A FORTRAN interface to the CODASYL database task group specifications

G. M. Stacey

Edinburgh Regional Computing Centre, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ.

A FORTRAN interface to the CODASYL Data Base Task Group specifications is presented which incorporates the concepts of a common sub-schema framework for all host languages supplemented by specific sub-schema host language interfaces. A central feature of the FORTRAN sub-schema specifications is the technique for mapping hierarchically structured schema records into non-hierarchically structured FORTRAN variables and arrays.

(Received November 1972)

1. Introduction

This paper presents a FORTRAN interface to the CODASYL Data Base Task Group database specifications as described in their April '71 Report (CODASYL DBTG, 1971). It has been prepared for the purpose of generating discussion with the belief that the CODASYL specifications should be considered in a multi-host language context. It incorporates the concept of a host language independent sub-schema framework common to all host languages, and in conjunction with which it is possible to define host language interfaces. FORTRAN was selected as the host language for this work due to its widespread use in the computing community at large.

The CODASYL Report introduces three special purpose database languages and it is worthwhile reviewing their functions. The Schema Data Description Language (Schema DDL) provides facilities to enable a unified description of a database to be specified which is intended to be controlled by a central Data Administrator and which is independent of the host languages interfacing with the database. The Sub-Schema Data Description Language (Sub-schema DDL) provides facilities to enable application-specific descriptions to be specified. These application-specific descriptions (sub-schemas) will describe only those parts of the database which are relevant to the application. They include descriptions of how data in the database is mapped into user work areas consisting of data structures of the particular host language. The duality of the sub-schema interface is apparent from its functions. A sub-schema must interface with schema data in selecting the sub-division of the database to be used. It must also interface with the host language in that all data is presented to the host language in the form of host language data structures. It is suggested that these two aspects are best treated independently and that the schema interface of host language sub-schemas should have a common and host language independent DDL. A complete sub-schema DDL would thus consist of the common sub-schema framework DDL plus the host language interface DDL. The Data Manipulation Language (DML) is concerned with the movement of data between the host language program user work areas and the database, and with associated functions. It should be particularly noticed that a DML may be independent both in function and syntax from the details of the host language data structures. This is true because the mapping between database and host language data structures is specified independently through the host language sub-schema DDL. Although it is possible to define different DML's for different host languages, a valuable aid to compatibility results from utilising a common DML.

In view of the above considerations the FORTRAN interface presented here interfaces to the CODASYL Schema DDL and uses the CODASYL COBOL DML (both as specified in CODASYL DBTG 1971). Further the FORTRAN sub-schema DDL is presented in the form of a framework intended to be common to all host languages, and a sub-schema FORTRAN interface.

2. The FORTRAN sub-schema specifications

FRAMEWORK 0.0 *Introduction*

These sub-schema specifications are presented in the form of a framework plus one, or more, host language interfaces.

The framework is intended to be common to all host languages whereas the host language interfaces are specific to the host language concerned. Comments are given on why particular parts of the specifications are (or are not) included in the framework. The framework and host language interface specifications are contained in appropriately numbered and headed (FRAMEWORK or host language name) sections. A sub-schema section is understood to consist of all appropriately numbered framework and host language interface sections and sub-sections.

Section C.4.0.0* is included in this section with the following alterations:

1. Delete sentence 1
2. Replace paragraph 5 (starting p153 line 17) by 'The Renaming Section consists of a single entry. The Area and Set Sections consist of an entry for each area or set to be included in the sub-schema being defined. Each such entry completely describes an area or set. The correspondence of entries in the Record Section is given in the appropriate host language interface introduction'.
3. Replace sentence 1 of paragraph 6 (starting p153 line 24) by 'Entries in the framework and, unless otherwise specified, in the host language interfaces consist of one or more clauses'.
4. Delete sentence 1 of paragraph 7 (starting p153 line 28)
5. Replace sentence 2 of paragraph 7 by 'For all entries a Complete Entry showing the general format of all the clauses included in the entry is shown'.
6. Add the following text to the end of the section 'Lists of DML statements occur in parts of the sub-schema specifications. For example in the specification of privacy locks which may apply to one or more DML statements (*c.f.* C.4.2.4.2). Where different host languages are involved there may also be differences in DML statements and hence in the DML lists to be specified. This event is catered for by specifying "DML-list-*n*", where *n* is an integer, in the syntax. A separate specification of the actual list may then be included in the host language interface of the sub-schema'.

