

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|-----------------------|--|
| 1. REPORT NUMBER NRL Report 8038 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) THE CODASYL DATA DESCRIPTION LANGUAGE: STATUS AND ACTIVITIES, APRIL 1976 | | 5. TYPE OF REPORT & PERIOD COVERED Interim report on a continuing NRL problem |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Frank A. Manola | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, D.C. 20375 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problem B02-24 RF-21-211-401 FR-22-242-401 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Va. 22217 | | 12. REPORT DATE November 22, 1976 |
| | | 13. NUMBER OF PAGES 34 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 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 Presented at the SHARE Working Conference on Data Base Management Systems, Apr. 26-30, 1976, Montreal, Canada. NTIS WGA Category 62. | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) CODASYL Data description language committee Data base Data description languages Data base management Data structures Data definition languages DDLC | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The CODASYL Data Description Language Committee (DDLC) held its first meeting on November 30, 1971. It was instituted to use the work reported on by the CODASYL Data Base Task Group (DBTG) in its April 1971 report as a base for development of specifications for a host-language independent data description language (the schema DDL). In addition, the DDLC formulated objectives of investigating related areas, such as the relationship between the schema DDL (used to describe the entire data base) and host-language dependent subschema DDL's (used | | |

(Continued)

20. Continued

to describe the parts of data bases known to specific application programs). However, the committee felt that its most immediate purpose was publication of a DDL specification as soon as possible, and that this could be best done by temporarily limiting its activities to clarification and minor extensions of the schema DDL as specified in the April 1971 DBTG report. With the publication of the *CODASYL Data Description Language Journal of Development*, in June 1973, this initial phase was concluded. Since the publication of this initial *Journal of Development*, there has been much activity in data-base and data-description technologies. The work has been international in scope and has been sponsored by computer societies, user groups, standards organizations, academic institutions, and governmental agencies. The CODASYL DDLC has informally (by the outside activities of its members) and formally (by its relations with other groups), continuously monitored these developments, and various changes in the schema DDL language specifications, to reflect the progress, made. This report briefly describes the activities of the CODASYL DDLC, its language specifications, and its future plans.

CONTENTS

| | |
|---|----|
| 1. INTRODUCTION | 1 |
| 2. DATA DESCRIPTION LANGUAGE COMMITTEE STRUCTURE | 2 |
| 2.1 Subschema Task Group | 3 |
| 2.2 Data Base Administration Working Group | 4 |
| 2.3 Working Group on Environment | 5 |
| 2.4 Data Manipulation Task Group | 6 |
| 3. CHANGES IN LANGUAGE SPECIFICATIONS | 6 |
| 3.1 DDL Categorization | 7 |
| 3.2 Specific DDL Changes | 11 |
| 4. TECHNICAL OBJECTIVES | 17 |
| 4.1 Improved Definitions of Data Manipulation Functions | 17 |
| 4.2 Extensions of the Set Construct | 18 |
| 4.3 Additional Tuning Statements | 18 |
| 4.4 Freer Syntax for Writing Schemas | 19 |
| 4.5 Additional Consistency Declaration Facilities | 19 |
| 4.6 Eliminating Undesirable Dependence on Tuning or Resource Allocation Elements | 19 |
| 4.7 Eliminating Tuning and Resource Allocation Statements from the DDL | 19 |
| 4.8 Other Objectives | 20 |
| 5. LIAISON ACTIVITIES | 20 |
| 5.1 British Computer Society (BCS) | 20 |
| 5.2 European Computer Manufacturers Association (ECMA) | 20 |
| 5.3 Information Processing Society of Japan (IPSJ) | 20 |
| 5.4 ANSI/X3/SPARC Study Group on Data Base Management Systems | 21 |
| 5.5 International Federation for Information Processing (IFIP) | 21 |
| 6. PUBLICATIONS | 21 |

| | |
|---|----|
| ACKNOWLEDGMENTS | 22 |
| REFERENCES | 22 |
| APPENDIX A — Organizations that Are or Have Been Members of the DDLC | 23 |
| APPENDIX B — DDL Categories | 25 |
| APPENDIX C — IFIP Conference Recommendations and DDLC Action | 29 |

THE CODASYL DATA DESCRIPTION LANGUAGE: STATUS AND ACTIVITIES, APRIL 1976

1. INTRODUCTION

The CODASYL Data Description Language Committee (DDLC) was instituted to take the work of the CODASYL Data Base Task Group (DBTG), as reflected in its April 1971 Report [1], as a base and to develop from it specifications for a host-language independent data description language (the schema DDL). The DDLC held its first meeting November 30, 1971. The DDLC formulated objectives to investigate certain related areas, such as the relationship between the schema DDL (which is used to describe the entire data base) and host-language dependent subschema DDL's (which are used to describe the parts of data bases known to specific application programs). At the time, however, the committee felt that its most immediate purpose was publication of a DDL specification as soon as possible, and that this was best accomplished by temporarily limiting its activities to clarification and only minor extensions of the schema DDL as specified in the April 1971 DBTG Report. With the publication of the *CODASYL Data Description Language Journal of Development, June 1973* [2], this initial phase of the DDLC's activities was concluded. Since the publication of this initial *Journal of Development* (JOD), there has been great interest, activity, and development in the data base and data description technologies. This activity has been international and has been sponsored by various institutions, including computer societies, user groups, standards organizations, academic institutions, and governmental departments. The CODASYL DDLC has informally (by the outside activities of individual members) and formally (by establishing relations with other groups and activities) continuously monitored these developments. Various changes have been made in the schema DDL language specifications to reflect the development of data base systems technology. This report describes briefly the current status and activities of the CODASYL DDLC, the status of the DDLC language specifications, and future directions for DDLC work.

Before presenting the basic material of this report, however, two disclaimers and a warning must be given. The first disclaimer is that the statements made are the responsibility of the author, not of the DDLC or CODASYL. The second disclaimer is that only those specifications contained in the June 1973 DDL JOD [2] are CODASYL-approved language specifications. Changes reported here as having been made by the DDLC are changes in the working JOD maintained by the DDLC for internal use, but are not yet official. Some of the reported changes may be "undone" or otherwise altered before approval and publication of the next official JOD. The warning is that an acquaintance with the June 1973 DDL JOD, or with the April 1971 DBTG Report, is necessary for a full understanding of the DDL changes reported here.

2. DATA DESCRIPTION LANGUAGE COMMITTEE STRUCTURE

The CODASYL Data Description Language Committee (DDLC) is a standing committee under the CODASYL organization. As such, it is responsible for the specifications of the DDL in the same way that the Programming Language Committee (PLC) is responsible for the specifications of the COBOL language. The DDLC has, since its beginning, been a focal point of much of the data-base-related activity in CODASYL. Membership on the DDLC is institutional; generally, each member is represented continuously by one or two persons. Organizations that have been members of the DDLC since its beginning are listed in Appendix A. Current DDLC members are listed below.

Aberdeen University, Scotland
B. F. Goodrich Company
Cincom Systems, Inc.
Consolidated Analysis Centers, Inc.
Control Data Corporation
Computer Sciences Corporation
Defense Communications Agency
Department of the Navy
Digital Equipment Corporation
General Electric Company
Honeywell Information Systems, Inc.
International Business Machines Corporation
International Computers, Ltd., U. K.
National Bureau of Standards
National Security Agency
Ohio State University
Philips-Electrologica, Netherlands
Scientific Control Systems, Ltd., U. K.
Southern Railway System
Sperry Univac Corporation
U. S. Air Force
U. S. Army
University of Florida

In addition to its member organizations, the DDLC maintains on its mailing list a number of other organizations that are applying for DDLC membership, are designated observers of DDLC's activities, or are cooperating with the DDLC in its development activities. These organizations are listed below.

