Definition of database transactions by the casual user

by FRED J. MARYANSKI* and C. STEVEN ROUSH**

Computer Science Department
Kansas State University
Manhattan, Kansas

INTRODUCTION

Database management systems have been in general use for more than a decade. However, only recently have advances in technology and reduction in cost made such systems feasible for the small enterprise. Experience with database systems over the years indicates that the definition of a schema, design and implementation of a set of application programs, and maintenance of the system are far from trivial tasks. Consequently, the ability to perform effectively any of the above three tasks has become a highly marketable skill.

The need to make database systems accessible to the casual, or nonprogramming, user was first addressed by Codd [9] and has been the subject of considerable recent work [6, 7, 10-13, 15-17, 19-21, 23-26]. Much of the effort in this area has centered upon the development of interactive query languages for the specification of ad hoc database transactions. In this document, we report the results of an effort to permit the casual user to specify a complete set of transactions for an enterprise. This transaction definition subsystem produces a set of "canned" transactions to which the user supplies any runtime parameters.

The transaction definition subsystem is one unit in an application development system designed to facilitate database utilization by the nonprogramming users. The goal of the project is to create a database system that can be comprehended and effectively utilized by a user whose enterprise cannot support a data processing professional. It is assumed that the user of the application development system is unfamiliar with the concepts of database management but highly knowledgeable concerning the data of the enterprise. The system is highly interactive and relies upon the user to supply all information on the data items, relationships among data items, and operations on the database.

In a prior effort a subsystem that permits a casual user to interactively create a third normal form schema has been developed [3, 14, 22]. The outputs of this subsystem are used in the transaction definition process. Figure 1 depicts the present status of the application development system.

The remainder of this paper concentrates upon the transaction definition subsystem. The next section contains an overview of the methodology employed to define the transactions. The structure of the subsystem is elaborated upon in the third section by describing the processing and interaction that take place. The concluding section describes possible future efforts in this area. A sample transaction definition session is included in the appendix.

OVERVIEW

The function of the transaction definition subsystem is to receive a description of a document and through interaction with the user, create a transaction which generates the document. The definition of a transaction is dependent upon the presence of a hierarchical description of the document that the transaction is to produce and a third normal formal schema which embodies all noncomputed data items on the document. The hierarchical description and schema are outputs of the data definition subsystem which must be completed prior to the execution of the transaction definition subsystem.

The document descriptor indicates all data items on the document and the hierarchical structure of the document. The hierarchical structure is a tree representing the logical organization of the document. The nodes of the tree are the unique lines of the document. Figure 2 illustrates the structure of the document descriptor. Detailed information on the construction of the document descriptor can be found in Reference [14].

The feature that distinguishes this approach to transaction definition from other methods is that the user need not be aware of the structure of the database when the transactions are defined. However, we must emphasize that the user is required to be thoroughly familiar with the meaning and organization of the data on the documents of the enterprise.

The transaction definition subsystem is highly interactive. The user is questioned in order to obtain information on the semantic nature of the transactions. Figure 3 depicts the structure of the transaction definition subsystem. The function of each of the modules is explained in the succeeding section.

The output of the transaction definition subsystem is a modified form of SEQUEL2 [5] code intended for execution by a relational database management system.

* Address: Digital Equipment Corp., Maynard, MA 01754
** NCR Corp., Witchita, Kansas 57226

