NBS HANDBOOK 113

A UNITED STATES
DEPARTMENT OF
COMMERCE
PUBLICATION

CODASYL
Data Description Language

Journal of Development
June 1973

U.S.
DEPARTMENT
OF
COMMERCE

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Nuclear Sciences[2] — Applied Radiation[2] — Quantum Electronics[3] — Electromagnetics[3] — Time and Frequency[3] — Laboratory Astrophysics[3] — Cryogenics[3].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute consists of a Center for Building Technology and the following divisions and offices:

Engineering and Product Standards — Weights and Measures — Invention and Innovation — Product Evaluation Technology — Electronic Technology — Technical Analysis — Measurement Engineering — Structures, Materials, and Life Safety[4] — Building Environment[4] — Technical Evaluation and Application[4] — Fire Technology.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consists of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Relations.

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.

[2] Part of the Center for Radiation Research.

[3] Located at Boulder, Colorado 80302.

[4] Part of the Center for Building Technology.

CODASYL Data Description Language

Journal of Development, June 1973

+Handbook no 113

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

Revision by Canadian Govt. entered [redacted] ICST for info E.C. 6/28/79
collection

U.S. DEPARTMENT OF COMMERCE, Frederick B. Dent, *Secretary*
U.S. NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, *Director*

Issued January 1974

Library of Congress Catalog Number: 73-600347

National Bureau of Standards Handbook 113

Nat. Bur. Stand. (U.S.), Handb. 113, 155 pages (Jan. 1974)

CODEN: NBSHA

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1974

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
(Order by SD Catalog No. C13.6/2:113). Price $1.70

FOREWORD

Under Public Law 89-306 (Brooks Bill) the Secretary of Commerce was given important responsibilities for improving the procurement, utilization, and management of computers and related information systems in the Federal Government. To carry out the Secretary's responsibilities under the Brooks Bill, the NBS Institute for Computer Sciences and Technology provides leadership and coordination for Government efforts in the development of voluntary commercial information processing standards.

A major problem in the use of electronic data processing equipment lies in the inability to state the data processing application in such a way that computer programs and data are developed and maintained with a minimum of time and programming effort. A common Data Description Language (DDL), independent of any make or model of computer, would do much to solve this problem. NBS believes that a common DDL could have a significant impact on the future development of functionally compatible data base management systems and should increase the portability of programs and data among different computer systems.

Since 1969, the Conference of Data Systems Languages (CODASYL) has been active in the development of a common DDL. The current activity within CODASYL on the development of a DDL is being conducted by the Data Description Language Committee (DDLC) composed of voluntary representatives from computer manufacturers and users in industry and the Federal Government.

The present publication represents a report to the DDL community from the CODASYL DDLC on the development of a common DDL through June 1973. The National Bureau of Standards is pleased to have the opportunity to make this information available through publication as an NBS Handbook.

R. M. Davis, Director
Institute for Computer
Sciences and Technology

ABSTRACT

This Journal of Development reports the work of the CODASYL Data Description Language Committee. The Committee was assigned the tasks of establishing "ways to aid the functions of data administration and systems administration". The Committee's charter included, "the provision of specifications for the declarations required to establish and maintain data base structures". As a step towards this purpose, the Journal contains three sections which treat the Background and History of the Data Description Language Committee, Major Concepts, and the specifications of the Data Description Language. The Committee based its work, in part, on the 1971 report of the Data Base Task Group Report.

The approved Data Description Language specifications contain the syntax and semantic rules that permit the description of the structure and contents of a data base in a language independent of, but common to, many other high level programming languages. The language specifications will have a significant impact on the development of functionally compatible data base management systems and will increase the portability of programs between different computer systems.

Though not part of the approved language specifications, the presentation of the major concepts will help in the understanding of the specifications. Similarly, the background and history information will help explain the evolutionary growth of the Data Description Language.

Key words: COBOL; CODASYL; Data Base Administration; Data Base Management; Data Base Task Group; Data Description Language.

TABLE OF CONTENTS

Note: Each of the above sections is prefaced by its respective table of contents.

TABLE OF CONTENTS

SECTION 1. PREFACE, BACKGROUND AND HISTORY

1.1 PREFACE

It is the pleasure of the Conference on Data Systems Languages (CODASYL) to present this, the first Journal of Development of its Data Description Language Committee. It is important to note that these language specifications are the result of a truly international effort on the parts of many dedicated people and of their respective companies.

We must also point out that this report constitutes but a first step in the development of a common data description language, independent of, but common to, many other high-level programming languages. The CODASYL organization and its Data Description Language Committee is fully committed to sustained improvement of these specifications through maintenance and extension as user and implementor alike learn more about the field of integrated data structures and their impact upon the information processing industry. Because of this commitment, we invite your comments and participation in this endeavor, while we pledge our support and cooperation to you in defining viable interfaces with other languages.

As you apply these specifications, remember that only Chapter III of the Journal contains "language specifications" and that any constraints implied or stated in Chapter II (Concepts) are not to be interpreted as "CODASYL approved" unless also stated or implied in Chapter III.

Release of this Journal of Development as a Type A release requires inclusion of the following disclaimer.

> *The reader is hereby notified that the following language specification has been approved by the Data Description Language Committee but may be a partial specification which relies on information appearing in many parts of the total specification. These specifications are dynamic in nature, and the changes reflected by this approved change may not correspond with the latest specification available.*
>
> *Because of the evolutionary nature of these specifications, the reader is further reminded that changes are likely to occur in the specifications released herein prior to a complete republication of the Data Description Language Journal.*

Anyone reproducing this release is requested to reproduce this preface and to include on each subsequent page a reference to the preface.

Please address any comment, proposal, or working paper on the subject to:

Chairman, D.D.L.C.
CODASYL
Box 124
Monroeville, Pa. 15146

J. L. Jones, Chairman
Executive Committee
Conference on Data Systems Languages

THE BACKGROUND TO AND HISTORY OF THE CODASYL DATA DESCRIPTION LANGUAGE COMMITTEE

On April 8, 1959, a small group of computer users and manufacturers, meeting at the University of Pennsylvania to review recent language developments in the area of business applications, concluded that the development of specifications for a machine independent common language for business problems might be feasible. The group asked the Department of Defense to host a meeting at which both the feasibility and desirability of a common business oriented language could be considered.

On May 28 and 29, 1959, over 40 representatives from users in private industry and in government, computer manufacturers, and other interested parties, met at the Pentagon. The group agreed that a major problem in the efficient utilization of electronic data processing equipment was the inability to state the data processing application in such a way that computer programs could be developed and maintained with a minimum of time and programming effort. With the goal of solving this problem in mind, the group agreed that the development of specifications for a common business oriented language was both desirable and feasible. At this meeting the concept of the Conference on Data Systems Languages (CODASYL) was agreed upon. CODASYL was established as (and is currently) an informal and voluntary organization of interested individuals, supported by their institutions, who contribute their efforts and expenses towards the ends of designing and developing techniques and languages to assist in data systems analysis, design, and implementation.

Three CODASYL committees were agreed upon at the May, 1959, meeting, and an executive committee was established to coordinate the work of the three. The formation of three committees was the result of differing points of view regarding the objectives of CODASYL. The group of representatives which became the Short Range Committee felt that the immediate need for a common language necessitated working within the state of the art as it then existed to develop the specifications for the language. A second view, which formed the basis for an Intermediate Range Committee and a Long Range Committee, was that a better understanding of the problems of data processing programming was necessary before a common language could be proposed.

The Short Range Committee, although initially assigned the task of studying existing business compilers and the experience of users of these compilers, developed the initial specifications for a Common Business Oriented Language (COBOL). In 1960, a COBOL Maintenance Committee was organized to make additions, clarifications, and changes to the language specifications, and to guide users and implementors of the language. The Maintenance Committee consisted of users' and manufacturers' groups which met both separately and jointly. In 1961 portions of the Intermediate and Long Range committees were combined by the Executive Committee to form the Development Committee comprised of a Systems Group and a Language Structures Group. These groups worked on the development of both languages and data processing techniques. "Decision Table Structured Language" produced by the Systems Group, and "Information Algebra", produced by the Language Structures Group, are examples of the type of work done by the Development Committee.

In January, 1964, the COBOL Maintenance Committee was reorganized to broaden its scope of activities. The separate user and manufacturer groups were combined into the COBOL Committee of three subcommittees: Language, Evaluation, and Publication. The work of the former COBOL Maintenance Committee was assumed by the Language Subcommittee. Additionally, the COBOL Language Subcommittee maintained liaison with the United States of America Standards Institute (USASI) and the International Organization for Standardization (ISO) in their development of COBOL Standards. The Publication Subcommittee was responsible for the production of official COBOL publications and liaison with USASI regarding the COBOL Information Bulletin (CIB). The Evaluation Subcommittee analyzed and evaluated implementations and user activities. In April, 1965, the Development Committee was reorganized as the Systems Committee and the Language Structures Committee. These two committees continued the work of developing languages and techniques.

In July, 1968, the Executive Committee reorganized the entire structure of CODASYL committees. The reorganization produced three standing committees: the Programming Language Committee (PLC), the Planning Committee, and the Systems Committee. These three committees exist currently. The purpose and objective of the Systems Committee is to build an expertise in, and to develop advanced languages and techniques for, data processing, with the aim of automating the processes of system analysis, design and implementation. The purpose and objective of the Planning Committee is to aid in CODASYL planning by gathering, assimilating and disseminating information from implementors and users pertaining to the goal of CODASYL. The purpose and objective of the Programming Language Committee is to develop programming language specifications which facilitate compatible and uniform source programs and object results, with continued reduction in the number of changes necessary for conversion or interchange of source programs and data. The PLC concentrates its efforts in the area of tools, techniques and ideas aimed at the programmer. The purpose and objectives of the COBOL Language

Subcommittee were assumed and extended by the PLC, which is responsible for the COBOL Journal of Development. The Executive Committee coordinates the activities of all standing committees and directs them in accomplishing the purpose of CODASYL.

At the June, 1965, meeting of the COBOL Language Subcommittee, W. G. Simmons of the United States Steel Corporation suggested that "list processing" be added to the agenda as a topic for future developmental work, and volunteered to organize a task force for the work. The first meeting of the List Processing Task Force was held in October, 1965; at this meeting Mr. Simmons assumed the responsibility for the preparation of a working paper on the topic. In November, 1965, Mr. Simmons presented to the subcommittee the proposal "List Processing Extension to Mass Storage" (USS-011165.00). The proposal detailed justification for the use of list processing techniques in file management, and suggested the changes and additions to the COBOL language specifications to accommodate processing techniques for defined chain relationships.

In March, 1966, the second meeting of the List Processing Task Force was held, and in May, 1966, Mr. Simmons was appointed chairman of the task force. The List Processing Task Force met regularly to solicit the opinions of interested parties, to examine many data base and file systems, and to produce working papers at all levels of detail from functional requirements to draft language specifications. In May, 1967, the List Processing Task Force voted to change its name to the Data Base Task Group (DBTG). Because the membership of the group changed constantly, a major amount of the DBTG's efforts was directed toward listening to and studying the views of as many persons as possible. Although the latter direction of effort affected the progress rate of the group, the direction was deemed a necessary aspect of the group's undertaking. The Integrated Data Store work by C. W. Bachman of General Electric Company and the Associative Programming Language work by G. G. Dodd of General Motors Research Laboratories were inspirational to much of the early work done by the DBTG.

In an effort to solicit endorsement of the DBTG objectives, and to solicit recommendations and/or guidance for future work, the DBTG presented to the COBOL Language Subcommittee, in January, 1968, an interim report entitled "COBOL Extensions to Handle Data Bases". At this joint meeting the DBTG presented a program which included an introduction to data base by Mr. Simmons, a history of Integrated Data Store by Mr. Bachman, and an introduction to data structures by Mr. Dodd. Discussion following the program revealed a consensus that the structure of a data base should be included in COBOL. The COBOL Language Subcommittee was directed to review the DBTG interim report in preparation for a joint meeting February 28, 1968.

During the February meeting the COBOL Langauge Subcommittee approved, for public distribution, the interim report by the DBTG. The report was subsequently published as a joint newsletter by the Special Interest Groups for Business Data Processing and for Programming Languages of the Association for Computing Machinery (ACM). The subcommittee requested that the DBTG produce functional and language specifications, and incorporate its specifications into the COBOL specifications. Following the joint session, a straw vote taken by the subcommittee showed the majority of members agreed to the statement, "COBOL needs the Data Base concept." Shortly after the February, 1968, meeting, G. Durand of Southern Railway System replaced Mr. Simmons as Chairman of the DBTG, and remained Chairman until his resignation at the end of 1968. In January, 1969, A. Metaxides of Bell Telephone Laboratories was appointed Chairman. In the interests of continuity, membership on the DBTG became stabilized and the group capitalized on its earlier research as it worked under the direction of Mr. Metaxides toward producing the functional and language specifications for the incorporation into COBOL of a data base structure.

At the Tenth Anniversary Conference of CODASYL held in May, 1969, CODASYL reaffirmed its role in bringing about the design, development and specification of common data systems languages appropriate to user needs and feasibility. At the meeting CODASYL specified the following items as its policy and objectives in areas needing immediate attention.

1. The development, design, and specification of common languages should be separated from the establishment of standards. In addition to continuing its work on COBOL, CODASYL will do development, design and specification of other common user languages.

2. Information on CODASYL activities will be disseminated much more broadly. The anniversary meetings of CODASYL will be continued on an annual basis in Washington, D.C. in May.

3. CODASYL will continue to work on improving and extending COBOL to further enhance its greater utility.

4. CODASYL recognizes that COBOL should be extended to permit procedural interaction with environmental subjects such as dynamic scheduling and other job control features.

5. It was pointed out at the Tenth Anniversary meeting that the achievement of commonality is conditioned by the variation in implementation of compilers and that some level of uniformity must be established. CODASYL recognizes this need and feels that it is within the

established national service responsibilities of the National Bureau of Standards.

6. A common data definition language to achieve program independence from data is needed and will be developed. This language must have the ability to:

   a. Specify the physical description of data as it has been stored,

   b. Specify the logical organization of data for more complex structures,

   c. Modify the stored representation of the physical and logical data description without unnecessarily affecting the programs processing the original data.

   The COBOL Data Division will be investigated as a base for the development of a common data definition language. Representatives interested in FORTRAN, PL/I, ALGOL, JOVIAL, etc., are to be invited to participate in the design and constructive review of this common data definition language.

7. CODASYL will pursue the development, design and specification of a generalized and independent common data base management language. This language is conceived to be one which allows maintenance of and retrieval from a data base with the user specifying only what is wanted and not how it is to be accomplished.

8. The work of the Data Base Task Group will be reviewed and the elements that are of a data manipulative (procedural) nature will be used to extend COBOL capabilities in handling of data bases. Those elements of this work that relate to data definition will be used with the COBOL Data Division to provide a basis for the common data definition language.

9. Due to the favorable response to the "Survey of Generalized Data Base Management Systems" published by the Systems Committee, CODASYL will continue to prepare and publish reports on areas of professional interest.

At the Tenth Anniversary Meeting, consideration was given the idea of separating the data description and data manipulation languages. Proponents of the idea suggested that a separation would allow data bases described by a data description language to be independent of the host languages used for processing the data. The idea received wide endorsement at the meeting and was the basis of the direction of efforts by the DBTG from May, 1969, to October, 1969, at which time the group presented to the CODASYL Programming Language Committee the October 1969 DBTG Report.

The October 1969 DBTG Report presented the recommendations of the Data Base Task Group to its parent committee. The recommendataions detailed the semantics and syntax of a Data Description Language and a Data Manipulation Language. The Data Description Language specified in the report is a language for describing a data base. The Data Manipulation Language is a language which, when associated with the facilities of a host language such as COBOL, PL/I, ALGOL, JOVIAL, FORTRAN ..., allows manipulation of data bases described by the Data Description Language. It was the hope of the DBTG, when submitting the October 1969 DBTG Report, that the Data Description Language ultimately would form the basis of an industry standard and that individual host languages would interface with it. The Report recommended that CODASYL form a standing committee to maintain and further develop the Data Description Language.

The semantics and syntax of the Data Manipulation Language detailed in the October 1969 DBTG Report were proposed not only as an extension to COBOL, but also as a prototype of the manipulative capabilities required in a host language. The Data Base Task Group held that the proposals contained in the October 1969 DBTG Report were applicable not only to COBOL, but to a number of other host languages. Thus the DBTG recommended immediate distribution of the report to the computing community.

The following organizations, as members of the Data Base Task Group, contributed to the preparation of the October 1969 DBTG Report:

Allstate
Bell Telephone Laboratories
Burroughs
General Electric
General Motors Research Laboratories
Honeywell
International Business Machines Corporation
National Bureau of Standards
The NCR Co.
RCA
Southern Railway System
Travelers Insurance
United States Steel
UNIVAC
URS Systems Corporation

(Inclusion in the above list does not necessarily imply that the organization endorsed the Report.)

The October 1969 DBTG Report was reviewed at the December, 1969, meeting of the Programming Language Committee. Mr. Metaxides presented and discussed various documents of comments on the DBTG Report and the IBM Minority Report. The Programming Language Committee approved the DBTG Report for publication and included in the Report a request for proposals for clarifications, change, etc. The Report was published as a Type B release. (A Type B release is a document which does not represent language specifications and is made available with the permission/approval/support of the committee of CODASYL for consideration and study by the computing community. The views presented do not necessarily reflect those of the members of CODASYL, the committee, or the sponsors of committee members.) The October 1969 DBTG Report was published on behalf of CODASYL by the Association for Computing Machinery, and over 3000 copies were distributed in the United States. The Report was published in Europe by the British Computer Society and by the International Federation for Information Processing (IFIP) Administrative Data Processing Group (IAG).

From the time of publication of the October 1969 DBTG Report to April, 1971, 179 proposals for changes and extensions to the report were acted upon by the Data Base Task Group. 130 of these proposals were accepted and incorporated in the next DBTG report, the April 1971 DBTG Report. The member organizations of the Data Base Task Group at the time the April 1971 DBTG Report was presented to the Programming Language Committee were:

Bell Telephone Laboratories
B. F. Goodrich Chemical Company
Computer Sciences Corporation
Control Data Corporation
Equitable Life Assurance Society
General Motors Research Laboratories
Honeywell Information Systems
International Business Machines Corporation
International Computers Limited
Montgomery Ward
The NCR Co.
RCA Corporation
United States Navy
UNIVAC

(Inclusion in the above list does not necessarily imply that the organization endorsed the Report.)

The April 1971 DBTG Report was reviewed at the May, 1971, meeting of the CODASYL Programming Language Committee. IBM and RCA presented qualifying statements opposing endorsement of the Report. The following series of motions, which were passed by the PLC during the discussion of the Report, reflect the action taken at the meeting.

> "Moved that PLC should review the Schema portion of the DBTG Report for the purpose of possible recommendation to the CODASYL Executive Committee for publication as a stand alone document separate from COBOL."

> "Moved that PLC recommend that the CODASYL Executive Committee establish an organization separate from PLC to prepare, review and maintain the document describing the Schema for separate publication."

> "Moved that the PLC finds the report of the DBTG meets PLC's requirements for a data base facility and directs that the description of the functions (Subschema and DML) described in the DBTG Report be converted to a form conforming to that specified in the Publications Guidelines regarding proposals to the CODASYL COBOL Journal of Development."

> "Moved that the Subschema and DML portion of the DBTG Report (PLC item 7102, DBTG - 71001) be referred to the DBTG for modification in accordance with the previous motion."

The April 1971 Report was published on behalf of CODASYL in the United States by the Association for Computing Machinery. The Report was published in Europe by the British Computer Society and by the International Federation for Information Processing Administrative Data Processing Group.

The Executive Committee accepted the recommendation of the PLC to establish an organization separate from the Programming Language Committee to prepare, review, and maintain the document describing the Schema for separate publication. In a press release dated June, 1971, CODASYL announced its intention to form the Data Description Language Committee (DDLC), a standing committee independent of, and equal in status to the Programming Language Committee. In the press release, the Executive Committee announced the following objective of the DDLC:

> "The new committee (i.e., DDLC) is charged with finalizing the specifications for a common DDL independent of any high level programming language. The committee's work, which will be based on the April, 1971 DBTG Report (in particular sections 1, 2, and 3) is seen as an evolutionary process much like the development of COBOL."

In July, 1971, the Data Base Task Group established subgroup, the DBTG Publication Group, held its first meeting to begin working toward the goal of converting the Subschema and DML specifications in the April 1971 DBTG Report to a proposal for change to the CODASYL COBOL Journal of Development (JOD). In December, 1971, the DBTG Publication Group became the Data Manipulation Language Task Group (DMLTG) of the PLC. The DMLTG was given responsiblity for completion of the PLC proposal which would add a data base facility to the COBOL JOD, and for liaison with the Data Description Language Committee. The DBTG retained its responsiblity for reviewing the work of the DMLTG, and developing long-range extensions to the COBOL Data Base Facility. The names of the task groups were subsequently changed, the DMLTG to the Data Base Language Task Group, and the DBTG to the Data Base Concepts Task Group.

The inaugural meeting of the CODASYL Data Description Language Committee (DDLC) was held November 30, 1971. Under the chairmanship of a former PLC Chairman, R. Kurz of NCR, currently of Southern Railway System, the committee prepared the proposed changes to the CODASYL Constitution to accommodate the organization of the Data Description Language Committee, and the Data Description Language Committee Bylaws. Among the changes recommended by the DDLC to the CODASYL Constitution was inclusion of the following purpose of the DDLC:

> "The CODASYL DDLC strives to establish ways to aid the functions of data administration and systems administration. This includes the provision of specifications for the declarations required to establish and maintain data base structures."

At the second meeting of the DDLC January 26-27, 1972, the committee amended and approved the changes to the CODASYL Constitution and the DDLC Bylaws, and proposed short and long range goals of the DDLC. Section 3 of the April 1971 DBTG Report was accepted by the DDLC as its "base document". In February, 1972, the CODASYL Executive Committee approved the changes to the CODASYL Constitution and the DDLC Bylaws.

At the third meeting of the DDLC March 14-15, 1972, the committee identified and agreed to the following objectives which are listed in order of importance.

1. The DDLC shall publish its language specification in a Journal of DDL Development akin to the Journal of COBOL Development published by PLC.

2. The DDLC shall maintain and extend its language specifications, i.e., the DDL. In this work the DDLC is applying to the base document its formal procedures for considering proposals to change its own language specifications.

3. In order to develop, maintain and extend the formal language specifications, the DDLC will investigate certain related areas. These investigations may lead to published documents separate from the Journal of DDL Development. Currently the following areas are being investigated.

   a. The purpose of the DDL in terms of its environment and possible methodologies for its use.

   b. Coordination with existing high level languages to determine, in particular, what special constraints on the DDL arise (if any).

