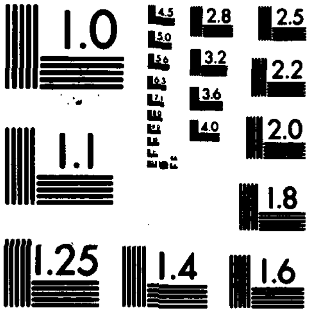
AD-A081 784   WHARTON SCHOOL PHILADELPHIA PA DEPT OF DECISION SCIENCES   F/G 9/2
AN EXTERNAL SCHEMA FACILITY FOR CODASYL 1978.(U)
1978   E K CLEMONS   N00014-75-C-0462
UNCLASSIFIED 78-10-03   NL

END
DATE
FILMED
4 80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AN EXTERNAL SCHEMA FACILITY FOR CODASYL 1978

ERIC K. CLEMONS

78-10-03

AN EXTERNAL SCHEMA FACILITY FOR CODASYL 1978

Eric K. Clemons

78-10-03

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Research supported in part by Office of Naval Research Contract N00014-75-C-0462.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
14 78-10-03

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER
9

4. TITLE (and Subtitle)
6 An External Schema Facility for CODASYL 1978.

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
78-10-03

7. AUTHOR(s)
10 Eric K. Clemons

8. CONTRACT OR GRANT NUMBER(s)
15 N00014-75-C-0462

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Decision Sciences
The Wharton School Univ. of Pennsylvania
Philadelphia, PA 19104

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Task NR049-272

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Department of the Navy
800 N. Quincy St., Arlington, VA 22217

12. REPORT DATE
11 1978

13. NUMBER OF PAGES
30

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
12 35

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

CODASYL, Data Description Language, external schema facility, virtual data base

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

~~We propose~~ as an extension to the CODASYL Data Description Language, a new external schema description language to provide an external schema facility more powerful than that presently offered by CODASYL subschemas is proposed. A powerful external schema facility that, in effect, offers applications programmers a virtual data base. Programmers will use records formatted, not like those stored in the common data base, but like the cognitive structures employed in program development. A possible language for external schema definition is introduced.

408 751

ABSTRACT

We propose as an extension to the CODASYL Data Description Language a new external schema description language to provide an external schema facility more powerful than that presently offered by CODASYL subschemas. We note that the present facility relies on records and sets that closely resemble those employed in the schema, with unfortunate implications for programmer productivity, data independence, and communications channel traffic. We propose instead a powerful external schema facility that, in effect, offers applications programmers a virtual data base. This data base, while derivable from the stored data base, is not constrained to resemble it in form; thus programmers will use records formatted, not like those stored in the common data base, but like the cognitive structures employed in program development. A possible language for external schema definition is introduced.

AN EXTERNAL SCHEMA FACILITY
FOR CODASYL 1978

1. Introduction

The CODASYL data description and data manipulation languages are rapidly emerging as de facto national data base standards; in fact, John Berg of the National Bureau of Standards, currently Acting Chairman of the newly reconstituted ANSI/X3/SPARC Data Base Study Group, states that the CODASYL DML and DDL will emerge as formal national standards by 1981 [2]. I have previously stated my reasons for considering these choices unfortunate, while acknowledging the need for any future DBMS standard to be downward compatible so as to provide continuing support for user application implementations based on these languages [8]. In this paper I present a more comprehensive solution to the problems inherent in the CODASYL specifications, an extension to the DDL that would permit development of an alternative DML with better support of applications programming.

My principal objection to the CODASYL 1978 specifications is the nature of the programmer interface. This is not a new objection; Engels, an original member of the Data Base Task Group, published his objections to this interface [11] within months of the publication of the DBTG Report [9]. The proposal of macro facilities to hide this interface from the programmer soon followed [17]. But, with the introduction by the original ANSI/X3/SPARC Study Group of the three-schema data base architecture, we now have the ability to provide a superior interface for applications programmers. In section 4 of this paper I will describe an external schema facility with far greater capabilities than the CODASYL subschema facility that it is intended to replace.

2. Background Material

2.1 A Review of the ANSI/SPARC Reports

By 1975, discussion of the "correct" data model -- hierarchical, network, or relational -- had become widespread, occasionally acrimonious, and futile. No single data model could be selected. With the publication in 1975 of the first ANSI/SPARC Data Base Study Group Report [1], one reason for this impasse became clear: There were three different users of data base systems, interacting with the systems at different levels, and with dramatically different needs.

