063-Areaid used for rec.

The area identifier is already in use as a record identifier within the entered data base identifier.

E064-Areaid used for set.

The area identifier is already in use as a set identifier within the entered data base identifier.

E400-Invalid statement syntax.

The transaction statement does not comply with DB-ADE syntax rules (see Figure 28).

E401-Invalid control or verb form.

The transaction statement contains neither a valid control form nor a valid verb form (see Figure 27).

E402-SORTKEY within area search invalid.

SORTKEY is not an allowable search technique for a FIND/ OBTAIN within a data base area.

E403-Record is not in specified area.

The record for which a FIND/ OBTAIN within a data base area is being issued is not stored within the specified data base area.

E404-Record is not a member in specified set.

The record for which a FIND/ OBTAIN within a set is being issued is not a member record within the specified set.

E405-Specified set is not sorted.

SORTKEY is not an allowable search technique for a FIND/ OBTAIN within an unsorted set.

E406-Specified set does not have NEXT pointers.

FIND/OBTAIN NEXT/FIRST within a set requires that the set have next pointers.

E407-Specified set does not have PRIOR pointers.

FIND/OBTAIN PRIOR/LAST within a set requires that the set have prior pointers.

E408-Specified record is not CALC.

FIND/OBTAIN CALC/DUP requires that the object record be stored with a location mode of CALC.

E409-Specified CALC record does not allow dups.

FIND/OBTAIN DUP is not allowable for a CALC record that does not allow duplicates.

E410-CONNECT/DISCONN not valid for D-I rule.

CONNECTS are allowed only for MM and OM sets. DISCONNects are allowed only on OA and OM sets.

E411-ACCEPT format requires set PRIOR pointers.

ACCEPT NEXT/PRIOR currencies within a set formats require that the object set have prior pointers.

E500-END is unassignable to IF or ELSE.

*** CURRENTLY NOT INVOKED ***

E501-STOP statement missing.	There is no STOP control statement to indicate the end of the transaction.
E502-Statement after first STOP ignored.	*** CURRENTLY NOT INVOKED ***
E503-IF or ELSE statement not closed.	*** CURRENTLY NOT INVOKED ***
E504-No IF for ELSE to belong to.	There is no IF control statement preceding the ELSE statement that has yet to be counter-conditioned by an ELSE statement.
E700-Dbkey of record type not previously ACCEPTed.	To use the FIND/OBTAIN using a saved dbkey as a direct access mechanism requires that the dbkey has been ACCEPTed by a prior statement into a storage field.
E701-Record type currency not prev. established.	Execution of the transaction statement requires that current of record type be previously established.
E702-Set currency not previously established.	Execution of the transaction statement requires that current of set be previously established.
E703-Area currency not previously established.	Execution of the transaction statement requires that current of area be previously established.
E704-Set partic. check must precede statement.	An "if member of set" check must precede an OBTAIN OWNER statement if the object record

participates in the object set as an optional or manual member.

E705-Preceding set partic. check for diff. set.

Reference error message for error code 704. The participation check preceding the OBTAIN OWNER statement is for a different object set.

E706-Object record must be current-of-run-unit.

Execution of the transaction statement requires that the object record be current-of-run-unit.

E707-Currency not est. in all rec OA, MA, Via set.

This is a currency requirement prior to the storing of an IDMS record.

E708-Current-of-run-unit not in specified set.

"If set member" and "If set empty" checks require that the current-of-run-unit record participate in the object set.

W001.

On a successful DB-ADE record add or change, precedes the "Files updated." message when an input-empty screen is redisplayed. Signifies that the set by which the record is stored VIA is not yet identified in DB-ADE files.

W700-Dbkey of rec not prev. accepted, -1 assumed.

On a STORE of a record type whose location mode is DIRECT, the record will be stored as close to the end of its area as is possible unless a dbkey is supplied for the STORE via a prior ACCEPT statement.

W701-Caution for mand. mems in sets owned by rec.

Highlighted when issuing an ERASE statement.

W702-Caution for DB-END-OF-SET on set walks.

When a transaction path has at least one FIND/OBTAIN within set statement, warns that an application program must handle the "on db-end-of-set" condition.

W703-Caution for DB-END-OF-SET on area walks.

When a transaction path has at least one FIND/OBTAIN within area statement, warns that an application program must handle the "on db-end-of-set" condition (end of area).

W704-Caution for DB-REC-NOT-FOUND on CALCs.

When a transaction path has at least one CALC/DUP statement, warns that an application program must handle the "on db-rec-not-found" condition.

Program abort diagnostic messages - require program changes and regeneration of run modules.