4. The following areas have been classified as longer-term objectives.

   a. The definition of generic functional terms for DDLs.

   b. Establishing guide lines for those wishing to establish a subset of the DDL.

   c. The definition of restructuring facilities in the DDL for application to a data base.

   d. Further development of the concept of a subschema and the necessary DDLs.

   e. The relation between the DDL and self-contained data manipulative capabilities.

   f. Investigation of possible changes to the base document's meta-language and syntax.

Since the fourth meeting of the DDLC in May, 1972, the committee worked toward accomplishment of its first objective, the preparation of this Data Description Language Journal of Development. To produce this document within a decided time frame, the DDLC limited its efforts primarily to clarification of, rather than extension to, the base document. The mode of operation of the DDLC during the preparation of the Journal of Development was, and currently is, based upon the submission and consideration of working papers, and change proposals directed toward the base document.

This Data Description Language Journal of Development is currently the base document of the Data Description Language Committee. Working papers and change proposals may be submitted to the DDLC by individuals who do not participate in any CODASYL activity including that of the DDLC. Although no specific proposal format is mandatory, the following guidelines are applied to proposal context:

1. Proposals made to specific points must cite all specific references.

2. Proposals of general nature should cite at least some specific instances.

3. Sufficient justification and motivation should be contained in the proposal to point out what the problem appears to be and why this proposal is a solution.

4. The proposal should include recommended specification changes with specific references where necessary.

Data Description Language Committee formal meetings are held every two months for a minimum of three days. Membership on the Data Description Language Committee is institutional in nature and resides in an organization rather than an individual. Membership is based upon the sponsoring institution's expressed support of CODASYL objectives and upon the availability of a suitable vacancy within the established membership limitation of 25. Any institution committed to using or implementing any language with an element of data description may apply to the Chairman of the Data Description Language Committee for membership. The structure of the committee is such that neither those members representing institutions considered to be primarily in the category of implementor nor those members representing institutions considered to be primarily in the category of user shall comprise two-thirds or more of the membership, and an institution may not have more than one membership.

Current member organizations of the Data Description Language Committee are:

Air Force Data Systems Design Center
Bell Telephone Laboratories
B. F. Goodrich Chemical Co.
Burroughs Corporation
Cincom Systems, Inc.
Control Data Corporation
Defense Communications Agency
Department of the Navy
Fireman's Fund American Insurance Co.
General Electric Company
General Motors Research Laboratories
Honeywell Information Systems Incorporated
International Business Machines Corporation
International Computers Limited
The MITRE Corporation
National Bureau of Standards
The NCR Co.
The Ohio State University
Philips-Electrologica, B.V.
Scientific Control Systems Limited
Southern Railway System
Sperry Univac Corporation
The University of Michigan
Xerox Corporation

Table of Contents

SECTION 2. MAJOR CONCEPTS

Page

2.0 MAJOR CONCEPTS

2.1 SCOPE AND PURPOSE

This Journal of Development contains a specification of a language to describe the structure and contents of a data base. This description is called a schema. The schema language represents one of several languages which data base designers, implementors and users will employ. Other languages include current procedural programming languages, for example, COBOL and FORTRAN, data manipulation languages, device media control languages, and languages to control the execution of work (data processing) on a computer system. The current procedural programming languages must contain the following elements to be used with a schema language controlled data base:

- A subschema language to describe a subset of the schema which is of interest to a particular application program. A subschema enables an application program to deal with a subset of the data in the data base. The subschema may also vary in certain respects from the schema with respect to particular elements in the data base.

- A data manipulation language (DML) used at execution time to handle all program interfaces to the data base.

The subschema language specifications and the DML specifications are outside the scope of this Journal. Where they are treated as extensions of a procedural programming language, they reflect the syntax and other characteristics of their "host".

In order to create and process data it is necessary to describe to the computer system the mapping of the data onto physical storage media. This is accomplished through a device media control language (DMCL). Specifications for a DMCL are not included in this Journal.

In order to execute work on a computer, including processing a data base, it is necessary to have a language to specify control of the work to be done. Such a language is commonly called a job control language. Specifications for a job control language are not included in this Journal.

These are a minimum set of languages for a data base system as envisaged in this Journal, and for which this schema language is intended. Other languages (for example, those of the functional, end user oriented type known as self contained languages) could also interface with a data base described by this schema language.

The schema language is a specification of a common data description language (DDL) which is independent of, but common to the other languages required for a data base system. It is expected that the schema DDL will have a significant impact on the development of functionally compatible data base management systems and will increase the portability of programs between different computer systems. While the widespread adoption of the schema DDL will not of itself fully achieve this objective, it does lay the foundation for the development and adoption of common subschema and data manipulation languages. The net result would be an increase in the portability of source programs. The portability of physical data base representations is not dealt with in this Journal.

2.2 TERMINOLOGY

2.2.1 SCHEMA DDL

The schema DDL is used for describing a DATA BASE, which may be shared by many programs written in many languages. This description is in terms of the names and characteristics of the DATA ITEMS, DATA AGGREGATES, RECORDS, AREAS, and SETS included in the data base, and the relationships that exist and must be maintained between occurrences of those elements in the data base.

A DATA ITEM is an occurrence of the smallest unit of named data. It is represented in a data base by a value.

A DATA AGGREGATE is an occurrence of a named collection of data items within a record. There are two kinds: vectors and repeating groups. A vector is a one dimensional sequence of data items, all of which have identical characteristics. A repeating group is a collection of data that occurs a number of times within a record occurrence. The collection may consist of data items, vectors, and repeating groups.

A RECORD is an occurrence of a named collection of zero, one, or more data items or data aggregates. This collection is specified in the schema DDL by means of a Record Entry. Each Record Entry in the schema for a data base determines a record type, of which there may be an arbitrary number of record occurrences (records) in the data base. For example, there would be one occurrence of a record of type PAYROLL-RECORD for each employee.

A SET is an occurrence of a named collection of records. The collection is specified in the schema DDL by means of a Set Entry. Each Set Entry in the schema for a data base determines a set type, of which there may be an arbitrary number of set occurrences (sets) in the data base. Each set type specified in the schema may have one record type declared as its owner record type, and one or more other record types declared as its member record types. Each set must contain one occurrence of its defined owner record type and may contain an arbitrary number of occurrences of each of its defined member record types. For example, if a set type QUALIFICATIONS was defined as having owner record type EMPLOYEE and member record types JOB and SKILL, each occurrence of set type QUALIFICATIONS must contain one occurrence of record type EMPLOYEE, and may contain an arbitrary number of occurrences of record types JOB and SKILL.

An AREA is a named collection of records which need not preserve owner/member relationships. An area may contain occurrences of one or more record types, and a record type may have occurrences in more than one area. A particular record is assigned to a single area and may not migrate between areas. An area may optionally be declared to be temporary. Temporary areas are local to a run unit, that is, they are created for that run unit, cannot be accessed by other run units, and disappear when the run unit terminates.

A SCHEMA consists of DDL entries and is a complete description of a data base. It includes the names and descriptions of all of the areas, set types, record types and associated data items and data aggregates as they exist in the data base and are known to the data base management system (DBMS). The DDL for developing a schema appears in Section 3.

A DATA BASE consists of all the records, sets and areas which are controlled by a specific schema. If an installation has multiple data bases, there must be a separate schema for each data base. Furthermore, the content of different data bases is assumed to be disjoint.

2.2.2 SUBSCHEMA DDL

In addition to the schema declarations, it is expected that each program will have access to a description of those areas, set types, record types, data items, and data aggregates of interest to it. Such a description is termed a subschema, and is not specified in this Journal.

2.2.3 THE DATA MANIPULATION LANGUAGES (DML)

A PROGRAM is a set or group of instructions.

A RUN UNIT is an execution of one or more programs.

A DATA MANIPULATION LANGUAGE (DML) is a language which the programmer uses to cause data to be transferred between his run unit and the data base. It is the intent of this DDL to provide a data structure suitable for multiple DML's. To date, the DML of the April '71 DBTG Report is the only specific DML specified by CODASYL. The DBTG DML is not a complete language by itself. It relies on a host language to provide a framework for it and to provide the procedural capabilities required to manipulate data.

The USER WORKING AREA (UWA) is conceptually a loading and unloading zone where all data provided by the DBMS in response to a call for data is delivered and where all data to be picked up by the DBMS must be placed. Each data item included in the subschema will be assigned a location in the UWA and may be referenced by the programs by its name as declared in the subschema.

2.3 CONCEPTUAL FRAMEWORK

2.3.1 EXAMPLE

This Journal is not a complete specification for a DBMS. However, it may be helpful to an understanding of the DDL to conceptualize a complete system. The system presented is for pedagogic purposes only and is illustrated by Diagram 1.

The numbered arrows in Diagram 1 trace a call for data by run-unit-1 and are explained in the following. Calls for data by other run units may be handled concurrently by the DBMS, but this is not shown in the diagram.

'1' using the DML statements, the run unit makes a call for data to the DBMS.

'2' the DBMS analyzes the call and supplements the arguments provided in the call itself with information provided by the schema for the data base, and the subschema referenced by the run unit originating the call.

'3' on the basis of the call for its services and information obtained from the schema and subschema, the DBMS requests physical I/O operations, as required to execute the call, from the Operating System.

'4' the Operating System interacts with the storage media containing the data base.

'5' data is transferred between the data base and the system buffers.

'6' the DBMS transfers data, as required to fulfill the call, between the system buffeis and the UWA of the run unit originating the call. Any required data transformations between the representation of the data as it appears in the data base (as declared in the schema) and the representation of the data as it appears in a run unit's UWA (as declared by the subschema) are handled by the DBMS.

'7' the DBMS provides status information to the run unit on the outcome of its call, for example, error indications.

'8' data in a run unit's UWA may be manipulated as required, using the facilities in the host language.

'9' the DBMS administers the system buffers. The system buffers are shared by all run units serviced by the DBMS. Run units interact with the system buffers entirely through the DBMS.

CONCEPTUAL DATA BASE MANAGEMENT SYSTEM

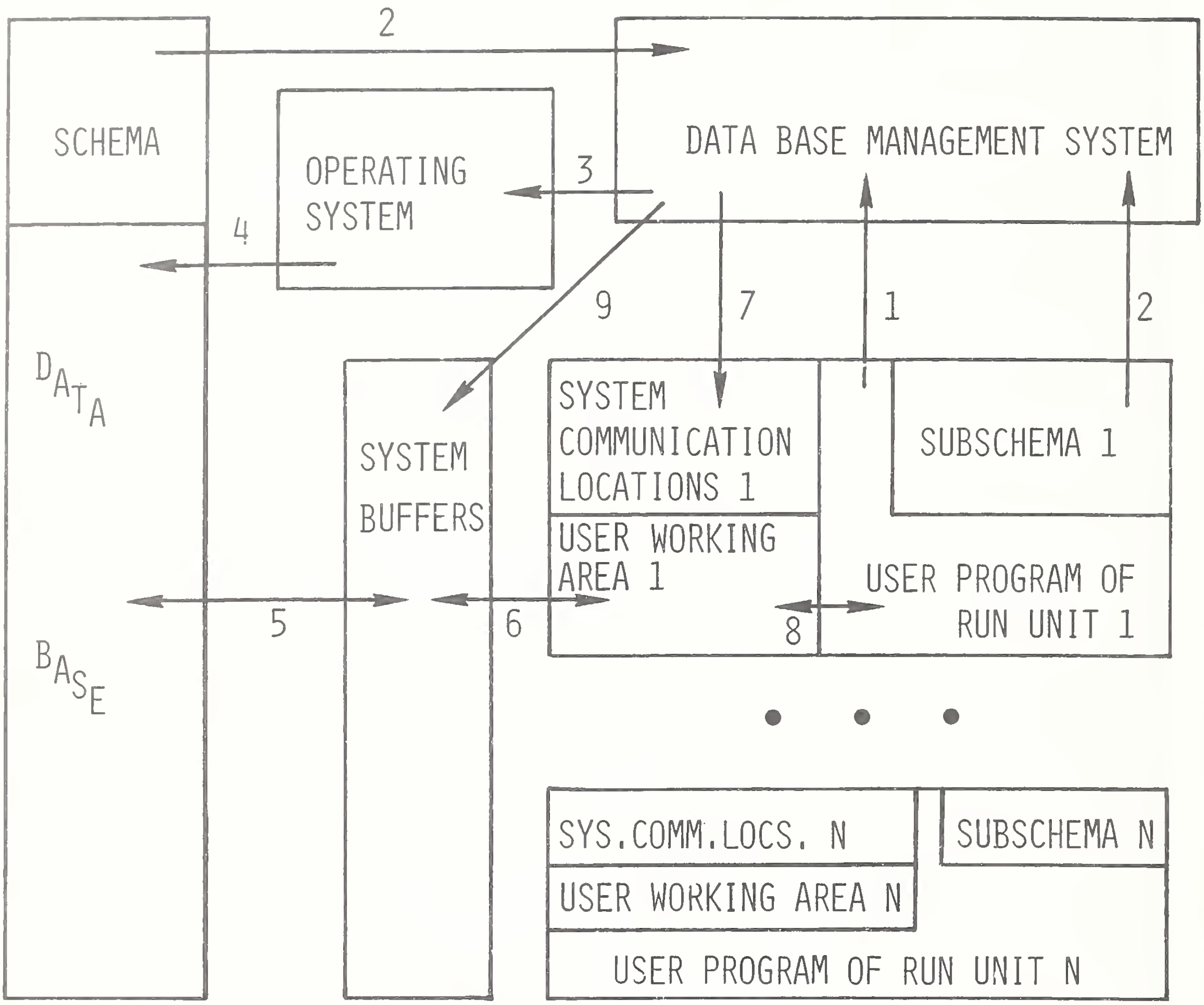


DIAGRAM 1

2.3.2 THE SCHEMA AND DEVICE INDEPENDENCE

No schema DDL entry includes references to the physical devices or media space. Thus, a schema written using the schema DDL is a description of the data base which is not affected by the devices and media used to store the data. The data base may, therefore, be stored on any combination of storage media which are supported in a particular DBMS. Some devices, such as magnetic tape, because of their sequential nature, may not allow full advantage to be taken of the facilities included in the DDL. Such devices are not precluded, however, and may be perfectly adequate for some of the data.

2.3.3 THE SCHEMA AND THE SUBSCHEMA

2.3.3.1 SCHEMA VS. SUBSCHEMA

The concept of separate schema and subschema allows the separation of the description of the entire data base from the description of portions of the data base known to individual programs. The concept is significant from several points of view:

- An individual programmer need not be concerned with the universe of the entire data base, but only with those portions of the data base which are relevant to the program he is writing. Since the data base may contain data which is relevant to, and shared by, multiple applications, this may ease the writing, debugging, and maintaining of programs.

- A run unit is limited to that portion of the data base that is known to it via its subschema. Therefore, the rest of the data base is insulated to a large extent from that run unit.

- A measure of data independence is provided for programs in that certain changes may be made to the schema for the data base, and the data base adjusted accordingly, without affecting existing programs using that data. This is possible because the subschema may vary in certain important aspects from the schema on which it is based and because programs are only dependent on the subschema. The degree of data independence achieved is entirely dependent on the capabilities of the DBMS for mapping between the schema and subschema data descriptions.

- A common language may be specified for defining a data base while allowing that part of the data base known to a program to be described in a manner which is oriented towards the conventions of the language in which that program is written. This permits the use of several languages, chosen on the basis of their suitability to a problem to be solved, to process the same data base.

2.3.3.2 VARIATIONS BETWEEN THE SCHEMA AND SUBSCHEMAS

While it is not the intent of this Journal to specify the details of a subschema DDL, either explicitly or implicitly, some possible variations are worth noting. For example in a subschema:

- At the data item level:
  a. Descriptions of specific data items may be omitted.

  b. Privacy locks may be changed.

  c. The characteristics of data items may be different.

  d. The ordering of data items may be changed.

- At the data aggregate level:
  a. Descriptions of specific data aggregates may be omitted.

  b. Privacy locks may be changed.

  c. The ordering of data aggregates may be changed.

  d. Vectors may be redefined as multi-dimensional arrays.

  e. Data items or data aggregates may be selected and given a group name.

  f. Additional structure mapping may be provided by the facilities of a particular subschema DDL.

- At the record level:
  a. Descriptions of specific record types may be omitted.

  b. Privacy locks may be changed.

  c. Descriptions of new record types composed of data items from other record types may be introduced.

- At the set level:
  a. Descriptions of specific set types may be omitted.

    b. Privacy locks may be changed.

    c. Different set selection criteria may be specified.

    d. Descriptions of specific member record types may be omitted.

- At the area level:

    a. Descriptions of specific areas and the included records may be omitted, while other occurrences of the same record type are included.

    b. Privacy locks may be changed.

The following additional points are important to an understanding of the concepts of schema and subschema.

- An arbitrary number of subschemas may be declared on the basis of any given schema.

- The declaration of a subschema has no effect on the declaration of any other subschema and subschemas may overlap one another.

- A user program references a subschema.

- The same subschema may be referenced by an arbitrary number of programs.

- Only the areas, records, data items, and sets described in the subschema referenced by a program may be used by that program.

- A program references a subschema that is consistent with its source language.

2.3.4 THE SCHEMA AND THE DML

The relationship between the DDL and a DML is the relationship between declarations and procedure. The declarations impose a discipline over the executable code and are to some extent substitutes for procedures written in the DML and the host language.

In order to specify the relationship between DDL declarations and DML functions a set of basic data manipulation functions must be defined which is DML and host language independent. Specific commands provided by a particular DML must be resolved into those basic functions. The resolution is defined by the implementor of the DML.

The basic data manipulation functions assumed in these specifications include the functions required to:

- Select records.
- Present records to the run unit.
- Add new records and relationships.
- Change existing records and relationships.
- Remove existing records and relationships.

2.3.5 SYSTEM SUPPORT FUNCTIONS

In addition to the conceptual framework described above, the specifications for a complete DBMS should include descriptions and language specification for:

- The utility or service routines which are required to support a data base in day to day operations. Examples of such routines are:

  a. Dump, edit, and print routines

  b. Load routines

  c. Preconditioning routines

  d. Garbage collection routines

  e. Statistical gathering and analysis routines

  f. Compare routines

- The data base recovery routines including activity logging, checkpoint and rollback.
- A language which permits modification of a schema or subschema and causes the changes to be reflected in the data base itself. Without such a language, changes to the schema can only be made by developing an entirely new schema and restructuring the data base in accordance with the new schema.
- The assignment of data to devices and media space, and specifying and controlling buffering, paging, and overflow. The term device media control language (DMCL) is used for these aspects of a DBMS.

This Journal does not include language specifications for any of these functions.

2.3.6 FACILITIES FOR DATA ADMINISTRATION

In an environment where a data base includes data which is shared by many user programs, it becomes necessary for the schema and perhaps the subschemas to be developed centrally. In such an environment a data base is, in a sense, a compromise between the needs of the various user programs. The data base will therefore require a means of mediating conflicting needs. This mediation is the prime responsibility of the Data Aministrator. The term Data Administrator is used to emphasize the human activity involved in the performance of this function.

The Data Administrator's function is to create and maintain both the data base and its schema in such a way that the data base may satisfy efficiently the data requirements of its user programs. This function may include the following tasks.

- Organizing, that is designing and assembling the data base.
- Monitoring the use and performance of the data base.
- Reorganizing and restructuring the data base so as to improve its performance.
- Recovering the data base after system malfunctions.

Organizing

While designing and building the data base a Data Administrator will:

- Write a schema describing the data base and input it to the DBMS.
- Load the data base.
- Assign privacy locks and issue privacy keys to users who need to use portions of this data base.
- Assign data to devices and media using a DMCL.

Monitoring

Effective use of the DBMS requires monitoring the activity of its users for usage, response, privacy breach, and potential reorganization. These activities require the Data Adminstrator to:

- Gather statistics on the usage of portions of the data base.
- Record an audit trail of changes to the data base to aid in recovering from system and user failures.

Reorganizing and Restructuring

As a result of information gained through monitoring or because of new information required in the data base, changes may be required. This requires the Data Administrator to:

- Modify the schema and compile the changes into the object version of the schema.
- Modify the data base to reflect changes in the schema.
- Remove inaccessible records and compact reusable storage space (garbage collecting).
- Reassign data to different devices and media (using a DMCL) based on time/space requirements.
- Edit portions of the data base.

Recovering

Various system failures will occasionally require that portions of the data base be restored to some previous condition. This requires the Data Administrator to:

- Dump portions of a data base onto alternate storage media.
- Restore portions of a data base using previously dumped versions or using audit trails gathered by DBMS monitoring facilities.

The individual tasks noted under the above headings are not intended to be an exhaustive list but merely to indicate the scope of the Data Administrator's functions.

2.3.7 DATA BASE PROCEDURES

At various points in the accessing of a data base, computations are required which are specific to that particular data base. Some examples of these computations are:

- Checking of privacy keys for validity.

- Computation of data item values as functions of other data item values.

- Searching algorithms.

- Compression and expansion of values of data items.

- Validation of values of data items.

- Systems instrumentation.

The routines which perform these computations are called data base procedures. They are stored in the system where they can be invoked by the DBMS when they are needed. The rules for writing data base procedures (that is, linkage conventions, allowable side effects, programming languages in which they are written, etc.) are implementor defined. There must be provision however for data base procedures to have access to the following information:

- Identity of the run unit from which control was transferred to the data base procedure.

- Identity of the declarative clause involved.

- Type of DML function from which control was transferred to the data base procedure.

- Type of entry or subentry that is the operand of the DML function from which control was transferred.

- All of the information available to the run unit from which control was transferred to the data base procedure. This includes access to the run unit's user working area, currency indicators, special registers and all data in the data base.

Data base procedures must also have the capability of returning values to the run unit from which control was transferred, to other run units and to other data base procedures.

2.3.8 REPRESENTATION OF DATA STRUCTURES

2.3.8.1 INTRODUCTION

One of the objectives of the DDL is to allow data to be structured in the manner most suitable to each application without requiring data redundancy. To achieve this, it must be possible to represent relationships by methods other than juxtaposition of records.

The schema DDL as described in Section 3 provides facilities to declare data structures among records in the data base. The set concept provides a structure representing a one to many relationship, and order within sets provides a sequential relationship. A wide variety of data structures including sequential, tree, and network relationships can be represented conveniently by these facilities. Sections 2.3.8.2, 2.3.8.3, and 2.3.8.5 show how the following types of data structures may be represented in the DDL:

- Sequential
- Trees
- Networks

In addition, the absence of structure may be represented by declaring records in the schema which do not participate in sets.