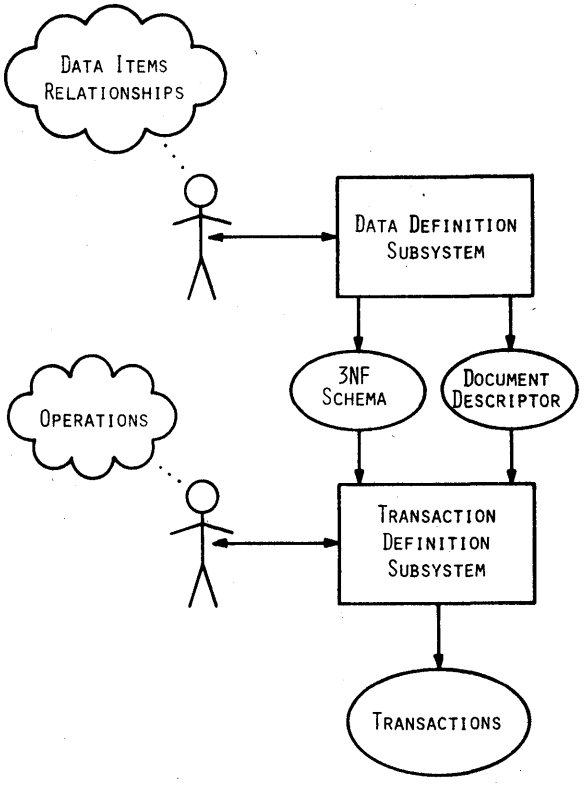
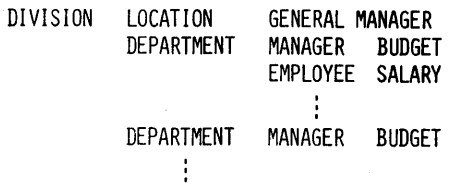
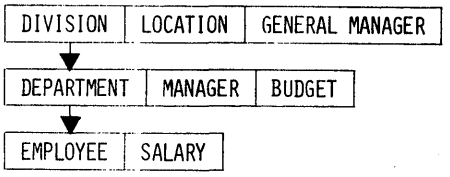


Figure 1—Application development system



A. SAMPLE DOCUMENT



B. HIERARCHICAL STRUCTURE

NAME	LEVEL	GROUP	FLAG
DIVISION	1	110	1 (KEY)
LOCATION	1	110	0 (NON-KEY)
GENERAL MANAGER	1	110	0
DEPARTMENT	2	210	1
MANAGER	2	210	0
BUDGET	2	210	0
EMPLOYEE	3	310	1
SALARY	3	310	0

C. DOCUMENT DESCRIPTOR

Figure 2—Document descriptor example

STRUCTURE

In this section we explain the workings of the transaction definition subsystem by detailing the functions of the modules pictured in Figure 3. For purposes of the presentation, the operation of the subsystem is explained initially for a retrieval transaction. The last portion of the section indicates the differences involved in the definition of an update transaction. A complete description of the transaction definition subsystem is available in Reference [18].

Initialization

The schema and document descriptor are read as input, and various internal tables are contructed using this information. The HANDLE-TRANSACTIONS module calls the PROCESS-LINE module in an iterative manner in order to generate the code for each document in the user's application system.

Identification of primary relation

A key concept of the transaction definition subsystem is the identification of the relation closest in terms of domains to the content of each line of the document to be produced. This relation is termed the *primary relation* for the line. The primary relation is used as the starting point for the navi-