The data base machine itself needed device-level detail including data descriptions, addresses, and access paths, to permit data retrieval and efficient system operation. The enterprise as a whole required logical completeness: a description of all entities of interest, their significant attributes, and their relationships. This need not include device detail needed for efficient operation, but must be sufficiently complete to support the enterprise's collection of diverse operations. And the individual applications programmer required logical simplicity: data structures perhaps less general than those provided to support the numerous and diverse needs of the full organization, but well suited to the specific program in which they were employed and closely matching the cognitive structures the programmer employed in its preparation.

Recognizing that these requirements might prove to be incompatible, the ANSI/X3/SPARC Study Group proposed a multi-level, multi-schema data base architecture. At the machine level a single internal schema provided details permitting efficient operation. For the enterprise a single conceptual schema provided an integrated description of data and their logical relationships. And, for each application program, an external schema was provided that offered a view of the data not necessarily similar in form to that of any other schema, but well suited to the specific programming task being performed.

In this paper we are not concerned with the details of the internal schema, since the existence of the conceptual schema makes the internal level of little consequence to the design of the programmer interface. We will also not be concerned with the selection of a conceptual schema that is in any sense "complete" or the best possible, primarily because no such design has been developed; rather, we shall assume that the pre-eminence of CODASYL implementations justifies for the moment the selection of the CODASYL model for the conceptual level.

2.2 Role of the External Schema

My principal area of interest has for several years been the design of external schema facilities. This level of the three schema architecture has the greatest effect on programmer productivity. It is the level with which the greatest number of users will interact, and thus the most important level for formal standardization. It is, unfortunately, the level at which both ANSI/SPARC and CODASYL offer the least guidance to DBMS designers.

The most important role of the external schema should be to facilitate programmer use of the data base. This can best be accomplished by providing <u>user views</u>, essentially virtual data bases that are closely designed to match the cognitive structures that the user employs in preparing his program rather than the data structures provided in the data base. The records in these virtual data bases must of course be derivable from the data actually stored; virtual data bases may be subsets, reformatted, with records combined in ways that simplify or eliminate the need for programmer navigation.

A single example will serve to make these concepts clear. We consider a simple data base: courses include multiple sections, students have taken many courses, and for each student in each course a single grade is stored. Irrespective of the model employed at the conceptual level -- hierarchical, network, or relational -- and irrespective of the programming language employed, an excessive amount of work may be required to prepare some simple queries. For example, to prepare a student's summary transcript with name and repeating term and term average information:

```
NAME

     TERM, AVERAGE
     TERM, AVERAGE
        . . .
     TERM, AVERAGE
```

using the CODASYL data base depicted in figure 1

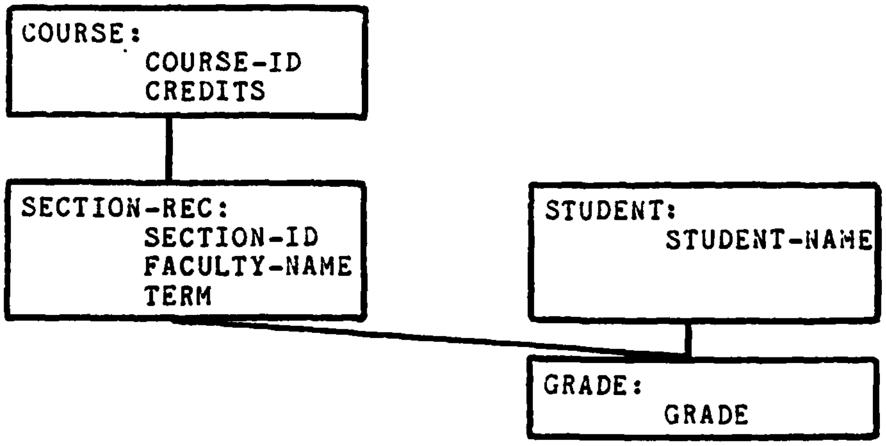


Figure 1 -- A CODASYL Data Base to Support Student Transcripts, Course Rosters, and Registration

