decsystem10

DATA BASE MANAGEMENT SYSTEM (DBMS-10) DATA BASE ADMINISTRATOR'S PROCEDURES MANUAL

digital

[illegible]

DEC-10-AAPMA-A-D

DECSYSTEM-10

DATA BASE MANAGEMENT SYSTEM(DBMS-10)

Data Base Administrator's Procedures Manual

This document reflects the software as of version 1 of DBMS-10.

For additional copies, order No. DEC-10-AAPMA-A-D from Software Distribution Center, Digital Equipment Corporation, Maynard, Mass.

digital equipment corporation · maynard. massachusetts

First Printing, September, 1973

CDP	DIGITAL	INDAC	PS/8
COMPUTER LAB	DNC	KA10	QUICKPOINT
COMTEX	EDGRIN	LAB-8	RAD-8
COMSYST	EDUSYSTEM	LAB-8/e	RSTS
DDT	FLIP CHIP	LAB-K	RSX
DEC	FOCAL	OMNIBUS	RTM
DECCOMM	GLC-8	OS/8	SABR
DECTAPE	IDAC	PDP	TYPESET 8
DIBOL	IDACS	PHA	UNIBUS

CONTENTS

FOREWORD

This manual describes DBMS-10 from the point of view of the Data Base Administrator, and as such, it is the reference manual for the system. It is not, though, intended to be a tutorial guide for beginning Data Base Administrators or DBMS-10 users. In addition, it is assumed that the reader has a knowledge of the COBOL language.

The Data Base Administrator is assumed to be familiar with the DECsystem-10 operating system and the editing and debugging languages available to him. Such familiarity must be acquired prior to attempting the undertaking of the role of Data Base Administrator.

Chapter 1 is dedicated to the major concepts of DBMS-10, Chapter 2 to the role of the Data Base Administrator, and Chapters 3 through 5 to the various languages known to the system. Chapter 6 details the data organization and access employed, Chapter 7 deals with the Schema Directory File, and Chapter 8 discusses recovery files and techniques. A number of appendices are provided which contain information essential to the system.

CHAPTER 1

MAJOR CONCEPTS

Since this manual is intended to be the primary reference document for Data Base Administrators working under the DECsystem-10 Data Base Management System (DBMS-10), it is essential that it contain not only the specifications for the languages peculiar to DBMS-10, but that it detail the terms and concepts embodied in these languages in such a way as to assist the Data Base Administrator in fully understanding the specifications themselves.

1.1 INTRODUCTION

A data base is a grouping or collection of interrelated data which has been structured and linked to permit the referencing and accessing of the data contained therein without regard to the physical means of storage. In other words, the program, and not the sequence in which data is stored, determines which data is to be accessed, when, and by whom.

As these data bases increase both in use and sophistication, their creation and management must be relinquished by individual application programmers in favor of greater coordination and centralized control. It has become too costly and/or impractical, in most instances, for individual programmers to create data bases for single applications on a one-to-one basis. The need at present is for data bases which are suitable for processing by, and available to, multiple applications.

The objective in developing DBMS-10 was to make this possible by providing features which:

(1) Allow data to be structured in the manner most suitable to each application, regardless of the fact that some or all of that data may be used by other applications; such flexibility being achieved without requiring data redundancy.

(2) Allow more than one run-unit to concurrently retrieve the data in the data base even while one run-unit is updating it.

(3) Provide and permit the use of a variety of search strategies against an entire data base or portions of a data base.

(4) Provide protection of the data base against unauthorized access of data and from destructive interaction of programs.

(5) Provide for centralized capability to control the physical placement of data.

(6) Provide device independence for programs.

(7) Allow the declaration of a variety of data structures ranging from those in which no connection exists between data items to network structures.

(8) Allow the user to interact with the data while being relieved of the mechanics of maintaining the structural associations which have been declared.

It is important to note that the Data Manipulation Language is not a universal language. Rather, it is an enhancement of COBOL and it can thus be categorized as a host language system. As such, its level of procedurality is about equal to that of COBOL and thus it is appropriate for use in programming that large class of problems for which COBOL is the most used and most suitable language.

1.2 DBMS-10 LANGUAGES

DBMS-10 provides three languages which function to allocate media space, to describe a data base or subset of a data base, and to transfer data between a data base and program. There are the Device Media Control Language (DMCL), the Data Description Languages (DDLs), and the Data Manipulation Language (DML), respectively.