CODASYL Programming Language Committee
British Computer Society (BCS)
BCS Advanced Programming Group
Bell Telephone Laboratories (observer)
Boeing Computer Services (observer)
Statskonsult, Sweden
European Computer Manufacturers Association (ECMA)
ECMA TC-22 (Data Base Management Systems)
Information Processing Society of Japan (DBLWG)

Academy of Sciences of the U.S.S.R.
NCR (observer)
Software Sciences Limited (observer)

Since 1973, the DDLC has formed task groups and working groups to study and make recommendations in several particularly important areas. In general, membership in the DDLC has not been a requirement for participation in these groups, nor has participation in itself conferred DDLC membership. These groups, the Subschema Task Group (SSTG), Data Base Administration Working Group (DBAWG), Working Group on Environment (WGE), and the Data Manipulation Task Group (DMTG), are briefly described in the following sections.

2.1 Subschema Task Group

The Subschema Task Group (SSTG) was created during the August 1973 meeting to further develop the subschema facility for data bases. Creation of the task group was prompted by consideration of several DDLC working papers. They indicated that subschema facilities that could be designed for a number of different host languages had many features in common. From this beginning came the current program of work for the SSTG. It is

1. To develop an approach to a common subschema framework for the most commonly used host languages (COBOL, FORTRAN, PL/1, ALGOL) and for the most commonly used data structures (networks, hierarchies, relations)
2. To develop a functional description of differences which should be allowed between schema and subschemas, and of the mappings that should be required to support these differences
3. To develop language specifications for the subschema framework and for schema-to-subschema mappings
4. Ultimately, to consider incorporating a subschema facility into those host language specifications whose developers deem a data base facility a necessity.

Ray Seth of American Can Company was appointed Chairman. The first SSTG meeting was held in April 1974. The SSTG has continued to meet regularly and has concentrated on the development work described in items 1. through 3. above specifically for network subschemas. The SSTG is now producing a report to the DDLC describing the results of this work.

It should be noted that the SSTG does not intend to replace the current COBOL subschema facility developed by the Programming Language Committee with its own subschema facility. The SSTG exists to develop the subschema facility in general. The language specifications to be produced by the SSTG are, as described above, for a schema-to-subschema mapping language (to the extent that this is appropriate), which does not currently exist, and for a subschema framework common to many host languages. These languages will almost certainly not be host-language dependent. Thus, they must be adapted for use with a particular host language by the developers of that language. It is

certainly not unreasonable to expect that if the SSTG developed enhanced subschema facilities, the Programming Language Committee might incorporate them in its COBOL subschema facility. However, this would be a decision for the PLC.

2.2 Data Base Administration Working Group

The Data Base Administration Working Group (DBAWG) was created to develop tools for the data base administrator to use in assuring efficient and reliable use of the data base. The group was established in recognition of the fact that there were data base administrative functions whose importance was acknowledged by the DDLIC (and earlier by the DBTG), but for which no language specifications were provided. As a result, the following program of work was adopted:

a. To examine the requirements for the objectives of the following data base administrative facilities:

- Control of data base system performance
- Mapping of data to storage
- Data base reorganization and restructure
- Backup and recovery
- Collection and analysis of usage statistics
- Control of data base procedures
- Other data base utilities

b. To develop a functional description of these facilities.

c. To develop language specifications where appropriate, on the basis of the functional descriptions.

The DDLIC at its December 1973 meeting decided to ask a working party of the British Computer Society (BCS) to become the nucleus of this group. The working party accepted this offer, but wished to remain within the BCS. Consequently, the group is termed a Working Group by the DDLIC. The Chairman of the DBAWG is Mr. J. S. Knowles of Aberdeen University. Within the BCS the DBAWG is a working party of the Advanced Programming Specialist Group whose chairman is Professor Peter King of Birkbeck College, London. Before undertaking the DBAWG work, the working party existed to study CODASYL's data base specifications and has submitted proposals to the PLC as well as to the DDLIC.

The DBAWG has continued to meet approximately once every two months in the United Kingdom. In June 1975 it published a report [3] containing the following six chapters:

1. Introduction
2. Concepts
3. Data Storage Control
4. Integrity Control
5. Statistics
6. Restructuring and Reorganization

At the June 1975 DDLC meeting in London, the DBAWG spent two days making a detailed presentation on the last four chapters. In subsequent DBAWG meetings a detailed plan of work was developed. It called for analysis of data base administration issues and drafting of working papers and proposals during the rest of 1975 and for submitting specific proposals to the DDLC in July 1976. The following items are scheduled to be presented to the DDLC by the DBAWG by July, 1976:

1. A working paper on data base management system (DBMS) architecture
2. Proposals to remove the dependence of other DDL elements on tuning and resource allocation elements, and a proposal to remove tuning and resource allocation elements from the DDL (some DBAWG proposals on this subject have already been acted on by the DDLC)
3. A working paper or proposal on a low-level Data Strategy Description Language (DSDL), which would incorporate tuning and resource allocation statements.

At its June 1975 meeting, the DDLC (in its review of the DBAWG report) asked the DBAWG to investigate a high-level DSDL in addition to the low-level DSDL. The high-level DSDL would allow specification by the data administrator of the tuning effects desired but would not dictate the methods used by the DBMS to achieve the effects. As a rough example of the difference between "high-level" and "low-level" statements, a high-level tuning statement for a key item might be "OPTIMIZE FOR DIRECT ACCESS," while a low-level statement for the same key might specify the type of hash-coding or indexing to be used in performing the optimization. Of course, other types of optimization statements (including even higher level statements) may be developed. The DBAWG has already done work on both types of tuning statements.

2.3 Working Group on Environment

The Working Group on Environment (WGE) is an informal subgroup of the DDLC, concerned with the architectural environment in which the DDL is meant to operate. The WGE was specifically formed to consider questions raised by the architecture proposed by the ANSI/X3/SPARC Study Group on Data Base Management Systems [4], whose work was first described to the DDLC by Mr. Charles Bachman in October 1973. The WGE was created shortly thereafter. Since then, the WGE has produced a number of working papers aimed at interpreting the intent of the current CODASYL schema/subschema architecture, the intent of various language components of the DDL within that architecture, and the architecture of the ANSI Study Group. To some extent, the DDLC's efforts in categorizing its language (described below) are the result of this activity. In addition, WGE members have followed the activities of the ANSI Study Group, informally exchanged working papers, and occasionally attended Study Group meetings. In October 1975 the WGE sent a letter to the Study Group asking for clarification of certain points of the Study Group's architecture and describing the WGE's views on assignment of various DDL constructs to various points in the ANSI architecture. Written response was received from Mr. Tom Steel, Chairman of the Study Group. More recently, Mr. Steel gave a presentation to the DDLC at their February 1976 meeting describing the status and future plans of the Study Group. The DDLC anticipates further liaison with the ANSI Study Group, as the Study Group evaluates both the COBOL data base facility

adopted by the PLC and the DDLC's own language specifications. With establishment of this closer liaison between DDLC and the ANSI Study Group, the WGE has suspended its activities.

2.4 Data Manipulation Task Group

The Data Manipulation Task Group (DMTG) was created at the October 1973 meeting to further develop data manipulation facilities. The suggested program of work for this task group, as defined in its charter, is as follows:

1. To develop a functional description of a DML appropriate to hierarchic data structures.

2. To develop a functional description of a DML appropriate to relational data structures.

3. To develop a functional description of host language independent enhancements to the data manipulation functions included in the April 1971 Data Base Task Group Report. Some suggested types of enhancement to the data manipulation functions are:

- Addition of more complex record selection expressions
- Addition of set level operations
- More control over DBMS update of currency indicators
- Selective OPEN statement
- Generalized statements
- More sophisticated locking mechanisms.