we require the code segment included in figure 2.

```
*    WORKING STORAGE INCLUDES A STRUCTURE OF THE FORM:
*              01  TERM-HISTORY, INDEXED BY POSIT.
*                  02  TERM-COUNT PICTURE  S99.
*                  02  TERM-ENTRY OCCURS 15 TIMES.
*                      03  TERM-ID       PICTURE  X(6).
*                      03  TERM-POINTS   PICTURE  S999.
*                      03  TERM-CREDITS  PICTURE  S999.
*                      03  TERM-AVERAGE  PICTURE  S9V99.

          . . .

FIND-SUMMARY-TRANSCRIPT.
     FIND STUDENT RECORD.
     PERFORM CLEAR-TABLE.
     MOVE Ø TO ERROR STATUS.
     FIND FIRST GRADE RECORD IN STU-GRD SET.
     PERFORM GRADE-LOOP THRU GRADE-LOOP-EXIT
          UNTIL ERROR-STATUS NOT EQUAL Ø.

     PERFORM SORT-TABLE-BY-TERM THRU SORT-TABLE-EXIT
          VARYING I FROM 1 BY 1
          UNTIL I GREATER THAN TERM-COUNT.

     PERFORM COMPUTE-TABLE
          VARYING I FROM 1 BY 1
          UNTIL I GREATER THAN TERM-COUNT.

*    SORT-TABLE-BY-TERM IS SIMPLE SORT, NOT SHOWN.
*    COMPUTE-TABLE SETS TERM-AVERAGE (I) =
*         TERM-POINTS (I) / TERM-CREDITS (I),
*         ALSO NOT SHOWN.
*    CLEAR TABLE ZEROES ALL ENTRIES IN TABLE, NOT SHOWN.

GRADE-LOOP.

     FIND NEXT GRADE RECORD IN STU-GRD SET.
     IF ERROR-STATUS NOT EQUAL Ø,
          GO TO GRADE-LOOP-EXIT.

     GET GRADE RECORD.

     FIND OWNER RECORD OF SEC-GRD SET.
     GET SECTION-REC RECORD.

     FIND OWNER RECORD OF CRS-SEC SET.
     GET COURSE RECORD.
     SEARCH TERM-HISTORY:
          AT END ADD 1 TO TERM-COUNT,

               MOVE TERM-COUNT TO POSIT,
               MOVE TERM TO TERM-ID (POSIT)

          WHEN TERM-ID (POSIT) EQUALS TERM, NEXT SENTENCE.

     CALCULATE TERM-POINTS (POSIT) =
          TERM-POINTS (POSIT) + GRADE.GRD*
          COURSE.CREDITS.

     ADD COURSE.CREDITS TO TERM-CREDITS
          (POSIT)

GRADE-LOOP-EXIT.
     EXIT.
```

Figure 2 -- DML Code Segment to Create Summary Transcript.

Clearly, we would prefer to say:

```
READ SUMMARY-TRANSCRIPT
```

where summary-transcript is a rather simple record in the user view, constructed by a rather powerful external schema facility.

3. Limitations of the Present CODASYL Subschema

As I have written elsewhere [7, 8], a major limitation of the current CODASYL specifications is the design of the subschema. In fact, the subschema so closely resembles the conceptual schema that it cannot truly be considered an external schema facility. Both must employ the network data model. More significant, both employ the same collection of record types and sets, although in the subschema some minimal reformatting may be provided. As a consequence, the programmer is still required to perform navigation to accomplish all but the most trivial of tasks. Also, there is little to protect the applications programs from changes to the conceptual schema; since the external schemas are also affected by most changes, these programs will need to be rewritten. We note, for example, that too often the addition of an unanticipated application to be supported will necessitate changing the conceptual schema from a simple hierarchy to a confluency representing a many-to-many relationship. Because of the close relationship between schema and subschema, programs processing this data base will need a different set of FIND and GET statements, and perhaps different control logic as well. Finally, since the subschema's ability to perform data aggregation and data reduction is so limited, these functions must be performed by the application program through host language statements; thus, in a distributed environment, a program that requires only averages will still have to request individual course and grade records, greatly increasing the necessary channel traffic.