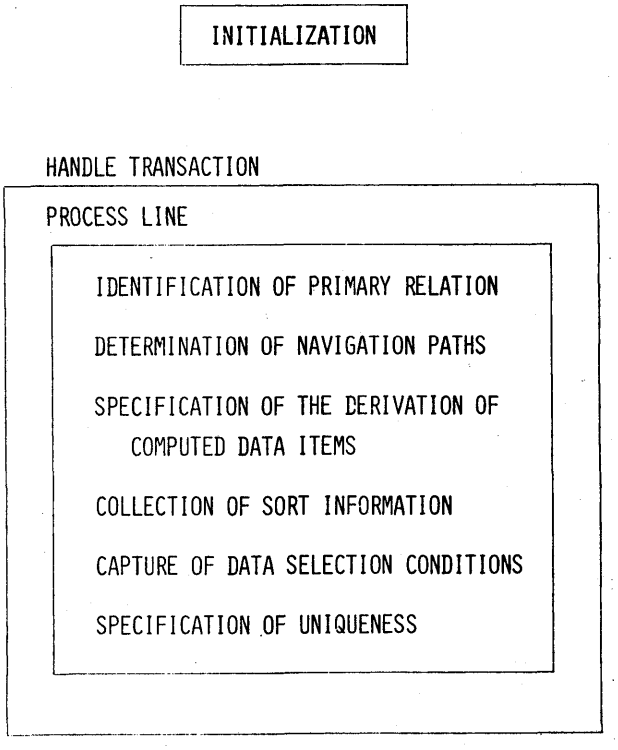


Figure 3—Structure of transaction definition facility

gation paths that link all data on the line. The determination of the primary relation is made without interaction using information in the schema and document descriptor. Figure 4 is a high level description of the primary relation identification algorithm. The primary relation is the starting point for all further processing by the transaction definition subsystem.

As indicated in the high level description, the algorithm compares the contents of the lines and the relations. It is possible that two or more relations may be equally close, according to the metric of the algorithm, to a given line. In this situation the program maintains multiple primary relations and begins navigation from all primary relations in order to cover all elements in a line of a document.

Determination of navigation paths

In order to produce a transaction without requiring that the user have a knowledge of the schema, the system must be able to determine navigation paths from the primary relation to relations containing all data items in the line being processed. The algorithm for the determination of a navigation path for retrieval operations is portrayed in Figure 5. The algorithm involves searching through relations beginning at the primary relations until all data items for the line have been reached. Although the implementation is different, the algorithm for the determination of the navigation paths is conceptually similar to Bernstein's membership algorithm [1, 2].

DETERMINE A REASONABLE SET OF CANDIDATE DETERMINANTS OF THE LINE

1. ELIMINATE ELEMENTS NOT MARKED AS POSSIBLE DETERMINANTS BY THE USER
2. ELIMINATE ELEMENTS NOT KEY TO ANY RELATION
3. IDENTIFY ELEMENTS WHICH ONLY APPEAR IN RELATIONS AS KEYS
4. ELIMINATE ELEMENTS WHICH NEVER APPEAR IN A CONCATENATED KEY WITH THE ABOVE ELEMENTS

DETERMINE A SET OF CANDIDATE RELATIONS

1. START WITH EVERY RELATION WITH ANY CANDIDATE DETERMINANT IN ITS KEY
2. ELIMINATE THOSE RELATIONS WITHOUT FULLY COVERED KEYS
3. ELIMINATE ANY CANDIDATE RELATION WHICH CAN BE COVERED BY ANOTHER

REMAINING RELATION IS THE PRIMARY RELATION

Figure 4—Primary relation algorithm

```
NAVIGATION
   FOR EACH ELEMENT INVOLVED
      IF FOUND IN PRIMARY RELATION
         STOP--THAT'S IT
      ELSE
         IF IT APPEARS IN ONLY ONE RELATION AS NONKEY
            STOP--THAT'S IT
         ELSE
            FOR EACH RELATION IN WHICH IT APPEARS AS NON-KEY
               IF ALL THE KEYS OF THAT RELATION ARE NOT
                     IN THE LINE
                  ELIMINATE THAT RELATION FROM CANDIDACY
               ENDIF
            ENDLOOP
            IF ONLY ONE CANDIDATE LEFT
               STOP--USE IT
            ELSE
               CAN'T BE DETERMINED
```

Figure 5—Algorithm for navigation paths

Specification of the derivation of computed data items

During the execution of the data definition subsystem, the user has identified all data items which are computed and consequently not included in the schema. This information is preserved in the document descriptor which is an input to the transaction definition subsystem. The user is requested to indicate the data items used in the derivation of the value of each computed data item. In the current version of the prototype only the operands, but not the operators, of the expression for the computed data item's value need be specified. Alternatives for the capture of the complete expression for a computed data item are being investigated.

Collection of sort information

The user is asked to supply any sort keys in major to minor order and the sorting sequence for each key.