FORTRAN INTERFACE 0.0 *Introduction*

These specifications describe a FORTRAN host language sub-schema interface.

*All references to the CODASYL DBTG report are of the form C followed by the section number.

In the case of the specifications for records, an entry exists for each record to be included in the sub-schema being defined. Each such entry completely describes a record and consists of a Complete Entry Skeleton representing the breakdown of the Record Description entry into its component entries. All component entries are specified using the same layout as described for entries in the sub-schema framework specifications.

The sub-schema identification and data divisions are translated into sub-schema object form by a FORTRAN sub-schema DDL translator separately from application programs. These object sub-schema may then be invoked by application programs written in 'database extended FORTRAN'.

A FORTRAN sub-schema consists of FORTRAN variables and arrays to be utilised in FORTRAN programs. FORTRAN variables and arrays declared in a sub-schema must not be declared in the normal non-executable statements of a FORTRAN application program using that sub-schema and must differ in name from any variables and arrays thus declared (or used implicitly in the main program).

FRAMEWORK 1.0 *Identification division*
Comment This division of the sub-schema specifications is concerned with defining and naming sub-schema in terms of database schema and as such should be host language independent
end of comment

Sections C.4.1.0 through C.4.1.3 are included in this section unchanged.

FRAMEWORK 2.0 *Data division*
Comment This division of the sub-schema specifications is concerned with naming and giving characteristics of areas, records and sets contained in the sub-schema.
end of comment

Section C.4.2.0 is included in this section unchanged except for the following alteration

Replace the last sentence by
'The area and set sections consist of an entry for each area or set to be included in the sub-schema being defined. The correspondence of entries in the Record section is given in the appropriate host language interface introduction'.

FRAMEWORK 2.1 *Renaming section*
Comment This section of the sub-schema specifications is concerned with the provision of new names in the sub-schema for named schema data. This may be to achieve conformity with the naming conventions of the host language or for convenience purposes.
end of comment

Section C.4.2.1 is included in this section unchanged except for the following alterations

Delete '(in this case COBOL)' from Function line 3.
Replace 'COBOL' in Syntax rules 1 and 3 by 'host language'
Replace Syntax rule 2 case *d* by 'names of data units (data aggregates and/or data items as appropriate) of the host language within each record type'.

FRAMEWORK 2.2 *Area section*
Comment This section of the sub-schema specifications is concerned with enumerating the areas of the schema that are to be included in the sub-schema and as such should be host language independent.
end of comment

Sections C.4.2.2 through C.4.2.2.2 are included in this section unchanged in any detail.

FRAMEWORK 2.3 *Record section*
Comment This section of the sub-schema specifications is concerned with records and in particular with how records are 'seen' by host languages. It is important that maximum flexibility be allowed here to cater for a multitude of host languages with a vast variation in data structure facilities. For this reason only the statement of function is included in the framework specifications.
end of comment

Function
To enumerate the records and subordinate data items within records of the schema that are to be included in the sub-schema, and to define how they are 'seen' by the host language.

By implication, to remove from view all other records and data items within records of the schema.

FORTRAN INTERFACE 2.3 *Record section*
Comment This FORTRAN interface section has some parts in common with the CODASYL COBOL sub-schema Record section (CODASYL DBTG, 1971) although these have not been enforced in the framework.
end of comment

Complete record description entry skeleton
RECORD SECTION
Record Control Entry.
[Data Description Entry.] · · · · ·

Complete record control entry
This is identical to that in C.4.2.3 except that '01' is replaced by 'RECORD'.

