Data Base Facility for COBOL 80

by MARGARET M. COOK
Department of the Navy
Washington, DC

INTRODUCTION

The American National Standards Committee (ANSC) X3J4 is responsible for developing the standard for the programming language COBOL. ANSC X3J4 is currently in the process of revising American National Standard Programming Language COBOL, X3.23-1974.[1] As part of this revision process, X3J4 is developing a candidate Database module for inclusion in the next COBOL Standard. The earliest date that a new COBOL Standard could be developed and approved is the year 1980, so this paper will refer to the revision of X3.23-1974 as COBOL 80.

Direction for developing the COBOL Standard is given to X3J4 by its parent committee, ANSC X3, the ANSI technical committee responsible for Computers and Information Processing. According to X3 direction, the source of the language specifications for COBOL 80 must be either the current COBOL Standard or the CODASYL COBOL Journal of Development.[2] Thus the only possible candidate for standardization of a Data Manipulation Language (DML) for COBOL is the CODASYL database facility. X3J4 cannot consider any other database model, such as relational, for standardization.

In order to develop a candidate Database module for COBOL 80, X3J4 has established a Database Task Group, X3J42, which has the responsibility of reviewing the database facility specifications in the CODASYL COBOL Journal of Development (JOD) and identifying those elements which are to be included in the candidate Database module. The JOD cutoff date for COBOL 80 has been established as January 1, 1978. Database elements in the JOD as of the cutoff date are the source of the database specifications which can be included in COBOL 80.

This paper discusses the structures which can be defined according to the CODASYL database specifications and the language elements which are available for standardization. The standardization effort by X3J4, and hence its subcommittee X3J42, is limited to that portion of a Database Management System (DBMS) which pertains to the defining of a Sub-schema and the Data Manipulation Language statements proposed for the host language COBOL. The portions of a DBMS which are outside the purview of COBOL, such as the Data Definition Language and the Device Media Control Language, are not candidates for standardization by X3J4. However, the functions and structures of the Data Definition Language, because of their close relationship to the Sub-schema definition, are discussed in this paper.

OVERVIEW OF THE DATA BASE FACILITY

The portions of the CODASYL database facility which are of concern to X3J4 are the database functions which permit an application program written in COBOL to communicate with the Database Management System. The functions required are a definition of that portion of a data base of interest to an application program and the procedural statements required to access the database.

The entire database is defined in the Schema. The Schema contains a description of inter-record relationships and the data items comprising each record in the database. Because the Sub-schema definition is a subset of the Schema definition, understanding the Schema and its capabilities is a prerequisite for understanding the Sub-schema definition.

The COBOL Sub-schema defines the portion of the database available to an application program. The part of the database that is not required by a given application area is not defined in the Sub-schema. The Sub-schema definition is a subset of the Schema definition but can define data items in a different manner than the Schema, as long as the definition is consistent with the Schema.

The COBOL Data Manipulation Language statements for the database facility contain update functions for data items and record relationships, and record and item retrieval functions. The DML statements are contained in the Procedure Division of a COBOL program and can only reference records and record structures which have been defined in the Sub-schema referenced by the COBOL program.

The Schema, COBOL Sub-schema and Procedure Division DML statements combined provide a database facility through the host language COBOL. The succeeding sections of this paper discuss each of these components in detail.

THE SCHEMA DEFINITION

The logical definition of the database is provided by the Schema. The Schema defines the record types in the database, the inter-record relationships, and the access control

mechanisms. The specifications for the Schema definition are contained in the CODASYL Data Definition Language Journal of Development.[3]

Record definition

The basic unit of access in the CODASYL database facility is the record. Each record type in the database is defined in the Schema. The record definition specifies the data items and data aggregates which comprise records of a particular record type. The smallest unit of data is the data item. In data subentries within a record definition, complete descriptions are given for each data item in the record. Among the data characteristics specified are the data type, that is, arithmetic, character or bit string, system generated identifier (database key), or implementor-defined extension; validation values for checking accuracy of data item contents when storing items in the database, and access control locks for retrieving, storing, or modifying data items. A named collection of data items in a record type is called a data aggregate and permits the definition of vectors and repeating groups.