2.3.8.2 SEQUENTIAL DATA STRUCTURES

A sequential data structure is an ordered collection of records. Such a structure may be represented in the DDL by a single set whose member records are ordered in the specified manner. The SYSTEM option of the OWNER clause of the Set Subentry obviates the necessity of declaring an unnecessary owner record type for a sequential structure. For efficiency in retrieving the records of a sequential data structure, the DDL provides a facility to indicate that the records are to be retrievable either in the forward or in the reverse order or both.

2.3.8.3 TREES

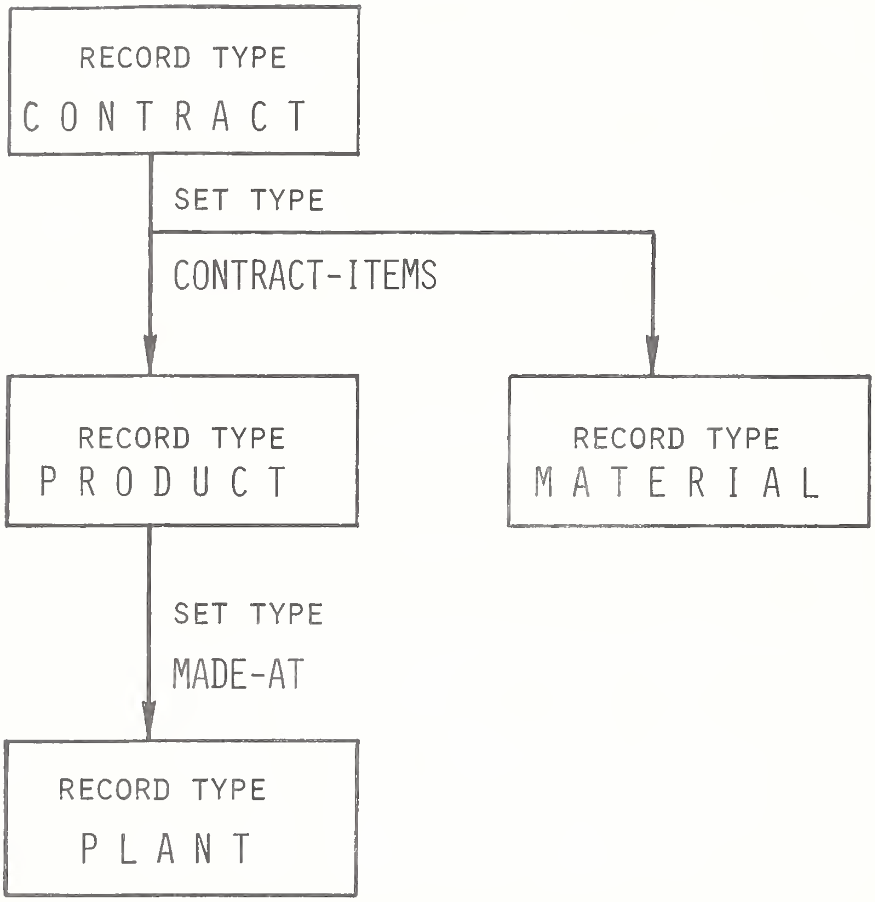
A tree structure is a hierarchical structure in which each record (except one called the root) is related to zero or more different records below it in the hierarchy and to exactly one record above it in the hierarchy. The root of the tree is the highest level record and is not related to any record above itself.

Diagram 2 is a representation of a tree data structure involving four record types and two set types. The records contain data about contracts whose completion requires products and materials and about the plants where products may be inventoried. There are two one to many relationships which give these records a hierarchical tree structure. The CONTRACT-ITEMS set type relates products and materials to contracts. The MADE-AT set type relates parts to the plants where that product is made.

These sets and records are shown diagramatically by:

- Each record type is shown as a box.
- Each set type is shown as a fork pointing from one box representing the owner record type of the set type to one or more boxes representing the member record types of the set type.

Since every set may have an arbitrary number of records as members and since an arbitrary number of set types is permitted, a tree of any breadth and depth may be represented.



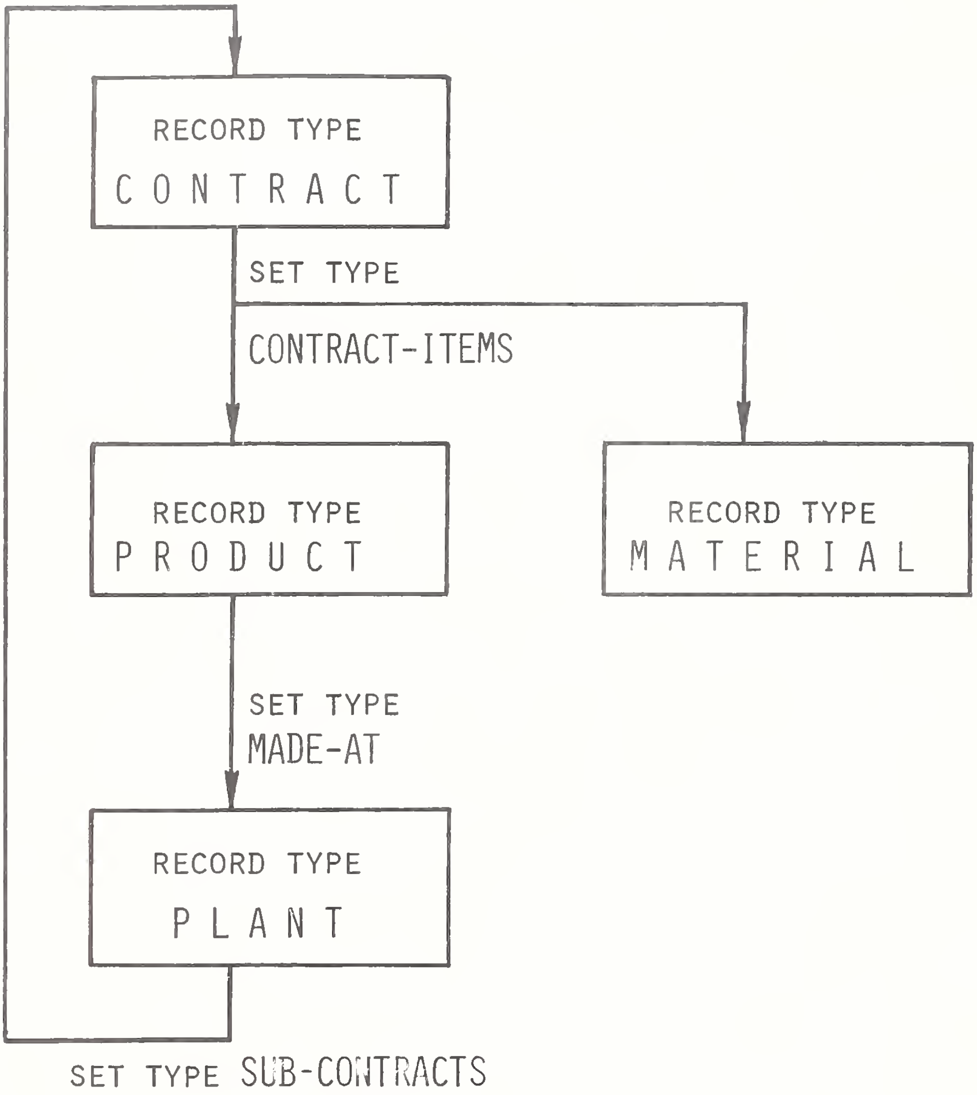
SET REPRESENTATION OF TREE DATA

DIAGRAM 2

2.3.8.4 CYCLES

Structures which represent cycles are permitted. A cyclic structure occurs when a series of different set types is declared such that each set type's owner is a member of the previous set type in the series. The membership of at least one of the member record types in a cyclic structure must be declared to be manual. (See Section 2.4.3.4, set membership).

Diagram 3 shows a data structure containing a cycle of set types. It shows the record types and set types of diagram 2 together with a new set type. The SUB-CONTRACTS sets specify which contracts have been let by each plant. Thus SUB-CONTRACTS, CONTRACT-ITEMS, and MADE-AT form a cycle of set types.



SET REPRESENTATION CONTAINING CYCLE

DIAGRAM 3

2.3.8.5 NETWORKS

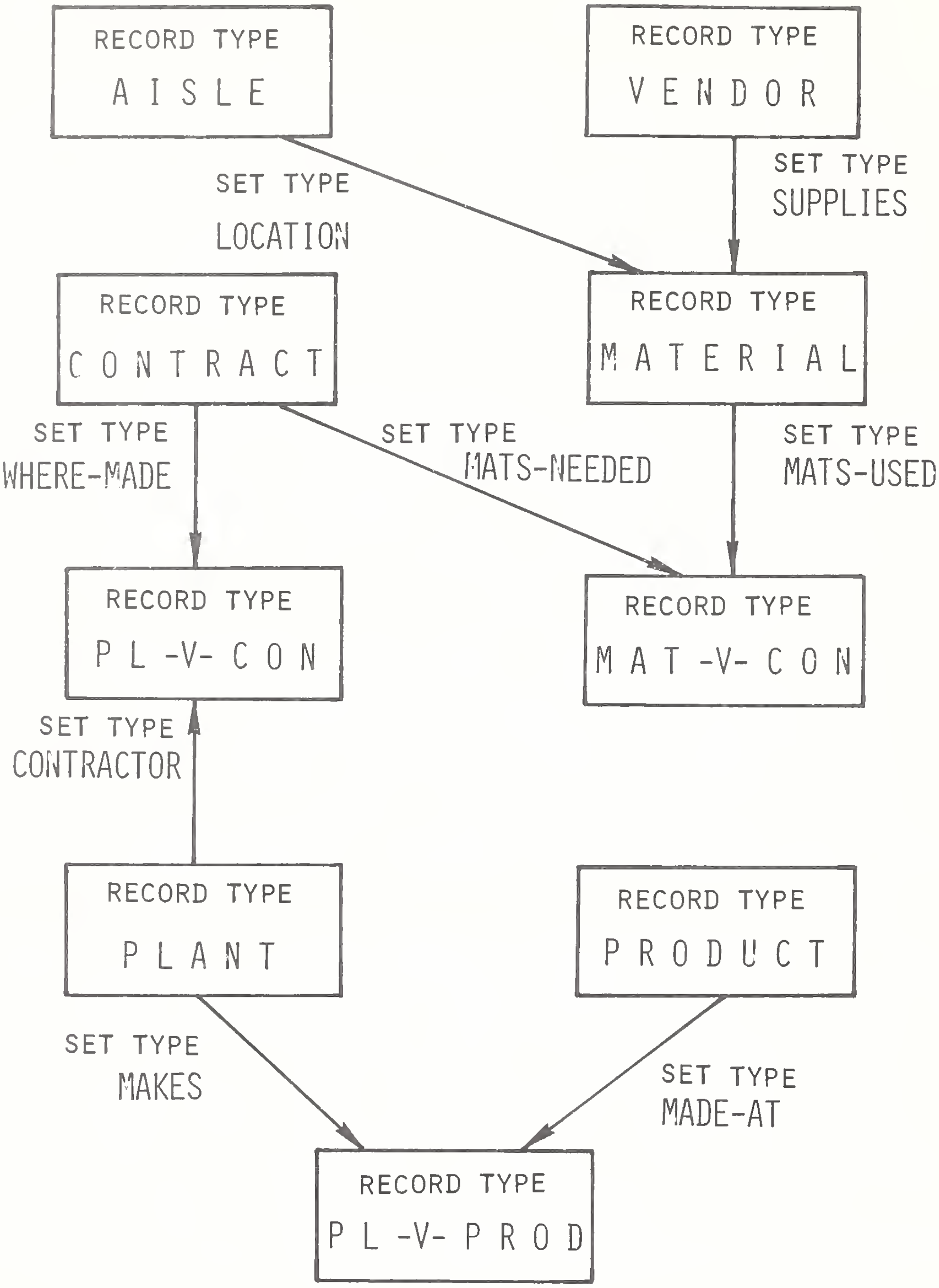
A more general data structure than the tree or cycle is the network. Whereas in a tree structure each record type participates in only a single set type as a member, in a network each record type may be a member of more than one set type. Thus, unlike tree structures, a network allows the representation of many to many relationships between records, and multiple classifications of records, without data redundancy. Diagram 4 illustrates these two situations.

An example of multiple classification is provided by the MATERIAL record type and the SUPPLIES and LOCATION set types. Each MATERIAL record may participate as a member of a SUPPLIES set and a LOCATION set. The SUPPLIES sets specify which materials are available from each vendor. The LOCATION sets specify the warehouse aisles where materials are stored. Thus materials are classified by vendor and by storage location (aisle) without repetition of the materials records.

Diagram 4 also contains several examples of many to many relationships between records. Such relationships exist between contracts and materials, contracts and plants, and plants and products. In using set types to represent many to many relationships, it must be remembered that each set represents a one to many relationship from owner to member but a one to one relationship from member to owner. Thus the representation of a many to many relationship requires two set types and three record types.

For example in the relationship between contracts and materials, the CONTRACT record type and the MATERIAL record type are the respective owners of the MATS-NEEDED set type and the MATS-USED set type. The MAT-V-CON record type is a member of both set types. The MATS-NEEDED set type represents which materials are needed for which contract; the MATS-USED set type represents which contracts are being supplied with a particular material. The MAT-V-CON record type in effect establishes the many to many relationship between contracts and materials. Each MAT-V-CON record is unique to a CONTRACT/MATERIAL pair and can thus be used to carry such data as the estimated and actual quantities of material required for a specific contract.

The other examples of many to many relationships in Diagram 4 are similar. In the case of the relationship between contracts and plants, the WHERE-MADE set type, the CONTRACTOR set type, and the PL-V-CON record type provide information about which plants are supplying a contract and the contracts on which a plant is working. The relationship between plants and products provides information about which plants are making a product, and the products made by a plant. Thus the set capability may be used to represent a complex network relationship.



SET REPRESENTATION OF NETWORK DATA

DIAGRAM 4

2.4 DDL FACILITIES

2.4.1 AREAS

The following are relevant to an understanding of the concept of an area:

- An area is a named subdivision of a data base.
- An arbitrary number of areas may be declared in a schema.
- Each area must be named.
- An area may be either permanent, or temporary and local to a run unit.
- Records may be assigned to areas independently of their set associations. A given record type or set type may have occurrences in multiple areas and a set may span areas.
- Each record must be associated with one and only one area. This association is permanent, in that a record may not change areas.

The concept of area allows the Data Administrator to subdivide a data base rather than considering the data base as a single unit. The use of areas allows the Data Administrator or the DBMS to control placement of an entire area to provide efficient storage and retrieval. The opening of areas by run units also gives implementors an opportunity to optimize access to the data base, since the run unit has narrowed the range of interest in the data base to a relatively small number of subdivisions of the entire data base. Areas are a convenient unit for recovery, as duplication or backup can be carried out selectively. Areas also provide a convenient natural subdivision for allowing certain unused portions of the data base to be saved in archival storage while the remainder of the data base is actively accessed. Mechanisms for mapping areas to media space are not specified in this Journal.

2.4.2 RECORDS

2.4.2.1 RECORD DESCRIPTION FACILITIES OF THE SCHEMA DDL

To describe data bases that can be manipulated through many host languages, the schema DDL must provide data formats and representations that can be mapped into the data formats and representations of the various host languages. Record description in the schema is, therefore, independent of any host language. The record description facilities of the subschema provide the link between the schema record description (the Record Entry) and the data formats and representations of a particular host language. The record description concepts of the schema DDL are as follows:

- Data Items: The smallest unit of named data is the data item. An occurrence of a data item is a representation of a value. The set of values that a data item can represent is called its range. The range of a data item is always restricted to one data type (and, if the type is arithmetic, to one base, scale, mode, and precision). The types of data are arithmetic, string, data base key, and implementor type.

    a. Arithmetic Data: An arithmetic data item is one that has a numeric value with characteristics of base, scale, mode, and precision. The value may be represented either in a numeric pictured form or in a coded form, that is, in an internal representation that is implementation dependent. A numeric pictured data item is a string of characters or bits that is given a numeric interpretation by means of the PICTURE clause of the Data Subentry. A data item in encoded form is described in terms of its base, scale, mode, and precision by means of the TYPE clause of the Data Subentry. Arithmetic data has either a decimal or binary base, a fixed point or floating point scale, and a real or complex mode. Fixed point data items are numbers for which the number of decimal or binary digits and the position of the decimal or binary point is specified. Floating point data items are numbers in the form of a mantissa and an exponent part. A complex data item consists of a pair of values. The first value is the real part of the complex number, and the second value is the imaginary part.

    b. String Data: String data is classified as either character string or bit string. The length of a string data item is equivalent to the number of characters (for a character string) or the number of binary digits (for a bit string) in the item. Character string data consists of a string of one or more characters of the data character set defined by the implementor. Bit string data consists of a string of one or more binary digits (0 and 1). The length of a string data item is fixed. However, the schema DDL provides for vectors whose elements can be string data. String data may be described by a PICTURE clause or a TYPE clause.

c. Data Base Keys: Each record in the data base is uniquely identified by a data base key. Items whose values are data base keys may be declared using the TYPE clause. The format and representation of data base keys is implementor defined.

d. Implementor Types: An implementor may provide for additional types of data items which can be declared by means of a TYPE clause.

- Data Aggregates: A named collection of data items within a record is known as a data aggregate. There are two kinds:

    a. Vectors: A vector is a one dimensional sequence of data items, all of which have identical characteristics. For example, all of the data items of the vector must have the same base, scale, mode, and precision. A vector is described in one Data Subentry by specifying the number of data items in the vector using the OCCURS clause of the Data Subentry. If the vector is not part of a repeating group the number of data items may be specified by the value of a data item in the same record.

    b. Repeating Groups: A repeating group is a collection of data that occurs a number of times within a record. The collection may consist of data items, vectors, and repeating groups; thus, repeating groups may be nested. A repeating group is described by a series of Data Subentries. Grouping is specified by means of level numbers and the number of occurrences is specified by use of an OCCURS clause. If a repeating group is not part of another repeating group, the number of occurrences of the group may be specified by the value of a data item in the same record.

- Data Subentry: A Data Subentry is the component of a schema which names and describes a data item, vector, component of a repeating group, or a repeating group. A Data Subentry consists of an optional level number, and a name for the component being defined followed by one or more clauses describing the characteristics of the component. Level numbers are specified in Data Subentries in order to define hierarchical relationships among the Data Subentries.

- Record: A record is a collection of data described by a Record Entry; the data content of a record is described by a series of Data Subentries. While both records and repeating groups are considered here as being collections of data that may occur a number of times, the concept of record in the schema differs from that of repeating group in several important ways:

a. Records, whether of one type or of several types, are related to one another by means of the set definitions in the schema, rather than by level numbers used for repeating groups. Since records are related to one another by sets instead of level numbers, complex network relationships may be defined among records which cannot be defined among repeating groups.

b. The record is the basic unit of access in the data base. Records in the data base are assigned data base keys which enable them to be directly accessed at any time if the data base keys are known. Repeating groups may only be accessed once the record occurrence within which the repeating group occurs is available to the run unit.

c. The number of occurrences of a particular record type need not be explicitly stated, either in the schema or as the value of a particular data item, as they must be for repeating groups. Rather, the number of occurrences depends on the DML functions which have been applied to the data base.

- Data-Base-Data-Name and Data-Base-Identifier: A data-base-data-name is a user defined name for a data item or data aggregate. The named data item or data aggregate need not be the subject of a Data Subentry. A data base identifier is a reference to a data item or a data aggregate declared in the schema, and is a reference to a data base data name.

2.4.2.2 PLACEMENT CONTROL

The objective of providing for control of relative placement of records is to increase efficiency by advising the DBMS of anticipated usage patterns of records. Thus, the schema DDL permits specification of the area or areas to which occurrences of a particular record type are to be assigned by the DBMS. The schema DDL also includes a clause which causes records being added to the data base to be stored near some other record in the data base. Conceputally, the effect of such clauses is to request the clustering of records which are required as a group to perform some procedure, thereby improving that procedure's performance. The declarations for controlling placement are the WITHIN clause and the LOCATION clause of the Record Subentry. Both affect the manner in which the DBMS assigns data base keys to records.

The fact that the schema DDL permits control over the relative placement of records does not necessarily have any physical connotations.

2.4.2.2.1 DATA BASE KEYS

In all data base management systems, it must be possible for the DBMS to distinguish each occurrence of a record from every other occurrence of a record in the data base. For this to be possible, a unique identifier must exist for each and every record in the data base.

This DDL assumes that such a unique identifier known as a data base key is assigned by the DBMS to a record when it is stored for the first time in the data base. It is assigned in accordance with the declarations for that record in the schema.

The data base keys may be:

- Supplied to the DBMS by a run unit or data base procedure.
- Generated by the DBMS from the data contents of the record.
- Assigned by a DBMS implementor algorithm.

A data base key once assigned to a record remains as the permanent identifier of that record until the record is deleted from the data base.

The permanence of data base keys must be guaranteed because data base keys may be made available to and be saved by run units and may be:

- Used for direct accessing.
- Referenced later in the execution of the same run unit.
- Re-input to a subsequent run unit in which they are referenced.

The mapping of data base keys onto media space is not specified in this Journal.

2.4.2.2.2 WITHIN

The WITHIN clause determines the areas into which occurrences of that record type may be stored. It also permits the precise area to be determined by a nominated data base procedure or by a run unit which stores the record. Once the area is determined the LOCATION clause will determine where in the area the record is to be placed.

2.4.2.2.3 LOCATION

The LOCATION clause allows control over the relative placement of records and over the algorithm used by the DBMS in assigning data base keys to records. Since data base keys uniquely identify records and are permanently assigned, the LOCATION clause can also control the retrieval process. There are various forms of the LOCATION clause. For example, a LOCATION clause may specify the record placement directly by providing the DBMS with the record's data base key, or may define the data base key from the data contents of the record.

2.4.3 SETS

2.4.3.1 CHARACTERISTICS OF SETS

The following is relevant to an understanding of the concept of a set:

- A set type is a named relationship between record types.
- An arbitrary number of set types may be declared in a schema.
- Each set type must be named and must have one owner record type. However, a special type of set which has exactly one occurrence and for which the DBMS is the owner may be declared. For convenience, this is known as a singular set.
- Each set type must have one or more member record types declared for it in the schema. This does not apply to set types specified to be dynamic - See Section 2.4.3.4.
- Each set type must have an order specified for it in the schema.
- Any record type may be declared in the schema as the owner of one or more set types.
- Any record type may be declared in the schema as a member of one or more set types.
- Any record type may be specified as both an owner of one or more set types and a member in one or more different set types.
- The capability for a record type to participate as both owner and member in the same set type is not supported by the DDL defined in this Journal.
- A set consists of an owner record and its member records if any.