Complete data description entry

$$\left\{\begin{matrix}\text{INTEGER}\\ \text{REAL}\\ \text{COMPLEX}\\ \text{LOGICAL}\end{matrix}\right\}\ [\text{ARRAY}]\ \text{precision specification}$$

data-base-data-name-1 (dimension specification)
[FORMAT IS (FORMAT SPECIFICATION)]

$$[PRIVACY\ \text{LOCK}\ [\text{FOR}\ \left\|\begin{matrix}STORE\\ GET\\ MODIFY\end{matrix}\right\|]\ \text{IS}$$

$$\left\{\begin{matrix}\text{PROCEDURE data-base-procedure}\\ \text{literal} - 1\\ \text{lock-name} - 1\end{matrix}\right\}$$

$$\left[\text{OR}\ \left\{\begin{matrix}PROCEDURE\ \text{data-base procedure} - 4\\ \text{literal} - 3\\ \text{lock-name} - 2\end{matrix}\right\}\right]\ldots]\ldots$$

Syntax rules for Complete record description entry
1. Syntax rule 1 of C.4.2.3
2. Syntax rule 2 of C.4.2.3
3. Syntax rule 5 of C.4.2.3 except for the exception

General rules for complete record description entry
1. General rule 1 of C.4.2.3
2. General rule 2 of C.4.2.3 is changed to read
   'A Data Description entry is used to describe an elementary data-item of the associated schema whether or not it is included in one or more repeating groups.'
3. General rule 3 of C.4.2.3

FORTRAN INTERFACE 2.3.1 *Data-base-data-name*
Function

To select data items of interest (whether or not they are included in one or more repeating groups) from a record defined in the schema, thereby implicitly removing from view all unnamed items of the record.

To create a FORTRAN non-hierarchical record description by including data items from the schema and specifying them in terms of FORTRAN variables and arrays.

General format

Data-base-data-name (dimension specification)

Syntax rules

1. Data-base-data-name may refer to any data-item declared for the record in the schema whether or not it is included in a repeating group.
2. The inclusion of the optional dimension specification clause denotes that a FORTRAN array is involved. This clause consists of a parameter list of integer constants in brackets which specifies the maximum index value for each dimension in the normal FORTRAN manner.
3. Any data-base-data-name which is declared for this record as a 'non-array' variable in the sub-schema must have been declared in the schema as a data-item not included in a repeating group.
4. Any data-item of the schema included in one or more repeating groups may be declared as an array of the sub-schema with number of dimensions inclusively between 1 and the number of nested repeating groups containing that data-item. The correspondence between the occurrences of the data-item in the database and the array elements of the sub-schema is determined on a one to one basis as follows:
   (*a*) if a one dimensional array is declared, the occurrences are stored in the array arranged by higher to lower repeating group occurrences. That is all occurrences of a data-item at the highest repeating group level and corresponding to a set of fixed indices for the lower repeating groups are stored in consecutive array elements.
   (*b*) if a two dimensional array is declared, the one dimensional arrangement above is modified by having the second dimension correspond to the index of the lowest repeating group in the schema corresponding to the data-item.
   (*c*) A further dimension causes the appearance of a dimension to correspond to the index of the next lowest repeating group with the left to right order of dimensions corresponding to higher to lower repeating groups. This rule applies each time a further dimension is added.
   (*d*) Within each repeating group level the occurrences in the array are indexed in the same order as in the database.
5. The array dimension sizes declared in the sub-schema must be sufficient to cater for all the database occurrences of the schema repeating group levels according to rule 4 above (and also with regard to rules 6 and 7).
6. If the data-item is specified in the schema by PICTURE or TYPE CHARACTER or TYPE BIT then it may be represented in the sub-schema by one dimension of an array. The conversion rules are specified in sections FORTRAN INTERFACE 2.3.2 and 2.3.3.
7. If rule 6 is being utilised in the case of data-items included in one or more repeating groups then rules 4 and 5 apply except that an extra dimension must exist and may be considered to correspond to an imaginary higher level repeating group.
8. Syntax rule 4 of C.4.2.3.1.
9. The number of dimensions allowed for FORTRAN arrays in a sub-schema is additionally restricted by the number allowed in the FORTRAN dialect being utilised as the host language.

FORTRAN INTERFACE 2.3.2 *FORTRAN variable type specification*

Function

To specify the type of the FORTRAN variable corresponding to the data-base-data-name in the sub-schema.

By implication to determine certain conversion rules between schema data-items and FORTRAN variables.