Capture of data selection conditions

The user is requested to indicate if he wishes all instances of the data items on the line or if he wishes to apply selection criteria to the data. Again, a reasonably straightforward interaction, as illustrated in Figure 6, is utilized to capture the data selection conditions.

Another limitation of the prototype affects the operands

```
DO YOU WANT THIS TRANSLATION TO INVOLVE EVERY
OCCURRENCE OF THE DATA FOR THIS LINE? (Y/N)
('N' IF THERE IS ADDITIONAL SELECTION CRITERIA)

                (ACCEPT ANSWER)

OF THESE ELEMENTS IN THE LINE

                (DISPLAY LINE)

WHICH IS INVOLVED IN THIS CONDITION?

                (ACCEPT ITEM USED IN SELECTION)

PLEASE ENTER CONDITIONAL OPERATOR (=, <, >, ETC.)

                (ACCEPT OPERATOR)

FOR NOW, THE 'RIGHT-HAND SIDE' WILL COME FROM THE
CRT
```

Figure 6—Capture of data selection conditions

of the selection expressions. In the case of selection criteria, one operand must be a data item on the line being processed. In the prototype, the other operand is restricted to being a value accepted from the keyboard. Many possibilities exist for the source and format of the second operand. The most difficult situation arises when the second operand must be retrieved from the database. The main difficulty here is the specification of the source of the operand by the user. This problem is currently under study.

Specification of uniqueness

Depending upon the selection criteria, an operation may retrieve tuples containing duplicate values. In some situations, the user may desire to observe only unique tuples satisfying the selection criteria. This option is provided interactively to the user. Based upon the response to the uniqueness question, a command that will retrieve either unique or duplicate tuples is generated.

Update transactions

As shown in Figure 7, the overall structure of the algorithms for the production of transactions that write to the database is similar to that of the algorithm for read-only transactions. In a retrieval transaction, it is necessary to establish the existence of navigation paths that reach at least one occurrence of each data item to be retrieved. When a data item is to be written (stored, updated, or deleted), all occurrences of that data item must be located.

In a third normal form database, the problem of multiple occurrences exists only for keyed data items. The existence of a third normal form schema implies that for a given set of keys, the tuple containing that data item must be uniquely determined [4, 8]. However, keyed items may exist in multiple relations. Therefore, the navigation path routines include the ability to locate all instances of keyed data items. Stored and deletion transactions will always operate upon keyed data items. At the present time, the ability to update a keyed data item is not provided.

Example

The appendix contains a comprehensive example of the input, user interaction, and output of the transaction definition subsystem. The information in the appendix should be self-explanatory. The only non-SEQUEL2 statements produced as output are the SELECT(NEXT) and SELECT(ALL) which are iterative retrieval statements. A SELECT(NEXT) statement in effect defines a loop in which one tuple is retrieved and then, in this example, the following ACCEPT and SELECT statements are executed. A SELECT(ALL) statement causes all tuples that satisfy the selection criteria to be retrieved. In this example, all grades will be listed on a student-by-student basis.

CONCLUSION

Summary

The application development system described here is oriented toward the casual user or the small businessman with the need for a computer but not a programming staff. The system captures the data from the user and then interactively synthesizes a third normal form schema and the transactions which operate upon the schema. A distinguishing feature of this system is that the user is not required to reference the schema in the definition of the transactions.