4. Ultimately, to consider the incorporation of the data manipulation facilities in appropriate host language specifications.

So far, the DMTG has not had a large enough membership to begin. Work on some of the subjects listed above is, however, going on outside CODASYL. In addition, the DDLC itself has taken up a number of these subjects directly. It was the intent of the DDLC that the DMTG concentrate on functional capabilities; any language specifications produced would be in the nature of a framework, which would be tailored to specific host languages or nonprocedural language interfaces. Thus, the role of the DMTG would be analogous to that of the SSTG in its specific subject area. In addition, both SSTG and DMTG were to provide feedback to the DDLC on the schema DDL and its ability to support advanced subschema and data manipulation facilities.

3. CHANGES IN LANGUAGE SPECIFICATIONS

Since the publication of the June 1973 DDL JOD, the DDLC has made a substantial number of changes to its language specifications. While not all of these changes are particularly significant (in that they involve no major changes in functionality), some of them are significant enough to be discussed in this report. These changes are the results of two general types of DDLC activity:

1. Analysis of the functionality and possible uses of the various DDL clauses, in order to categorize them.
2. Various suggested enhancements to specific parts of the DDL.

The more significant changes in the DDL are described briefly in the following sections.

3.1 DDL Categorization

The schema DDL defined by the DDLC is viewed primarily as a tool to be used in design, creation, and further development or maintenance of a data base. In its development of the DDL, the DDLC has found that categorization (or classification) of the statements of the language, according to a defined set of criteria, is a very useful technique. In this categorization, the language statements are examined for common properties which may be used to assign them to the appropriate category. Once these language statements with common properties have been placed in a category the common properties may be set aside in the interests of studying the differences among the statements.

The DDLC has categorized the DDL in two ways, one for presentation (grouping) in the JOD and one for analysis. In the first of these categorization schemes, clauses of the schema DDL are classified, listed, and presented in accordance with the data construct types that the clauses are used to describe. Thus, clauses exist to describe records, data items, sets, and the schema itself. This method of classifying clauses is useful because it groups together clauses that describe a single data construct, and it allows examination of their roles in describing that construct. This method was used in grouping the language specifications in the 1973 JOD, and it continues to be used in the present JOD.

The second of these categorization schemes, and the more important for analyzing and refining the DDL, is based on the basic function of the clause, as described in the language specifications. The basic function of a clause (or phrase of a clause) and its appropriate category assignment is based on the answers to the following questions:

1. What kind of function does the declaration of language elements in the category being defined provide?
2. Is inclusion of language elements from this category required in each schema? If not, what does the absence of a declaration mean?
3. *Must* specifications of language elements in this category depend on declarations in any other category; or *may* this be the case; or even, must this specifically not be the case?
4. What relation has the declaration of a language element from this category with a subschema. In particular, is redeclaration possible? If so, does it replace the schema declaration or add to it?

5. What impact will modification of the schema declaration of a language element from this category have on existing subschemas, existing application programs, and the contents and organization of the data base?

The DDLC considered that a meaningful categorization should

1. Help data administrators use the language when designing, implementing, and maintaining data base systems.
2. Facilitate the use of the language specifications by other groups concerned with language design, particularly the design of subschemas and data manipulation languages.
3. Aid the DDLC's understanding of the language and hence the rigor of its specifications.
4. Help implementors of data base software to understand the specifications of the DDL.

The categories distinguished by the DDLC were schema, structure, validation, DML interface, access control, measurement, tuning, resource allocation, and administration. Some of the major categories will be described briefly in the following paragraphs.

The *schema category* consists of language elements whose function is to declare characteristics of a schema considered as a collection of declarations. Thus, declarations in the schema category have no direct relation with the data in a data base, but only with its descriptor (the schema). For example, functions that belong to the schema category are the identification of a schema and the control over schema operations (such as displaying the schema).

The *structure category* consists of elements whose function is to declare the types of data constructs that will be referenced in other schema or subschema declarations and—depending on the actual subschema involved—in data manipulation statements. These data construct types may also be referenced via other languages interfacing with the schema, such as storage mapping languages. For example, the structure category contains those language constructs used to define data items, records, sets, and set orders. The structure declarations of a schema declare the overall data base structure on which other, application oriented structures may be mapped by a subschema mapping language. Modification of structure declarations in a schema has the following effect on existing subschemas and application programs:

1. If the change does not involve data referenced by a subschema, that subschema and the application programs working with it are not affected.
2. If the change does involve data referenced by a subschema, the mapping of the application-oriented structure on the schema structure must also change. If the latter change is possible, then the associated application programs are not affected. If the latter change is not possible, then the application-oriented structure itself must change; this is likely to invalidate the logic of the associated application programs.

Modification of structure declarations—whenever existing data constructs are involved—causes modification of the data in the data base. Also, modification of structure

declarations always requires modification in the organization of the data stored in the data base. Depending on the implementation, such a modification may be realized, for instance, by changing the mapping between the schema data constructs and the stored data constructs or by changing the stored data constructs themselves. Structure declarations take precedence over declarations in all the other categories except the schema category, in the sense that declarations in the other categories and the specifications for them refer to the structure declarations of the types of data constructs.

The *validation category* consists of elements whose function is to declare rules restricting the set of values that may be assigned to, or the relationships that may exist between, occurrences of the various types of data constructs that have been declared. For example, such declarations may require that values of a data item type lie within a certain range, or they may provide criteria for ensuring that a member record is associated with the correct owner record. Validation declarations are not required. If validation declarations are absent, changes to values or relationships in the data base will not be restricted except by rules inherent to the types of data constructs involved. Validation declarations cannot be overridden in a subschema; however, by means of a subschema language, additional restrictions may be declared. The effect of modifying validation declarations in a schema on existing application programs is that certain DML functions that had been executing successfully before modification may fail afterwards, or vice versa. In the former case, the application programs may have to be modified to deal with the possible failure of those DML functions. Modification of validation declarations may invalidate data in the data base if additional constraints on values or relationships are introduced.

The *access control category* consists of elements whose function is to declare rules that safeguard against unauthorized operations on occurrences of the various types of data constructs that have been declared. For example, a clause that declares an access control lock for the retrieval of certain data items belongs to the access control category. Access control declarations are not required. The safeguards declared in a schema cannot be bypassed by a subschema. However, by providing proof that it is authorized itself, a subschema may enable an application program to perform otherwise unauthorized data access operations. A subschema language may also provide facilities for declaration of additional safeguards. The effect of changing access control declarations in a schema upon existing subschemas and application programs is one of success or failure. Modification of access control declarations causes no modification of data in the data base.

The *tuning category* consists of elements whose function is to declare information for use by the DBMS to improve the performance of application programs operating on the data base or to decrease the cost of storing the data in the data base. Different DBMS's may react quite differently to such declarations. One might adjust the organization of the stored data in accordance with the declarations; another might not react at all because it allows no options or because the declarations are overridden by information gathered by the system itself. For example, tuning declarations may provide knowledge about how the data base will be used (e.g., the SEARCH KEY clause) or populated, or they may offer a choice between alternative techniques for representing the data construct occurrences (e.g., indexed versus nonindexed sets). Tuning declarations are optional; the absence of any declarations will cause the DBMS to assume suitable defaults. A subschema language may also provide tuning declarations; however, these would have effect only during the lifetime of run units using the subschema and only for the data referenced by

the subschema. Changing tuning declarations in a schema may affect the economic feasibility of existing subschemas and application programs. Note that the logic and results of application programs are not directly affected, only their efficiency. Modification of tuning declarations causes no modification of the data in the data base. However, as pointed out above, modification of tuning declarations may well cause reorganization of the data. Declarations in the structure and validation categories, and specifications for them, should not refer to any declarations in the tuning category.