General format

$$\left\{\begin{matrix} INTEGER \\ REAL \\ COMPLEX \\ LOGICAL \end{matrix}\right\} \text{[ARRAY] precision specification}$$

Syntax rules

1. The form of precision specification is implementation defined and will depend on the FORTRAN dialect in use. It specifies the storage allocated to the variable and in conjunction with the variable type determines the storage representation.
2. The variable types correspond to the types of variable available in FORTRAN.
3. The presence of ARRAY implies that one schema data-item corresponds to a number of elements of the sub-schema array as per FORTRAN INTERFACE 2.3.1, syntax rules 6 and 7.

General rules

The following rules are expressed in terms of the schema general rules of C.3.3.9 and C.3.3.12.

1. Conversions between TYPE arithmetic data-items and FORTRAN arithmetic variables obey rule 18 of C.3.3.12. All conversions are between one data-item and one arithmetic variable.
2. Conversions between TYPE arithmetic data-items and FORTRAN LOGICAL arrays are prohibited.
3. Rule 8 of C.3.3.12.
4. Conversions between TYPE BIT and single FORTRAN arithmetic variables obey rules 13 and 14 of C.3.3.12.
5. Conversions between TYPE BIT and FORTRAN LOGICAL arrays operate on the basis that a FORTRAN LOGICAL array is a special representation of a bit string.
6. Conversions between TYPE BIT and FORTRAN arithmetic arrays operate on the basis that each variable in the array is considered to be a packed bit string.
7. Conversions between all data-items and single FORTRAN LOGICAL variables are prohibited.
8. Conversions between TYPE CHARACTER and single FORTRAN LOGICAL arrays obey rules 10 and 11 of C.3.3.12.
9. Conversions between TYPE CHARACTER and single FORTRAN arithmetic variables or FORTRAN arithmetic arrays obey the FORMAT clause of section FORTRAN INTERFACE 2.3.3.
10. Conversions between PICTURE data-items and FORTRAN LOGICAL arrays obey rules 10 and 11 of C.3.3.12 (as rule 8 above). This is according to rule 12 of C.3.3.9 except that PICTURE numeric data-items are included.
11. Conversions between PICTURE numeric data-items and single FORTRAN arithmetic variables obey rules 16 and 17 of C.3.3.12 in conjunction with rule 12 of C.3.3.9

except under the conditions of rule 13 below.

12. Conversions between a PICTURE numeric data-item and a FORTRAN arithmetic array is prohibited except under the conditions of rule 14 below.
13. Conversions between PICTURE character data-items and single FORTRAN arithmetic variables or FORTRAN arithmetic arrays obey the FORMAT clause of FORTRAN INTERFACE 2.3.3 (as rule 9 above). This is according to rule 12 of C.3.3.9.
14. If a FORMAT clause is specified regarding conversion between a PICTURE numeric data-item and a single FORTRAN arithmetic variable or FORTRAN arithmetic array, then rules 11 and 12 above do not apply and the FORMAT clause is obeyed.

FORTRAN INTERFACE 2.3.3 *The FORMAT clause*
Function
To specify conversions which apply for certain special schema to sub-schema correspondences. In particular, most conversions between single schema data-items and FORTRAN arithmetic arrays are handled by this clause.

General format
FORMAT IS (format specification)

Syntax rules

1. The conditions under which a FORMAT clause may be used are specified in General rules 9, 12, 13, and 14 of FORTRAN INTERFACE 2.3.2.
2. The FORMAT specification is as appears in the FORTRAN Language but is interpreted differently in some respects. These are as follows:
   (*a*) the / has no effect
   (*b*) the concept of a FORTRAN unit record does not apply.
   (*c*) the A format specification accepts the whole internal character representation of the implementation machine rather than the card character set. (Note that this condition usually applies when using FORMAT in FORTRAN with auxiliary storage media rather than the card reader).
   (*d*) apart from the above, the FORMAT clause will operate according to the normal rules of the use of FORMAT in FORTRAN.

FORTRAN INTERFACE 2.3.4 *PRIVACY LOCK*
Function
To specify the privacy locks which apply to the use of a record and to the use of data-items included in a record.

General format
This is identical to that in C.4.2.3.5 except that the FORTRAN DML statements may be different.

Syntax rules
These are identical to those in C.4.2.3.5.

General rules
General rules 1 to 7 of C.4.2.3.5 apply.
General rule 8 of C.4.2.3.5 applies except for the sentence on repeating groups.