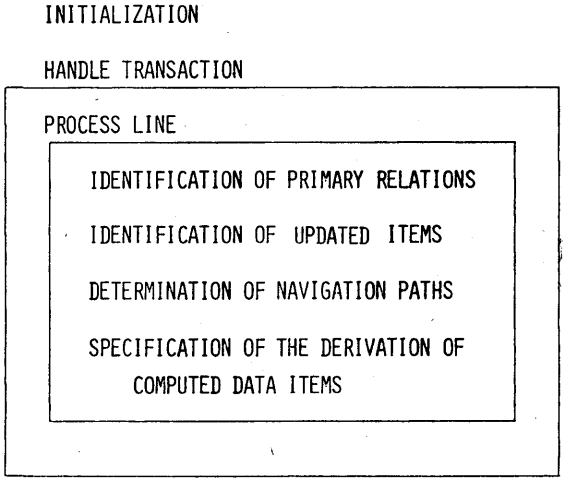


Figure 7—General organization of write transactions

Further work

At present the system's final output is a set of transactions in a modified form of SEQUEL2. Extensions are planned to incorporate maintenance utilities into the system. The addition of these features would permit the user to construct an entire application system.

As mentioned in the body of this paper, certain limitations presently exist in terms of the operations available within the transactions that may be defined. These restrictions are a reflection of the prototype status of the system and are expected to disappear as further work on the system transpires.

An important feature of any system with a high degree of interactivity is the understandability of the questions by the user. An effort has been made to involve representative users in the design of the interactive phases of the system. It is hoped that the effect of this involvement will be easy understanding of the system by the casual user. If the user can interact easily with the system, then the database and transactions can be designed without requiring special training.

REFERENCES

1. Beeri, C. and Bernstein, P. A., "Computational Problems Related to the Design of Normal Form Relational Schemas," *ACM TODS* (4, 1), Mar. 1979, pp. 30-59.
2. Bernstein, P. A., "Synthesizing Third Normal Form Relations from Functional Dependencies," *ACM TODS* (1, 4), Dec. 1976, pp. 277-298.
3. Buser, R. H. and Kusnyer, S. K., "Automatic Synthesis of Third Normal Form Relations," M. S. Report, Comp. Sci. Dept., Kansas State U., Dec. 1977.
4. Chamberlin, D. D., "Relational Data-Base Management Systems," *Computing Surveys* (8, 1), Mar. 1976, pp. 43-66.
5. Chamberlin, D. D. et al., "SEQUEL2: A Unified Approach to Data Definition, Manipulation, and Control," *IBM Journal R&D* (20, 6), Nov. 1976.
6. Chang, S. K. and Ke, J. S., "Translation of Fuzzy Queries for a Relational Database System," Knowledge Systems Lab., U. of Illinois at Chicago Circle, Mar. 1978.
7. Christensen, M. A. and Herndon, M. A., "QUEASY: The Design and Implementation of a Management Information System for Casual Users," *ACM Annual Conf.*, Dec. 1978, pp. 230-233.
8. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *CACM* (13, 6), June 1970, pp. 377-387.
9. Codd, E. F., "Seven Steps to Rendezvous with the Casual User," in *Data Base Management*, (J. W. Klimbie and K. L. Koffeman, eds.), North-Holland, Apr. 1974, pp. 179-200.