1.2.1 The Device Media Control Language (DMCL)

The DMCL is the language used by the Data Base Administrator to allocate storage space on mass storage devices for the physical placement of a data base. This is accomplished by equating a data base area name to a monitor filename, and specifying how large a file must be created to contain the area. Actually, the DMCL is not a self-contained language; it is an extension to the Schema DDL, and it must pass through the DDL processor at the same time that the Schema DDL is processed.

1.2.2 The Data Description Languages (DDLs)

The DDLs are the languages used for describing a data base (schema definition), or that portion of a data base known to a program (sub-schema definition). These descriptions provide the definitions for the data-items, data-aggregates, records, sets, and areas included in the data base, as well as the relationships that exist and must be maintained between occurrences of these elements in the data base.

A data-item is the smallest unit of named data in a data base. An occurrence of a data-item is represented by a value. Data-items may be alphanumeric or numeric (either fixed or floating point).

A data-aggregate is a named collection of data items within a record. It may be a one dimensional ordered collection of data-items with identical characteristics (a vector), or it may be a repeating group--i.e., a collection of data-items, vectors, and/or other repeating groups which occur an arbitrary number of times.

A record is a named collection of data-items and/or data-aggregates which is viewed as being contiguous. When a record is described in a schema, this description also defines the record as a record type--i.e., the name of the record and the record type are synonymous. An arbitrary number of occurrences of a record can be present in a data base for any record description in the schema for that data base. This distinction between the actual occurrences of a record and the type of the record is an important one.

A set is a named collection of record types which describes the logical relationships existing between these record types. Every set type must have one record type declared as its owner record and can have zero, one, or many record types declared as member records. Each occurrence of a set must contain one occurrence of its owner record type, but it may contain an arbitrary number of occurrences of each of its member record types. Set occurrence ordering is independent of the physical placement of particular records.

An area is a named sub-division of the addressable storage space in a data base and may contain occurrences of both records and sets. Areas may be opened by a run-unit with usage modes which permit, or forbid, concurrent run-units to open the same area. An area may be declared in a schema to be a temporary area--i.e., each run-unit opening it is given a unique occurrence of the area which is made available for reuse at the termination of the run-unit.

The concept of area permits the Data Base Administrator to subdivide a data base rather than viewing the data base as a single entity. Efficient storage and retrieval can be accomplished through the use of areas. This concept, when used properly, fosters optimization of data base access since the run-unit becomes interested in only a select region of the data base. Areas also provide a convenient unit for recovery, since duplication or backup can be performed selectively--e.g., areas with minimal updating then would have to be FAILSAFEd less often than those undergoing constant updating. Likewise, some information might be maintained for retrieval only--e.g., income tax tables. Thus areas also provide a convenient means for separating data according to usage.

A data base consists of all the record occurrences, set occurrences, and areas which are controlled by a specific schema. In the case of multiple data bases, there must be a separate schema for each data base. Furthermore, the contents of different data bases are disjoint, and the intersection of two or more data bases is not permitted. Different data bases and nondata-base files may exist, though, on the same physical and logical storage structures known to the DECsystem-10 monitor.

A schema consists of Schema DDL entries (see Chapter 4) which describe all of the areas, set types, record types, data-items, and data-aggregates as they exist in the data base. A schema is <u>not</u> data itself, but a description of data. It is established and maintained by the Data Administrator as a main storage file which imposes discipline over the entire data base. It can be extended as the data base grows and refines.

A sub-schema selects those areas, set types, record types, data-items, and data-aggregates known to a given program(s) and describes them in the form in which they are known to those specific programs using a Sub-Schema DDL. Since the only host language currently associated with DBMS-10 is COBOL, only the COBOL Sub-Schema DDL has been developed and is discussed in Chapter 5.

1.2.3 The Data Manipulation Language (DML)

The DML is the language used by programmers to access data in the data base. It is not a complete language by itself, but is a host-language extension. In other words, the DML relies on the host language (in the current situation, COBOL) to offer the framework from which the DML can provide the interface with the data base. In an application program, the DML commands and the host language statements coexist freely, and the distinction between them is merely conceptual. From the programmer's point of view, then, he is using one, unified language that has the capabilities of both the host language and the DML. The host language, then, is the language used to manipulate data in primary storage, and the DML is the interface language with the data base. Chapter 3 of the DBMS-10 Programmer's Procedures Manual contains the specifications for the COBOL DML.

1.2.4 Language Relationships

As already stated (Section 1.2.1), the DMCL is an extension of the Schema DDL. Whereas the latter describes the physical and logical layout of the data base, the former specifies physical storage space necessary to accommodate the data base. The relationship between DDL and DML is the relationship between declarations and procedure. The declarations impose a discipline over the executable code and are to a large extent substitutes for procedures written in the DML and the host language; that is, they are implicit procedures which may be invoked by the execution of DML commands.