- A record cannot be in more than one occurrence of the same set type.

- A set includes exactly one occurrence of its owner. In fact, the existence of the owner record in the data base establishes the set.

- A set which contains only an occurrence of its owner record is known as an empty set.

- A set may have an arbitrary number of occurrences of each of the member record types declared for it in the schema.

2.4.3.2 ORDERING OF SETS

Each set type declared in the schema must have an order specified for it. The effect of specifying an order for a set type is to cause the DBMS to insert member records into a set in such a way that the logical order defined for that set type is maintained. The logical order of the member records of a set is completely independent of the physical placement of the records. Thus, the same member records could participate in occurrences of two different sets and be ordered differently in each of those sets. Records which own sets may only be ordered in their capacity as members of other sets.

The member records of each occurrence of a given set type may be ordered in one of several ways:

- In ascending or descending sequence based on the values of specified keys. The keys specified may be data items in each of the member records, the member records' names or their data base keys, or some combinations of these.

- In the order resulting from inserting new member records into the set:

  a. First in the sequence of member records.

  b. Last in the sequence of member records.

  c. After or before another record which is selected by the run unit storing or inserting the record in the set.

- In the order most convenient to the DBMS.

2.4.3.3 INDEXED SETS AND SEARCH KEYS

Any set type declared to be sorted may also be declared to be indexed. This causes the DBMS to build an index, on the basis of the sort control keys specified, for each occurrence of that set type. No control is provided in the schema DDL over the development of an index; however, the index may be named. It is assumed that implementors of the DBMS will provide for such control.

An arbitrary number of search keys may also be declared for a set type regardless of whether it is sorted or not. The arguments for such search keys must be data items included in the member records of the set. The declaration of a search key causes the DBMS to develop some form of indexing of the member records for each set in which member records participate. The term indexing as used in this Journal means any technique which does not involve a complete scan of the member records involved. It is not restricted to an index in the usual sense. Some control over the type of indexing developed is provided in the schema DDL.

Where a set type has been declared to be indexed, or search keys have been specified for its members, functions or procedures which require a search to be performed on the basis of any argument for which indexing exists will automatically employ the available indexing.

2.4.3.4 SET MEMBERSHIP

A record type may have different kinds of membership in different set types.

Automatic or manual membership refers to the insertion of a member record into a set. Automatic means that membership in a set is established by the DBMS when a record is stored. That is, whenever an occurrence of a record type declared to be an automatic member of a set type is added to the data base, it will be inserted into (made a member of) the appropriate occurrences of all the set types in which it has been declared as an automatic member.

The addition to the data base of an occurrence of a record type declared to be a manual member of a set type will not cause it to be made a member of any occurrence of the set types in which it has been declared as a manual member. Manual means that membership in a set is established by a run unit by means of a DML function which causes the record, already stored in the data base, to become a member of a set.

Mandatory or optional membership refers to the removal of a member record from a set. Mandatory means that once a record becomes a member of any occurrence of a set type it will always be a member of one or another occurrence of that set type. The set of which the record is a member may be changed by using an appropriate DML function.

Optional means that membership of a record in a set is not necessarily permanent. A record may be removed, by using an appropriate DML function, from a set in which it is defined as an optional member.

It is not possible to access a record through a set from which it has been removed even though the record is still in the data base and is accessible via other sets.

Alternatively, any record may become a member of a dynamic set. A dynamic set differs from an ordinary set in the following respects.

- A dynamic set type must not have any member record types declared for it in the schema.
- Any record may be made a member of a dynamic set or removed from that set by executing an appropriate DML function.

2.4.3.5 MAINTENANCE OF SET RELATIONSHIPS

The housekeeping associated with the establishment and maintenance of sets is a responsibility of the DBMS.

Such action is required whenever:

- A record is added to or deleted from the data base.
- A record is inserted into or removed from a set.
- A record is modified in a way which changes its logical position within a set.
- A record is modified in a way which changes the set in which it participates.

Programmers are not involved in the mechanics of this process but may need to initialize with appropriate values those data items which are required by the DBMS to perform its functions. Such data items are named in the schema.

2.4.3.6 SET SELECTION

In general, there will be more than one set in the data base for each set type specified in the schema. It is therefore necessary to provide a means for identifying the proper set when member records of that set type are stored into or retrieved from the data base.

The SELECTION clause of the Member Subentry controls the strategy to be followed by the DBMS in selecting a specific set from the universe of sets in the data base. The DBMS automatically invokes the prespecified strategy whenever a DML function requires a specific set to be selected.

A separate SELECTION clause is required for each member record type/set type pair. The SELECTION clause provides for naming a series of sets which form a continous path through the data structure of the data base to the desired set. For each set on this path, the SELECTION clause names data items whose values are used by the DBMS to control the selection of specific sets.

For all sets along the path, other than the first named set, the DBMS limits its search to the member records of the previously selected set. Thus, it selects each set by selecting its owner record in its capacity as a member record of the previous set in the path. The occurrence of the first named set must be uniquely identifiable by the DBMS.

The SELECTION clause also provides for naming a data base procedure which performs the required selection.

2.4.4 PROTECTION OF DATA

The schema includes provision for the protection of data in the "social environment" of a shared data base. A shared data base is one which contains data relevant to many aspects of an organization's operations or one in which for any other reason the data is shared by multiple programs or applications. In this type of environment, two kinds of protection are required:

- Protection against unauthorized access of data for which the term privacy is used.

- Protection against inconsistent and unreasonable data for which the term integrity is used.

To some extent the mechanisms for providing privacy and ensuring the integrity of data overlap, but for the most part they are quite separate.

Of course, such protection cannot guard against unsocial behavior on the part of individual run units. If, for example, a run unit is authorized to access and delete a particular record and does so without regard for the fact that it is the owner of a set which has members required by other programs, the DBMS can offer little protection. Such action is really a logical error, and though intelligent design of the data base can minimize the effect, it can only be avoided by creating an awareness among programmers of their social responsibilities to other users of the data base.

2.4.4.1 PRIVACY OF DATA

Protection of the data in the data base is furnished through a mechanism of privacy locks which are specified in the schema and privacy keys which must be provided by a run unit seeking to access or alter data which is protected by means of privacy locks.

The schema DDL provides for declaring privacy locks at the schema, area, record, data item, data aggregate, set and member levels. At each of these levels, the schema DDL provides for locks for specific DML functions at that level.

A privacy key is a single value of implementor defined size and type, and may be a constant, the value of a variable, or the result of a procedure. A privacy lock is either a value (constant or variable), which is simply matched against a privacy key value, or a procedure, which is called and given access to the pertinent privacy key (see section 2.3.7, DATA BASE PROCEDURES). If the procedure returns, it gives a result of yes or no. Beyond that, the action of such a procedure is dependent on the implementor and the Data Administrator and is not specified in this Journal. Some possibilities are:

- Perform a calculation on the privacy key to check its validity.

- Open a conversation with a person at a terminal, to ask a number of questions before granting access.

- Write a new privacy key value in the run unit.

- In case of a violation, log the pertinent information and possibly send an alarm to a security console.

- In case of a repeated violation, abort the run unit.

- Suspend the run unit until access could be granted.

- Disconnect a user terminal, and call back before granting access.

Note that for many of these possibilities, the procedure must have knowledge of the identity of the run unit. The way in which this knowledge is passed to the procedure is not specified in this Journal.

While protection of privacy is probably the chief use of privacy keys and locks, it is by no means the only one. The mechanism can also be used, for example, to help ensure the consistency of inter related data and to prevent errors, by locking out clearly inconsistent, meaningless, or incorrect actions.

2.4.4.2 INTEGRITY OF DATA

The DDL provides for the specification of the data structure relationships which are to be maintained by and for all programs in such aspects as, for example, set membership. Provision is also made for checking the validity of a data item whenever a value is changed or a new value is stored in the data base. In addition there is provision for the naming of data base procedures and for causing the DBMS to invoke those procedures whenever a run unit attempts to update nominated records or sets. This, for example, enables the Data Administrator to check any update or series of updates applied to the data base.

Table of Contents

SECTION 3. THE DATA DESCRIPTION LANGUAGE Page

3.0 SYNTAX AND ENVIRONMENT

3.0.1 INTRODUCTION

This section contains the complete specifications of the data description language for writing the schema.

The schema written in the DDL consists of four entry types which serve to:

- Identify the schema (Schema Entry).
- Define areas (Area Entry).
- Define records (Record Entry).
- Define sets (Set Entry).

For each area, record type, and set type in the schema, a separate entry is required. There must be only one Schema Entry in the schema. The following rules apply to the sequence of the various entry types in the schema:

- The Schema Entry must be the first entry.
- An Area Entry must precede the Record Entry for each record type in that area.
- A Record Entry must precede the Set Entry for each record type in that set type.

An entry consists of one or more clauses which describe its attributes. In an entry describing a record or a set, clauses are grouped into subentries. Subentries may be repeated within an entry. Both entries and subentries are terminated by a period.

The specifications for an entry consist of the following:

- A narrative description of the function of the entry.
- An Entry Skeleton representing the organization of the entry into clauses, or into subentries as applicable. Where the Entry Skeleton consists of subentries a Subentry Skeleton is shown representing the organization of the subentry into clauses.
- The general formats of the entry, that is the general format of each of the clauses which may be specified in the entry.

- A separate description of each clause.

The description of each clause consists of the following:

- A narrative description of its function.
- Its general format.
- The syntax rules which apply.
- The general rules which apply.

A general format is the arrangement of the elements which make up a clause. Clauses in an entry or subentry may be written in any sequence provided the first clause names the entry or subentry. A syntax rule amplifies or restricts the usage of the elements within a general format. A general rule amplifies or restricts functions attributed to a general format or to its constituent elements.

The notation used in all formats and the rules which apply to all formats are:

- The elements which make up a clause consist of upper case words, lower case words, special symbols and special characters.

- All underlined upper case words are required when the format is used.

- Upper case words which are not underlined are optional words and need not be used.

- Lower case words are generic terms which must be replaced by appropriate names or values.

- The meaning of enclosing a portion of a general format in special symbols as follows is:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \begin{array}{l} \text{at least no occurrences} \\ \\ \text{at most one occurrence} \end{array}$$

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \quad \begin{array}{l} \text{at least one occurrence} \\ \\ \text{at most one occurrence} \end{array}$$

$$\left\| \begin{matrix} a \\ b \\ c \end{matrix} \right\| \quad \begin{array}{l} \text{at least one occurrence} \\ \\ \text{at most one occurrence of each} \end{array}$$

An ellipsis (that is, '...') indicates repetition is allowed. The portion of the format which may be repeated is determined by the '[' or '{' which logically matches the ']' or '}' to the immediate left of the '...'.

3.0.2 CHARACTER SET

The character set for the DDL consists of 51 characters which include letters of the alphabet, digits and symbols. Five of these symbols have been reserved for future use. The specification of this character set defines those characters which may be used in writing a schema (but see paragraphs 3.0.3 and 3.0.4). It is in no way intended to restrict the use within the data base of any allowable characters in the character set defined by the implementor. The character set for the DDL includes:

Character	Name
0, 1, ..., 9	digits
A, B, ..., Z	letters
	space or blank
+	plus sign
-	minus sign or hyphen
,	comma
;	semicolon
.	period or decimal point
"	quotation mark
(	left parenthesis
)	right parenthesis
$	dollar sign

The reserved symbols are:

=	equals sign
>	greater than symbol
<	less than symbol
/	stroke
*	asterisk

3.0.3 WORDS

A word is a sequence of not more than 30 characters. Each character is selected from the set 'A' ... 'Z', '0' ... '9', '-' except that the '-' may not appear as the first or last character of a word.

Reserved Words

Reserved words are a list of words that may be used, but must not appear as user defined names. The types of reserved words are described below.

- A key word is a word that is required when the format in which the word appears is used. Within each format, such words are upper case and underlined.

- Within each format, upper case words that are not underlined are called optional words and may appear at the user's option. The presence or absence of each optional word does not alter the translation. Misspelling of an optional word, or its replacement by another word of any kind, is not allowed.

- The following is a list of reserved words with their abbreviations enclosed in parentheses:

ACTUAL
ALL
ALLOWED
ALTER
ALWAYS
AND
ARE
AREA
AREA-ID
ASCENDING (ASC)
AUTOMATIC (AUTO)
BINARY (BIN)
BIT
BY
CALC
CALC-KEY
CALL
CHARACTER (CHAR)
CHECK
CLOSE
COMMENT
COMPLEX
COPY
CURRENT
DATA-EASE-KEY (DBKEY)
DECIMAL (DEC)
DECODING
DEFINED
DELETE
DESCENDING (DESC)
DIRECT
DISPLAY
DUPLICATES (DUP)
DURING
DYNAMIC
ENCODING
EQUAL

FIND
FIRST
FIXED
FLOAT
FOR
GET
IDENTIFIED
IMMATERIAL
IN
INDEX
INDEXED
INSERT
INSERTION
IS
KEY
KEYS
LAST
LINKED
LOCATION (LOC)
LOCK
LOCKS
MANDATORY (MAND)
MANUAL
MEMBER
MEMBERS
MODE
MODIFY
NAME
NEXT
NONEXCLUSIVE (NEXCL)
NOT
NULL
OCCURS
OF
ON
OPEN
OPTIONAL (OPT)

ORDER
OWNER
PERMANENT
PICTURE (PIC)
PRIOR
PRIVACY
PROCEDURE (PROC)
PROCESSABLE
PROTECTED (PROT)
RANGE
REAL
RECORD
RECORD-NAME
REMOVE
RESULT
RETRIEVAL (RETR)
SCHEMA
SEARCH
SELECTION
SET
SORTED
SOURCE
STORE
SYSTEM
TEMPORARY (TEMP)
THEN
THIS
THRU
TIMES
TO
TYPE
UPDATE
USING
VALUE
VIA
VIRTUAL
WHERE

ERROR
EXCLUSIVE (EXCL)
OR
WITHIN

Names

A name may be in either of two forms. In the normal form, a name is a word beginning with a letter. In the alternate, or escape form, a name is a string of any allowable characters in the computer's character set, delimited by the dollar sign. The string may include the dollar sign symbol itself if the symbol is written twice consecutively for each of its occurrences.

Types of names include:

- Data-base-data-name
- Record-name
- Area-name
- Set-name
- Lock-name
- Index-name
- Data-base-procedure
- Support-function
- Implementor-name
- Schema-name

3.0.4 LITERALS

A literal is a string of characters whose value is implied by the ordered set of characters of which the literal is composed. Every literal belongs to one of two types, numeric and nonnumeric.

Nonnumeric Literals

A nonnumeric literal is defined as a string of any allowable characters in the computer's character set, of any length, delimited by quotation marks. This includes the quotation mark itself which, however, must be written twice consecutively for each of its occurrences within the string. The value of a nonnumeric literal is the string of characters itself, excluding the delimiting quotation marks. Any spaces enclosed in the quotation marks are characters rather than separators; each such space is part of the nonnumeric literal and is part of its value.

Numeric Literals

A numeric literal is defined as a string of characters chosen from the digits '0' through '9', the plus sign, the minus sign, the decimal point, and the character 'E'. Numeric literals may be expressed in two forms, fixed point and floating point decimal. The rules for formation of numeric literals are as follows:

- A literal must contain at least one digit.

- A fixed point literal must not contain more than one sign character. If a sign is used, it must appear as the leftmost character of the literal. If the literal is unsigned, the literal is positive.

- A fixed point literal must not contain more than one decimal point. The decimal point may appear anywhere within the literal except as the rightmost character. If the literal contains no decimal point, the literal is an integer.

- The word 'integer' appearing in a general format represents a numeric literal containing neither the decimal point nor the character 'E'.

- The character 'E' may only be be used with floating point decimal literals.

- A floating point literal consists of two fixed point numeric literals separated by the character 'E'. The first literal (mantissa) may contain a decimal point; the second literal (exponent) must be an integer.

- If a literal conforms to the rules for the formation of numeric literals, but is enclosed in quotation marks, it is a nonnumeric literal.

- The value of a numeric literal is the algebraic decimal quantity represented by the characters in the numeric literal.

Literal Constant: NULL.

A data item may have a value of null, meaning that either its value has never been otherwise specified, or that its value is irrelevant. The representation of the NULL value is not defined in this Journal.

3.0.5 COMMENTS

Comments may be included for documentation purposes. They must be introduced by the reserved word COMMENT and the comment is delimited by the quotation mark character. Comments may appear wherever the blank character may be used as a separator. No blank character may appear between the word COMMENT and the first quotation mark. The quotation mark character must be written twice consecutively if it is to be included in a comment.

3.0.6 PUNCTUATION

The following punctuation characters are used.

- One or more consecutive spaces, when not contained in a comment or delimited string, is a separator.

- The comma is used as a separator.

- The period is a delimiter for an entry or subentry and is required.

- A semicolon may be used to separate clauses.

3.0.7 DATA-BASE-DATA-NAMES

A data-base-data-name is a user defined word that names a data item or data aggregate. When used in a general format 'data-base-data-name' may not be subscripted or qualified unless specifically permitted by the rules for that format. The named data item or data aggregate need not be the subject of a Data Subentry.

3.0.8 DATA-BASE-IDENTIFIERS

A data-base-identifier is a reference to a data item or data aggregate declared in the schema. It consists of a data-base-data-name followed, as required, by the syntactically correct combination of subscripts and qualifiers necessary to achieve uniqueness of reference.

Qualification

Where the same data-base-data-name is declared in more than one Record Entry, its use as a data-base-identifier may have to be qualified to achieve uniqueness. Syntax rules will specify when qualification is necessary. A name can be qualified even though it does not need qualification.

Subscripting

Subscripts can be used only when reference is made to a data item within a data aggregate. The subscript must be an integer.

The lowest possible subscript value is 1. This value refers to the first occurrence of the data-base-data-name referenced. When more than one subscript is required they are written left to right in the order of increasing data aggregate level numbers.

Format

The format of a data-base-identifier in the DDL for the schema is:

$$\text{data-base-data-name}[\text{integer-1}[,\text{integer-2}]\ldots]\left[\begin{Bmatrix}\underline{\text{OF}}\\ \underline{\text{IN}}\end{Bmatrix}\ \text{record-name}\right]$$

3.0.9 SYSTEM ENVIRONMENT

As mentioned previously in the Concepts Section, a DBMS based upon the concepts and language specified herein should include various capabilities to permit the Data Administrator to organize, monitor, reorganize and restructure the data base as it evolves with changing requirements. In addition the Data Administrator requires certain facilities to enable him to use, maintain and develop each schema under his control. These specifications presume the existence of the following generic schema operations for this purpose:

a. ALTER

This operation permits the alteration of all the schema with the exception of the privacy lock clauses.

b. COPY

This operation permits the extraction of information from the schema for the purpose of constructing a subschema.

c. DISPLAY

This operation permits viewing of the schema with the exception of the privacy locks.

d. LOCKS

This operation allows the viewing, creating, or changing of privacy locks.

3.0.10 DDL REFERENCES TO DML FUNCTIONS

In the specification of the DDL it is assumed that certain basic functions may be performed on the described data. This results in various interactions between data descriptions and data manipulation functions. Where relevant, these interactions are indicated. The basic DML functions assumed are of the following generic types:

Update Functions

a. STORE, DELETE:

a record into (from) the data base.

b. INSERT, REMOVE:

a record into (from) a set.

c. MODIFY:

that is, change data in a record.

d. ORDER:

that is, logically reorder the records in a set.

Retrieval Functions

a. FIND:

that is, locate a specific record in the data base.

b. GET:

that is, fetch data from a record.

Control Functions

a. OPEN, CLOSE:

areas of the data base for (from) user processing in retrieval or update modes.

The DDL specifications also assume that the open function can be qualified so as to restrict the availability of areas to concurrent run units. Thus an exclusive open dedicates an area to a run unit and a protected open permits concurrent run units to acquire only retrieval privileges. A nonexclusive open places no restrictions on concurrent run units.

The basic DML functions listed above may in any specific DML be combined to form more complex functions. For example FIND and GET could be combined to form a READ or FETCH function, INSERT and REMOVE could be combined to form a SWITCH sets function and FIND could be combined with any of the update functions. Other variations are also possible.

3.1.0 SCHEMA ENTRY

Function

To name and associate certain facilities with the schema which describes the data base.

Schema Entry Skeleton

SCHEMA clause

ON clause

PRIVACY clause

General Format of Entry

$$\underline{\text{SCHEMA}}\ \text{NAME IS schema-name-1}$$

$$\left[\ \text{;}\underline{\text{ON}}\ [\underline{\text{ERROR}}\ \text{DURING}]\ \left[\left\|\begin{array}{l}\underline{\text{ALTER}}\\ \underline{\text{COPY}}\\ \underline{\text{DISPLAY}}\\ \underline{\text{LOCKS}}\end{array}\right\|\right]\ \underline{\text{CALL}}\ \text{data-base-procedure-1}\ \right]\ldots$$

$$\begin{array}{l}\left[\ \text{;}\underline{\text{PRIVACY}}\ \text{LOCK}\ \left[\text{FOR}\ \left\|\begin{array}{l}\underline{\text{LOCKS}}\\ \underline{\text{DISPLAY}}\\ \underline{\text{COPY}}\\ \underline{\text{ALTER}}\end{array}\right\|\right]\ \text{IS}\ \left\{\begin{array}{l}\text{literal-1}\\ \text{lock-name-1}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-2}\end{array}\right\}\right.\\ \qquad\left.\left[\underline{\text{OR}}\ \left\{\begin{array}{l}\text{literal-2}\\ \text{lock-name-2}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-3}\end{array}\right\}\right]\ldots\ \right]\ldots\end{array}$$

3.1.1 ON

Function

To specify the procedure to be executed whenever a schema operation is performed.

General Format

$$\underline{\text{ON}}\ [\underline{\text{ERROR}}\ \text{DURING}] \left[\left\| \begin{array}{l} \underline{\text{ALTER}} \\ \underline{\text{COPY}} \\ \underline{\text{DISPLAY}} \\ \underline{\text{LOCKS}} \end{array} \right\|\right]\ \underline{\text{CALL}}\ \text{data-base-procedure-1}$$

Syntax Rules

1. A separate ON clause may be written for each schema operation.

2. The same data-base-procedure may be specified in different ON clauses.

General Rules

1. The procedure named by data-base-procedure-1 is invoked as soon as the specified schema operation is performed. If more than one procedure is applicable they are invoked in the order in which they are specified in the Schema Entry, but a procedure named in an ON clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an ON ERROR clause will be entered only if, during the performance of the schema operation, the DBMS detects an error that it intends to report.