In addition to data items whose contents are supplied by the user, the definition of derived data items is also permitted in the CODASYL Schema. Derived data items are data items whose contents are generated by the DBMS according to specifications in the data subentry. There are two types of derived data items, actual and virtual. The DBMS generates the contents of an actual derived data item during a store or modify and the contents are stored as part of the record. A virtual derived data item is generated by the DBMS when a record is retrieved from the database. The item does not exist in the database but is available to a user program as if it were stored in the database.

Within the Schema definition an arbitrary number of record types may be defined, and there is a record description for each type of record in the database. For any given record type defined in the Schema, there may be none, one, or an arbitrary number of occurrences of records in the database.

Set definition

A Set is the basic structure by which inter-record relationships are defined in a CODASYL Schema. Unfortunately, the use of the word set is confusing to those familiar with set as a mathematical entity. The misuse of the letters "SET" according to Steele is "That linguistic atrocity perpetrated by the DBTG Report wherein the nineteenth, fifth, and twentieth letters of the Roman alphabet are used in that order as the name of a peculiar object. This may seem harsh, but the point at issue represents a prize example of the manner in which the information processing sciences generate confusion for themselves and others by casual misuse of words."[4]

A set type in the Schema is defined by a single owner record type and one or more member record types. A record type may be defined to be the member of more than one set type, and a member of one set type may be the owner of a different set type. The set type relationship permits the definition of network structures in the CODASYL database facility.

There are restrictions on the record relationships permitted in the Schema set definition. A record type cannot be defined as both owner and member of the same set type, and a set type may not have more than one owner record type. Thus the many-to-many owner-member relationship is not directly supported by the CODASYL structure.

Set membership

There are two classes of set membership defined for each record type. The first class concerns the insertion of records into a set, and the second class concerns the removal of records from a set. For the insertion of records the member record types can be either automatic or manual. If automatic membership is specified, membership in a set is established by the DBMS when a record is created in the database. If manual membership is specified, a Data Manipulation Language statement is required to connect an already existing record in the database to its owner record.

For the removal of records from a set, the member record types can be either mandatory or optional. If mandatory membership is specified, once a record is a member of a set it cannot be removed from membership in that set type. If optional membership is specified, a member record can be disconnected from a set of the given set type by the execution of a DML statement.

Set selection

The method for the selection of a specific set from the totality of sets in the database is specified in the set selection criteria. The identifiers are specified which are to be used in navigating from owner record to member record in a series of sets on a continuous path, where the member of a given set is the owner of the next set in the path. The values of the named data items are used to select a specific set in the database.

Set order

Each set type defined in the Schema must have an order specified for the member records of a set of the given set type. The member records may be ordered by ascending or descending keys. The keys can be data items, member record names, or system generated identifiers. The member records may also be ordered based upon the insertion order of new member records into the set, for example, first, last, or before another member record selected by an application program.

REALMS

In the Schema the database can be defined to be logically divided into realms (areas). A realm is similar to a COBOL file in that a Data Manipulation Language statement is required before a program can access records within a given realm. The record type definition specifies the realms in which the records of a particular type may appear. An individual record can be associated with only one realm, and a record may not change realms.

There may be an arbitrary number of realms defined in the Schema, and records can be assigned to realms independently of their inter-record relationships (set structure). An owner record and member records of a set may be contained in the same realm or in many different realms. Similarly, a given record type or set type may have occurrences which appear in the same realm or in many different realms.

COBOL SUB-SCHEMA DESCRIPTION

A COBOL Sub-schema definition specifies the data items, record types, set types and realms which can be accessed by a COBOL program. A COBOL Sub-schema contains an individual program's view of the database. Only those data items, records, sets, and realms described in a COBOL Sub-schema may be accessed in a COBOL program referencing that Sub-schema. The portions of a database beyond the scope of an individual program are not defined in the Sub-schema referenced by the COBOL program and are not accessible to the program.

There may be an arbitrary number of COBOL Sub-schemas described for the Schema, and definitions for different Sub-schemas may reference the same Schema entries. A COBOL program references only one Sub-schema, but an arbitrary number of programs may reference the same Sub-schema.

COBOL Sub-schema relationship to schema

A COBOL Sub-schema definition is a subset of the database description given by the Schema. A Sub-schema specifies the data items, records, sets and realms which can be referenced by a COBOL program. A Sub-schema cannot define new record types or set types, and all data defined in the Sub-schema must have been previously described in the Schema.