In summary, we note that the form of the CODASYL subschema facility has the following results:

1. It limits programmer productivity by requiring navigation to perform almost all inter-record associations; this has been found to be the most difficult aspect of data retrieval in any language and using any data model [12]

2. It limits data independence because changes to the conceptual schema necessitated by addition of new applications will alter the existing external schemas as well [8, 15]

3. It increases channel traffic by limiting the data reduction that can be performed by the remote data base machine

4. Design of an External Schema Facility

4.1 Introduction to a Traditionalist's External Schema Facility

The external schema facility proposed is traditional, in that it does not change the way we view the programming task; it does not add new features such as operational naming [16], generalization hierarchies, or higher level language statements to perform implied iteration. Rather, it changes the way we view the external schema facility. In fact, in terms of programming languages it represents a step backwards: We use the external schema to hide from the programmer the fact that he is employing a DBMS and remove all DML statements from the host language.

Our intention is to provide each programmer with a virtual data base composed of virtual files suited to a particular application. Interaction with the data now requires only a traditional file-oriented READ, and we employ the external schema facility to transform this read request into the appropriate sequence of FIND and GET statements and utility calls to perform sorting and data reduction.

It appears impossible to design a truly universal data base interface; the requirements for accounting applications are simply too different from the requirements of a three dimensional display system offering variable perspective drawing to support architectural design. We have concentrated, therefore, on an interface to support common data base uses: data retrieval, report generation, and update.

After years of commercial programming and intensive study of applications requirements, I believe that the interface provided for individual applications programmers need support no data structure more complex than hierarchies. This is not to say that for the enterprise, at the conceptual level, hierarchies are sufficient; rather, the networks at the conceptual level are required by the incompatible collection of hierarchies needed by the application programs. For example, one application to generate transcripts may require the following data structure:

```
STUDENT
     TERM
          COURSE, GRADE
          COURSE, GRADE

            . . .

     TERM
          COURSE, GRADE

            . . .
```

A related application to provide class lists for faculty would require

```
FACULTY-NAME, COURSE, SECTION
     STUDENT-NAME

       . . .

     STUDENT-NAME.
```

The conceptual schema to support both applications is not a hierarchy but a network, as shown in figure 1.

This does not mean that any programmer requires the full plex structure. More significant, it does not imply that any single programmer be forced to perform complex inter-record navigation to reconstruct information objects of interest, such as transcripts and course listings. Surely there exist data that do represent graphs, such as transportation and telecommunications networks; for such applications the data must indeed contain network information, such as the costs and capacities of transportation between two locations, but I remain unconvinced that even in such cases network structured data, with a network structured schema, provide a superior programmer interface.

4.2 Restructuring to Produce Hierarchies

If we are to offer individual applications programmers an adequate and useful external schema facility based on hierarchies, then we must provide an adequate facility for constructing useful hierarchical virtual records from the stored data. In particular associations between records or among several records must be made, permitting required data to be accessed; individual items required for virtual records must be defined, permitting values to be retrieved or constructed; and the virtual record must be structured as the user requires. We will treat structuring in this section; data access and item definition are treated in the following sections.

Three concepts are important in understanding hierarchical data structures: extent, entry type, and content. Extent may be single or all; a structure with single extent has only a single subordinate entry, while a structure with all extent has repeated subordinate entries. Thus, if we wish to associate in a single record a student name and all his course grades, all extent is required; if each record is to contain only a single course grade then single extent is required.

An individual entry in a hierarchy may be either simple or grouped. Thus an entry may correspond to information about a single course taken if it is simple, or about all courses taken in a single term if the entry has grouped structure. Extent and entry descriptions may be combined; for example, all extent and grouped entries would permit data to be grouped by term and entries for all terms to be combined in a single record.

The information content of a structure may be complete, or only summary information may be contained. For example, the data structure may contain a complete record of courses taken and grades received, or it may contain only term averages.

The attributed for extent, entries, and content may be combined to yield a 2x2x2 classification of data structures. In the resulting eight-way classification only seven combinations are of use; the combination simple, single, summary is readily seen to be of no interest. Examples of each of the seven types of structures are shown as figure 3.

```
R. J. Nash     Fall 78     DS15    A
R. J. Nash     Fall 78     DS25    A
R. J. Nash     Fall 78     STAT1   B
```

a.) three records, SIMPLE, SINGLE, COMPLETE, each reporting on a single student in a single course

```
R. J. Nash     3.6
     Fall 78   DS15     A
     Spr 78    DS1      A
     Fall 78   DS25     A
     Spr 78    DS2      A
     Fall 78   ECON2    A
     Fall 77   ACT1     B
     Fall 77   FIN1     B
     Fall 78   STAT1    B
     Fall 77   ECON1    B
     Spr 78    FIN2     A
```