- IFsub exceeds IF table max. entries.
- Pathing table filled up in X direction.
- Transaction If not found in If table -- internal error.
- Pathing table overcompressed in Y-direction.
- Program problem -- Allpathtable is filled to maximum.
- Program problem -- Settable is filled to maximum.
- Program problem -- Currentofareatable is filled to maximum.
- Program problem -- Currentofsettable is filled to maximum.
- Program problem -- Currentofrecordtable is filled to maximum.
- Program problem -- Pathentrytable is filled to maximum.
- Program error, PAT record for block-path not

IX. Location of DB-ADE Demonstration System.

In compliance with thesis preparation guidelines, the following:

- the two DB-ADE system object code diskettes,
- two DB-ADE source code diskettes,
- a DB-ADE data files demonstration diskette,
- DB-ADE program source listings,

are on file in the office of thesis advisor:

Professor John C. Wiginton
Department of Industrial Engineering
Packard Laboratory 19
Lehigh University
Bethlehem, Pa. 18015.

X. Directions for Further Research.

The following represent the author's current thoughts on possible future extensions of DB-ADE system functionality.

AUTOMATED DATA DESCRIPTION LANGUAGE (DDL) INTERFACE TO THE INTEGRATED DATA DICTIONARY (IDD)

An expansion of the attributes that must be specified for the entity types (i.e., areas, records, sets) defined in the Data Base Definition Subsystem can provide the syntax requirements for the source code input of the Data Description Language into a mainframe IDMS Integrated Data Dictionary (IDD). This expansion, plus the development of an automated load of Data Base Definition Subsystem data into the proper mainframe IDD source statement library, can eliminate the clerical effort required to code and load the DDL source statements into the source statement library for subsequent processing by the DDL compiler.

VOLUME PROOFING

A new DB-ADE program module can gather the minimum, most likely, and worst case volume estimates provided for record and set entities, and insure that the estimates for structurally interrelated entities are reasonably compatible.

PHYSICAL DATA BASE ACCESS PERFORMANCE EVALUATOR

Statistical models can be developed and programmed within new DB-ADE program modules that will evaluate the performance (i.e., efficiency) of a transaction against the defined data base structure. Performance criteria must first be developed. Logical performance measurements might be logical I/O counts, physical I/O counts, and the number of physical IDMS page reads per physical or logical IDMS page write (page update). The Data Base Definition Subsystem must be expanded to accept the appropriate data base physical attributes. This evaluator module will allow the user to predict performance and to fine-tune the data base structure and transactions as required.

DATA BASE PATHFINDER

A set of program modules can be developed that will determine the transactions that can be implemented to acquire a set of required data. This will first require an expansion of the Data Base Definition Subsystem Record

Definition Program to allow the defining of data elements for each record. Secondly, a new version of the Transaction Maintenance Program must be developed that requests that the data elements that are to be updated/retrieved by the transaction be identified. A program module will then be able to determine, based upon the data base structure, all feasible paths that will obtain or update the data elements. A marriage of the aforementioned Physical Data Base Access Performance Evaluator and this module can possibly enable the program module to recommend the best access path based upon the relative efficiencies of the feasible paths.

- BIBLIOGRAPHY -

Atre, S., Data Base: Structured Techniques for Design, Performance, and Management. New York: John Wiley and Sons, 1980.

Borland International, Inc., Turbo Pascal: Version 2.0 Reference Manual, 5th edition, October 1984.

Borland International, Inc., Turbo Toolbox: Power Tools for Turbo Pascal, 1st edition, July 1984.

Borland International, Inc., Turbo Tutor: A Turbo Pascal Tutorial, 1st edition, November 1984.

Cullinane Corporation, IDMS Concepts and Facilities, 1977.

Cullinane Corporation, IDMS Programmer's Reference Guide, Revision 1, Release 5.0, November 1978.

Cullinet Software, Inc., IDMS Programmer's Reference Guide - COBOL, Revision 1.1, Release 5.7, December 1983.

King, Richard Allen, The IBM PC-DOS Handbook. Berkeley, California: SYBEX, Inc., 1983.

Lieberman, Arthur Zale. "Methodology for the Automation of the Audit Process Involving the Evaluation of the Plan of Internal Control." Ph.D. dissertation, University of Arizona, 1977. Published by Xerox University Microfilms, Ann Arbor, Michigan.

Martin, James, Computer Data-Base Organization. 2nd edition. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.

VITA

The author was born in Palmerton, Pennsylvania on January 18, 1956, the son of John Allen and Mary Branch. The author lived there until graduating with honors from Palmerton Area High School in June of 1973. From 1973 to 1977 he attended Rutgers University, New Brunswick, New Jersey, from which he graduated magna cum laude in June of 1977 with a B.A. in Economics. Since June of 1977 he has been employed in the Management Information Services department of Air Products and Chemicals, Inc., Allentown, Pennsylvania, where he currently holds the title of Senior Systems Analyst.