10. Codd, E. F., "How About Recently," in *Databases: Improving Usability and Responsiveness* (B. Shneiderman, ed.), Academic Press, 1978, pp. 3-28.
11. Greenblatt, D. and Waxman, J., "A Study of Three Database Query Languages," in *Databases: Improving Usability and Responsiveness* (B. Shneiderman, ed.), Academic Press, 1978, pp. 77-97.
12. Harris, L. R., "The ROBOT System: Natural Language Processing Applied to Data Base Query," *ACM Annual Conf.*, Dec. 1978, pp. 165-172.
13. Hendrix, G. G. et al., "Developing a Natural Language Interface to Complex Data," *ACM TODS* (3, 2), June 1978, pp. 105-147.
14. Hunt, W. O., "Interactive Generation of Functional Dependencies," M. S. Report, Comp. Sci. Dept., Kansas State U., Dec. 1977.
15. Lozinskii, E. L., "Performance Considerations in Relational Data Base Design," in *Databases: Improving Usability and Responsiveness* (B. Shneiderman, ed.), Academic Press, 1978, pp. 273-294.
16. Powell, P. B. and Thompson, P., "Natural Language and Voice Output for Relational Data Base Systems, *ACM Annual Conf.*, Dec. 1978, pp. 585-595.
17. Reisner, P., "Use of Psychological Experimentation as an Aid to Development of a Query Language," *IEEE Trans. on Soft. Eng.* (SE-3, 3), May 1977, pp. 218-229.
18. Roush, C. S., "A User-oriented Transaction Definition Facility For a Relational Database System," M. S. Report, Comp. Sci. Dept., Kansas State U., Aug. 1979.
19. Shen, S. N. T., "A Semantic Approach in Designing Relational Data Base," *ACM Annual Conf.*, Dec. 1978, pp. 596-601.
20. Shneiderman, B., "Improving the Human Factors Aspect of Database Interactions," *ACM TODS* (3, 4), Dec. 1978, pp. 417-439.
21. Sorenson, D. G. and Wald, J. A., "PICASSO-An Aid to an End-User Facility," *ACM SIGMOD Conf.*, Aug. 1977, pp. 30-39.
22. Stevens, T. J., "Implementation of a Text Editor for the Third Normal Formal Synthesis System," M. S. Report, Comp. Sci. Dept., Kansas State U., Dec. 1977.
23. Waltz, D. L., "An English Language Question Answering System for a Large Relational Database," *CACM* (21, 7), July 1978, pp. 526-539.
24. Weiner, J. L., "Deriving Data Base Specifications from User Queries," *Berkeley Workshop on Distributed Data Management and Computer Networks*, May 1977, pp. 182-195.
25. Zloof, M. M., "Query-By-Example," *National Computer Conf.*, Vol. 44, May 1975, pp. 431-437.
26. Zloof, M. M. and de Jong, S. P., "The System for Business Automation (SBA): Programming Language," *CACM* (20, 6), June 1977, pp. 385-396.