b.) one record, SIMPLE, ALL, COMPLETE, having cumulative average as well as individual grades

```
R. J. Nash     3.6
```

c.) one record, SIMPLE, ALL, SUMMARY, having cumulative average but no individual grades

```
R. J. Nash     3.6
     Fall 78   3.75    DS15    A
                       DS25    A
                       STAT1   B
                       ECON2   A
```

d.) one record, GROUPED, SINGLE, COMPLETE, displaying cumulative average, as well as courses and term average for a single term

```
R. J. Nash     3.6
     Fall 77   3.0     ACT1    B
                       FIN1    B
                       ECON1   B
     Spr 78    4.0     DS1     A
                       DS2     A
                       FIN2    A
     Fall 78   3.75    DS15    A
                       DS25    A
                       STAT1   B
                       ECON2   A
```

e.) one record, GROUPED, ALL, COMPLETE, displaying cumulative average, as well as courses and term average for each term

```
R. J. Nash     3.6
     Fall 78   3.75
```

f.) one record, GROUPED, SINGLE, SUMMARY, displaying cumulative average as well as term average for a single term

```
R. J. Nash     3.6
     Fall 77   3.0
     Spr 78    4.0
     Fall 78   3.75
```

g.) one record, GROUPED, ALL, SUMMARY, displaying cumulative average as well as term average for each term

Figure 3 -- Examples of Each of Seven Record Types in Classification of Hierarchies.

By specifying multi-level groupings, it is possible to define more complex structures and provide more general summaries. For example, we may wish to group courses by school (arts, engineering, business) and within school by term, to provide term averages by school and overall school averages. And, as shown in the following section, structured data elements may themselves comprise structured elements, permitting records of great complexity to be declared. In fact, with the addition of order specification to this seven-way classification, a general external schema facility of substantial utility has been developed.

4.3 Access Information

Of course, the data included in virtual user records is derived from data present in the stored data base; therefore it is necessary that data be retrieved to permit user records to be constructed. We feel that it is necessary to specify in the mapping definition precisely which records are to be accessed, rather than to permit the external schema facility itself to determine which records are required. This provides generality: access conditions of complexity, based on set associations and record content, may be used when necessary. It requires the author of the map to specify which access path to use when multiple paths exist, and it provides a small additional level of data independence by requiring access specification, even when only a single path exists. In short, mandatory access specification increases the probability that the data retrieved are actually the data desired.

In a well designed CODASYL implementation, most of the frequently used inter-record associations will be between records related by set occurrence. Therefore, we expect that the most important means of access path specification will be through set membership or ownership. Since the records desired may compose a subset of members of a set, qualification may also be necessary.

4.4 Data Item Specification

It is necessary to specify all desired data elements in the user record since not all data in the stored records are to be included in the user record and not all items in the user record are directly retrievable from the stored records. Data items are of three types:

1. real items, retrieved from the stored data base
2. virtual elementary items, calculated from the stored data base using standard functions, according to computations specified
3. virtual structured items containing additional data elements

Since the components of virtual structured items may themselves be virtual structured items, complex records may be constructed as needed.

5. An Extension to DDL: A Schema to External Schema Mapping Language

5.1 Introduction to a Mapping Language

The mapping language proposed is based upon the classification of hierarchies introduced in section 4.2. The language will specify all information needed to map stored data base records into the desired virtual user records. Structuring information, which determines the form of the user record, is described in section 5.2. Access information, which determines which data base records will be used, is described in section 5.3, and data item definition, which determines the actual content of the user record, is described in section 5.4.

5.2 Structuring Information

Structuring information specified in the map determines the class of hierarchy being defined; that is, the options specified here determine extent, entries, and content. The form of a structure definition is:

```
STRUCTURE structure-name:  (option-list)
     access-statement.
     data-item-definitions.
     END structure-name.
```

A structure may correspond to a user record definition, like an 01-level COBOL entry, or to a complex entry within a user record. The option-list specifies single or all extent, simple or grouped entries, and complete or summary only content. Defaults are single, simple, complete. A single access statement is required corresponding to each defined structure; without an access statement, there is no need to define a structure. Data item definitions are optional; there may be zero, one, or more data items defined in a structure. Defined items may be elementary, or they may be structures containing access statements and data item definitions.