2. If no schema operations are specified the procedure is invoked whenever any of the listed schema operations is performed.

3.1.2 PRIVACY

Function

To specify the privacy lock(s) for certain operations which apply to the schema.

General Format

```
                    ┌      ║LOCKS  ║┐      ┌literal-1                        ┐
PRIVACY LOCK        │FOR   ║DISPLAY║│  IS {  lock-name-1                      }
                    │      ║COPY   ║│      └PROCEDURE data-base-procedure-1  ┘
                    └      ║ALTER  ║┘

     ┌   ┌literal-2                        ┐┐
     │OR{  lock-name-2                      }│ ...
     └   └PROCEDURE data-base-procedure-2  ┘┘
```

Syntax Rules

1. A separate PRIVACY clause may be stated for each restricted schema operation (LOCKS, DISPLAY, COPY, ALTER). However, the same operation must not be specified in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more operations.

3. All literals must conform to the implementor defined data characteristics of privacy locks.

General Rules

1. The literals and the content of the lock-names are privacy locks, to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which, when given access to a privacy key, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR clause is omitted all literals, lock-names, or procedures apply to all operations.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the PRIVACY clause is known.

6. If no PRIVACY clause has been specified for an operation, then the use of that operation is without restriction.

7. The privacy locks associated with each restricted operation (ALTER, COPY, ...) must be satisfied in order to perform that restricted operation on the schema.

3.1.3 SCHEMA

Function

To name the schema.

General Format

SCHEMA NAME IS schema-name-1

Syntax Rules

1. Schema-name-1 must be unique among the schema-names known to the DBMS.

General Rules

1. The schema named by schema-name-1 consists of the DDL entries that appear after the SCHEMA clause and before an implementor defined 'END SCHEMA' indicator.

3.2.0 AREA ENTRY

Function

To name and give certain characteristics of an area within the data base.

Area Entry Skeleton

AREA clause

  TEMPORARY clause

  ON clause

  PRIVACY clause

General Format of Entry

See next page

General Format of Area Entry

```
AREA NAME IS area-name-1

[;AREA IS TEMPORARY]

                   ┌║ ┌   ┌ ┌║EXCLUSIVE    ║ ┌UPDATE   ┐┐    ┐┐║┐
                   │║ │   │ │║PROTECTED    ║ └RETRIEVAL┘│ ...││║│
[;ON [ERROR DURING]│║OPEN│FOR│ └║NONEXCLUSIVE ║            │    ││║│
                   │║ │   │  ║EXCLUSIVE    ║               ││║│
                   │║ │   │  ║PROTECTED    ║               ││║│
                   │║ └   └  ║NONEXCLUSIVE ║               ┘┘║│
                   │║CLOSE                                      ║│
                   └                                             ┘

     CALL data-base-procedure-1] ...


                          ┌  ║[EXCLUSIVE]                                ║┐
                          │  ║[PROTECTED] RETRIEVAL                      ║│
[;PRIVACY LOCK FOR │  ║[EXCLUSIVE]                                ║│ IS
                          │  ║[PROTECTED] UPDATE                          ║│
                          └  ║ support-function-1[,support-function-2]...║┘

    ┌literal-1                        ┐ [    ┌literal-2                        ┐]   ]
    │lock-name-1                      │ [ OR │lock-name-2                      │] ...] ...
    └PROCEDURE data-base-procedure-2┘ [    └PROCEDURE data-base-procedure-3┘]   ]
```

3.2.1 AREA

Function

To name an area within the data base.

General Format

AREA NAME IS area-name-1

Syntax Rules

1. Area-name-1 must be unique among area-names within the schema.

2. At least one area-name must be specified in a schema.

General Rules

1. If only one area-name is specified in the schema, then the area named by that area-name and the data base are equivalent.

3.2.2 ON

Function

To specify the procedure to be executed when an area is opened or closed.

General Format

$$\underline{\text{ON}}\ [\underline{\text{ERROR}}\ \text{DURING}]\ \left[\left\|\begin{array}{l} \underline{\text{OPEN}}\ \left[\text{FOR}\left\{\begin{array}{l} \left\{\left\|\begin{array}{l}\underline{\text{EXCLUSIVE}}\\ \underline{\text{PROTECTED}}\\ \underline{\text{NONEXCLUSIVE}}\end{array}\right\|\ \left\{\begin{array}{l}\underline{\text{UPDATE}}\\ \underline{\text{RETRIEVAL}}\end{array}\right\}\right\}\ \ldots \\ \left\|\begin{array}{l}\underline{\text{EXCLUSIVE}}\\ \underline{\text{PROTECTED}}\\ \underline{\text{NONEXCLUSIVE}}\end{array}\right\| \end{array}\right\}\right] \\ \underline{\text{CLOSE}} \end{array}\right\|\right]$$

$$\underline{\text{CALL}}\ \text{data-base-procedure-1}$$

Syntax Rules

1. A separate ON clause may be written for each type of OPEN and for the CLOSE function.

2. The same data-base-procedure may be specified in different ON clauses.

3. In the optional FOR phrase, each of the words UPDATE and RETRIEVAL may appear at most cnce.

4. The FOR phrase must not be used in an entry for a temporary area.

General Rules

1. The procedure named by data-base-procedure-1 is invoked whenever the specified function is executed on the area in which this clause appears.

2. If OPEN or CLOSE is not specified, the procedure is invoked whenever any OPEN or CLOSE function is executed for the area.

3. If OPEN is specified and the FOR option is not stated, the procedure is invoked whenever any type of OPEN function is executed for the area.

4. If the FOR option is stated, but UPDATE or RETRIEVAL is not specified, the procedure is invoked whenever the specified type of OPEN function is executed regardless of whether the open is for update or retrieval.

5. If the FOR option is stated and UPDATE or RETRIEVAL or both are also specified, then any included qualifier words (that is, EXCLUSIVE, PROTECTED or NONEXCLUSIVE) apply only to the first following use of the words UPDATE or RETRIEVAL.

6. The procedure is invoked immediately before control is returned to the run unit. If more than one procedure applies to the execution of a function, the procedures are invoked in the order in which they are stated in the schema, but a procedure named in an ON clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an ON ERROR clause will be entered only if, during the performance of the specified function, the DBMS detects an error that it intends to report.

3.2.3 PRIVACY

Function

To specify the privacy locks which apply to the use of an area.

General Format

```
                       ┌     ║┌EXCLUSIVE┐                                ║┐
                       │     ║└PROTECTED┘ RETRIEVAL                      ║│
                       │     ║                                           ║│
PRIVACY LOCK           │FOR  ║┌EXCLUSIVE┐                                ║│ IS
                       │     ║└PROTECTED┘ UPDATE                         ║│
                       │     ║                                           ║│
                       └     ║ support-function-1[, support-function-2]...║┘

      ┌literal-1                          ┐
      │lock-name-1                        │
      └PROCEDURE data-base-procedure-1    ┘

      ┌    ┌literal-2                        ┐┐
      │OR  │lock-name-2                      ││...
      └    └PROCEDURE data-base-procedure-2  ┘┘
```

Syntax Rules

1. A separate PRIVACY clause may be stated for each usage mode (EXCLUSIVE or PROTECTED RETRIEVAL or UPDATE, or nonrestricted RETRIEVAL or UPDATE) and/or for the various support-functions. However, the same usage mode or support-function must not be specified in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more options included in this clause.

3. All literals must conform to the implementor defined data characteristics of privacy locks.

4. Support-functions are implementor defined names for the utility functions of, for example, loading, copying, patching, or dumping a data base.

General Rules

1. The literals and the content of the lock-names are privacy locks, to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which, when given access to a privacy key, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR phrase is omitted all literals, lock-names, or procedures apply to any use of the area.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the PRIVACY clause is known.

6. If a PRIVACY clause has not been specified for a given usage mode or support-function, then the use of that usage mode or support-function on the area being described, is without restriction.

7. The privacy locks associated with the EXCLUSIVE RETRIEVAL, PROTECTED RETRIEVAL, RETRIEVAL, EXCLUSIVE UPDATE, PROTECTED UPDATE, and UPDATE clauses must be satisfied by a run unit to enable it to open the area with the corresponding usage mode.

8. The privacy locks associated with the support-function names must be satisfied in order to execute the specified support-function on the area being described.

3.2.4 TEMPORARY

Function

To specify that the area is temporary.

General Format

AREA IS TEMPORARY

Syntax Rules

None

General Rules

1. A temporary area is not shared among concurrent run units. Any run unit which makes reference to an area defined as temporary is allocated a private, unique occurrence of that area. This is true even when multiple run units refer to the same area-name.

2. When a CLOSE function is executed on a temporary area or the run unit terminates, records and sets in the area are no lcnger accessible and the space occupied by the temporary area may be made available for re-use by the DBMS.

3. Records in a temporary area cannot participate, either as owner or member records, in sets which contain records that are not in temporary areas.

3.3.0 RECORD ENTRY

Function

To name and give certain characteristics of records and their subordinate data items within a data base.

Record Entry Skeleton

```
Record Subentry

   [Data Subentry] ...
```

Record Subentry Skeleton

```
RECORD clause

    LOCATION clause

    WITHIN clause

    ON clause

    PRIVACY clause
```

Data Subentry Skeleton

```
data-base-data-name clause

    PICTURE clause

    TYPE clause

    OCCURS clause

    RESULT clause

    SOURCE clause

    CHECK clause

    ENCODING/DECODING clause

    ON clause

    PRIVACY clause
```

General Format of Record Subentry

```
RECORD NAME IS record-name-1

                   ⎧DIRECT ⎧data-base-data-name-1 ⎫                                           ⎫
                   ⎪       ⎩data-base-identifier-1⎭                                           ⎪
                   ⎪                                                                          ⎪
;LOCATION MODE IS  ⎨CALC [data-base-procedure-1] USING data-base-identifier-2                 ⎬
                   ⎪     [,data-base-identifier-3]...DUPLICATES ARE [NOT] ALLOWED             ⎪
                   ⎪                                                                          ⎪
                   ⎪VIA set-name-1 SET                                                        ⎪
                   ⎪                                                                          ⎪
                   ⎩SYSTEM                                                                    ⎭

         ⎧area-name-1 [{,area-name-2}... AREA-ID IS data-base-data-name-2   ⎫
;WITHIN  ⎨                 [USING PROCEDURE data-base-procedure-2]]         ⎬
         ⎩AREA OF OWNER                                                      ⎭

                          ⎡⎡INSERT⎤⎤
                          ⎢⎢REMOVE⎥⎥
 ⎡                        ⎢⎢STORE ⎥⎥                                  ⎤
 ⎢;ON [ERROR DURING]      ⎢⎢DELETE⎥⎥ CALL data-base-procedure-3        ⎥ ...
 ⎣                        ⎢⎢MODIFY⎥⎥                                  ⎦
                          ⎢⎢FIND  ⎥⎥
                          ⎣⎣GET   ⎦⎦

                    ⎡        ⎡⎡INSERT⎤⎤
                    ⎢        ⎢⎢REMOVE⎥⎥
 ⎡                  ⎢        ⎢⎢STORE ⎥⎥    ⎧literal-1                          ⎫
 ⎢;PRIVACY LOCK     ⎢FOR     ⎢⎢DELETE⎥⎥ IS ⎨lock-name-1                        ⎬
 ⎣                  ⎢        ⎢⎢MODIFY⎥⎥    ⎩PROCEDURE data-base-procedure-4    ⎭
                    ⎢        ⎢⎢FIND  ⎥⎥
                    ⎣        ⎣⎣GET   ⎦⎦

       ⎡   ⎧literal-2                          ⎫⎤     ⎤
       ⎢OR ⎨lock-name-2                        ⎬⎥ ... ⎥...
       ⎣   ⎩PROCEDURE data-base-procedure-5    ⎭⎦     ⎦
```

General Format of Data Subentry

```
 [level-number-1] data-base-data-name-1

[;PICTURE IS "{character-string-picture-specification-1}"]
             {numeric-picture-specification-1         }

              {||{BINARY }||                                }
              {||{DECIMAL}||                                }
              {||{FIXED  }||  [integer-1[,integer-2]]       }
              {||{FLOAT  }||                                }
              {||{REAL   }||                                }
              {||{COMPLEX}||                                }
[;TYPE IS     {                                             } ]
              { {BIT      } [integer-3]                     }
              { {CHARACTER}                                 }
              {                                             }
              {  DATA-BASE-KEY                              }
              {                                             }
              {  implementor-name                           }

[;OCCURS {integer-4                } TIMES]
         {data-base-identifier-1   }

[;IS {ACTUAL } RESULT OF data-base-procedure-1
     {VIRTUAL}

   ||  ON THIS RECORD                              [                                    ]||
   ||                                              [                                    ]||
   ||  ON ALL MEMBERS OF set-name-1                [                                    ]||
   ||                   [,set-name-2]...           [ USING data-base-identifier-2       ]|| ]
   ||                                              [     [,data-base-identifier-3]...   ]||
   ||  {ON record-name-1                }          [                                    ]||
   ||  {    [,record-name-2]...         }...       [                                    ]||
   ||  {    OF set-name-3               }          [                                    ]||

[;IS {ACTUAL } AND SOURCE IS data-base-identifier-4 OF OWNER OF set-name-4]
     {VIRTUAL}
```

General Format of Data Subentry (continued)

```
[                ||PICTURE                                                 ||]
[                ||                                                        ||]
[ ;CHECK IS      ||data-base-procedure-2                                   ||]
[                ||                                                        ||]
[                ||VALUE [NOT] literal-1 [THRU literal-2]                  ||]
[                ||              [,literal-3 [THRU literal-4]] . . .       ||]


[ ;FOR {ENCODING} [ALWAYS] CALL data-base-procedure-3 ] ...
[      {DECODING}                                     ]


[                       [||STORE ||]                                  ]
[ ;ON [ERROR DURING]    [||GET   ||]  CALL data-base-procedure-4      ] ...
[                       [||MODIFY||]                                  ]


[                    [    ||STORE ||]     {literal-5                          }
[ ;PRIVACY LOCK      [FOR ||GET   ||] IS  {lock-name-1                        }
[                    [    ||MODIFY||]     {PROCEDURE data-base-procedure-5    }
                                                                               ]
      [    {literal-6                          }]                              ]
      [ OR {lock-name-2                        }] ...                          ] ...
      [    {PROCEDURE data-base-procedure-6    }]                              ]
```

3.3.1 CHECK

Function

To inhibit data conversion or to specify a validity-checking procedure to be executed whenever a value is changed or added to the data base.

General Format

```
           ||PICTURE                                               ||
           ||                                                      ||
CHECK IS   ||data-base-procedure-1                                 ||
           ||                                                      ||
           ||VALUE [NOT] literal-1 [THRU literal-2]                ||
           ||                [, literal-3  [THRU literal-4]] . . . ||
```

Syntax Rules

1. Literals specified in the VALUE option must appear in ascending order according to the implementor's collating sequence.

2. The subject of the CHECK clause must be described with a PICTURE, TYPE, or SOURCE clause.

General Rules

1. If PICTURE is specified, data conversion will not occur. Therefore, the characteristics of the data item as described in the subschema must match the characteristics of the data item as described in the schema.

2. If VALUE is specified, the value of the data item in the user working area is checked against the individual values or ranges specified. An individual value is specified by a literal. A range is specified by two literals separated by THRU. The value of the data item satisfies the VALUE option if it is equal to a specified literal, or if it is greater than or equal to the literal on the low end of a range and less than or equal to the literal on the high end of a range. The value is valid if it satisfies the VALUE option and NOT is omitted or if it does not satisfy the VALUE option and NOT is included.

3. If data-base-procedure-1 is specified, validity checking is performed by the named procedure.

4. If a CHECK clause is used with the VALUE or data-base-procedure-1 options, and the data item which is the subject of the CHECK clause is neither an actual or virtual result data item nor an actual or virtual source data item, the specified validity checking occurs whenever

a new value of the data item is placed in the data base as a result of a STORE function or whenever a value of the data item is changed as a result of a MODIFY function.

5. If a CHECK clause is used with the VALUE or data-base-procedure-1 options, and the data item which is the subject of the CHECK clause is a virtual result or source data item, the specified validity checking occurs whenever a GET function involving the data item is performed. If the data item is an actual result or source data item, the specified validity checking occurs whenever the data item is updated. The rules describing the conditions under which an actual result or source data item is updated are included in the general rules of the RESULT or SOURCE clauses.

6. If the CHECK clause is used with both the VALUE and data-base-procedure-1 options, the VALUE check is performed first. If the value of the data item is not valid, the procedure is not invoked.

7. If the data item is also the subject of other procedures, such as a DBMS conversion routine or the data base procedure specified in a RESULT clause, these procedures are invoked before validity checking occurs. A data base procedure specified in an ON clause for the data item is invoked after validity checking.

8. If an invalid value is detected the DBMS will report an error.

3.3.2 DATA-BASE-DATA-NAME

Function

To name a data item or data aggregate and indicate its structural level within the record.

General Format

[level-number-1] data-base-data-name-1

Syntax Rules

1. Data-base-data-name-1 must be unique among the data-base-data-names declared for this record type.

2. Level-number-1 is an unsigned decimal integer greater than 0 and less than 100.

3. A Data Subentry must include exactly one of the following:

   a. A PICTURE clause.
   b. A TYPE clause.
   c. A SOURCE clause.
   d. An OCCURS clause.
   e. An OCCURS clause and a PICTURE clause.
   f. An OCCURS clause and a TYPE clause.

   In addition it may include any other clauses appearing in the General Format for a Data Subentry unless such use is explicitly prohibited in the rules of those clauses.

4. If and only if a Data Subentry includes an OCCURS clause, but neither a PICTURE nor a TYPE clause, the subentry must be followed by one or more subentries with higher valued level-numbers.

General Rules

1. A data-base-data-name followed by one or more clauses constitutes a Data Subentry. The content of a record is defined by a series of zero or more Data Subentries.

2. A Data Subentry names and describes a data item, vector or repeating group. Additional subentries are required to name and describe the components of a repeating group.

3. If level-number-1 is not specified, level 1 is assumed.

4. A data item is described by a Data Subentry that includes a PICTURE, TYPE, or SOURCE clause, but no OCCURS clause.

5. A vector is described by a Data Subentry that includes an OCCURS clause and either a PICTURE or TYPE clause.

6. A repeating group is described by a Data Subentry that includes an OCCURS clause, but no PICTURE or TYPE clause. The components of the repeating group, which may, in turn, be repeating groups, are described by subsequent subentries.

3.3.3 ENCODING/DECODING

Function

To specify the procedure to be executed whenever a data item requiring special conversion is retrieved or updated.

General Format

$$\text{FOR } \left\{\begin{array}{l}\underline{\text{ENCODING}}\\ \underline{\text{DECODING}}\end{array}\right\} \text{ [}\underline{\text{ALWAYS}}\text{] } \underline{\text{CALL}} \text{ data-base-procedure-1}$$

Syntax Rules

1. An ENCODING and a DECODING clause may be used for the same data item.

2. The subject of the ENCODING/DECODING clause must be described with a PICTURE or TYPE clause.

3. The subject of an ENCODING clause must not be described with a VIRTUAL RESULT clause.

General Rules

1. The procedure named by data-base-procedure-1 is invoked in lieu of a standard conversion. Therefore, the procedure is invoked at the point in the execution of a DML function that the standard conversion would be invoked. If the optional word ALWAYS is not used, the procedure is invoked only if the characteristics of the data item vary between the schema and subschema.

2. If ENCODING is specified, the procedure is invoked when a new value of the data item is placed in the data base as a result of a STORE function or when a value of the data item is changed as a result of a MODIFY function. The procedure is passed the data item in the form in which it appears in the user working area. The result of the procedure is the data item in the form in which it is stored in the data base.

3. If DECODING is specified, the procedure is invoked when a GET function is executed for the data item. The procedure is passed the data item in the form in which it is stored in the data base. The result of the procedure is the data item in the form in which it appears in the user working area.

3.3.4 LOCATION

Function

To control the assignment by the DBMS of data base keys to records.

General Format

```
                   ⎧ DIRECT ⎧data-base-data-name-1 ⎫                                              ⎫
                   ⎪        ⎩data-base-identifier-1⎭                                              ⎪
                   ⎪                                                                              ⎪
                   ⎪ CALC [data-base-procedure-1] USING data-base-identifier-2                    ⎪
LOCATION MODE IS   ⎨      [,data-base-identifier-3] ...DUPLICATES ARE [NOT] ALLOWED               ⎬
                   ⎪                                                                              ⎪
                   ⎪ VIA set-name-1 SET                                                           ⎪
                   ⎪                                                                              ⎪
                   ⎩ SYSTEM                                                                       ⎭
```

Syntax Rules

1. Data-base-identifier-1 must be qualified with a record-name and must refer to a data item defined as a data base key.

2. Data-base-identifier-2, data-base-identifier-3,... must refer to data items included in the record type being described.

3. Set-name-1 must be a set type in which the record type is defined as being a member.

General Rules

1. By its appearance in a LOCATION clause data-base-data-name-1 is treated as a data base key and is not part of a record.

2. The DBMS assigns a data base key to a record when a DML function which stores the record in the data base is executed.

3. The assignment of data base keys is subject to the overall constraint of the WITHIN clause for the same Record Subentry. The same data base key is not assigned to more than one record in the data base.

4. If SYSTEM is specified the DBMS uses an implementor defined method of assigning a data base key.

5. If DIRECT is specified the contents at execution time of the data item associated with the DIRECT phrase is used by the DBMS in assigning a data base key. The contents must be a data base key or a null value.

6. If CALC is specified the contents at execution time of the data item(s) associated with the CALC phrase are used by the DBMS in assigning a data base key. If data-base-procedure-1 is not specified the DBMS develops a data base key using its standard key transformation algorithm. If data-base-procedure-1 is specified the DBMS develops a data base key using the named procedure.

7. If the DUPLICATES ARE ALLOWED phrase is specified the DBMS will permit more than one record with identical values for the data items associated with the CALC phrase to be stored in the data base.

8. If the DUPLICATES ARE NOT ALLOWED phrase is specified the DBMS will not permit more than one record with identical values for the data items associated with the CALC phrase to be stored in the data base.

9. If VIA set-name-1 SET is specified the DBMS assigns a data base key to the object record as though it were to become a member of an occurrence of the set type named in the VIA phrase. The set selection and set ordering criteria defined for the set type named are consulted by the DBMS when assigning a data base key.

3.3.5 OCCURS

Function

To define a vector or repeating group by specifying the number of times the data item or group occurs within a record.