The *resource allocation category* consists of elements that name and control the assignment of occurrences to organizational units to be used by the DBMS in allocating and managing its actual resources. For example, an area may be defined, with records assigned to it, as a convenient unit for mapping to a storage device of particular characteristics. Resource allocation declarations are optional; the absence of declarations will cause the DBMS to assume suitable defaults. Resource allocation declarations are not part of subschema languages. Changing resource allocation declarations in a schema may affect the economic feasibility of existing subschemas and application programs. Note that the logic and results of application programs are not directly affected, only their efficiency, unless the programs contain DML functions dependent on allocation units. Modification of resource allocation declarations causes no modification of the contents of the data base. Declarations in the structure or validation categories, and the specifications for them, must not refer to declarations in the resource allocation category.

It is fair to say that the categorization of its language by the DDLC has had a substantial effect on DDLC activities. The categories have led to more thorough and precise discussions of the purpose of proposed language constructs and to useful reevaluation of existing constructs. The increased insight into the nature of many of the language constructs provided by the categories has stimulated the following DDLC activities:

1. Elimination of undesirable dependence between constructs in different categories (e.g., between structural and tuning constructs).
2. Reorganization of language constructs to reflect their intended category (e.g., the separation of constructs that simultaneously apply to validation and tuning into separate constructs).
3. Proposals to move the "less logical" language constructs (e.g., those for tuning and resource allocation) from the DDL to other languages.
4. Clarification of the intended meaning of the language constructs.

It is expected that the end product of such activities will be a language that is more functional and easier to use and understand.

Categorization of language constructs is explicitly stated in the language specifications. Each category is defined in the concepts section of the JOD in a manner similar to that above. The definition includes expected effects on subschemas and programs when the declaration is changed and the intended interactions between constructs in different categories. Further, in the language specifications themselves, each category is briefly defined again, and a complete assignment of language constructs to categories is provided. This assignment is attached as Appendix B.

3.2 Specific DDL Changes

In addition to, and to some extent because of, its activity in categorizing the DDL, the DDLC has made a number of significant changes in the DDL. These are grouped into general categories and described below.

3.2.1 *Enhancements of Facilities for Declaration of Value-Based Structures*

Three DDL enhancements fall into this general category, the IDENTIFIER clause, the STRUCTURAL CONSTRAINT clause, and SELECTION BY STRUCTURAL CONSTRAINT, which is a new option of the SELECTION clause.

The IDENTIFIER clause in the record declaration provides for the declaration of user-specified, unique identifiers for a record type. A given unique identifier may consist of one data item, or of some combination of data items, in the record. The general format of the IDENTIFIER clause is shown below.

IDENTIFIER IS {data-identifier-1} ...

More than one IDENTIFIER clause may be specified for the same record type.

The STRUCTURAL CONSTRAINT clause in the set member declaration provides for the declaration that the set membership of a particular member record type in the set is to be constrained to a set occurrence that has owner data item values identical to the values of specified data items in the member record. The general format of the STRUCTURAL CONSTRAINT clause is shown below.

STRUCTURAL CONSTRAINT IS {data-identifier-1 EQUAL TO data-identifier-2} ...

where data-identifier-1 ... must be data items in the owner record type, and data-identifier-2 ... must be data items in the member record type.

More than one STRUCTURAL CONSTRAINT clause may be specified for the same member record type. Note that the combination of data items specified by data-identifier-1 ... need not necessarily be an IDENTIFIER of that owner record type, although they may be. The DBMS will prohibit the participation of a record of the member record type as a member of a set of the set type being constrained, unless the constraint is satisfied by the values of the identified data items in the owner and member records involved. This integrity check will be performed during any data manipulation function that either creates a new member of any set of the specified type or that changes the value of any of the data items specified in the constraint. Such constraints may be applied to sets with MANUAL as well as AUTOMATIC members.

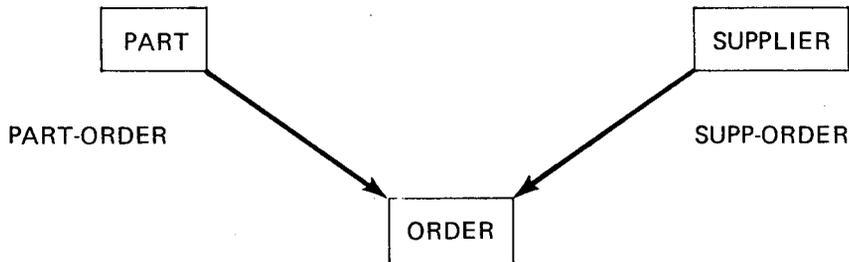
The SELECTION BY STRUCTURAL CONSTRAINT option of the SELECTION clause provides a simplified form of the SELECTION clause for use when a structural constraint exists and the conditions for set selection are the same as the structural constraint. This is often the case for value-based structures. In this case, the owner data

items specified in the STRUCTURAL CONSTRAINT clause must be declared as an IDENTIFIER of the owner record. The general format of this option is shown below (note that this is the third format of the SELECTION clause).

SET SELECTION IS BY STRUCTURAL CONSTRAINT

This option allows for automatic construction of set occurrences that satisfy the structural constraint.

With these new facilities, the structural declarations for the "standard" part-supplier-order example (excluding set orders) would be as shown in Fig. 1.



RECORD PART
 IDENTIFIER P#
 P#, NAME, COST

RECORD SUPPLIER
 IDENTIFIER S#
 S#, NAME, LOCATION

RECORD ORDER
 IDENTIFIER O#
 O#, S#, P#, DATE, QUANTITY

SET PART-ORDER
 OWNER PART
 MEMBER ORDER
 INSERTION AUTOMATIC RETENTION MANDATORY
 STRUCTURAL P# OF ORDER EQUAL P# OF PART
 SELECTION STRUCTURAL

SET SUPP-ORDER
 OWNER SUPPLIER
 MEMBER ORDER
 INSERTION AUTOMATIC RETENTION MANDATORY
 STRUCTURAL S# OF ORDER EQUAL S# OF SUPPLIER
 SELECTION STRUCTURAL

Fig. 1 — Structural declarations for the "standard" part-supplier-order example (excluding set orders)

3.2.2 *Removal of Dependence on Tuning Constructs*

Four basic changes in the DDL fall into this general category. These are (1) removal of the dependence of other clauses on the LOCATION MODE clause, (2) removal of the data base key construct and its associated constructs, (3) addition of a WITHIN ANY AREA option, and (4) removal of the TEMPORARY AREA facility.

Two specific changes fall under the general heading of removal of dependence on LOCATION MODE. The first is that the DUPLICATES ARE NOT ALLOWED phrase has been removed from the LOCATION MODE CALC option. This means that CALC no longer defines an identifier, but only a storage option. If it is intended that the CALC-key also be a unique identifier, it must be so defined using the IDENTIFIER clause. The second change is that the SELECTION clause (Format 1) has been altered to allow set selection based on any IDENTIFIER. Thus, at the entry level of the SELECTION clause, there are now the following options: SELECTION by SYSTEM, APPLICATION, IDENTIFIER, and using the SELECTION clause defined for another member record.