Important differences in the definitions for the Schema and COBOL Sub-schema are permitted in the CODASYL database facility. The Database Management System uses the Schema and COBOL Sub-schema definitions to perform the data conversion required for access by a COBOL program.

The Sub-schema references data items and records in the Schema by their names given in the Schema. The names for identical data items in the Sub-schema and Schema can differ if they are the subject of an Alias Description entry, in which case use of the Sub-schema names is equivalent to use of the Schema names. The COBOL program always references the data-names defined in the Sub-schema.

Only a portion of a Schema record need be defined in the Sub-schema record and the data items in the Sub-schema may be specified in a different order. The data mapping is performed by matching data-names and record-names to the data-names and record-names in the Schema. The mapping is performed at the elementary item level, and the data description in the Sub-schema is the description used by the COBOL program.

The characteristics of data items defined in the Sub-schema can differ from the Schema specifications as long as the results are consistent with the Schema definition. Data transformation of equivalent data items is performed according to the MOVE statement rules for elementary sending and receiving items. Alternately, a database procedure could be defined in the Schema which performs the required transformation, in which case the MOVE statement rules do not apply.

Data item validation is performed by the DBMS as it stores data into the database or retrieves data from the database. The data item validation values specified in the Sub-schema may be different from the validation values specified in the Schema, but the Sub-schema specification must be within the range of the Schema specification. The DBMS checks the contents of data items when retrieving them from the database and does not retrieve items outside of the Sub-schema specified range.

Access control locks for a given realm, set type, record type, or data item can be declared in either the Schema or Sub-schema, or in both. If an access control lock for a given entity has been declared in both the Schema and Sub-schema, the Sub-schema access control lock overrides the Schema lock and the Sub-schema access control lock must be satisfied by the COBOL program.

The set description in the Sub-schema permits the specification of the arguments to be used by the DBMS in defining the set selection criteria. If set selection criteria are defined for a record type in the Sub-schema, they replace the set selection criteria defined in the Schema. If set selection criteria for a record type are not defined in the Sub-schema, then the Schema set selection criteria remain in effect.

Data independence

Although the data items defined in the Sub-schema must be defined in the Schema, important differences are permitted which give COBOL programs data independence. The definition of data in the database is contained in the Schema which is a step removed from the COBOL program's view of the database as given in the Sub-schema. Because of the differences between the Schema and Sub-schema which are permitted in the CODASYL database facility, changes can be made to the Schema without making any modifications to the programs accessing the database.

Modification of the Schema may require modification of the Sub-schemas based on the Schema. If the changes redefine the set structures or remove data items from a record type, then changes to the Sub-schemas referencing these structures and items must be made. If the changes are the addition of new data items, record types, and set structures which are not referenced by the Sub-schema, or if the data descriptions are changed, Sub-schema modifications are not required.

COBOL Sub-schema divisions

A COBOL Sub-schema source program consists of three divisions: Title, Mapping, and Structure. The Title Division specifies the name of the Sub-schema and the name of the Schema to which the Sub-schema is related, the Sub-schema access control key which must satisfy the Schema access control lock in order for the Sub-schema to be placed in the Sub-schema library, and the access control locks associated with the use of the Sub-schema. The Mapping Division contains the Alias Section where Sub-schema names not defined in the Schema are equated to names defined in the Schema. Identifiers, realm-names, record-names, and set-names may be named in the Alias Section.

The Structure Division contains the Realm Section, the Set Section, and the Record Section. This division specifies the portion of the database that is to be made available to a COBOL program and the structure of that portion of the database. The Realm Section specifies the Sub-schema realms and the access control locks associated with the use of the realms. The Set Section specifies the Schema set types to be made available to the Sub-schema, the set selection criteria, and access control locks associated with the use of the Sub-schema sets. The Record Section specifies the Schema record types and the data items within the record types that are to be made available to a Sub-schema. Access control locks associated with the use of records and data items, and the data item characteristics are also specified in the Record Section.

Compilation and binding time of COBOL Sub-schema

The Schema, COBOL Sub-schema and COBOL program are required to operate as a unit in order to perform database functions. The time of compilation of the Sub-schema and the binding time of the Schema, Sub-schema, and COBOL program as a run unit are extremely important. Some of the possible times for binding are COBOL compile time, COBOL Sub-schema compile time, execution time, or any other implementor-defined time.