General Format

$$\underline{\text{OCCURS}} \left\{ \begin{array}{l} \text{integer-1} \\ \text{data-base-identifier-1} \end{array} \right\} \text{TIMES}$$

Syntax Rules

1. The OCCURS clause cannot be used in the same subentry as a RESULT or SOURCE clause.

2. The value of integer-1 must be greater than 0.

3. The data item referred to by data-base-identifier-1 must be previously defined as an integer in the same record type as the subject data aggregate.

4. Data-base-identifier-1 can be used only if the subject data aggregate is not a component of a repeating group.

5. Data-base-identifier-1 cannot refer to a data item which is part of a data aggregate defined with an OCCURS data-base-identifier-1 clause.

General Rules

1. The OCCURS clause is used to describe a data aggregate. The component elements of a vector are described by a PICTURE clause or a TYPE clause in the same subentry at the OCCURS clause. The components of a repeating group are described by subsequent subentries.

2. Integer-1 or the value of data-base-identifier-1 specifies the number of occurrences of the data item or group of data items. The use of integer-1 indicates that the number of occurrences is the same for all records. The use of data-base-identifier-1 defines a data item or group of data items of a variable number of occurrences. For a given record, the number of occurrences of the data item or group of data items is given by the value of data-base-identifier-1

in that record. The value must be a positive integer or zero.

3. The components of a repeating group are specified by means of subsequent subentries which have a level number which is greater than the level number of the subentry for the repeating group. Components of the same level must have the same level number.

3.3.6 ON (DATA)

Function

To specify the procedure to be executed when specified DML functions are performed on a data aggregate or data item.

General Format

$$\underline{\text{ON}}\,[\underline{\text{ERROR}}\ \text{DURING}] \left[\left\| \begin{array}{l} \underline{\text{STORE}} \\ \underline{\text{GET}} \\ \underline{\text{MODIFY}} \end{array} \right\| \right] \underline{\text{CALL}}\ \text{data-base-procedure-1}$$

Syntax Rules

1. A separate ON clause may be written for each DML function or group of functions.

2. The procedure named by data-base-procedure-1 may be specified in different ON clauses.

General Rules

1. If STORE is specified, the procedure is invoked whenever a new value of the data item is placed in the data base as a result of a STORE function. If MODIFY is specified, the procedure is invoked whenever a value of the data item is changed as a result of a MODIFY function. If GET is specified, the procedure is invoked whenever a value of the data item is placed in the user working area as a result of a GET function. If no DML functions are specified, the procedure is invoked whenever a value of the data item is the object of any of the functions listed.

2. The procedure is invoked immediately before control is returned to the run unit. If more than one procedure applies to the execution of a DML function, the procedures are invoked in the order in which they are stated in the schema, but a procedure named in an CN clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an CN ERROR clause will be entered only if, during the performance of the specified function, the DBMS detects an error that it intends to report.

3.3.7 ON (RECORD)

Function

To specify the procedure to be executed when specified DML functions are performed on a record.

General Format

```
                   ┌‖INSERT‖┐
                   │‖REMOVE‖│
                   │‖STORE ‖│
ON [ERROR DURING]  │‖DELETE‖│ CALL data-base-procedure-1
                   │‖MODIFY‖│
                   │‖FIND  ‖│
                   └‖GET   ‖┘
```

Syntax Rules

1. A separate ON clause may be written for each DML function or group of functions.

2. The procedure named by data-base-procedure-1 may be specified in different ON clauses.

General Rules

1. The procedure is invoked whenever a run unit issues one of the specified functions for an occurrence of the record. If no functions are specified, the procedure is invoked whenever the record is the object of any of the functions listed.

2. The procedure is invoked immediately before control is returned to the run unit. If more than one procedure applies to the execution of a DML function, the procedures are invoked in the order in which they are stated in the schema, but a procedure named in an ON clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an ON ERROR clause will be entered only if, during the performance of the specified function, the DBMS detects an error that it intends to report.

3.3.8 PICTURE

Function

To describe the characteristics of a data item.

General Format

$$\underline{\text{PICTURE}}\ \text{IS}\ \ "\left\{\begin{array}{l}\text{character-string-picture-specification-1}\\ \\ \text{numeric-picture-specification-1}\end{array}\right\}"$$

Syntax Rules

1. A picture specification, either character string or numeric, is composed of one or more characters enclosed in quotation marks. The characters that may be used in a picture specification are the character specifiers A and X; the digit specifiers 1, 2, 3, and 9; the point specifiers V, . (period), and P; the sign specifiers S and T; the exponent specifiers K and E; and a repetition factor, which is an unsigned decimal integer enclosed in parentheses.

2. Character-string-picture-specification-1 must include at least one A or X. Numeric-picture-specification-1 cannot contain the characters A or X.

3. The allowable combinations of picture characters and the parts of a numeric picture specification are described in the general rules.

General Rules

1. The picture characters A, X, P, 1, 2, 3, and 9 may be followed by a repetition factor, which is an unsigned decimal integer constant, n, enclosed in parentheses, to indicate repetition of the character n times. (If n is zero, the character is ignored.) For example, PICTURE "9(4)V9(5)" is equivalent to PICTURE "9999V99999".

2. Character-string-picture-specification-1 is used to describe a character string data item. A character string data item consists of a string of one cr more characters of the data character set defined bv the irplementor.

3. Each character in character-string-picture-specification-1 describes the corresponding character in the data item. The picture characters that may be used are as follows:

A specifies that the associated position may contain any alphabetic character or a blank character.

X specifies that the associated position may contain any character of the data character set defined by the implementor.

9 specifies that the associated position may contain any decimal digit.

4. Numeric-picture-specification-1 is used to describe an arithmetic data item that is represented in numeric pictured form. (The TYPE clause is used to describe an arithmetic data item that is represented in an implementor defined coded form.) There are four types of numeric pictured arithmetic data and four corresponding numeric picture specifications: fixed point decimal, fixed point binary, floating point decimal, and floating point binary.

5. Numeric-picture-specification-1 may consist of one, two, or three parts. The description of a fixed point number or the mantissa of a floating point number has a whole part and, optionally, a fractional part. For floating point numbers, an additional part is required to describe the exponent. Each part of numeric-picture-specification-1 must contain at least one picture character that specifies a digit position, and each part of the same numeric picture specification must contain the same digit specifier; that is, either 1, 2, 3, or 9.

6. Decimal numeric pictured data is a string of characters that represents an arithmetic value or a character string value depending on its use. The data item consists of decimal digits. A fixed point decimal numeric pictured data item may contain a decimal point and an overpunched digit or a separate sign. A floating point decimal numeric pictured data item may contain a decimal point, two signs and the character E.

7. Binary numeric pictured data is a string of bits that represents an arithmetic value or a bit string value, depending on its use. The data item consists only of binary digits, either signed or in 1's or 2's complement form, with an assumed binary point.

8. Digit Specifiers:

The picture characters 1, 2, 3, and 9 are used in numeric picture specifications to describe digits.

1 specifies that the associated position in the data item contains a binary digit.

2 specifies that the associated position in the 2's complement data item contains a binary digit.

3 specifies that the associated position in the 1's complement data item contains a binary digit.

9 specifies that the associated position in the data item contains a decimal digit.

9. Point Specifiers:

The picture characters V, P, and . (period) are used in numeric picture specifications to describe the position of the radix point.

V specifies that a decimal or binary point is assumed at this position in the associated data item.

. specifies that the associated position in the data item actually contains a decimal point. This picture character may only be used for decimal numeric pictured data.

P specifies an assumed scaling position and is used only when the assumed decimal or binary point is either more than one position to the left of the first actual position in the data item or more than one position to the right of the last actual position in the data item.

Numeric-picture-specification-1 may contain either no point specifier, one V, one decimal point, or one group of P's. A V may be used in combination with P's, but since the use of P's implies an assumed point (to the left of P's on the left or to the right of P's on the right) the V is redundant. If no point specifiers are used (in a fixed point numeric picture specification or in the mantissa part of a floating point numeric picture specification) a V is assumed on the right. A point specifier cannot be used in the exponent part of a floating point numeric picture specification.

10. Sign Specifiers:

The picture characters S and T are used to specify signs in numeric pictured data.

S specifies that a separate sign always appears in the associated position of the data item.

T specifies that the associated position of the data item will contain a digit overpunched with the sign of the number.

Only one sign specifier can be used in a fixed point numeric picture specification. A floating point numeric picture specification may contain two sign specifiers, one for the mantissa part and one for the exponent. The sign specifier T can only be used in a decimal numeric picture specification. The sign specifier S can be used in a binary numeric picture specification, but only in combination with the digit specifier 1. If the sign specifier S is used, it must appear to the left of all digits in the mantissa part and to the left of all digits in the exponent part of a floating point numeric picture specification, and either to the right or left of all digit positions of a fixed point numeric picture specification. The sign specifier T can be used in any digit position and it also serves as a decimal digit specifier.

Use of the sign specifier S in a decimal numeric picture specification means that the associated position of the data item contains a plus sign character if the value of the number is greater than or equal to zero; otherwise, it contains a minus sign character. Use of the sign specifier S in a binary numeric picture specification means that the associated position of the data item contains the binary digit 0 if the value of the number is greater than or equal to zero; otherwise, it contains the binary digit 1. The representation of overpunched signs is implementor defined. If a data item contains an overpunched sign, the overpunch is part of the character string value of the data item. If no sign specifiers are used, the data item is assumed to be positive.

11. Exponent Specifiers:

The picture characters K and E delimit the exponent part of a numeric picture specification that describes a floating point number. The exponent is always the last part of a floating point numeric picture specification. The picture characters K and E cannot appear in the same specification.

K specifies that the exponent part appears to the right of the associated position. It does not specify a character in the numeric pictured data item.

E specifies that the associated position contains the letter E, which indicates the start of the exponent. It cannot appear in a binary numeric picture specification.

The value of the exponent is always adjusted in the character string or bit string value so that the first significant digit of the mantissa appears in the position associated with the first digit specifier of the specification. The value zero is represented by zero exponent and zero mantissa, with positive signs if required.

12. Unless explicitly prohibited by the CHECK clause, the characteristics of a data item as defined in the schema may differ from the characteristics of the data item as defined in a subschema. When the characteristics differ, a conversion occurs from the schema defined characteristics to the subschema defined characteristics whenever a GET function is issued for the data item; and a conversion occurs from the subschema defined characteristics to the schema defined characteristics whenever the data item is involved in a STORE or MODIFY function. A data item described by character-string-picture-specification-1 is subject to the conversion rules defined for character strings by the general rules of the TYPE clause. Rules for conversion of numeric pictured arithmetic data to coded arithmetic data are also defined by the general rules of the TYPE clause. Except for the situation defined by the General Rule 13 below, whenever a numeric pictured arithmetic data item is involved in a conversion, it is first converted to a coded arithmetic form. Conversion then proceeds in accordance with the general rules of the TYPE clause.

13. If both the schema and subschema define a fixed point arithmetic data item in numeric pictured form and the source and target description differ only in regard to the number of digits and/or the position of the radix point, then, in accordance with the numeric picture specification of the target, insignificant zero digits will be appended or removed and rounding will occur if necessary. If the precision of the target is not adequate to account for all significant digits in the whole part of the number, the conversion does not occur, and an error is reported. Whenever a conversion does not occur, the value in the data base and the user working area remains unchanged.

3.3.9 PRIVACY (DATA)

Function

To specify the privacy locks which apply to the use of a data item or data aggregate.

General Format

$$\underline{\text{PRIVACY}}\ \text{LOCK}\ \left[\ \text{FOR}\ \left\| \begin{array}{l}\underline{\text{STORE}}\\ \underline{\text{GET}}\\ \underline{\text{MODIFY}}\end{array}\right\| \right]\ \text{IS}\ \left\{ \begin{array}{l}\text{literal-1}\\ \text{lock-name-1}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-1}\end{array}\right\}$$

$$\left[\ \underline{\text{OR}}\ \left\{ \begin{array}{l}\text{literal-2}\\ \text{lock-name-2}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-2}\end{array}\right\}\right]\ \ldots$$

Syntax Rules

1. A separate PRIVACY clause may be stated for each DML function. However, the same DML function must not be stated in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more of the DML functions included in this clause.

3. All literals must conform to the implementor defined data characteristics for privacy locks.

General Rules

1. The literals and the content of the lock-names are privacy locks, to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which, when given access to a privacy key, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, the lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR phrase is omitted all literals, lock-names, or procedures apply to all DML functions included in the relevant format.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are

processed in the order listed until the outcome of the PRIVACY clause is known.

6. If a PRIVACY clause has not been specified for a DML function, then unless other PRIVACY clauses apply, the use of that function on occurrences of the data item or data aggregate being described is without restriction.

7. The privacy locks associated with the various DML functions (STORE, GET,...) must be satisfied in order to execute the respective DML function on the data item or data aggregate (or any of its components) to which the privacy lock applies.

3.3.10 PRIVACY (RECORD)

Function

To specify the privacy locks which apply to the use of a record type.

General Format

$$\underline{\text{PRIVACY}}\ \text{LOCK}\left[\ \underline{\text{FOR}}\ \left\|\begin{array}{l}\underline{\text{INSERT}}\\ \underline{\text{REMOVE}}\\ \underline{\text{STORE}}\\ \underline{\text{DELETE}}\\ \underline{\text{GET}}\\ \underline{\text{MODIFY}}\\ \underline{\text{FIND}}\end{array}\right\|\ \right]\ \text{IS}\left\{\begin{array}{l}\text{literal-1}\\ \text{lock-name-1}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-1}\end{array}\right\}$$

$$\left[\ \underline{\text{OR}}\ \left\{\begin{array}{l}\text{literal-2}\\ \text{lock-name-2}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-2}\end{array}\right\}\right]\ldots$$

Syntax Rules

1. A separate PRIVACY clause may be stated for each DML function. However, the same DML function must not be stated in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more of the DML functions included in this clause.

3. All literals must conform to the implementor defined data characteristics for privacy locks.

General Rules

1. The literals and the content of the lock-names are privacy locks, to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which, when given access to a privacy key, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, the lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR phrase is omitted all literals, lock-names, or procedures apply to all DML functions included in the relevant format.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the PRIVACY clause is known.

6. If a PRIVACY clause has not been satisfied for a DML function, then unless other PRIVACY clauses apply, the use of that function on occurrences of the record being described is without restriction.

7. The privacy locks associated with the various DML functions (INSERT, REMOVE,...) must be satisfied in order to execute the respective DML function on the record (or any of its components) to which the privacy lock applies.

3.3.11 RECORD

Function

To name a record type in the schema; that is, to specify a generic name for all occurrences of the record type in the data base.

General Format

RECORD NAME IS record-name-1

Syntax Rules

1. Record-name-1 must be unique among the record-name of the schema.

2. At least one record-name must be specified in the schema.

General Rules

None.

3.3.12 RESULT

Function

To specify that the value of a data item is established by the execution of a procedure and to control the times at which that procedure is invoked to materialize the value.

General Format

```
IS {ACTUAL } RESULT OF data-base-procedure-1
   {VIRTUAL}

  || ON THIS RECORD                          [                                            ] ||
  ||                                         |                                            | ||
  ||                                         |                                            | ||
  || ON ALL MEMBERS OF set-name-1            |                                            | ||
  ||                  [,set-name-2]...       | USING data-base-identifier-1               | ||
  ||{                                        |       [,data-base-identifier-2]...         | ||
  ||{ON record-name-1            }           |                                            | ||
  ||{   [,record-name-2]...      }...        |                                            | ||
  ||       OF set-name-3         }           [                                            ] ||
```

Syntax Rules

1. The data item which is the subject of this clause is the target data item, and must not be a data aggregate or a component of a data aggregate.

2. The data-base-identifiers must be the names of Data Subentries in the Record Entries for the record type explicitly or implicitly specified in the ON phrase of this entry.

3. The record-names must be defined as member record types of set-name-3.

4. All set names specified must be the names of set types, each of whose owner is the record type which contains the target data item.

5. All set-names must be the names of different set types.

6. If VIRTUAL RESULT is specified none of the ON phrases in the general format is allowed.

General Rules

1. The value of the target data item is established by the execution of the procedure named by data-base-procedure-1 and cannot be altered except by that procedure.

2. All data-base-identifiers are parameters to the procedure. They are the names of Data Subentries of the object record types.

3. If RECORD is specified the object record is the occurrence of the record type in which the target data item is defined.

4. If set-name-1, set-name-2,... is specified, the object records are all member records of those sets of the named set types whose owner record is the occurrence of the record type in which the target data item is defined.

5. If set-name-3 is specified, the object records are the member records whose types are named and whose owner record is the occurrence of the record type in which the target data item is defined.

6. If the target data item is the subject of a VIRTUAL RESULT clause, its value is established by the procedure at the time a GET function, which involves that data item, is executed, or whenever a value for that data item is required by the DBMS.

7. If the target data item is the subject of an ACTUAL RESULT clause, its value is maintained by the DBMS in its materialized form at all times.

8. In the absence of a USING phrase the procedure establishes the value of an ACTUAL RESULT item as follows:

   - If RECORD is specified, the value is established when the object record is stored in the data base and whenever it is modified.

   - If any set-names are specified the value is established whenever any of the object records are inserted into, or removed from occurrences of the set types named, or when they are stored, deleted, or modified.

9. When a USING phrase is specified the procedure establishes the value of an ACTUAL RESULT item as in General Rule 8 except that when any object record is stored or modified the value of the ACTUAL RESULT item is not established unless one of the items specified in the USING phrase is stored or modified, or unless any object record is switched to a new set.

3.3.13 SOURCE

Function

To specify that the value of a data item is to be the same as the value of another data item and to control the times at which the DBMS materializes the value.

General Format

$$\text{IS} \left\{ \begin{array}{l} \underline{\text{ACTUAL}} \\ \underline{\text{VIRTUAL}} \end{array} \right\} \text{AND } \underline{\text{SOURCE}} \text{ IS data-base-identifier-1 OF } \underline{\text{OWNER}} \text{ OF set-name-1}$$

Syntax Rules

1. If this clause is used, the only other clauses permitted in the Data Subentry are the PRIVACY clause, CHECK clause and the ON clause.

2. If data-base-identifier-1 is defined with a VIRTUAL SOURCE or VIRTUAL RESULT clause, then the optional word ACTUAL is illegal.

3. The record type in which this data item is being described must be a member of set-name-1 and data-base-identifier-1 must refer to a data item included in the owner record type of set-name-1.

4. This clause must not be applied to data aggregates or their components.

General Rules

1. The characteristics of the data item being described are the same as those ascribed to data-base-identifier-1.

2. If the record which includes the data item being described is a member of an occurrence of set-name-1, the DBMS is responsible for ensuring that the value of that data item is equal to the value of data-base-identifier-1 in the owner record of that set.

3. If the record which includes the data item being described is not currently a member of an occurrence of set-name-1, then the value of the data item is null.

4. The value of a data item defined with a VIRTUAL SOURCE clause is established by the DBMS whenever a GET function which involves that data item is executed and whenever a value for that data item is required by the DBMS.

5. The value of a data item defined with an ACTUAL SOURCE clause is maintained by the DBMS in its materialized form at all times.

6. The value of the SOURCE data item may be directly modified if this data item is used in a SELECTION clause to govern set selection. The SELECTION clause will be used to determine the correct set when this data item is stored, or a possibly different set when this data item is modified.

3.3.14 TYPE

Function

To describe the characteristics of a data item.

General Format

```
           ⎧ ⎧BINARY ⎫                                     ⎫
           ⎪ ⎩DECIMAL⎭                                     ⎪
           ⎪ ⎧FIXED  ⎫                                     ⎪
           ⎪ ⎩FLOAT  ⎭  [integer-1 [,integer-2]]           ⎪
           ⎪ ⎧REAL   ⎫                                     ⎪
           ⎪ ⎩COMPLEX⎭                                     ⎪
TYPE IS    ⎨                                               ⎬
           ⎪ ⎧BIT      ⎫                                   ⎪
           ⎪ ⎩CHARACTER⎭ [integer-3]                       ⎪
           ⎪                                               ⎪
           ⎪  DATA-BASE-KEY                                ⎪
           ⎪                                               ⎪
           ⎩  implementor-name                             ⎭
```

Syntax Rules

1. Integer-1 and integer-3 must be unsigned decimal constants with value greater than zero. The maximum value is implementor defined.

2. If FLOAT is specified, integer-2 must not be specified.

General Rules

1. The TYPE clause is used to define an arithmetic data item, a string data item, or a data base key. Implementor-name means that the implementor may provide for one or more additional types of data items. For arithmetic data, the TYPE clause is used to specify a coded arithmetic data item as opposed to a data item in numeric pictured form. The representation of a coded arithmetic data item is implementation defined. A coded arithmetic data item is described in terms of its base, scale, mode, and precision.

2. BINARY and DECIMAL are used to specify the base of a coded arithmetic data item as either binary or decimal. If the base is not specified, DECIMAL is assumed.

3. FIXED and FLOAT are used to specify the scale of a coded arithmetic data item as either fixed point or floating point. Floating point data items have a mantissa and an exponent part. Integer-1 is used to specify the number of binary or

decimal digits in the mantissa of the number. The precision of fixed point data is given by integer-1 and integer-2 where integer-1 specifies the total number of binary or decimal digits and integer-2 is the scale factor. If the scale is not specified, FIXED is assumed.

4. REAL and COMPLEX are used to specify the mode of a coded arithmetic data item as either real or complex. A complex data item is a pair of values. The first value is the real part of the complex number and the second value is the imaginary part. Integer-1 and integer-2 specify the precision of both the real part and the imaginary part. If the mode is not specified, REAL is assumed.

5. Integer-1 and integer-2 are used to specify the precision and scale respectively of a coded arithmetic data item. Integer-1 is used to specify the minimum number of significant binary or decimal digits to be maintained for all values of a fixed point data item or for the mantissa of a floating point data item. Integer-2 is used to specify the scale factor of fixed point data items (the assumed position of the binary or decimal point). A negative scale factor, -n, describes an integer, with the point assumed to be located n places to the right of the rightmost actual digit. A positive scale factor, n, describes an arithmetic data item with the point assumed to be located n places to the left of the rightmost actual digit. A zero scale factor describes an arithmetic data item with the point assumed to be located immediately to the right of the rightmost actual digit. If FIXED is specified and integer-2 is omitted, it is assumed to be zero. If both integer-1 and integer-2 are omitted, integer-2 is assumed to be zero and the assumed value of integer-1 is implementor defined.

6. BIT and CHARACTER are used to specify a string data item. BIT specifies a bit string, that is, one or more binary digits. CHARACTER specifies a character string, that is, one or more characters of the data character set defined by the implementor.

7. Integer-3 is used to specify the length of a string data item; that is, the number of bits or characters in the string. If integer-3 is omitted, its value is assumed to be 1.