The data base key facility was removed, largely because of its mixed logical and physical connotations and because of the lack of any control mechanism over its use. With removal of data base keys, the DIRECT option of the LOCATION MODE clause was also removed, as well as the DATA-BASE-KEY option at the entry level of the SELECTION clause. As a possible substitution for the logical aspects of the data base key facility, the DDLC is considering optional facilities for the declaration of "system-generated identifiers" (i.e., facilities for specifying that one or more data items within a record are to have guaranteed unique values generated for them by the DBMS). Unlike data base keys, however, these items will have data-administrator-supplied names, will be part of the logical content of the records, and will (presumably) be accessible to application programs like any other data items. The physical aspects of the data base key facility, having to do with placement control within the data base, will probably be incorporated into other tuning or resource allocation statements, or into the DBAWG's proposed DSDL (see Sec. 2.2).

The new WITHIN ANY AREA option allows application programs more freedom from the area construct. When WITHIN ANY AREA is specified, the DBMS performs area assignment, and run-units need not supply the DBMS with information regarding areas. Such facilities reflect the DDLC's categorization of the WITHIN and AREA constructs as being concerned with resource allocation rather than logical structure.

The TEMPORARY AREA facility was removed because the intended facility could be better provided by alternative schema and subschema facilities.

3.2.3 *Miscellaneous Changes to Existing Facilities*

Under this general category are five specific changes: addition of a FIXED set membership option, addition of the POSTPONED RESULT data item, addition of a BEFORE/AFTER CALL procedure facility, deletion of the ENCODING/DECODING clause, and addition of the ability to use the record type as a general sort control key and to define the "collating sequence" of the record types.

FIXED set membership is a set membership option similar to the MANDATORY and OPTIONAL set membership options. If a member record type is specified as a FIXED member, then once an occurrence of this record type becomes a member of any set occurrence of the defined set type, it must remain a member of that particular set occurrence until it is deleted from the data base. This is unlike the MANDATORY option, in that a MANDATORY member must remain a member of *some* occurrence of the defined set type, but it may be switched from one such set occurrence to another. The FIXED member may not be switched. Figure 2 illustrates this difference for a personnel data base. There may be a rule that a PERSON must always be assigned to some DEPARTMENT. However, PERSON records may be reassigned to different occurrences of the DEPTPERS set to reflect personnel transfers. Thus, MANDATORY is the appropriate set membership option for the PERSON record type. However, JOB HISTORY records are unique to individuals, and should never be moved from one PERSON (and thus from one PERSHIST set) to another. Accordingly, FIXED is the appropriate set membership option for the JOB HISTORY record type. This provides the data administrator with a simple means of guarding against accidental or malicious misuse of the data base through set reassignment of JOB HISTORY records.

POSTPONED RESULT data is a type of derived data similar to the ACTUAL and VIRTUAL RESULT data provided in the June 1973 JOD and in the DBTG Report. The motivation for this new type of derived item is as follows. When the value of a RESULT data item is derived by means of a procedure that is at all lengthy, the data administrator will wish to make some attempt to minimize the number of times the procedure is invoked. If he declared the item as VIRTUAL, then the value would be recalculated every time the item is accessed, even if no change at all is made to the data from which it is derived. If, on the other hand, he made it an ACTUAL item, then it would be recalculated every time a change was made to any of the data from which it was derived, even if no access at all was made to the RESULT item itself. The POSTPONED facility provides for an intermediate category of derived item whose value is stored in the data base, but whose recalculation is postponed until the first access to it after a change is made to the data base which necessitates that recalculation.

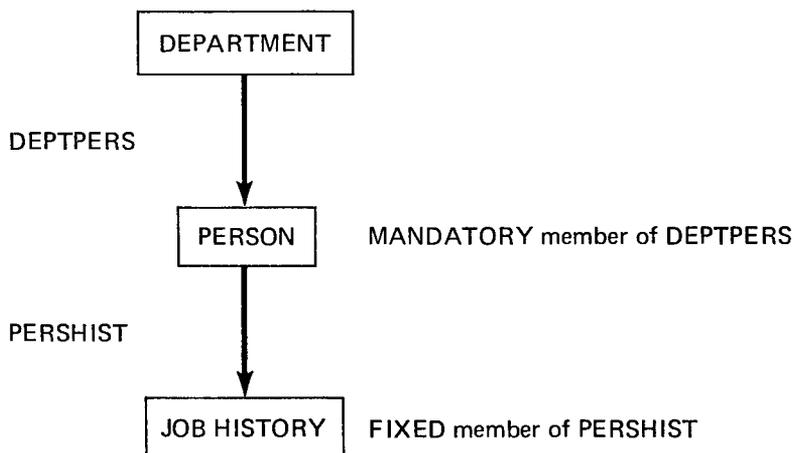


Fig. 2 — FIXED and MANDATORY set membership

The BEFORE/AFTER CALL procedure facility allows the data administrator to more precisely control when a CALL procedure (formerly an ON procedure) is invoked relative to the time it is triggered by the invocation of a specific data manipulation (DM) function. In the June 1973 JOD, the general format for the CALL data base procedure facility was given as

ON [ERROR DURING] [“stack” of DM functions] CALL data-base-procedure-1

The present general format is

CALL procedure-name-1 $\left\| \begin{array}{l} \text{BEFORE} \\ \text{ON ERROR DURING} \\ \text{AFTER} \end{array} \right\|$ [“stack” of DM functions]

with the words BEFORE and AFTER having their obvious meanings.

The ENCODING/DECODING clause was deleted from the DDL because, after analysis of its possible uses, the DDLC felt that it should not appear in the schema. The DDLC felt that, if the clause was to be used for conversion between schema and subschema formats, it should be in the subschema, because different subschemas may require different formats and thus perform different conversions. On the other hand, if the clause was to be used for encryption or data compression, there would have to be a rule requiring a corresponding DECODING clause for each ENCODING clause (which did not then exist). In addition, more efficient data compression was likely from procedures that processed complete records, or larger units of data, than from procedures that processed single data items. The DDLC also felt that the effect of ENCODING could be achieved by using CALL BEFORE STORE and of DECODING by using CALL AFTER GET.

The ability to use the record type as a general sort control key, and to define the “collating sequence” of the record types for this purpose, involves changes in both KEY and ORDER clauses of the DDL. In declaration of a sort control key using the KEY clause, each component of the key (there may be several) may be declared using the syntax:

$$\left\{ \begin{array}{l} \text{ASCENDING} \\ \text{DESCENDING} \end{array} \right\} \left\{ \begin{array}{l} \text{data-identifier-1} \\ \text{RECORD-TYPE} \end{array} \right\}$$

Within the ORDER clause, the ORDER may then be defined as

ORDER ... SORTED BY DEFINED KEYS RECORD-TYPE SEQUENCE IS
{record-name-1}...

The record-type sequence defines the “collating sequence” for the RECORD-TYPE entries in the sort control keys defined using the KEY clause.

3.2.4 *Name Change*

The DDLC has changed the name of clauses concerned with the control of access to specific constructs in the data base (and to the schema itself) from PRIVACY clauses to ACCESS-CONTROL clauses. This was done primarily in recognition of the fact that the term "privacy" has come to have a somewhat different meaning than that for which the clauses are often used, particularly with the passage of the Privacy Act of 1974 in the United States. "Privacy" is felt to be something that people have (or ought to have), rather than something that computers or data processing systems automatically provide. While clauses that control access to data in a data base (e.g. DDLC's ACCESS-CONTROL clauses) may be used to support the concept of privacy, they may also be used in other ways. Accordingly, the DDLC felt that it might be misleading to refer to these clauses as PRIVACY clauses.

3.2.5 *Other Changes Considered*

In addition to making the above-described changes in the DDL specifications, the DDLC has considered in its discussions a number of other issues that appear to be of continuing interest. These issues are often brought up in published papers and public comment concerning the DDL. Some of these issues continue to be on the DDLC technical agenda. Some of the more prominent of these issues, along with the action taken by the DDLC with respect to them, are described below.