5.3 Access Information

Associated with each structure is a data access statement; the number of records retrieved plus the structuring options specified determine the structure of the declared user record. Access information may be based on set ownership or membership, or on qualification using record content.

The form of the access statement is:

```
ACCESS recordname1 RECORD
      [ Qualification ]
[ ⎧ MEMBER                OWNED BY                      ⎫ ]
[ ⎨          IN setname                record-reference ⎬ ]
[ ⎩ OWNER                 OF                            ⎭ ]
```

Both qualification and set specification are optional. The outermost structure sometimes will employ neither, as when creating virtual records corresponding to all data base records of a given type:

```
ACCESS STUDENT RECORD.
```

Sometimes it will employ qualification, when creating virtual records corresponding to a subset of the data base records:

```
ACCESS STUDENT RECORD
     WHERE MAJOR = 'DEC SCI'.
```

Sometimes access will employ set specification:

```
ACCESS GRADE
     MEMBER IN STU-GRD OWNED BY STUDENT.
```

And sometimes access will employ both set specification and qualification:

```
ACCESS GRADE
     WHERE GRD = 4
     MEMBER IN STU-GRD OWNED BY STUDENT.
```

When the access statement is not in the outermost structure declaration, it is expected that set specification will be used. To retain compatability with our relational interface, access based on qualification without set specification is permitted, though for performance reasons it is discouraged. The precise form of qualification is of little theoretical interest; we use relational and Boolean operators and a syntax based on SEQUEL. The only slight complication arises because CODASYL records are not normalized and may include repeating values; therefore a distinction must be made between any value satisfying a condition and all values satisfying the condition.

Note that records used in constructing user records are retrieved in a top-down order; that is, a record for the outermost structure is accessed first, then records for its directly contained structures, followed by the successive levels of contained structures. Thus, any data used in qualification must be either from the record being accessed or from higher level structures; likewise, any records used in set specification must have been accessed in a higher-level structure.

Record-reference has the form:

```
structurename.recordname
```

although if a record type is referenced in only a single access statement, specification of the containing structure name may be omitted. Record reference may itself employ set specification. Thus to access COURSE records corresponding to GRADE records previously accessed, we write:

```
ACCESS COURSE RECORD
     OWNER IN CRS-SEC OF SECTION
     OWNER IN SEC-GRD OF GRADE.
```

Finally, we note that it is often useful to construct user views that are based on other, previously defined views. Therefore, we permit records accessed in a structure declaration to be either data base records or virtual user records; the only restriction on use is that since set relationships are defined only for real records, they must not be employed to identify desired user records.

5.4 Data Item Definition

Data items in user records are of three types, and therefore three forms of data item definition are provided. For elementary virtual items, definition has the form:

```
item-name: (description)
     arithmetic-expression
```

The arithmetic expression can be any well-formed arithmetic expression and may employ built-in functions such as MAX, COUNT, SUM, and AVERAGE. The description is optional; it specifies which terms are summary values and, when grouped entries are requested, specifies at which level summaries are to be taken. Note that all terms used in arithmetic expressions must be single-valued; thus, if terms from contained structures are employed, all intervening structures must have single extent requested.

Just as data retrieval in virtual record construction was performed top-down, data item computation is performed bottom-up. This imposes a restriction on the use of terms in arithmetic expressions: while terms used may be real or virtual, all virtual terms must be from a contained lower level structure.

Also, we note that in some instances when a single data value is to be selected from an accessed record, we may choose to introduce a defined data item rather than add another contained structure. Thus, to retrieve the term of a section corresponding to a previously accessed grade, we may write:

```
TERM:  ACCESS SECTION RECORD
            OWNER IN SEC-GRD OF GRADE
            SELECT TERM.
```

Real data items may be defined as were virtual items; the arithmetic expression then is the simplest possible, containing only the single term. Alternatively, if several real items are to be contained in a single structure, they may be defined by selection from the record accessed within the structure.

Virtual data items are themselves structures, with definition as described in section 5.2.

Only data items explicitly defined or selected will be included in the constructed user record.