8. DATA-BASE-KEY defines a data item designed to hold a data base key. A data base key is a unique identifier of a record. The representation of data base keys is implementor defined.

9. Unless explicitly prohibited by the CHECK clause, the characteristics of a data item as defined in the schema may

differ from the characteristics of the data item as defined in a subschema. Where the characteristics differ, a conversion occurs from the schema defined characteristics to the subschema defined characteristics whenever a GET function is issued for the data item; and a conversion occurs from the subschema defined characteristics to the schema defined characteristics whenever the data item is involved in a STORE or MODIFY function. The following general rules specify the conversion rules in terms of the language used to describe data items in the schema.

10. If the source form of a data item is a bit string and the target form is a character string, the bit 1 becomes the character 1, and the bit 0, the character 0.

11. If the source form of a data item is a character string and the target form is a bit string, the characters 1 and 0 become the bits 1 and 0. If the character string contains characters other than 0 and 1, the conversion does not occur and an error is reported.

12. If the source form of a data item is string and the target form is a string of greater length, the value is extended on the right with blanks for character strings, zeros for bit strings. If the target is a string of shorter length, the value is truncated on the right. If truncation removes only blank characters or zero bits, the operation is completed. On a STORE or MODIFY function, if truncation removes any nonblank characters or nonzero bits, the conversion does not occur and an error is reported. On a GET function, if truncation removes any nonblank characters or nonzero bits, conversion does occur and a warning is reported.

13. If the source form of a data item is a bit string and the target form is coded arithmetic, the bit string is interpreted as an unsigned binary integer and is converted to the base, scale, mode, and precision of the target. Insignificant zero digits will be appended or removed in accordance with the precision of the target. If the precision of a fixed point target is not large enough to account for all significant digits, the conversion does not occur and an error is reported.

14. If the source form of a data item is coded arithmetic and the target form is a bit string, the absolute arithmetic value is converted, if necessary, to real and then to fixed point binary. Zero bits will be appended to or removed from the left of the value in accordance with the length of the bit string. If the length of the string is not adequate to account for all significant digits, the conversion does not occur and an error is reported.

15. Since there is no standard for the character representation of numbers, conversion from coded arithmetic to character string and conversion from character string to coded arithmetic are implementor defined. If the precision or length of the target does not conform to the implementor defined rules, the conversion does not occur and an error is reported. If the source is a character string which does not conform to the implementor defined rules for the character string representation of numbers, the conversion does not occur and an error is reported.

16. If the target is arithmetic data described with a numeric picture specification, the source must be either in coded arithmetic form or in a form that can be converted to coded arithmetic. If the source is a character string which does not conform to the implementor defined rules for the character string representation of numbers, the conversion does not occur and an error is reported. If the source is also numeric pictured data, it is converted, if necessary, to coded arithmetic form as stated in the general rules of the PICTURE clause. If the target is described as decimal numeric pictured data, the coded arithmetic value is converted to character representation. If the target is described as a binary numeric pictured data, the coded arithmetic value is converted to bit representaion. Insignificant zero digits will be appended or removed from either end of the value in accordance with the precision and scaling factor of the target. If the scaling factor of the target is less than the scaling factor of the source, rounding occurs in the least significant digit. If the precision of the target is not adequate to account for all significant digits in the whole part of the number, the conversion does not occur, and an error is reported.

17. If the target is coded arithmetic and the source is in numeric pictured form, the value is converted to the appropriate internal representation. Insignificant zero digits may be appended or removed and rounding will occur, if necessary, in the least significant digit. If the precision of the target is not adequate to account for all significant digits in the whole part of the number, the conversion does not occur, and an error is reported.

18. If both the target form and the source form of the data item are coded arithmetic, the value is converted, if necessary, to the base, scale, mode, and precision of the target. If a complex value is converted to a real value, the result is the real part of the complex value. If a real value is converted to a complex value, the result is a complex value that has the real value as the real part and zero as the imaginary part. Insignificant zero digits may be appended or removed and rounding will occur, if necessary, in the

least significant digit. If the precision of the target is not adequate to account for all significant digits in the whole part of the number, the conversion does not occur, and an error is reported.

19. Whenever a conversion does not occur, the value in the data base and the user working area remains unchanged.

3.3.15 WITHIN

Function

To define to the DBMS the areas in which occurrences of a record type may be stored and to provide a means of differentiating between such areas.

General Format

$$\underline{\text{WITHIN}}\left\{\begin{array}{l}\text{area-name-1}\left[\begin{array}{l}\left\{,\text{area-name-2}\right\}\cdots\ \underline{\text{AREA-ID}}\ \text{IS data-base-data-name-1}\\ \qquad\left[\text{USING}\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-1}\right]\end{array}\right]\\ \text{AREA OF}\ \underline{\text{OWNER}}\end{array}\right\}$$

Syntax Rules

1. The area-names must be the names of areas for which an Area Entry for each is included in the schema prior to this entry.

2. If OWNER is specified, the LOCATION clause in the Record Subentry must specify VIA set-name, where the owner of the referenced set type is not SYSTEM.

General Rules

1. By its appearance in a WITHIN clause, data-base-data-name-1 is implicitly defined to be a data item that contains a character string that conforms to the rules for the formation of area-names.

2. When only one area-name is specified, all records of the type being described will be stored in the named area (area-name-1).

3. When more than one area-name is specified, the contents of data-base-data-name-1 determine the area into which a record is stored.

4. The content of data-base-data-name-1, which must be either one of the area-names specified in the WITHIN clause or a null value, must be set before the object record can be stored. If the content is an area-name the record is stored in the named area. If the content is a null value the DBMS selects, using an implementor defined method, one of the areas named in the WITHIN clause; this selection is subject to the constraints, if any, of the LOCATION clause.

5. The procedure named by data-base-procedure-1 will be invoked by the DBMS and must return a valid area name or a null value in data-base-data-name-1 as defined in General Rule 4 above.

6. If OWNER is specified, the record will be stored in the same area as the owner of the selected occurrence of the set type named in the LOCATION clause.

7. The record's area assignment remains constant regardless of changes in its set membership.

3.4.0 SET ENTRY

Function

To name and give certain characteristics of the sets within a data base.

Set Entry Skeleton

Set Subentry

[Member Subentry] ...

Set Subentry Skeleton

```
SET clause

    OWNER clause

    DYNAMIC/PRIOR clause

    ORDER clause

    ON clause

    PRIVACY clause
```

Member Subentry Skeleton

```
MEMBER clause

    KEY clause

    SEARCH clause

    SELECTION clause

    ON clause

    PRIVACY clause
```

General Format of Set Subentry

```
SET NAME IS set-name-1

 ;OWNER IS {record-name-1}
           {SYSTEM       }

[;SET IS  ||DYNAMIC            ||]
[         ||PRIOR PROCESSABLE  ||]

 ;ORDER IS {PERMANENT} INSERTION IS
           {TEMPORARY}

   {FIRST                                                                    }
   {LAST                                                                     }
   {NEXT                                                                     }
   {PRIOR                                                                    }
   {IMMATERIAL                                                               }
   {SORTED [INDEXED [NAME IS index-name-1]]                                  }
   {                                                                         }
   {      {BY DATA-BASE-KEY                                                 }}
   {      {BY RECORD-NAME                                                   }}
   {      {WITHIN RECORD-NAME                      [FIRST]                  }}
   {      {BY DEFINED KEYS [DUPLICATES ARE         [LAST ] ALLOWED]         }}
   {      {                                        [NOT  ]                  }}

[;ON [ERROR DURING] [||ORDER ||]                                  ]
[                   [||INSERT||] CALL data-base-procedure-1       ]...
[                   [||REMOVE||]                                  ]

[;PRIVACY LOCK [FOR [||ORDER ||]] IS {literal-1                          }
[              [    [||INSERT||]]    {lock-name-1                        }
[              [    [||REMOVE||]]    {PROCEDURE data-base-procedure-2    }
[              [    [||FIND  ||]]
[
[   [ OR {literal-2                          }]   ]
[   [    {lock-name-2                        }]...]...
[   [    {PROCEDURE data-base-procedure-3    }]   ]
```

General Format of Member Subentry

```
MEMBER IS record-name-1 {MANDATORY} {AUTOMATIC} [LINKED TO OWNER]
                        {OPTIONAL } {MANUAL   }

   [DUPLICATES ARE [NOT] ALLOWED FOR data-base-identifier-1
          [,data-base-identifier-2]...]...

[                 {ASCENDING }
[ ;[RANGE] KEY IS {DESCENDING} data-base-identifier-3
[
   [   [ASCENDING ]                        ]
   [ , [DESCENDING] data-base-identifier-4 ] ...

                   [FIRST]                                 ]
   [DUPLICATES ARE [LAST ] ALLOWED] NULL IS [NOT] ALLOWED  ]
                   [NOT  ]                                 ]

[
[;SEARCH KEY IS data-base-identifier-5[,data-base-identifier-6 ]...
[
   [      {CALC                              }]
   [USING {INDEX[NAME IS index-name-1]       }]
   [      {PROCEDURE data-base-procedure-1   }]
    DUPLICATES ARE [NOT] ALLOWED ] ...
```

Format 1

```
 ;SET SELECTION [FOR set-name-1] IS
    THRU set-name-2 OWNER IDENTIFIED BY
      | SYSTEM                                                            |
      | CURRENT OF SET                                                    |
      | DATA-BASE-KEY [EQUAL TO {data-base-identifier-7}]                  |
      |                         {data-base-data-name-1 }                  |
      { CALC-KEY      [ EQUAL TO {data-base-identifier-8}                ] }
      |               |          {data-base-data-name-2 }                |  |
      |               |            [,data-base-identifier-9]...         |  |
      |               [            [,data-base-data-name-3 ]            ]  |
      | MEMBER record-name-2 SELECTION                                    |
   [ THEN THRU set-name-3                                                      ]
   [   {WHERE OWNER IDENTIFIED BY data-base-identifier-10                }     ]
   [   {   [           {data-base-identifier-11          }]              }...  ]...
   [   {   [ EQUAL TO  {data-base-data-name-4            }]              }     ]
   [   {   [           {PROCEDURE data-base-procedure-2  }]              }     ]
```

Format 2

```
 ;SET SELECTION IS BY PROCEDURE data-base-procedure-3

 [                      [||INSERT||]                            ]
 [ ;ON [ERROR DURING]   [||REMOVE||] CALL data-base-procedure-4 ]...
 [                      [||FIND  ||]                            ]


 [                      ||INSERT||    {literal-1                      }
 [ ;PRIVACY LOCK [ FOR  ||REMOVE||] IS {lock-name-1                    }
 [               [      ||FIND  ||]    {PROCEDURE data-base-procedure-5}

      [    {literal-2                       }]     ]
      [ OR {lock-name-2                     }] ... ]...
      [    {PROCEDURE data-base-procedure-6 }]     ]
```

3.4.1 DYNAMIC/PRIOR

Function

To specify that any record type, defined in the schema, may be a member of the set type and/or to specify that occurrences of the set type are to be processed in the PRIOR direction.

General Format

```
SET IS ||DYNAMIC          ||
       ||PRIOR PROCESSABLE||
```

Syntax Rules

1. If DYNAMIC is used then the Set Entry may not contain any Member Subentries. That is, no member record types may be declared for the set type.

General Rules

1. If DYNAMIC is used then any record except for a record of the type declared to be the owner record type of this set type, may be made a member of a single occurrence of this set type. A record may only appear once in a given set. All membership is implicitly OPTIONAL MANUAL. The SET SELECTION for all members is implicitly THRU CURRENT.

2. The PRIOR option causes the DBMS to select preferentially for this set type an implementation method which allows a set to be processed as efficiently in the backward direction as in the forward direction.

3.4.2 KEY

Function

To specify the sort control key for a member record of a sorted set. To control the insertion into a set of those member records that contain duplicate values for the specified sort control key. To control the insertion into any set of those member records that contain a null value for the specified sort control key.

General Format

$$[\underline{\text{RANGE}}]\ \underline{\text{KEY}}\ \text{IS}\ \left\{\begin{matrix}\underline{\text{ASCENDING}}\\ \underline{\text{DESCENDING}}\end{matrix}\right\}\ \text{data-base-identifier-1}$$

$$\left[\ ,\ \left[\begin{matrix}\underline{\text{ASCENDING}}\\ \underline{\text{DESCENDING}}\end{matrix}\right]\ \text{data-base-identifier-2}\right]\ldots$$

$$[\underline{\text{DUPLICATES}}\ \text{ARE}\ \left[\begin{matrix}\underline{\text{FIRST}}\\ \underline{\text{LAST}}\\ \underline{\text{NOT}}\end{matrix}\right]\ \text{ALLOWED}]\ \underline{\text{NULL}}\ \text{IS}\ [\underline{\text{NOT}}]\ \text{ALLOWED}$$

Syntax Rules

1. The KEY clause must be specified in all Member Subentries of any Set Entry which includes the ORDER IS SORTED clause with the DEFINED option. The KEY clause must not be specified if the Set Entry does not include the ORDER IS SORTED clause; nor may it be specified if the Set Entry includes the ORDER IS SORTED clause with the DATA-BASE-KEY option. The KEY clause is optional if the Set Entry includes the ORDER IS SORTED clause in any other form.

2. The data-base-identifiers must refer to data items declared in the Record Entry for the record type named in the MEMBER clause of this Subentry.

3. If the Set Entry includes the ORDER IS SORTED clause with the DEFINED option, corresponding data items must be specified in the KEY clauses of all member record types; the corresponding data items must have identical data characteristics and must also match in terms of whether ASCENDING or DESCENDING is specified for them. Two data items to which identical TYPE or PICTURE clauses apply have identical data characteristics.

4. The DUPLICATES phrase must be specified if the ORDER IS SORTED clause in the Set Entry does not include any DUPLICATES phrase. The DUPLICATES phrase is optional if the ORDER IS SORTED clause in the Set Entry includes the optional DUPLICATES ARE ALLOWED phrase. In all other cases the DUPLICATES phrase must not be specified.

General Rules

1. The data-base-identifiers are the key items which together constitute the sort control key for the member record type named in the MEMBER clause of this Subentry. The key items are stated in the KEY clause in order of decreasing significance, that is, data-base-identifier-1 is the major key item and data-base-identifier-2,... are minor key items. The value of the key is considered to be the juxtaposition of all key item values in major to minor sequence.

2. The ASCENDING option applies to a key item, if in the KEY clause the data-base-identifier that identifies the key item is preceded by the keyword ASCENDING and the keyword DESCENDING does not appear between that instance of the keyword ASCENDING and the data-base-identifier; otherwise the DESCENDING option applies to the key item. The sorted sequence of the member records within a set is from the lowest (highest) value of the key item to the highest (lowest) value, if the ASCENDING (DESCENDING) option applies to that key item.

3. If the ORDER IS SORTED clause is specified in the Set Entry and no KEY clause is specified, the data base key of the member record is considered to be the ascending key item.

4. The collating sequence is implementor defined.

5. The RANGE option controls the effect of the use of the sort control key as an argument for set selection. If RANGE is specified and if all data items specified in the KEY clause are also specified in a SELECTION clause, an equality match between the key item value(s) (which are in the record to be selected) and the input argument value(s) is not required for a record to be selected as being the owner of the sought set; in such a case a match will occur as described in General Rule 6. In all other cases an equality match between the key item value(s) in the record and the input argument value(s) is required for that record to be selected.

6. Under the circumstances described in General Rule 5 and regardless of whether the key is composed of only ascending key items or only descending key items or a mixture of both, a match between the key value and the input value, which is considered to be the juxtaposition of the input argument

values for the key items in major to minor sequence, will occur in accordance with the following:

a. If the input value equals the value of any range key occurrence, then a match occurs on that specific range key occurrence.

b. If the input value is less than the lowest value of any range key occurrence, then a match occurs on the range key occurrence with the lowest value.

c. If the input value is greater than the largest value of any range key occurrence, then no match occurs.

d. If the input value lies between two adjacent range key values, then a match occurs on the range key occurrence with the larger value.

7. If the optional DUPLICATES ARE NOT ALLOWED phrase is specified, the DBMS rejects the insertion into any given set of those member records that are of the same type and have the same nonnull values for the specified key items as a record that is already a member of that set. This may occur during an attempt to store a new member record in the data base or to insert an existing record into a set, or to modify the value of a key item.

8. If the optional DUPLICATES ARE FIRST or DUPLICATES ARE LAST phrase is specified, member records will be inserted into the set sequence before or after, depending on the option specified, any existing member records in the set that are of the same type and have the same values for the specified key items.

9. If the optional DUPLICATES ARE ALLOWED phrase is specified, the insertion point in the set sequence of member records relative to any existing member records in the same set, that are of the same type and have the same values for the specified key items, is unpredictable.

10. If the DUPLICATES phrase is not specified, the action to be taken is controlled by the DUPLICATES phrase that is included in the ORDER IS SORTED clause in the Set Entry.

11. If the NULL IS NOT ALLOWED phrase is specified, the DBMS rejects the insertion into any set of those member records having a null value for one or more of the specified key items.

12. If the NULL IS ALLOWED phrase is specified, the DBMS allows the insertion into any set of those member records having a null value for one or more of the specified key items.

3.4.3 MEMBER

Function

To specify the name of a record type, the occurrences of which may be members in occurrences of the set type named in this Set Entry.

To specify the type of membership in that set type and optionally to check and reject the insertion within the same set of those member records that have duplicate values for specified data items.

General Format

$$\underline{\text{MEMBER}}\ \text{IS record-name-1} \left\{\begin{array}{l}\underline{\text{MANDATORY}}\\ \underline{\text{OPTIONAL}}\end{array}\right\} \left\{\begin{array}{l}\underline{\text{AUTOMATIC}}\\ \underline{\text{MANUAL}}\end{array}\right\} \text{[LINKED TO } \underline{\text{OWNER}}\text{]}$$

$$\Big[\ \underline{\text{DUPLICATES}}\ \text{ARE} \left[\underline{\text{NOT}}\right] \text{ALLOWED FOR data-base-identifier-1}$$

$$\text{[,data-base-identifier-2]...}\Big]\text{...}$$

Syntax Rules

1. Record-name-1 must be previously defined in a Record Entry.

2. This clause must not be used if the DYNAMIC clause is specified in the Set Entry.

3. The data-base-identifiers must refer to data items included in the record type named by record-name-1.

4. Record-name-1 cannot be the name of the record type specified in the OWNER clause of this Set Entry.

5. If the record types named in a series of Set Entries are such that the resulting structure forms a cycle, then at least one of the MEMBER clauses involved must specify MANUAL. Further the Record Entry for one such member record type must have a LOCATION clause which is not via a set type included in the cycle.

General Rules

1. If AUTOMATIC is used, then an occurrence of the record type named by record-name-1 is inserted into (made a current member of) the selected occurrence of the set type named in this Set Entry, when the record is added to the data base. If MANUAL is used, adding an occurrence of the record type named by record-name-1 to the data base will not cause that record to become a current member of any occurrence of the set type named in this Set Entry. Membership in the set is established by a run unit by means of an INSERT function.

2. If MANDATORY is used, then once an occurrence of the record type referenced by record-name-1 is made a member of any occurrence of the set type named in this Set Entry, it will always be a member of one or another set of that set type. Such a record cannot be the object of a REMOVE function; it may however be switched between sets of the same set type by a MODIFY function. If OPTIONAL is used, the membership is not permanent in the above sense and can be cancelled by means of a REMOVE function.

3. If the DUPLICATES NOT ALLOWED phrase is used the DBMS will reject the insertion into any given set of those member records with duplicate values for the data items specified in this clause. This may occur during an attempt to store a new record in the data base, or to insert an existing record into a set or to modify the value of such a data item.

4. The optional LINKED TO OWNER phrase causes the DBMS to select preferentially for the set type, whose declaration contains this Member Subentry, an implementation method which allows the OWNER record of the set containing an occurrence of this member record to be accessed directly from that member record. The LINKED TO OWNER phrase may not be applied to member record types of singular set types.

5. A MEMBER clause must be specified for each record type that can participate as a member in the set type being described.

6. More than one record type can be declared as a member of any given set type.

7. A record type can be defined as a member in more than one set type. It may also be defined as an owner in one or more set types.

8. The DUPLICATES NOT ALLOWED phrase must be repeated for each data item or concatenation of data items for which duplicate values are not allowed. The data-base-identifiers included

in any single DUPLICATES NOT ALLOWED phrase will be concatenated.

9. Each member record participates in at most one occurrence of each set type for which it is declared a member record type. That is, it may be associated with no more than one owner record for each set type for which it may be a member. A record may only appear once in a given set.

3.4.4 ON (MEMBER)

Function

To specify the procedure to be executed when specified DML functions are performed on the record as a member of an occurrence of the set type named in the Set Subentry.

General Format

$$\underline{\text{ON}}\ [\underline{\text{ERROR}}\ \text{DURING}]\ \left[\left\|\begin{array}{l}\underline{\text{FIND}}\\ \underline{\text{INSERT}}\\ \underline{\text{REMOVE}}\end{array}\right\|\right]\ \underline{\text{CALL}}\ \text{data-base-procedure-1}$$

Syntax Rules

1. A separate ON clause may be written for each DML function cr group of functions.

2. The procedure named by data-base-procedure-1 may be specified in different ON clauses.

General Rules

1. The procedure is invoked whenever a run unit issues one of the specified functions for the record as a member of an occurrence of the set type named in the Set Subentry. If no DML functions are specified, the procedure is invoked whenever a run unit issues a FIND, INSERT, or REMOVE function for the record as a member of an occurrence of the set type named in the Set Subentry.

2. The procedure is invoked immediately before control is returned to the run unit. If more than one procedure applies to the execution of a DML function, the procedures are invoked in the order in which they are stated in the schema, but a procedure named in an ON clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an ON ERROR clause will be entered only if, during the performance of the specified function, the DBMS detects an error that it intends to report. Procedures referenced in a Set Subentry are invoked prior to those in the Member Subentry.

3.4.5 ON (SET)

Function

To specify the procedure to be executed when specified DML functions are performed on a set.

General Format

$$\underline{\text{ON}}\ [\underline{\text{ERROR}}\ \text{DURING}] \left[\left\| \begin{array}{l} \underline{\text{ORDER}} \\ \underline{\text{INSERT}} \\ \underline{\text{REMOVE}} \end{array} \right\| \right] \underline{\text{CALL}}\ \text{data-base-procedure-1}$$

Syntax Rules

1. A separate ON clause may be written for each DML function or group of functions.

2. The procedure named by data-base-procedure-1 may be specified in different ON clauses.

General Rules

1. The procedure is invoked whenever a run unit issues one of the specified functions for an occurrence of the set type named in the Set Subentry. If no DML functions are specified, the procedure is invoked whenever a run unit issues an ORDER, INSERT, or REMOVE function for such a set.