Allow the same record type to be both owner and member in the same set type. This facility, often referred to as the "recursive set" or "unicycle," has been the subject of three proposals to the DDLC. In all three cases, the proposals were referred back to their authors for further work. The DDLC has, in general, been receptive to this idea. However, before the facility is added to the DDL, all problems related to DML operations on such sets, and the integration of such sets into the rest of the DDL, must be solved. The latest proposal was considered at the February 1976 meeting, and it was felt then that the addition of the facility should proceed to some extent in parallel with further development of the facility of cycles of set types, and that the facility should be added in such a way that "recursive" sets have all the facilities of present sets. A number of useful working papers have been presented, and it may be possible to incorporate this facility before the next JOD publication.

Eliminate the repeating group within records. The repeating group capability within records, provided by the OCCURS clause in the DDL, has occasionally been denounced as a "storage" concept, having no place in the DDL. At least one proposal for its removal has been received. When this proposal was considered, however, most members did not see the repeating group as a storage concept, but rather one that facilitated certain types of mappings to subschemas. For example, if all subschema records are COBOL records with repeating groups, the schema-to-subschema mapping of those records is obviously easier to define if the repeating groups can be specified in the schema. It is also true that a facility for controlling the number of member records in a set, corresponding to the facility for controlling the number of occurrences of a repeating group in a record, does not exist in the DDL, and thus removal of the repeating group would decrease the functionality of the DDL. Once logical set population control facilities are available in the DDL, it may be appropriate to again consider removing repeating groups.

Change the name of the CODASYL "set" construct. It has occasionally been argued that the term "set" used for the DDL's interrecord relationship construct may be confused with the mathematical term "set." Both are relevant in discussions of data base management, and it is therefore argued that some change in terminology is needed. One proposal suggested "coset" as an alternative. This was not approved for a number of reasons, one of which was that "coset" is also a mathematical term. However, the DDLC did not rule out a change in terminology if a suitable alternative could be found.

4. TECHNICAL OBJECTIVES

In the coming year, the DDLC will consider technical input from three major internal sources: the DDLC itself, the DBAWG, and the SSTG. The current DDLC agenda includes proposals or working papers on the major subjects discussed in the following sections.

4.1 Improved Definitions of Data Manipulation Functions

The June 1973 DDL JOD and the present specifications contain definitions of a set of "basic" data manipulation (DM) functions assumed to be possible on structures defined using the DDL. Thus, for example, such generic types of functions as FIND, INSERT, and STORE are defined, but without specific syntax and without the implication that they necessarily perform the same functions as implemented functions with the same names. These DM functions are referred to in the specifications in describing the effects of certain DDL declarations on operations on the data base (e.g., when derived data values are created, when certain validity checking is performed, etc). Thus, these functions must be properly defined to allow proper definition of the DDL. In addition, the functions provide a conceptual interface to the DBMS, which may be used in implementation. For example, implementors of a DBMS based on the schema DDL could choose a different set of basic DM functions according to individual requirements. They could then map their own sets of DM functions onto that defined in the DDL specifications, and vice-versa, to properly relate the specifications and the actual DM functions implemented. Similarly, complex DML commands provided in host or query languages by an implementor could be defined by resolving them into a sequence (or, possibly, multiple sequences that can be executed in parallel) of basic DM functions.

The DDLC has found that the present set of DM function definitions is not precise enough for its growing requirements. As a result, the DDLC is engaged in an effort to more precisely define them. In addition, the present set of DM functions is defined such that the execution of any single DM function must transform the data base from a state consistent with that data base's schema into another such consistent state. However, the DDLC is also considering the definition of what are termed "primitive" DM functions. A primitive DM function is one that processes the same types of data constructs as the basic DM functions now defined, but that may, when executed, transform the data base from a state consistent with its schema to one that is not, or vice-versa.

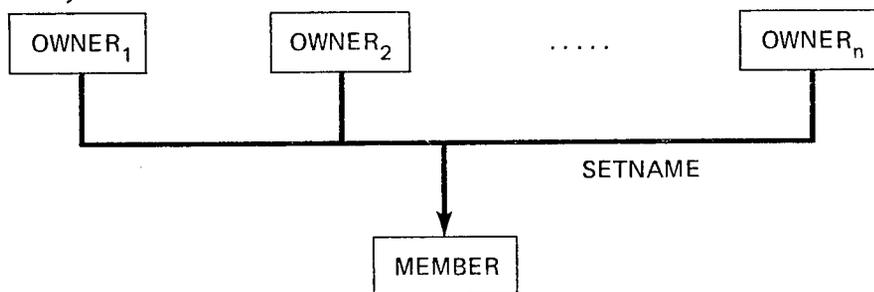
As an example, a "basic STORE" function applied to a record that is declared an AUTOMATIC member of a set will insert that record in the appropriate set in addition to storing the record in the data base. A corresponding primitive function might be a

“create record” function which stores a record in the data base without checking that the data values contained comply with the provisions of the DDL’s CHECK clause, and without inserting the record into any set in which it is declared to be an AUTOMATIC member.

Present DDL specifications do not contain definitions of any primitive DM functions. However, the DDLC recognizes that it may be necessary to specify such primitive functions for future work, including the more detailed specification of the data base procedure facility, the incorporation of the DBAWG’s work on lower-level facilities, and the incorporation of more complex structural validity-checking facilities. Included in these primitive functions would have to be functions that inform the DBMS when to suspend and reinitiate validity checking on structures created using other primitive functions.

4.2 Extensions to the Set Construct

As noted in Sec. 3.2.5, the DDLC has considered a number of proposals on the most frequently discussed extension, the recursive set facility. Another extension, which has been discussed in several working papers, is to allow alternate owner record types for the same set type. For such a set type, the owner record of a particular set could be of several different types, but the restriction to only one owner record occurrence per set occurrence would still be enforced. The data structure diagram for such a set type might have the form shown in Fig. 3. Other extensions have been mentioned, such as removing the restriction that a record occurrence can be a member of only one set occurrence of a given type at a time, or that a set may have only one owner record occurrence. However, no formal input on these subjects has been received by the DDLC.



where $OWNER_1, \dots, OWNER_n$ are alternate owners of the set type “SETNAME.”

Fig. 3 — A set type with alternate owners

4.3 Additional Tuning Statements

These include such statements as search keys at the record level for optimizing access to individual records irrespective of set relationships (i.e., so-called “out-of-the-blue” access) and declarations of probable populations of sets, areas, etc. Major questions about such statements are whether they should be part of the schema DDL or be included in another language (discussed below) and whether they should be “low level” or “high level” (see Sec. 2.2).

4.4 Freer Syntax for Writing Schemas

The present schema DDL syntax is rather rigid in that it insists, for example, that all declarations pertaining to a particular record type be grouped together with the record name specification and that set declarations follow record declarations. A number of proposals to allow a freer grouping of DDL constructs have been discussed by the DDLC, and some are on the current agenda. Such freer grouping would, for example, allow all ACCESS-CONTROL clauses to be grouped together, all tuning and resource allocation statements to be grouped together, etc. This should assist the data administrator by allowing him to write the schema in whatever order he wishes.

4.5 Additional Consistency Declaration Facilities

An example of this type of declaration would be a facility for declaring the logical population of a set (e.g., that a particular set type must have exactly one member of each of the declared member record types). Some proposals along these lines have been discussed. The DDLC has found, however, that definition of more primitive DM functions would assist in the definition of such facilities.

4.6 Eliminating Undesirable Dependence on Tuning or Resource Allocation Elements

Steps in this direction have been taken, as noted in Sec. 3.2.2. This will continue, particularly in connection with the activity described below, concerning removal of tuning and resource allocation elements from the DDL.