1.3 OPERATIONAL ARCHITECTURE

The DBMS-10 operational environment includes the Data Base Control System (DBCS), the User Working Area (UWA), and the run-unit.

1.3.1 The Data Base Control System (DBCS)

The DBCS is the object-time module of DBMS-10, controlling the placement and access of data in the data base. It provides the interface between the application program and the data base by associating monitor relative address blocks within the data base media files with page locations in the data base.

It must be possible for the DBCS 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 be assigned by the DBCS to each and every record occurrence in the data base. This unique identifier is known as a database key.

A database key is assigned by the DBCS to a record occurrence when it is stored for the first time in the data base. It is assigned in accordance with:

(1) The declarations for that record in the schema;

(2) Arguments, if any are required, supplied by the run-unit adding the record to the data base.

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

The permanence of database keys is an important element in the integrity of the data base. Database keys are made available to and may be saved by run-units and:

(1) Used for direct accessing;

(2) Referenced later in the execution of the same run-unit;

(3) Be re-inputted to a subsequent run-unit in which they are referenced.

Database keys are mapped by the DBCS to physical addresses. It is important to note, however, that positional record selection expressions which specify an area-name are interpreted in terms of database keys, (that is, NEXT within area-name means the record occurrence with the next higher database key in the named area).

1.3.2 The User Working Area (UWA)

Conceptually, the UWA is a loading and unloading zone where all data provided by the DBCS in response to a call for data is delivered and where all data to be picked up by the DBCS must be placed. Each program has its own UWA. The data in the UWA of a program is not disturbed except in response to the execution of a DML command or by the user program's host language procedures. There is no implication that UWA locations are contiguous.

The UWA is set up by the DBCS in accordance with the invoked sub-schema. Each data-item included in the sub-schema will be assigned a location in the UWA and may be referenced by its name as declared in the sub-schema. Data-items included in the data base, but not in the sub-schema invoked, cannot be referenced.

The DBCS must also provide for a number of System Communication Locations (SCL). These locations are used for run-unit/system interaction and are assigned space by the DBCS. The most important of these locations are the currency status indicators which are defined in Section 1.6. Seven other locations are also provided: AREA-NAME, RECORD-NAME, ERROR-STATUS, ERROR-SET, ERROR-RECORD, ERROR-AREA, and ERROR-COUNT.

1.3.3 The Run-Unit

In this manual, the words 'run-unit' and 'program' are often used interchangeably as if there were a one-to-one correspondence between run-units and programs. However, strictly speaking, a program is a set or group of instructions while a run-unit is an execution of one or more programs where the precise correspondence between programs and run-units depends on the operating system. The important fact to note is that regardless of which of these two terms is used in the manual there is exactly one UWA, one set System of Communication Locations, and one set of Currency Status Indicators per run-unit. This means that, from the point of view of DBMS-10, each run-unit is a separate entity which it is servicing.

1.4 THE SCHEMA AND THE SUB-SCHEMA

1.4.1 Concepts

The concept of separate schema and sub-schema 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:

(1) 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 be important to ease the writing, debugging, and maintaining of programs.

(2) A program is limited to the subset of the schema that is known to it via its sub-schema. To a large extent, this automatically ensures the privacy and integrity of the rest of the data base from that program.

1.4.2 Variations

A sub-schema may differ from a schema of which it is a subset in several important respects:

(1) At the data-item level:

-- Descriptions of specific data items may be omitted.
-- The ordering of data-items may be changed.

(2) At the data-aggregate level:

-- Descriptions of specific data-aggregates may be omitted.
-- The ordering of data-aggregates may be changed.

(3) At the record level:

-- Descriptions of specific record types may be omitted.

(4) At the set level:

-- Descriptions of specific set types may be omitted.

(5) At the area level:

-- Descriptions of specific areas may be omitted.

A sub-schema must, however, be a consistent and logical subset of the schema from which it is drawn. Specific rules are included in Chapter 5.

The following additional points are also important to an understanding of the concept of the schema and sub-schema.

(1) An object version of the source code schema may be "compiled" independently of any user program or any sub-schema.

(2) Object versions of a source code sub-schema may be "compiled" independently of any user program and stored in a library.

(3) An arbitrary number of sub-schemas may be declared on the basis of any given schema.

(4) The declaration of a sub-schema has no effect on the declaration of any other sub-schema and sub-schemas may overlap one another.

(5) Each sub-schema must be named.

(6) A user program invokes a sub-schema.

(7) The same sub-schema may be invoked by an arbitrary number of programs.