FORTRAN INTERFACE 2.3.5 *Record names*
Function
To define records which are to be included in the sub-schema. This facility is identical to that in C.4.2.3.6.

FORTRAN INTERFACE 2.3.6 *WITHIN*
Function
To define and restrict the selection of occurrences of the record named.
This facility is identical to that in C.4.2.3.10.

FRAMEWORK 2.4 *SET section*
Comment This section of the sub-schema specifications is concerned with enumerating the sets of the schema that are to be included in the sub-schema and as such should be host language independent.
end of comment
Sections C.4.2.4 through C.4.2.4.3 are included in this section unchanged except for the following alterations
C.4.2.4 and C.4.2.4.2—PRIVACY LOCK clauses

$$\text{change} \left\| \begin{matrix} ORDER \\ FIND \\ REMOVE \\ INSERT \end{matrix} \right\| \text{to } \|\text{DML-list-1}\|$$

Replace General rule 6 of C.4.2.4.2 by
'The privacy locks associated with the various statements of DML-list-1 must be satisfied in order to execute the respective statement on an occurrence of the set being described'.

FORTRAN INTERFACE 2.4 *SET section*
For this interface the DML-list-1 of FRAMEWORK 2.4 is given by DML-list-1 ≡ (*ORDER, FIND, REMOVE, INSERT*).

3. Examples of schema to FORTRAN mappings
This section provides examples of some of the more important aspects of the mappings defined in the FORTRAN sub-schema specifications presented in this paper. A number of examples are given of particular data mappings within the context of a single data record. This data was chosen for illustrative purposes only and apologies are given for any lack of realism. The data consists of information on shoe supplies as seen from the wholesale aspect. The schema record description is given in **Fig. 1.**
We will now consider each data entry of the record description in turn. The entries named MANUFTRS and STYLEINF correspond to groups of data items (data aggregates) which repeat a number of times. MANUFTRS represents a breakdown of the record data by manufacturer (with exactly three existing). STYLEINF represents a breakdown of the data by shoe style within manufacturer. In this case the number of occurrences of shoe style is variable and is specified by the value of STYLECNT (one value per manufacturer). MANUNAM is a data item specifying (for each occurrence) the name of a manufacturer. STYLENAM and STYLEPRICE are data items specifying (for each occurrence) the name and price respectively of a shoe style. Data for a sample record occurrence is given in **Fig. 2,** with the appropriate schema name shown against each data item occurrence.
The schema record description of Fig. 1 is mapped into a FORTRAN user work area by the FORTRAN sub-schema record description given in **Fig. 3.** Any problems of the length

```
RECORD IS SHOE SUPPLIES.
  Ø2 MANUFTRS OCCURS 3 TIMES.
     Ø3 MANUNAM PICTURE IS 'X(8)'.
     Ø3 STYLECNT TYPE IS BINARY FIXED 15.
     Ø3 STYLEINF OCCURS STYLECNT TIMES.
        Ø4 STYLENAM TYPE IS CHARACTER 8.
        Ø4 STYLEPRICE TYPE IS BINARY FLOAT 2Ø.
```

Fig. 1 Schema record description of shoe supplies data

DATA	*SCHEMA DATA NAMES*		
COMFORT▽		MANUNAM	
2		STYLECNT	
FLAIR▽▽▽	STYLENAM	STYLEINF	MANUFTRS
8.0	STYLEPRICE		
PRIDE▽▽▽	STYLENAM	STYLEINF	
6.5	STYLEPRICE		
LOBIS▽▽▽		MANUNAM	
3		STYLECNT	
ELEGANT▽	STYLENAM	STYLEINF	
9.2	STYLEPRICE		MANUFTRS
CASUAL▽▽	STYLENAM	STYLEINF	
5.5	STYLEPRICE		
FASHION▽	STYLENAM	STYLEINF	
7.5	STYLEPRICE		
MAKERITE		MANUNAM	
2		STYLECNT	
STRONG▽▽	STYLENAM	STYLEINF	MANUFTRS
7.5	STYLEPRICE		
LONGLIFE	STYLENAM	STYLEINF	
8.0	STYLEPRICE		

Fig. 2 Data for a sample record occurrence of the shoe supplies record

of FORTRAN variable names are ignored but in practice would be handled using the renaming facility.