5.5 An Example of A Mapping Definition

The user record to be defined is the summary transcript introduced in section 2. The mapping definition to produce this transcript is provided in figure 4. We note, but are largely unconcerned by, the fact that the map is quite difficult to read. The external schema facility is intended to provide an interface for applications programmers permitting more easy access to stored information. It is not necessary that preparation of mapping definitions be simple for these programmers; indeed, we are not certain that this would be valuable since we do not intend for these maps to be prepared by programmers. However, alternate formats for definition might prove preferable.

```
STRUCTURE TRANSCRIPT:
     ACCESS STUDENT WHERE MAJOR = 'DEC SCI'.
     SELECT STUDENT-NAME.

*    TERM-LEVEL STRUCTURE, CONTAINS ALL TERM SUMMARIES
     STRUCTURE TERM-DATA:  (SUMMARY, ALL, GROUP BY TERM,
          ORDER BY TERM).
          ACCESS GRADE RECORD
               MEMBER IN STU-GRD OWNED BY STUDENT.
          TERM:
               ACCESS SECTION RECORD
                   OWNER IN SEC-GRD OF GRADE.
               SELECT TERM.
          CREDITS:
               ACCESS COURSE RECORD
                   OWNER IN CRS-SEC OF SECTION
                   OWNER IN SEC-GRD OF GRADE.
               SELECT CREDITS.
          TERM-AVERAGE:  (TERM SUMMARY)
               SUM (GRD*CREDITS)/SUM (CREDITS).
          TRANS-AVERAGE:  (SUMMARY)
               SUM (GRD*CREDITS)/SUM (CREDITS).
          END TERM-DATA.
     END TRANSCRIPT.
```

Figure 4 -- Definition of Map to Construct Summary Transcript.

6. Remaining Difficulties

A number of significant difficulties remain, relating to implementation, update, and user performance. There are significant design problems that must be surmounted if virtual data bases are to be produced on demand, rather than produced off line and stored. These problems are most troublesome when the map employs ORDERED BY specification, particularly when the ordering terms are virtual items, or when the user record definition itself exploits other virtual user records.

Most external schema definitions will employ maps that are not invertible, and this is known to cause difficulty when the resulting user records must be updated [4, 6]. It is of course necessary that changes to the user records be captured as changes to the corresponding stored data base records, yet the changes to be made cannot always be determined if the maps are not invertible. In fact, analysis of this problem to date has been restricted to the simpler relational model; update of a CODASYL data base which according to the 1978 specifications may have very complex set selection requirements will no doubt prove at least as difficult.

Of course, if an enhanced external schema facility is proposed to replace the present subschemas provided by CODASYL in order to provide better support of applications programmers, it is certainly desirable to verify that improved programmer performance does indeed result. Human factors experimentation in data model and programming language design has proved exceedingly difficult. Nevertheless, some valuable studies are available as examples, and it is necessary that some evaluation of the proposed interface be performed.

7. Concluding Remarks

The interested reader is referred to earlier works for a more complete treatment of hierarchies and recursive hierarchies [5]. An outline of a design for implementation is also presented [5].

We are presently engaged in developing a prototype external schema facility with acceptable machine performance, and it should be completed by spring of 1979. Performance data, user reactions, and a demonstration should be available before the SIGMOD conference in May 1979.

ACKNOWLEDGEMENTS

I would like to acknowledge my debt to my colleagues at the University of Pennsylvania, Rob Gerritsen and Frank Germano; to my co-panelists at the Fourth Very Large Data Base Conference, John Berg, Frank Manola, and Diane C. P. Smith; and to my co-participants at the N.B.S. Three-Schema Feasibility Workshops, particularly Don Chamberlin, Henry Lefkowitz, and Carlo Zaniolo. Their thinking has influenced my thinking. Obviously, biases, errors, and omissions in this paper are my own.

REFERENCES

1. "ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report 75-02-08". FDT--Bulletin of the ACM SIGMOD, Vol. 7, No. 2, 1975.

2. Berg, J. L. "Implementing a Framework for DBMS Standards". Unnumbered Working Paper, Institute for Computer Sciences and Technology, National Bureau of Standards, Gaithersberg, MD., March 1978.

3. Berg, J. L. Unpublished letter to ANSI/X3/SPARC Study Group membership, Document No. ANSI/X3/SPARC/DBS-SG-78-6, November 1978.