(8) Only the areas, records, data-items, and sets included in the sub-schema invoked by a program may be referenced by that program.

1.4.3 Placement Control

The Schema DDL provides for control over the relative placement of records. The objective of providing for control of relative placement of records is to increase efficiency by advising DBMS-10 of anticipated use 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 DBMS-10. Where more than one such area is specified, final selection takes place in a run-unit. The Schema DDL also includes a clause which causes record occurrences being added to the data base to be stored near some procedure, thereby reducing overall access times.

The fact that the Schema DDL permits control over relative placement of records does not necessarily have any physical connotations. What is required of DBMS-10 in response to this type of declaration is for it to attempt to allocate storage space in a manner which tends to optimize access times. The ability to achieve some optimization of access times is important when considered in the environment of the delays in accessing data occasioned by arm movement and rotational delay in many of the currently available direct access storage devices.

1.4.4 Device Independence

All interfaces of the DML with data in the data base are at the symbolic or logical level. Application programs written using the DML are thus device independent. In addition, programs receive and place all data in their User Working Area and the DBCS is responsible for all aspects of physical input/output including buffering. No Schema DDL entry includes references to the physical devices or media space. Thus, a schema written using the Schema DDL is a logical description of the data base and is not affected by the devices and media used to store the data. The data base may, therefore, be stored on any combination of secondary storage devices which are supported by DBMS-10.

1.5 PROTECTION OF DATA

1.5.1 Privacy and Integrity

The Schema and Sub-Schema DDLs and the DML include the provision for the protection of data in the environment of a shared data base--i.e., one which contains data relevant to, and shared by, multiple programs or applications. In this type of environment, two kinds of protection are required:

(1) Protection against unauthorized access of data, for which the term privacy is used in DBMS-10.

(2) Safeguarding the data from destructive interaction of programs, 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. This protection, needless to say, cannot guard against unsocial behavior on the part of individual programs. If, for example, a program is authorized to access and delete a particular record and does so without regard for the fact that this same record is the owner of a set which has members required by other programs, DBMS-10 can offer no protection. Such action is in reality a logical error and can only be avoided by the Data Base Administrator creating an awareness among programmers of their actions on others.

1.5.2 Privacy of Data

The data in the data base is protected through a mechanism of privacy locks which are specified in the schema and sub-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 declared at the schema level, the sub-schema level, and the area level. A privacy lock is a single value, up to thirty characters in length, and may be a constant or the value of a variable. A privacy key is a value (again, either constant or variable) which is simply matched against a privacy lock value.

In case of a violation, DBMS-10 aborts the run-unit and records the violation. When repeated violations occur, the Data Base Administrator should take positive action to remedy the situation (see Section 2.2). While protection of privacy is the foremost use of privacy locks and keys, it is by far not the only one. This mechanism can also be used, for example, to help ensure the consistency of interrelated data, and to prevent errors by locking out clearly inconsistent, meaningless, or incorrect actions.

1.5.3 Integrity of Data

When run-units are permitted to interact with the same data concurrently, they require protection against each other. No provision has been included in the DML for a run-unit to selectively lock record occurrences and make them unavailable to concurrent run-units. Had such a feature been included, then a deterioration of the performance of DBMS-10 would have resulted.

Provision has been included, though, for giving a run-unit exclusive or protected update rights over one or more areas. No concurrent run-unit can gain access to the areas over which a run-unit has acquired exclusive update rights. Protected update is a less restrictive form of exclusive update which prevents concurrent update but allows concurrent retrieval. These various rights are obtained by executing the OPEN imperative in a program which specifies the desired right, and they are relinquished simply by closing the area.

The exclusive update option means that changes to an area are being done in place and therefore no other run-unit can be permitted to access that area. If the run-unit aborts before closing the area, the area cannot be accessed again. To guard against this, the Data Base Administrator should keep at least one additional copy of every area as backup. The protected update option means that changes are being made in a separate area that is logically (but not physically) equivalent to the area being updated. These changes will only be physically made when the run-unit performing the update relinquishes control by closing the area. If the run-unit aborts before closing the area, all of its changes are discarded.

1.6 RECORDS

A record--i.e., an occurrence of a record type--is viewed as a contiguous collection of data stored in the data base. The description of record type in the schema includes a location mode which determines the placement of the occurrences of that record type in the data base and the form of reference to be used in order to access these occurrences.

1.6.1 Location Mode

The Schema DDL provides for the following access methods to be specified in the location mode clause for each record entry appearing in the schema.