The results for a program using the database facility could vary depending upon the binding time. For example, if a Sub-schema and COBOL program are compiled and bound together with a Schema as a run unit, and the Schema definition is then changed, the results of the execution could be quite different from compiling and binding the Sub-schema and COBOL program with the Schema after the changes are made to the Schema.

COBOL 80 Sub-schema subsetting

If a COBOL database facility is included in COBOL 80, it is quite probable that the COBOL 80 Sub-schema will be a subset of the COBOL Sub-schema specifications in the CODASYL COBOL Journal of Development. The X3/SPARC Database Systems Study Group has expressed concern over declarations in the Schema being overridden by declarations in the Sub-schema. Votes taken by ANSC X3J4 indicate that this concern is shared by X3J4.[5] It is probable that the specifications for access control locks for realms, set types, record types and data items will not be contained in the COBOL 80 Sub-schema. If access control locks were permitted for these resources, they would take precedence over access control locks specified in the Schema for the same resources.

The set selection criteria specified in the Sub-schema replaces the set selection criteria in the Schema and may not be included in the COBOL 80 Sub-schema for that reason. In the case of the set selection criteria, there is sentiment in X3J42 that the Sub-schema override of the Schema is not a problem and set selection criteria should be included in the COBOL 80 Sub-schema.

DATA DIVISION SUB-SCHEMA SECTION

If the database facility is included in COBOL 80, the Sub-schema Section will be added to the Data Division. The Sub-schema Section consists of a Sub-schema entry which specifies the COBOL Sub-schema that is to be accessed by the COBOL program, and the access control key which must satisfy the access control lock associated with the Sub-schema.

Only one Sub-schema entry is permitted per COBOL program. Multiple Sub-schemas per program are not presently permitted by the CODASYL Journal of Development and hence cannot be permitted by COBOL 80.

PROCEDURE DIVISION DML STATEMENTS

The Procedure Division Data Manipulation Language statements provide an interface to the database for a COBOL program. The Procedure Division DML statements provide the capability to update the database, to retrieve records and data items from the database, and to control the availability of realms to the COBOL program.

Update function statements

There are DML statements which provide update functions for both set structures, records, and data items within a record. Records are stored in the database by executing a STORE statement. The record stored becomes a member

of each set type in which it has been declared in the Schema to be an automatic member, and the record becomes the owner of an empty set (set with no members) for each set type for which it has been defined to be an owner. Records are removed from the database by executing an ERASE statement. The ERASE statement can remove one record, or with the use of additional options, all records within a set structure for which the current record is the owner record.

A record becomes a member in one or more sets for which the record type has been defined to be a manual member by executing a CONNECT statement. A record is removed from one or more sets for which its record type has been defined to be optional by executing a DISCONNECT statement.

The MODIFY statement is used to change the contents of one or more data items in a record, the set membership of a record, or to change both data items within a record and the set membership of a record. Execution of a MODIFY statement without the ONLY option causes the DBMS to replace in the database the contents of data items within the current record with the contents of the record area associated with the record. Derived data items may be indirectly changed by execution of a MODIFY statement. If the INCLUDING option or the ONLY option is specified in a MODIFY statement, the set membership of the current record is changed in accordance with the set ordering criteria of each set type.

The ORDER statement permits the logical reordering of members of the set of which the current record is either an owner or a member record. The logical reordering may apply to only the current run unit, or the reordering of the set member records may occur in the database. The object records of an ORDER statement may be all the records referenced by a given record-name, all records in which a given data-name is defined, or all records that are members of the set containing the current record.

Retrieval function statements

The two DML statements which are used to locate records and retrieve all or part of a located record are the FIND and GET statements. The FIND statement is used to locate a specific record in the database which then becomes the current record of the run unit. Execution of the FIND statement does not retrieve the selected record but identifies it as the current record for use in subsequent DML statements. The record selection expression in the FIND statement specifies the criteria to be used in determining the object record of a FIND statement. The record selection criteria may be based on record contents, system generated identifiers, or a record's relationship with other records.

The GET statement is used to move all or part of a record from the database into its associated record area where it can be accessed by other Procedure Division statements in a COBOL program. The GET statement without any identifier specified or with identifier consisting of a record-name, moves the portion of a database record defined in the Subschema into its associated record area. If identifiers referencing data items are specified, the contents of those data items are moved into the associated record area. Data items not referenced are unchanged in the record area. In all cases the current record of the run unit is the database record accessed.