4.7 Eliminating Tuning and Resource Allocation Statements from the DDL

The DDLC has received proposals and working papers on removal to the DBAWG's DSDL (see Sec. 2.2) of PRIOR PROCESSABLE and LINKED TO OWNER declarations, SEARCH KEY and INDEX declarations (including the new proposal for record-level search keys), and the LOCATION MODE clause. Needless to say, the elimination of dependence on tuning and resource allocation statements is underway, and is a necessary preliminary to the total removal of such statements from the DDL. However, some members of the DDLC question the distinction between such statements being a "separate language" and their being a different category in the same language (as described in Sec. 3.1), assuming that in either case undesirable dependencies are removed and the meaning of their being in a different category is well defined. During the DDLC's consideration of the above proposals and working papers on specific statements, many members felt that they would prefer to see what the "separate language" (DSDL) would look like before removing these facilities from the DDL. Accordingly, it is likely that some more definite decision will be made at the July 1976 DDLC meeting, at which time the DBAWG will present their ideas for the DSDL.

4.8 Other Objectives

As noted, both the DBAWG and SSTG are expected to contribute material to the DDLC agenda. An additional element of the technical input will be proposals (as required) to resolve any differences determined to exist between DDL specifications and those of the COBOL data base facility adopted by the CODASYL Programming Language Committee (PLC). Other general topics of interest to the DDLC (some of which are explicitly reflected in the DDLC's agenda or plans for the future) are

1. Activation of the DMTG to investigate functional capabilities of enhanced data manipulation functions (e.g., support of Boolean operations and of nonprocedural languages)
2. Improvement of the access control mechanism
3. Improvement of the data base procedure control mechanism
4. Consideration of improvements suggested by external (to DDLC) sources (e.g., through the liaison activities discussed below).

5. LIAISON ACTIVITIES

The DDLC maintains liaison with a number of groups that contribute valuable input to the DDLC's technical activities. This liaison activity is briefly described in the following sections.

5.1 British Computer Society (BCS)

As described in Sec. 2.2, the BCS has maintained a continuing interest in CODASYL's work on data bases for some time, one effect of which was the establishment of the DBAWG, a joint group of BCS and DDLC. In addition, the DDLC also plans to establish some liaison with the BCS Data Dictionary Systems Working Party.

5.2 European Computer Manufacturers Association (ECMA)

ECMA has submitted numerous proposals to the DDLC in the past, and the DDLC continues to maintain a useful liaison with ECMA. Recently, ECMA formed a new Technical Committee, TC-22 on Data Bases, and the DDLC has established formal liaison with that group.

5.3 Information Processing Society of Japan (IPSJ)

The DDLC established formal liaison with the IPSJ's Data Base Language Working Group last year after a period of informal communications. This working group has already submitted a request for clarification of a particular DDL-related issue which was acted upon by the DDLC; the request proved useful by illustrating the existence of a problem.

5.4 ANSI/X3/SPARC Study Group on Data Base Management Systems

As described in Sec. 2.3, the DDLC has maintained informal liaison with the Study Group's activities via the Working Group on Environment. The DDLC anticipates that the Study Group's comments on the schema DDL will prove to be valuable input to the DDLC's consideration of various changes to the DDL, as well as on the placement of various language components in a DBMS architecture.

5.5 International Federation for Information Processing (IFIP)

IFIP is really not a "liaison activity" of the DDLC in the same sense as the preceding groups. However, it is listed here on the strength of recent conferences on data-base-related subjects [5,6], which contained papers of direct relevance to the DDL. The latter conference had as its specific aim "an in-depth technical evaluation of CODASYL DDL." In addition to the normal objectives of a conference, one of the objectives of this specific conference was development of a set of specific recommendations to the DDLC concerning possible modifications of its language specifications for the DDL. Such a list of recommendations was produced, and more details of this process may be found in the conference proceedings [6]. Volunteers were solicited at the conference to produce specific proposals or working papers to the DDLC for each item on the list. The list of recommendations prepared by the IFIP conference is presented in Appendix C, with a brief description of DDLC activity with respect to each item. In some cases, DDLC consideration of the subject was a direct result of input from the volunteer at the IFIP conference. In other cases, DDLC consideration of the item was the result of input from a DDLC member, who may or may not have been affected by the IFIP conference results (many of the ideas considered there have also been suggested elsewhere).

The DDLC engages in this liaison activity because it is interested in getting outside comments on its DDL specifications, in disseminating the products of its activities, and in finding out about related activities elsewhere. To a great extent, the ability of DDLC to actively solicit outside comments and to disseminate the results of its interim work is limited by the voluntary nature of the CODASYL organization. However, each CODASYL-approved language specification contains an invitation to submit to CODASYL comments on those specifications. The DDLC takes that invitation seriously and is prepared to take any reasonable action to cooperate with persons or organizations seriously interested in improving the DDL [7].

6. PUBLICATIONS

At the September 1976 meeting, the DDLC will again consider whether to publish a revised JOD. It is hoped that enough progress will have been made in the technical work on the agenda to justify a new publication. While it is not expected that a publication in the near future would include any significant input from SSTG, it would include input (perhaps a substantial amount) from DBAWG. In addition to a revised JOD, it may be that the SSTG report to DDLC would be suitable for publication. If so, however, this would be published "for information only," as was the initial DBAWG report [3] to the DDLC.

ACKNOWLEDGMENTS

I must thank the entire DDLC for their "involuntary" help in preparation of this report, not only because the accomplishments I report here are those of the entire DDLC, but also because much of the wording and examples come from sections of the draft JOD and from DDLC proposals and working papers on associated material. I also thank Mr. Stanley Wilson, of the Naval Research Laboratory, and Mr. Roger Holliday, of International Business Machines Corporation, for their comments.

REFERENCES

1. *CODASYL Data Base Task Group, April 1971 Report*, available through the Association for Computing Machinery, 1133 Avenue of the Americas, New York, N.Y. 10036.
2. *CODASYL Data Description Language Journal of Development, June 1973*, National Bureau of Standards Handbook 113, U.S. Government Printing Office (SD Catalog No. C13.6/2:113), Washington, D.C., Jan. 1974.
3. *Data Base Administration Working Group, June 1975 Report*, available from the British Computer Society, 29 Portland Place, London W1N 4AP.
4. ANSI/X3/SPARC Study Group on Data Base Management Systems, "Interim Report," Feb. 8, 1975, in *FDT* (bulletin of ACM-SIGMOD) 7(2), available through Association for Computing Machinery, 1133 Avenue of the Americas, New York, N.Y. 10036.
5. J. W. Klimbie and K. L. Koffeman, eds., *Data Base Management*, North Holland/American Elsevier, New York, 1974 (*Proceedings of the IFIP Working Conference on Data Base Management, Cargese, Corsica, France, Apr. 1-5 1974*).
6. B. C. M. Douque and G. M. Nijssen, eds., *Data Base Description*, North Holland/American Elsevier, New York, 1975 (*Proceedings of the IFIP TC-2 Special Working Conference, Wepion, Belgium, Jan. 13-17, 1975, "An In-Depth Technical Evaluation of CODASYL DDL"*).
7. Address CODASYL DDLC correspondence to

Emile B. Broadwin
Chairman, DDLC
Computer Sciences Corporation or
650 N. Sepulveda Blvd.
E1 Segundo, CA 90245

Frank A. Manola
Secretary, DDLC
Code 5403
Naval Research Laboratory
Washington, DC 20375