APPENDIX

1. Document Descriptor to STUDRECD Document

Name	Level	Group	Flag
STUD#	1	110	1
STUDNAME	1	110	0
GRADDATE	1	110	0
GPA	1	110	0
STUD#	2	210	1
COURSWORK	2	210	1
COURSGRADE	2	210	0

2. Relations

R01 STUD# STUDNAME GRADDATE GPA
R02 STUD# COURSWORK COURSGRADE

3. User Dialog with Transaction Definition Subsystem

```
 TDS:        STARTING NEW TRANSACTION          (REPORT)
 TDS:                      TRANSACTION-NAME:          STUDRECD
 TDS:     STUD#        STUDNAME        GRADDATE        GPA
 TDS: WHICH DO YOU WANT TO DISPLAY
 TDS:               Y—YES, THIS ITEM
 TDS:               N—NOT, THIS ITEM
 TDS:               A—ALL OF THE REST OF THE ITEMS
 TDS:               S—STOP, NONE OF THE REST OF THE ITEMS
 TDS: STUD#
USER:   Y
 TDS: STUDNAME
USER:   Y
 TDS: GRADDATE
USER:   Y
 TDS: GPA
USER:   Y
 TDS: DO YOU WANT THIS TRANSACTION TO INVOLVE EVERY
 TDS: OCCURRENCE OF THE DATA FOR THIS LINE? (Y/N)
 TDS: ('N' IF THERE IS ADDITIONAL SELECTION CRITERIA)
USER:   N
 TDS: OF THESE ELEMENTS IN THE LINE:
 TDS:     STUD#        STUDNAME        GRADDATE        GPA
 TDS: WHICH IS INVOLVED IN THIS CONDITION?
USER:     STUD#
 TDS: PLEASE ENTER CONDITION OPERATOR (=, <, >, ETC.)
USER:   =
 TDS: FOR NOW, THE 'RIGHT-HAND SIDE' WILL COME FROM
 TDS: THE CRT
 TDS: OF THESE ELEMENTS IN THE LINE:
 TDS:     STUD#        STUDNAME        GRADDATE        GPA
 TDS: WHICH IS INVOLVED IN THIS CONDITION?
USER:
 TDS:     STUD#        STUDNAME        GRADDATE        GPA
 TDS: DO YOU WANT THIS LINE SORTED? (Y/N)
USER:   Y
 TDS: ENTER SORT KEYS MAJOR TO MINOR
 TDS: STUD#
USER:   Y
 TDS: ASCENDING OR DESCENDING? (A/D)
USER:   A
 TDS: ARE THERE ANY MORE SORT KEYS? (Y/N)
USER:   N
 TDS:     STUD#        COURSWORK        COURSGRADE
 TDS: WHICH DO YOU WANT TO DISPLAY
 TDS:               Y—YES, THIS ITEM
 TDS:               N—NOT, THIS ITEM
 TDS:               A—ALL OF THE REST OF THE ITEMS
 TDS:               S—STOP, NONE OF THE REST OF THE ITEMS
 TDS: STUD#
USER:   N
 TDS: COURSWORK
```

```
USER:    Y
 TDS:  DO YOU WANT THIS TRANSACTION TO INVOLVE EVERY
 TDS:  OCCURRENCE OF THE DATA FOR THIS LINE? (Y/N)
 TDS:  ('N' IF THERE ARE ADDITIONAL SELECTION CRITERIA)
USER:   N
 TDS:  OF THESE ELEMENTS IN THE LINE:
 TDS:      STUD#          COURSWORK          COURSGRADE
 TDS:  WHICH IS INVOLVED IN THIS CONDITION?
USER:      STUD#
 TDS:  PLEASE ENTER CONDITION OPERATOR (=, <, >, ETC)
USER:    =
 TDS:  FOR NOW, THE 'RIGHT HAND SIDE' WILL COME FROM
 TDS:  THE CRT
 TDS:  OF THESE ELEMENTS IN THE LINE:
 TDS:      STUD#          COURSWORK          COURSGRADE
 TDS:  WHICH IS INVOLVED IN THIS CONDITION?
USER:   COURSWORK
 TDS:  PLEASE ENTER CONDITION OPERATOR (=, <, >, ETC)
USER:    =
 TDS:  FOR NOW, THE 'RIGHT HAND SIDE' WILL COME FROM
 TDS:  THE CRT
 TDS:  OF THESE ELEMENTS IN THE LINE:
 TDS:      STUD#          COURSWORK          COURSGRADE
 TDS:  WHICH IS INVOLVED IN THIS CONDITION?
USER:
 TDS:      COURSWORK          COURSGRADE
 TDS:  DO YOU WANT THIS LINE SORTED? (Y/N)
USER:    Y
 TDS:  ENTER SORT KEYS MAJOR TO MINOR
 TDS:  COURSWORK
USER:    Y
 TDS:  ASCENDING OR DESCENDING? (A/D)
USER:    A
 TDS:  ARE THERE ANY MORE SORT KEYS? (Y/N)
USER:    N
 TDS:  DO YOU WANT TO DISPLAY ALL VALID RECORDS,
 TDS:            OR JUST THOSE THAT ARE UNIQUE? (A/U)
USER:    A
```

4. Output

```
TRANSACTION-NAME        STUD-RECD
ACCEPT PARAMETER-01     FROM CRT
SELECT (NEXT)
    R01.STUD
    R01.STUDNAME
    R01.GRADDATE
    R01.GPA
FROM
    R01
WHERE
    R01.STUD# = PARAMETER-01
ORDERED BY
    R01.STUD#     ASCENDING

ACCEPT PARAMETER-01 FROM CRT
ACCEPT PARAMETER-02 FROM CRT
```

```
SELECT (ALL)
    R02.COURSWORK
    R02.COURSGRADE
FROM
    R02
WHERE
    R02.STUD# = PARAMETER-01
        AND    R02.COURSWORK = PARAMETER-02
ORDERED BY
    R02.COURSWORK     ASCENDING
```