2. The procedure is invoked immediately before control is returned to the run unit. If more than one procedure applies to the execution of a DML function, the procedures are invoked in the order in which they are stated in the schema, but a procedure named in an ON clause containing the optional word ERROR will be invoked prior to a procedure named in an ON clause which does not contain the word ERROR. A procedure named in an ON ERROR clause will be entered only if, during the performance of the specified function, the DBMS detects an error that it intends to report.

3.4.6 ORDER

Function

Either to specify the insertion point of a member record within a set and thereby define the order of sequential progression or to declare to the DBMS that it may maintain occurrences of this set type in any order.

General Format

```
ORDER IS { PERMANENT } INSERTION IS
         { TEMPORARY }

  { FIRST                                                                  }
  { LAST                                                                   }
  { NEXT                                                                   }
  { PRIOR                                                                  }
  { IMMATERIAL                                                             }
  {                                                                        }
  { SORTED [ INDEXED [NAME IS index-name-1] ]                              }
  {                                                                        }
  {    { BY DATA-BASE-KEY                                                } }
  {    { BY RECORD-NAME                                                  } }
  {    { WITHIN RECORD-NAME                     [ FIRST ]                } }
  {    { BY DEFINED KEYS [DUPLICATES ARE [ LAST  ] ALLOWED]             } }
  {    {                                        [ NOT   ]                } }
```

Syntax Rules

1. If the SORTED phrase is specified with the DEFINED option, then a KEY clause must be stated for each Member Subentry ıor this set type.

2. The SORTED phrase must not be used if the DYNAMIC clause is used for this Set Entry.

3. If the declaration includes the phrase NAME IS index-name-1, then index-name-1 must not be referenced in any other declarations within the schema.

4. If the SORTED phrase is present the ORDER clause must specify PERMANENT.

General Rules

1. ORDER FIRST refers to the position within the set that immediately follows the owner record; this is a reversed chronological sequencing; the last member record inserted into the set becomes the first member of the set.

2. ORDER LAST refers to the position within the set that immediately precedes the owner record. This is a chronological sequencing; the newest member record becomes the last member in the set.

3. ORDER PRIOR and ORDER NEXT refer to insertion points relative to the member record of the set most recently selected by the run unit. If this record is the owner record, ORDER PRIOR is equivalent to ORDER LAST and ORDER NEXT is equivalent to ORDER FIRST.

4. The SORTED phrase allows specification of a set order based on the record names or the data base keys of the member records of the set, or on the values of the key items specified in the KEY clauses for the member records of the set. The collating sequence is implementor defined.

5. The optional INDEXED phrase, if used, causes the implementor to determine and generate the necessary index or indexes. The index will be controlled by the key items specified in the KEY clauses appearing in the Member Subentries for this Set Entry. A name can be given to the index to enable it to be referenced in the device media control language.

6. The optional WITHIN RECORD-NAME phrase allows records to be sorted without regard to the order of other record types in the set. This does not mean that there is an implied major sort by record type. It means only that when a given type of record is considered independently of any other member record type, it is in sequence by its own sort control key. The sort control keys are specified by the KEY clause for each of the member record types. If the KEY clause is not used for any member record type, the data base keys of the occurrences of that record type are used as ascending key items.

7. The optional DATA-BASE-KEY phrase specifies that the member records of a set are kept in ascending sequence by their data base key.

8. The optional DEFINED phrase specifies that the member records in a set are to be maintained in a single sequence regardless of the number of different member record types specified in the Set Entry. The corresponding sort control keys are specified in the KEY clauses for each member record type.

9. The optional RECORD-NAME phrase specifies that the record names of the member records are used as the major key items. Minor key items are specified by the KEY clauses for each member record type. If the KEY clause is not used for any member record type, the data base keys of the occurrences of that record type are used as ascending key items.

10. In the ORDER clause the words TEMPORARY and PERMANENT control the effect on the set of the ORDER function. If the word TEMPORARY is specified, the effect of the ORDER function may or may not be local to the run unit at the run units option. If the word PERMANENT is specified the effect of the ORDER function can only be local to the run unit. Use of the ORDER function has no effect on the ORDER clause and new records will be added to the set as specified in the ORDER clause, that is, FIRST, LAST , NEXT, PRIOR, IMMATERIAL.

11. If the optional DUPLICATES ARE NOT ALLOWED phrase is specified, the DBMS rejects the insertion into any given set of those member records that have the same nonnull values for the specified key items as a record that is already a member of that set. This may occur during an attempt to store a new member record in the data base or to insert an existing record into a set, or to modify the value of a key item.

12. If the optional DUPLICATES ARE FIRST or DUPLICATES ARE LAST phrase is specified, member records will be inserted into the set sequence before or after, depending on the option specified, any existing member records in the set that have the same values for the specified key items.

13. If the optional DUPLICATES ARE ALLOWED phrase is specified, the insertion point in the set sequence of member records relative to any existing member records in the same set, that have the same values for the specified key items, is unpredictable, unless the action to be taken is controlled by a specified DUPLICATES phrase that is included in a KEY clause of the pertaining Member Subentry.

14. Use of the form ORDER IS IMMATERIAL informs the DBMS that member records participating in an occurrrence of this set type are to be maintained in the order most convenient to the DBMS.

3.4.7 OWNER

Function

To specify the name of a record type, each occurrence of which establishes the existence of an occurrence of the set type named in this Set Subentry.

General Format

$$\underline{\text{OWNER}}\ \text{IS} \left\{\begin{array}{l}\text{record-name-1}\\ \underline{\text{SYSTEM}}\end{array}\right\}$$

Syntax Rules

1. Record-name-1 must be previously declared in a Record Entry.

General Rules

1. A record type may be specified as an owner in more than one Set Entry. It may also be defined as a member in one or more Set Entries.

2. The OWNER IS SYSTEM clause defines a singular set. A singular set has exactly one occurrence and no user specified owner record type.

3.4.8 PRIVACY (MEMBER)

Function

To specify the privacy locks which apply to a member record type of a set type.

General Format

$$\underline{\text{PRIVACY}}\ \text{LOCK}\ \left[\ \text{FOR}\ \left\|\begin{array}{|l|}\underline{\text{FIND}}\\ \underline{\text{INSERT}}\\ \underline{\text{REMOVE}}\end{array}\right\|\ \right]\ \text{IS}\ \left\{\begin{array}{l}\text{literal-1}\\ \text{lock-name-1}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-1}\end{array}\right\}$$

$$\left[\ \underline{\text{OR}}\ \left\{\begin{array}{l}\text{literal-2}\\ \text{lock-name-2}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-2}\end{array}\right\}\right]\ \ldots$$

Syntax Rules

1. A separate PRIVACY clause may be stated for each DML function. However, the same function must not be specified in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more DML functions.

3. All literals must conform to the implementor defined data characteristics for privacy locks.

General Rules

1. The literals and the content of the lock-names are privacy locks to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which, when given access to a privacy key, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR clause is omitted, all literals, lock-names, or procedures apply to all functions included in the format.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the PRIVACY clause is known.

6. If a PRIVACY clause has not been specified for a DML function, then unless other PRIVACY clauses apply, the use of that function on an occurrence of this record type as a member of the set type named in the Set Subentry is without restriction.

7. The privacy locks associated with the various DML functions (INSERT, REMOVE and FIND) must be satisfied in order to execute the respective function on an occurrence of this record type as a member of the set type named in the Set Subentry.

3.4.9 PRIVACY (SET)

Function

To specify the privacy locks which apply to the use of a set type.

General Format

$$\underline{\text{PRIVACY}}\ \text{LOCK}\left[\ \text{FOR}\ \left\|\begin{array}{l}\underline{\text{FIND}}\\ \underline{\text{ORDER}}\\ \underline{\text{INSERT}}\\ \underline{\text{REMOVE}}\end{array}\right\|\ \right]\ \text{IS}\ \left\{\begin{array}{l}\text{literal-1}\\ \text{lock-name-1}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-1}\end{array}\right\}$$

$$\left[\ \underline{\text{OR}}\ \left\{\begin{array}{l}\text{literal-2}\\ \text{lock-name-2}\\ \underline{\text{PROCEDURE}}\ \text{data-base-procedure-2}\end{array}\right\}\right]\ldots$$

Syntax Rules

1. A separate PRIVACY clause may be stated for each DML function However, the same function must not be specified in more than one PRIVACY clause.

2. The same literal, lock-name, or data-base-procedure may be specified for one or more DML functions.

3. All literals must conform to the implementor defined data characteristics for privacy locks.

General Rules

1. The literals and the content of the lock-names are privacy locks to be matched with the pertinent privacy key. The procedures named are privacy lock procedures which when given access to a privacy kev, either return a yes or no result, or do not return at all.

2. By their appearance in a PRIVACY clause, lock-names are treated as data items with implementor defined data characteristics.

3. If the optional FOR phrase is omitted, all literals, lock-names, or procedures apply to all functions included in the format.

4. A value of null for any literal or lock-name is equivalent to the omission of the entire clause in which it occurs.

5. Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the PRIVACY clause is known.

6. If a PRIVACY clause has not been specified for a DML function, then unless other PRIVACY clauses apply, the use of that function on occurrences of the set type being described is without restriction.

7. The privacy locks associated with the various DML functions (ORDER, INSERT...) must be satisfied in order to execute the respective function on any occurrence of the set type being described.

3.4.10 SEARCH

Function

To declare to the DBMS that for each occurrence of a specific set type, an index is required of all of its member records of a given type.

To specify the type of indexing and the data items for which indexing is required.

To optionally check and reject the insertion within the same set of those member records that contain duplicate values for the specified search keys.

General Format

```
SEARCH KEY IS data-base-identifier-1 [,data-base-identifier-2]...

 ┌        ┌CALC                                 ┐ ┐
 │ USING ┤ INDEX [NAME IS index-name-1]         ├ │ DUPLICATES ARE [NOT] ALLOWED
 │        └PROCEDURE data-base-procedure-1      ┘ │
 └                                                ┘
```

Syntax Rules

1. The data-base-identifiers must refer to data items included in the record type named in the Member Subentry of this Set Entry.

2. If the declaration includes the optional NAME phrase then index-name-1 must not be referenced in any other declarations within the schema.

General Rules

1. A search key may aopear as an argument in a SELECTION clause or FIND function. Where such arguments have been declared with a SEARCH clause, the indexing provided will be used to speed the required search.

2. The data items specified in one SEARCH clause will be concatenated to form a single search argument. The SFARCH clause must be repeated for each search argument for which indexing is to be provided.

3. If the optional word CALC is specified in the USING option of the SEARCH clause, the DBMS's standard key transformation algorithm is used in the selection of the sought record.

4. If the USING phrase is not specified or if the optional word INDEX is specified, the DBMS's standard indexing mechanism is used in the selection of the sought record. The NAME phrase is provided to simplify references to specific indexes in the device media control language.

5. If the optional word PROCEDURE is specified, the procedure named by data-base-procedure-1 is used in the selection of the sought record.

6. If the DUPLICATES ARE NOT ALLOWED phrase is used, the DBMS will reject the insertion into any given set of those member records with duplicate values for the specified search keys. This may occur during an attempt to store a new record in the data base, or to insert an existing record into a set or to modify the value of a data item declared to be a search key.

7. If the DUPLICATES ARE ALLOWED phrase is used, the record selected on the basis of an argument specified as a search key will be the first record encountered which satisfies the argument.

3.4.11 SELECTION

Function

To define the rules governing the selection of the appropriate occurrence of a set type for the purpose of inserting or accessing a member record.

General Format

Format 1

```
SET SELECTION [FOR set-name-1] IS

    THRU set-name-2 OWNER IDENTIFIED BY
              ⎧ SYSTEM                                                        ⎫
              ⎪ CURRENT OF SET                                                ⎪
              ⎪ DATA-BASE-KEY[EQUAL TO {data-base-identifier-1}]              ⎪
              ⎪                        {data-base-data-name-1  }              ⎪
              ⎨                        ⎡          {data-base-identifier-2} ⎤  ⎬
              ⎪ CALC-KEY               ⎢EQUAL TO  {data-base-data-name-2 } ⎥  ⎪
              ⎪                        ⎢  [,data-base-identifier-3]...     ⎥  ⎪
              ⎪                        ⎣  [,data-base-data-name-3 ]        ⎦  ⎪
              ⎪ MEMBER record-name-1 SELECTION                                ⎪
              ⎩                                                               ⎭
   ⎡THEN THRU set-name-3                                                 ⎤
   ⎢       ⎧WHERE OWNER IDENTIFIED BY data-base-identifier-4⎫            ⎥
   ⎢       ⎪    ⎡           {data-base-identifier-5         }⎤ ⎪          ⎥
   ⎢       ⎨    ⎢EQUAL TO   {data-base-data-name-4          }⎥ ⎬...       ⎥...
   ⎢       ⎪    ⎣           {PROCEDURE data-base-procedure-1}⎦ ⎪          ⎥
   ⎣       ⎩                                                ⎭            ⎦
```

Format 2

```
SET SELECTION IS BY PROCEDURE  data-base-procedure-2
```

Syntax Rules

1. Set-name-1 is the name of the set type of whose Set Entry this clause is a part.

2. Data-base-identifier-1 must be declared as a data base key.

3. If the CALC-KEY option is specified, the owner record type of the set type referenced by set-name-2 must have a location mode of CALC. The LOCATION clause must specify the DUPLICATES NOT ALLOWED phrase. Data-base-identifier-2, data-base-identifier-3,... must have identical data characteristics to those of the calc keys as specified in the LOCATION clause.

4. Data-base-identifier-4 is a declared data item in the owner record of set-name-3.

5. Data-base-identifier-5 or the result of the procedure named by data-base-procedure-1 must have identical data characteristics to those of data-base-identifier-4.

6. Set-name-2, set-name-3,... must form a continuous path in the sense that the owner of set-name-3 is a member of set-name-2..., with set-name-2 as a start point, or root. In that path the same set name must not appear more than once, nor may it appear in any SELECTION clause referenced as a result of the use of the MEMBER option except as the subject of the referenced SELECTION clause. The last set type named must be the subject of the Set Entry of which this clause is a part.

7. The data items referenced by data-base-identifier-4 must together uniquely identify the owner of set-name-3. A DUPLICATES NOT ALLOWED phrase must be declared for those data items considered as a group in the KEY clause, in the MEMBER clause, or in a SEARCH clause for the owner of set-name-3 as a member of set-name-2.

8. If the EQUAL TO phrase of the DATA-BASE-KEY or the CALC-KEY phrase is not stated, a location mode of DIRECT or CALC respectively must be specified for the owner record type of set-name-2.

9. If the SYSTEM option is stated, set-name-2 must have an OWNER IS SYSTEM clause specified in its Set Subentry.

10. Record-name-1 must be declared as a member of set-name-2.

11. If the MEMBER option is used then the SELECTION clause declared for record-name-1 as a member of set-name-2 must either itself not use the MEMBER option or by use of the MEMBER option must refer, possibly through several other SELECTION clauses, to a separate SELECTION clause which does not use the MEMBER option.

General Rules

1. Data-base-data-name-1 is treated as a data base key.

2. Data-base-data-name-2, data-base-data-name-3,... are treated as data items having identical data characteristics to those of the CALC keys as specified in the LOCATION clause in the Record Entry for the owner record type of the set type referenced by set-name-2.

3. Data-base-data-name-4 is treated as a data item having identical data characteristics to those of data-base-identifier-4.

4. The SELECTION clause for the appropriate member record type and set type combinations will govern the selection of a specific set for the purpose of inserting or accessing a member record.

5. Prior to the execution of any function involving selection, nonnull values must be supplied for the data items specified in the EQUAL TO phrase(s), or in the LOCATION clause if one is implied.

6. The SYSTEM option causes the DBMS to select from set-name-2 within the data base the singular occurrence that exists for this set type.

7. The CURRENT option causes the DBMS to select from set-name-2 within the data base that occurrence most recently selected by the run unit.

8. The DATA-BASE-KEY option causes the DBMS to select from set-name-2 within the data base that occurrence whose owner record has a data base key equal to the data base key contained in data-base-identifier-1, data-base-data-name-1, or in the absence of the EQUAL TO specification, the parameter specified in the LOCATION MODE IS DIRECT clause for the owner record type.

9. The CALC-KEY option causes the DBMS to select from set-name-2 within the data base that occurrence whose owner record has a CALC key equal to the CALC key contained in data-base-identifier-2, data-base-identifier-3,... or data-base-data-name-2, data-base-data-name-3,... or in the absence of the EQUAL TO specification, the parameters as specified in the LOCATION MODE IS CALC clause for the owner record type.

10. The MEMBER option causes the DBMS to select from set-name-2 within the data base that occurrence as specified by the SELECTION clause for record-name-1 as a member of set-name-2.

11. When only the THRU set-name-2 phrase is specified, the occurrence of set-name-1 is selected as described in General Rules 6 through 10.

12. When both the THRU set-name-2 phrase and one or more THEN THRU phrases are specified the occurrence of set-name-2 is selected as described in General Rules 6 through 10. For each subsequent set in the path, the THEN THRU phrase causes the DBMS to select the owner record of set-name-3... in its

capacity as a member of the selected occurrence of the previously named set type such that the owner record has a value for data-base-identifier-4 equal to the value contained in data-base-identifier-5, data-base-data-name-4, or the value which is the result of the procedure named by data-base-procedure-1.

13. Format 2 applies when the set to be selected is identified by the procedure named by data-base-procedure-2. The procedure must uniquely identify an occurrence of the set type defined by the Set Entry of which this Member Subentry is a part.

3.4.12 SET

Function

To name a set type in the schema, that is to specify a generic name for all occurrences of the set type in the data base.

General Format

SET NAME IS set-name-1

Syntax Rules

1. Set-name-1 must be unique among the set-names of the schema.

General Rules

None.

☆ U. S. GOVERNMENT PRINTING OFFICE :1974--542-653/115

FORM NBS-114A (1-71)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS HB-113	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

CODASYL Data Description Language Journal of Development
June 1973

5. Publication Date
January 1974

6. Performing Organization Code

7. AUTHOR(S) CODASYL Data Description Committee
Local Contact: John Berg x3485

8. Performing Organization

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.
640-1113

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

Same as No. 9.

13. Type of Report & Period Covered
Interim

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

This Journal of Development reports the work of the CODASYL Data Description Language Committee. The Committee was assigned the tasks of establishing "ways to aid the functions of data administration and systems administration". The Committee's charter included, "the provision of specifications for the declarations required to establish and maintain data base structures". As a step towards this purpose, the Journal contains three sections which treat the Background and History of the Data Description Language Committee, Major Concepts, and the specifications of the Data Description Language. The Committee based its work, in part, on the 1971 report of the Data Base Task Group Report.

The approved Data Description Language specifications contain the syntax and semantic rules that permit the description of the structure and contents of a data base in a language independent of, but common to, many other high level programming languages. The language specifications will have a significant impact on the development of functionally compatible data base management systems and will increase the portability of programs between different computer systems.

Though not part of the approved language specifications, the presentation of the major concepts will help in the understanding of the specifications. Similarly, the background and history information will help explain the evolutionary growth of the Data Description Language.

17. KEY WORDS (Alphabetical order, separated by semicolons)
COBOL; CODASYL; Data Base Administration; Data Base Management; Data Base Task Group; Data Description Language.

18. AVAILABILITY STATEMENT

☒ UNLIMITED.

☐ FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NTIS.

19. SECURITY CLASS (THIS REPORT)
UNCLASSIFIED

20. SECURITY CLASS (THIS PAGE)
UNCLASSIFIED

21. NO. OF PAGES
155

22. Price
$1.[illegible]

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH reports National Bureau of Standards research and development in physics, mathematics, and chemistry. Comprehensive scientific papers give complete details of the work, including laboratory data, experimental procedures, and theoretical and mathematical analyses. Illustrated with photographs, drawings, and charts. Includes listings of other NBS papers as issued.

Published in two sections, available separately:

- **Physics and Chemistry (Section A)**

Papers of interest primarily to scientists working in these fields. This section covers a broad range of physical and chemical research, with major emphasis on standards of physical measurement, fundamental constants, and properties of matter. Issued six times a year. Annual subscription: Domestic, $17.00; Foreign, $21.25.

- **Mathematical Sciences (Section B)**

Studies and compilations designed mainly for the mathematician and theoretical physicist. Topics in mathematical statistics, theory of experiment design, numerical analysis, theoretical physics and chemistry, logical design and programming of computers and computer systems. Short numerical tables. Issued quarterly. Annual subscription: Domestic, $9.00; Foreign, $11.25.

DIMENSIONS, NBS

The best single source of information concerning the Bureau's measurement, research, developmental, cooperative, and publication activities, this monthly publication is designed for the layman and also for the industry-oriented individual whose daily work involves intimate contact with science and technology —*for engineers, chemists, physicists, research managers, product-development managers, and company executives.* Annual subscription: Domestic, $6.50; Foreign, $8.25.

NONPERIODICALS

Applied Mathematics Series. Mathematical tables, manuals, and studies.

Building Science Series. Research results, test methods, and performance criteria of building materials, components, systems, and structures.

Handbooks. Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications. Proceedings of NBS conferences, bibliographies, annual reports, wall charts, pamphlets, etc.

Monographs. Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

National Standard Reference Data Series. NSRDS provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated.

Product Standards. Provide requirements for sizes, types, quality, and methods for testing various industrial products. ,These standards are developed cooperatively with interested Government and industry groups and provide the basis for common understanding of product characteristics for both buyers and sellers. Their use is voluntary.

Technical Notes. This series consists of communications and reports (covering both other-agency and NBS-sponsored work) of limited or transitory interest.

Federal Information Processing Standards Publications. This series is the official publication within the Federal Government for information on standards adopted and promulgated under the Public Law 89–306, and Bureau of the Budget Circular A–86 entitled, Standardization of Data Elements and Codes in Data Systems.

Consumer Information Series. Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service (Publications and Reports of Interest in Cryogenics). A literature survey issued weekly. Annual subscription: Domestic, $20.00; foreign, $25.00.

Liquefied Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $20.00. Send subscription orders and remittances for the preceding bibliographic services to the U.S. Department of Commerce, National Technical Information Service, Springfield, Va. 22151.

Electromagnetic Metrology Current Awareness Service (Abstracts of Selected Articles on Measurement Techniques and Standards of Electromagnetic Quantities from D-C to Millimeter-Wave Frequencies). Issued monthly. Annual subscription: $100.00 (Special rates for multi-subscriptions). Send subscription order and remittance to the Electromagnetic Metrology Information Center, Electromagnetics Division, National Bureau of Standards, Boulder, Colo. 80302.

Order NBS publications (except Bibliographic Subscription Services) from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.