Appendix A
ORGANIZATIONS THAT ARE OR HAVE BEEN MEMBERS OF THE DDLC

Aberdeen University, Scotland
Bell Telephone Laboratories
B. F. Goodrich Company
Blue Cross/Blue Shield
Boeing Computer Services, Inc.
Burroughs Corporation
Cincom Systems, Inc.
Computer Sciences Corporation
Consolidated Analysis Centers, Inc.
Control Data Corporation
Defense Communications Agency
Department of the Navy
Digital Equipment Corporation
Fireman's Fund American Insurance Company
General Electric Company
General Motors Corporation
Honeywell Information Systems, Inc.
International Business Machines Corporation
International Computers Limited, England
Manufacturer's Hanover Trust Company
The MITRE Corporation
National Bureau of Standards
National Security Agency
NCR Corporation
The Ohio State University
Philips-Electrologica B.V., The Netherlands
RCA Corporation
Scientific Control Systems Ltd., England
Southern Railway System
Sperry Univac Corporation
U. S. Air Force
U. S. Army
University of Florida
University of Michigan
Xerox Corporation

Appendix B DDL CATEGORIES

Nine functional categories are appropriate to the schema DDL. When viewed as functional entities, each syntactic element of the schema falls into one of the categories. The categories, their definitions, and the schema elements that compose the categories are listed below. When a whole clause is in a single category, it is designated by the name of the clause followed by the word "clause," and syntactic elements subordinate to the clause are not listed. When the entire clause is not in a single category, then the clause name is listed followed by the subordinate elements that fall in the same category. Thus, clause names appear more than once when they have syntax and semantics for more than one functional category.

Schema

The schema category identifies a schema and declares its characteristics. The syntactic elements of this category include:

- ACCESS-CONTROL clause (schema)
- CALL clause (schema)
- SCHEMA NAME clause

Structure

The structure category names the data structures described by the schema. The syntactic elements of this category include:

- Data-name clause
- DYNAMIC clause; DYNAMIC
- KEY clause; ASCENDING, DESCENDING, RECORD-TYPE, DUPLICATES FIRST, LAST, SYSTEM-DEFAULT
- MEMBER clause
- OCCURS clause
- ORDER clause; FIRST, LAST, NEXT, PRIOR, SYSTEM-DEFAULT, SORTED, WITHIN RECORD-TYPE, DUPLICATES FIRST, LAST, SYSTEM-DEFAULT
- OWNER clause
- RECORD NAME clause
- SET NAME clause

Validation

The validation category declares rules that constrain the occurrences of the data structures declared in the structure category. The syntactic elements of this category include:

- CHECK clause
- DUPLICATES clause
- IDENTIFIER clause
- INSERTION clause
- KEY clause; DUPLICATES NOT, NULL IS NOT ALLOWED
- ORDER clause; DUPLICATES NOT
- PICTURE clause
- RESULT clause; RECORD, MEMBERS, ON, OF, USING
- SEARCH clause; DUPLICATES NOT
- SOURCE clause; OWNER
- STRUCTURAL clause
- TYPE clause

DML Interface

The DML interface category declares procedures that may be invoked by a DML function and parameters to be supplied to these procedures. The syntactic elements of this category include:

- KEY clause; RANGE
- SELECTION clause
- WITHIN clause; AREA-ID

Access Control

The access control category declares authorization mechanisms for access to and change to the occurrences of the data structures declared in the structure category. The syntactic elements of this category include:

- ACCESS-CONTROL clause (except schema)

Measurement

The declarations of the measurement category direct the DBMS in collecting data about data base use, population, etc. There are at present no syntactic elements in this category.

Tuning

The tuning category declares guidelines for data base organization to assist in tuning data base performance. The syntactic elements of this category include:

DYNAMIC clause; PRIOR PROCESSABLE
LINKED clause
LOCATION clause; CALC, USING, VIA, SYSTEM-DEFAULT
ORDER clause; PERMANENT, TEMPORARY, INDEX NAME
RESULT clause; ACTUAL, POSTPONED, VIRTUAL
SEARCH clause; USING, CALC, INDEX NAME, PROCEDURE
SOURCE clause; ACTUAL, VIRTUAL

Resource Allocation

The resource allocation category names organizational units appropriate for managing system resources and controls the assignment of occurrences of the declared data structures to these units. The syntactic elements of this category include:

AREA NAME clause
WITHIN clause

Administration

The administration category names and provides for the invocation of DBA supplied procedures. The syntactic elements of this category include:

CALL clause (except schema)

Appendix C
IFIP CONFERENCE RECOMMENDATIONS AND DDLC ACTION

- a. *Allow the same record type to be both owner and member in a given set type.*—Proposals on this subject, including one from the IFIP volunteer, have been considered by the DDLC, as discussed in Sec. 3.2.5.
- b. *Eliminate repeating groups.*—A proposal on this subject from the IFIP volunteer was discussed and voted down, as discussed in section 3.2.5.
- c. *Allow specification of multiple identifiers in the record declaration.*—Input from the IFIP volunteer and from within the DDLC was received on this subject; this facility is now in the DDL (the IDENTIFIER clause).
- d. *Introduce into the record declaration a SEARCH KEY clause for optimization of access to record occurrences with designated item values.*—Input was received from the IFIP volunteer, and a modified version of this proposal is currently being debated, as discussed in section 4.3.
- e. *Provide an option in the DML for a record occurrence based only on designated item values.*—Suggested DML syntax for this facility was contained in the proposal received for item d above; the DDLC in general supports this idea. However, DML syntax is not within the jurisdiction of the DDLC, but rather the PLC.
- f. *Revise the SET SELECTION clause to accommodate the possibility that set selection might take place on identifiers other than that which is the CALC-key.*—This facility is now in the DDL.
- g. *Restrict the set to have only one member record type.*—IFIP input was received on this subject; however, the proposal was voted down.
- h. *Consolidate the SEARCH KEY and SORTED INDEX clauses within the set declaration.*—No input has been received on this; however, it is likely that some consolidation will take place in connection with DBAWG's proposal for a DSDL.
- i. *Improve the record selection power in the SET SELECTION clause, including existential quantifiers.*—No input has been received.
- j. *Reexamine the facilities for item type declaration.*—The DDLC has considered this an important area for some time; however, little work has been done on this subject.
- k. *Allow cardinality constraints on set occurrences within the set declaration.*—The DDLC has considered some proposals on this subject, as discussed in Sec. 4.5.

FRANK A. MANOLA

- l. *Consider extensions of the set attributes including VIRTUAL, PHANTOM, and an extended DYNAMIC set.*—No IFIP input has been received on this subject; however, the DDLC has considered extensions to DYNAMIC sets, and has just completed discussion of a preliminary proposal on VIRTUAL sets.
- m. *Base conceptual schema on units of information (e.g. binary relations).*—No input has been received.
- n. *Develop better integrity checks.*—This is under development.
- o. *Move various clauses to a 'storage structure language.'*—This is currently under study, in connection with the DBAWG's proposed DSDL.
- p. *Rework SOURCE/RESULT clause to specify the execution sequence.*—The POSTPONED RESULT facility may be considered work on this subject; preliminary input on more precise specification of SOURCE/RESULT timing has been received.
- q. *Extend SOURCE/RESULT clause.*—Input from the IFIP volunteer was received on this subject, and some extensions (e.g. POSTPONED RESULT) were adopted as a result.
- r. *Allow multilevel subschemas.*—This facility was one of the first considered by the WGE in its study of the CODASYL architecture and is being developed by the SSTG.
- s. *Make higher level operations available in the DML.*—This is within the province of the DMTG in terms of the development of functional descriptions of such operations; however, the production of syntax specifically for the COBOL DML is within the province of the PLC.
- t. *Provide for alternate owner of a set.*—This subject is under study, as discussed in Sec. 4.2.