4. Bernstein, P. A. and Dayal, U. "On the Updatability of Relational Views". Proceedings of the Fourth International Conference on Very Large Data Bases, Berlin, West Germany, September 1978, pp. 368-377.

5. Clemons, E. K. Design of a User Interface for a Relational Data Base, Dissertation, School of Operations Research, Cornell University, 1976.

6. Clemons, E. K. "An External Schema Facility to Support Data Base Update". Databases: Improving Usability and Responsiveness, ed. Ben Schneiderman, Academic Press, New York, 1978, pp. 371-398.

7. Clemons, E. K. "The External Schema and CODASYL". Proceedings, Fourth International Conference on Very Large Data Bases, Berlin, West Germany, September 1978, p. 130.

8. Clemons, E. K. "Rational Data Base Standards: An Examination of the 1978 CODASYL DDLC Report". Decision Sciences Working Paper 78-10-02, University of Pennsylvania, 1978.

9. "CODASYL Data Base Task Group April 71 Report". ACM, New York, 1971.

10. "CODASYL Data Description Language Committee Journal of Development", 1978.

11. Engels, R. W. "An Analysis of the April 1971 Data Base Task Group Report". Proceedings ACM SIGFIDET Workshop 1971, ACM, New York, pp. 69-92.

12. Lochovsky, F. and Tsichritzis, D. "User Performance Considerations in DBMS Selection". Proceedings, ACM SIGMOD Workshop, Toronto, Canada, August 1977, pp. 128-134.

13. Manola, F. "On Relating the CODASYL Database Languages and the ANSI/SPARC Framework". Proceedings, Fourth International Conference on Very Large Data Bases, Berlin, West Germany, September 1978, p. 132.

14. Manola, F. "A Review of the 1978 CODASYL Database Specifications". Proceedings, Fourth International Conference on Very Large Data Bases, Berlin, West Germany, September 1978, pp. 232-242.

15. Smith, D. C. P. "Conversion and the CODASYL Framework". Proceedings, Fourth International Conference on Very Large Data Bases, Berlin, West Germany, September 1978, pp. 133-134.

16. Smith, J. M. and Smith, D. C. P. "Integrated Specifications for Abstract Systems",Technical Report UUCS-77-112, Computer Science, University of Utah, 1977.

17. Taylor. R. W. "Data Administration and the DBTG Report". Proceedings ACM SIGMOD Workshop 1974, ACM, New York, pp. 431-444.

18. Tsichritzis, D. and Klug, A. "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems". AFIPS Press, Montvale, N. J., 1977.

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation Center
(12 copies)
Cameron Station
Alexandria, VA 22314

Office of Naval Research
Code 102IP
Arlington Virginia 22217

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605

New York Area Office

715 Broadway - 5th Floor
New York, NY 10003

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington DC 20380

Office of Naval Research
Code 458
Arlington, VA 22217

Office of Naval Research
(2 copies)
Information Systems Program
Code 437
Arlington VA 22217

Office of Naval Research
Branch Office
495 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office Pasadena
1030 East Green Street
Pasadena CA 91106

Naval Research Laboratory
(6 copies)
Technical Information Division
Code 2627
Washington DC 20375

Office of Naval Research
Code 455
Arlington VA 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation & Mathematics Dept.
Bethesda, MD 20084

Mr. Kim B. Thompson
Technical Director
Information Systems Division
(OP-911G)
Office of Chief of Naval Operations
Washington, DC 20350

Professor Omar Wing
Columbia University
in the City of New York
Dept. of Electrical Engineering
and Computer Science
New York, NY 10027

Commander, Naval Sea Systems Command
Department of the Navy
Washington, D.C. 20362
ATTENTION: (PMS30611)

Captain Richard L. Martin, USN
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
(OP-916D)
Office of Chief of Naval Operations
Washington, DC 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick, NJ 08903
Attn: Dr. Henry Voos

Defense Mapping Agency
Topographic Center
ATTN: Advanced Technology
Division
Code 41300 (Mr. W. Mullison)
6500 Brookes Lane
Washington, D.C. 20315

Major J.P. Pennell
Headquarters, Marine Corps
Washington, D.C. 20380
ATTENTION: Code CCA-40

Professor Mike Athans
Massachusetts Institute of Technology
Dept. of Electrical Engineering and
Computer Science
77 Mass. Avenue
Cambridge, MA 02139