As a result of invoking the FORTRAN sub-schema of Fig. 3 in a FORTRAN program the FORTRAN variables (or arrays) with the names STYLECNT, STYLEPRICE, MANUNAM and STYLENAM will be effectively declared in that program. The combined contents of these variables then corresponds to the contents of an occurrence of a database record as seen by the program utilising the sub-schema. The programmer would process the contents of the user work area using normal FORTRAN statements and would call a DML statement whenever he wished to interact with the database by, for instance, obtaining in his user work area a new record occurrence from the database or updating a record occurrence in the database with the contents of this user work area.

Each of the FORTRAN variables in the sub-schema of Fig. 3 illustrates a different sub-schema mapping option and we will consider them in turn using the sample data of Fig. 2. First, however, a comment is required on the exclusion of MANUFTRS and STYLEINF from the sub-schema. These two names correspond to data aggregates in the schema record. That is, each one is composed of more than one named data item. For instance STYLEINF consists of STYLENAM and STYLEPRICE. Data aggregates are excluded from the FORTRAN sub-schema by the present specifications. This is indicated in the specifications by: FORTRAN INTERFACE 2.3, General rule 2; FORTRAN INTERFACE 2.3.1, Function and Syntax rule 1.

```
RECORD SHOESUPPLIES
  INTEGER*2 STYLECNT(3)
  REAL STYLEPRICE(15)
  INTEGER ARRAY MANUNAM(2, 3)
    FORMAT IS (2A4)
  INTEGER ARRAY STYLENAM(8, 5, 3)
    FORMAT IS (8A1)
```

Fig. 3 FORTRAN sub-schema record description of shoe supplies data

STYLECNT is a data item appearing in a single repeating group in the schema. It is represented in the sub-schema by an array of one dimension. The lack of the optional ARRAY qualifier in the data description entry of the sub-schema implies that one database data item occurrence is mapped into one FORTRAN variable (in this case an array element). The effect of transferring the sample data of Fig. 2 from the database to the user work area is shown in **Table 1** and involves the use of the following sub-schema specification items: FORTRAN INTERFACE 2.3.1, Syntax rules 4,4(*a*), 4(*d*) and 5; FORTRAN INTERFACE 2.3.2, General rule 1.

STYLEPRICE is a data item appearing in two nested repeating groups in the schema. It is represented in the sub-schema by an array of one dimension. Thus two database dimensions must be mapped into a single user work area dimension. The lack of the optional ARRAY qualifier in the data description entry of the sub-schema implies that one database data item occurrence is mapped into one FORTRAN variable (in this case an array element). The effect of transferring the sample data of Fig. 2 from the database to the user work area is shown in **Table 2** and involves the use of the following sub-schema specification items: FORTRAN INTERFACE 2.3.1, Syntax rules 4,4(*a*), 4(*d*) and 5; FORTRAN INTERFACE 2.3.2, General rule 1.

The ordering by higher to lower repeating groups is illustrated in Table 2 by the adjacency of all occurrences of STYLEPRICE of the STYLEINF repeating group for each occurrence of the lower MANUFTRS repeating group.

MANUNAM is a data item appearing in a single repeating group in the schema. It is represented in the sub-schema by an array of two dimensions. The ARRAY qualifier in the data description entry of the sub-schema implies that one database data item occurrence is mapped into a number (more than one) of array elements within one array dimension. From considering FORTRAN INTERFACE 2.3.1, Syntax rules 6 and 7 in conjunction with Syntax rules 4,4(*a*), 4(*b*), 4(*d*) and 5 we see that the first user work area dimension is used to split the individual schema data items and that the second dimension enumerates the three occurrences of the single database repeating group involved. The mapping between one database

Table 1 Contents of STYLECNT in user work area using data from Fig. 2

STYLECNT(3)	
Element No.	*Value*
1	2
2	3
3	2

Table 2 Contents of STYLEPRICE in user work area using data from Fig. 2

STYLEPRICE (15)	
Element No.	*Value*
1	8·0
2	6·5
3	9·2
4	5·5
5	7·5
6	7·5
7	8·0
8–15	Undefined

data item occurrence and an array dimension is handled, as specified in FORTRAN INTERFACE 2.3.2, General rule 13, by the FORMAT clause. This clause is specified in FORTRAN INTERFACE 2.3.3. In this case, each 8 character manufacturers name has been divided into two elements of 4 characters by the FORMAT 2A4. Note that this FORMAT applies to all uses of this sub-schema record and cannot be altered by a FORMAT specification in a host language program. **Table 3** illustrates the MANUNAM mapping using the sample data of Fig. 2.