Control function statements

There are two DML statements which control a COBOL program's access to database realms. The READY statement prepares one or more realms for processing and specifies whether the records in a realm can be accessed and modified, or only accessed. The READY statement also indicates whether the records in a realm are for the exclusive use of the run unit, or if other run units are allowed to access the realm but prevented from updating the specified realm.

The FINISH statement terminates the availability of one or more realms to the run unit. The individual realms to be terminated can be specified by name, or a set name can be referenced in which case all realms which contain the owner or members of the named set are terminated.

The DML includes the COBOL IF statement with database conditions as the conditional expression to be tested. There are three database conditions which may be tested: Tenancy, Member, and Nullity. The Tenancy condition determines if the correct record of the run unit is the owner, member, or either an owner or a member of a set of a given set type. The Member condition determines if a given set has any member records. The Nullity condition is used to determine if a specified data item has the null attribute.

USE declarative statement

The USE declarative statement for the COBOL data base facility specifies the procedures to be used in producing access control keys for satisfying Schema and Sub-schema access control locks pertaining to functions performed on realms, records, sets or identifiers. The USE statement also specifies the procedures to be executed on database exception conditions.

Concurrent run units

There are three DML statements which are included in the COBOL database facility to monitor concurrent usage of records by different run units. The three COBOL statements are KEEP, FREE, AND REMONITOR. The entire area of monitored mode and the functions of these verbs are expected to change from those specified in the Journal of Development in the Fall of 1977, so these statements will not be further explained. ANSC X3J4 has agreed that the KEEP, FREE, and REMONITOR statements, and the extended monitored mode as they are specified in the Fall 1977 COBOL JOD are not suitable for standardization and hence will be excluded from COBOL 80.

COBOL 80 DML statements subsetting

As mentioned earlier in the Sub-schema explanation, ANSC X3J4 is concerned about Sub-schema declarations replacing Schema declarations, and Sub-schema declarations being overridden by programmer action. Votes taken by X3J4 give an indication of the possible subsetting which will occur in determining the DML functions suitable for standardization.[5,6] The ORDER, KEEP, FREE, and REMONITOR statements have been identified as statements which will not be included in COBOL 80. Also, it is probable that the USE statement for data base functions will be subset for COBOL 80.

There are many problem areas which CODASYL is trying to resolve before the cutoff date. Among the data base features which have been identified as problem areas are the NULL concept (Is NULL a value or an attribute?), the results of MODIFY on derived data items not available to the Sub-schema, and the cascading effect of the ERASE ALL statement. Some of these problems may be eliminated by subsetting the database facility.

CONCLUSION

There are many obstacles in the path of the standardization of a database facility for the language COBOL. The Database Task Group X3J42 must develop a candidate Database module based on the CODASYL COBOL Journal of Development 1978. The proposed module must then be approved for publication for public comment as a draft proposed standard, either as a separate document or as a module of COBOL 80. The public comments are reviewed by X3J4, and changes made to the draft Database module based upon the public comments. The final Database module specifications must then be approved by letter ballot by X3J4 and their parent committee X3. After approval by X3, the Database Module is forwarded to the ANSI Board of Standards Review for approval of the publication of the Database module as an American National Standard.

Even if a Database module is approved for inclusion in COBOL 80, much work is still required before there will be a database standard. The inclusion of a database facility in COBOL 80 would only standardize the COBOL Sub-schema and COBOL Data Manipulation Language statements. There would then exist the anomaly of a standard for the COBOL database facility without a standard for the Data Definition Language. Standardization of a COBOL database facility is meaningless without a Data Definition Language standard.

REFERENCES

1. *American National Standard Programming Language COBOL, X3.23-1974,* American National Standards Institute Incorporated, New York 1974.
2. *CODASYL COBOL Journal of Development 1976* (page changes through May 1977), Department of Supply and Services, Ottawa 1976.
3. *CODASYL Data Definition Language Journal of Development 1973,* National Bureau of Standards Handbook 113, U.S. Government Printing Office, Washington, D.C. 1973.
4. Steele, T. B., "Data Base Standardization A Status Report," *IFIP TC-2 Working Conference,* January 1975.
5. Minutes ANSC X3J4 Meeting Number 90, CBEMA, Washington, D.C., July 1976.
6. Minutes ANSC X3J4 Meeting Number 91, CBEMA, Washington, D.C., September 1976.