STYLENAM is a data item appearing in two nested repeating groups in the schema. It is represented in the sub-schema by an array of three dimensions. As in the case of MANUNAM, the ARRAY specification is present, implying that the first dimension will be used to split a database data item occurrence into a number of array elements. From FORTRAN INTERFACE 2.3.2, General rule 9 we see that the FORMAT clause applies to the mapping. In this case the FORMAT of 8A1 causes each of the 8 character database data item occurrences to be arranged as one character (plus appropriate space character fillers) in each of eight array elements. The second dimension then corresponds to the higher STYLEINF repeating group occurrences and the third dimension to the lower MANUFTRS repeating group occurrences. This involves FORTRAN INTERFACE 2.3.1, Syntax rules 6 and 7 in conjunction with Syntax rules 4, 4(*a*), 4(*b*), 4(*c*), 4(*d*) and 5. This is shown in **Table 4** where the sets of data values corresponding to variation of the first array subscript with fixed second and third array subscripts are shown by vertical displacement within the appropriate position of the matrix of the second and third dimensions.

Table 3 Contents of MANUNAM in user work area using data from Fig. 2

MANUNAM (2, 3)

Dimension 2	1	2	3
Dimension 1			
1	COMF	LOBI	MAKE
2	ORT∇	S∇∇∇	RITE

Table 4 Contents of STYLENAM in user work area using data from Fig. 2

STYLENAM (8, 5, 3)

Dimension 3	1	2	3
Dimension 2			
	F∇∇∇	E∇∇∇	S∇∇∇
	L∇∇∇	L∇∇∇	T∇∇∇
	A∇∇∇	E∇∇∇	R∇∇∇
1	I ∇∇∇	G∇∇∇	O∇∇∇
	R∇∇∇	A∇∇∇	N∇∇∇
	∇∇∇∇	N∇∇∇	G∇∇∇
	∇∇∇∇	T∇∇∇	∇∇∇∇
	∇∇∇∇	∇∇∇∇	∇∇∇∇
	P∇∇∇	C∇∇∇	L∇∇∇
	R∇∇∇	A∇∇∇	O∇∇∇
	I ∇∇∇	S∇∇∇	N∇∇∇
2	D∇∇∇	U∇∇∇	G∇∇∇
	E∇∇∇	A∇∇∇	L∇∇∇
	∇∇∇∇	L∇∇∇	I ∇∇∇
	∇∇∇∇	∇∇∇∇	F∇∇∇
	∇∇∇∇	∇∇∇∇	E∇∇∇
		F∇∇∇	
		A∇∇∇	
		S∇∇∇	
3		H∇∇∇	
		I ∇∇∇	
		O∇∇∇	
		N∇∇∇	
		∇∇∇∇	
4	UNDEFINED		
5	VALUES		

Concluding remarks

The publishing of the CODASYL Data Base Task Group April 1971 Report represented an important landmark in the development of database systems and the specifications contained therein will undoubtedly have considerable influence on database systems and practices in the future. It is therefore important that these specifications are considered in the context of existing major languages other than COBOL. It is hoped that this paper will stimulate discussions on the relevance of the CODASYL proposals to FORTRAN in particular and other languages in general and will encourage progress towards a truly multi-host language database system.

Acknowledgements

The author acknowledges his debt to the CODASYL Data Base Task Group for their work as embodied in their April 1971 Report.

The author would also like to thank the CODASYL Data Description Language Committee for providing the opportunity for presentation of the sub-schema framework concept at their UK meeting in October 1972, and individual members of that committee for their encouragement of this work.

Thanks are also due to various members of the Edinburgh Regional Computing Centre for their helpful comments during preparation of the sub-schema specifications and to the referee for suggesting the inclusion of examples.

Reference

CODASYL Data Base Task Group Report, April 1971.