1.6.1.1 DIRECT Location Mode - Retrieval of an occurrence of the record type is based on the unique identifiers (database keys) assigned by the DBCS to each record occurrence in the data base. In this method to be used, the database keys of the record to be selected must be made available by the run-unit to the DBCS and thus must have been saved previously by the run-unit. This is possible because the database keys of the current record of the run-unit are always made available to the run-unit by the DBCS and are stable for the life of the record.

1.6.1.2 CALCULATION Location Mode - Retrieval is based on the values supplied by the run-unit for the data-names which are contained in the sought record and which have been declared as CALC keys. The DBCS transforms the values so provided into a unique identifier and retrieves the record on the basis of that identifier. The algorithm used in this transformation is discussed in Chapter 6.

1.6.1.3 VIA Location Mode - Retrieval depends on the relationships which have been established for the sought record by the DBCS on the basis of the set declarations in the schema. Functionally, two separate steps are involved:

(1) Selection by the DBCS of the appropriate occurrence of the named set;

(2) Selection of the sought record from among the member record occurrences of the selected set occurrence.

The process of set occurrence selection is controlled by declarations for the sought record in its capacity as a member record of the named set. These declarations specify:

(1) The strategy to be used for selection.

(2) The data names which the run-unit must initialize with the values that will be used by DBMS-10 for set occurrence selection. Where, however, the strategy is simply that set selection will be handled by user programs, no data names can be specified.

Set occurrences are selected by means of selecting their owner record occurrences, and a strategy specified to the DBCS must include sufficient information for a unique owner record occurrence, and thus set occurrence, to be selected. When the owner record occurrence is not unique on the basis of the selection criteria specified for it and a hierarchy of sets exists, the strategy must include selection criteria for the owner record itself in its capacity as a member of another set, and so on up the hierarchy until sufficient uniqueness is established. From the viewpoint of the DBCS uniqueness is achieved:

(1) When an owner record is to be selected by the user program or when it is to be selected independently of its set associations; that is, using the DIRECT or CALCULATION access method.

(2) When occurrence selection involves a cycle and there is a starting point for the cycle, in the sense that the owner of at least one set in the cycle is selected independently of the other sets in the cycle. It may be selected as above or through its membership in a set not involved in the cycle. Occurrences of successive sets in the cycle can then be selected based on selection criteria for those sets. Selection terminates before a second occurrence of the starting point set is selected. That is, the selection process proceeds in the sequence of the cycle and does not traverse the cycle more than once.

1.6.2 Accessing of Records

While the Schema DDL provides the means to specify the access methods required at the record level and at the set level, the DML provides commands which allow interaction with such specifications and with any relationships which have been declared. The following items are important to an understanding of the approach taken in the accessing of records.

(1) The DML provides for the selection of individual records.

(2) Depending on the entry in the schema for a record, a record occurrence may be accessed on the basis of any of the data-item values in the record, or on the basis of a concatenation of such data-items values.

(3) The DML is not an inquiry language.

(4) A record may always be accessed through any of the sets in which it participates as a member.

(5) A record may always be accessed by a complete scan of an area.

(6) The DML distinguished between the selection of a record and its delivery to a run-unit.

(7) A record is selected by means of a FIND command and its associated record-selection-expression. The record most recently selected is known as the current record of the run-unit.

(8) Values of data-items from a record are made available to a run-unit in its UWA. The data-items that appear in a run-unit's UWA depend on the invoked sub-schema.

(9) All of the available methods of accessing records may be used in a given program.

(10) All of the available methods of accessing records may be applied to the same data base or area.

1.6.3 Record-Selection-Expressions

As stated earlier search arguments for selecting records from a data base are provided by run-units in the form of record-selection-expressions. There are three main types of record-selection-expressions:

(1) Record-selection-expressions based on identifiers.

(2) Record-selection-expressions based on currency.

(3) Record-selection-expressions that are positional.

1.6.3.1 Record-Selection-Expressions Based on Identifiers - All record occurrences in the database can be retrieved on the basis of their DBCS-assigned unique identifiers. In addition, if the declared location mode is CALCULATED, any record of that type may be retrieved by specifying its record name and the value of the data-item specified as the CALC key.

1.6.3.2 Record-Selection-Expressions Based on Currency - For the duration of each run-unit the DBCS maintains currency status information on the identity of the last record occurrence accessed by the run-unit of:

(1) Each record-type known to the run-unit (that is, current of record-name).

(2) Each set-type known to the run-unit (that is, current of set-name). Note that the current of set-name may be any record which participates in the set whether as an owner or as a member record.

(3) Each area-name known to the run-unit (that is, current of area-name).

(4) Any record-type known to the run-unit (that is, current of run-unit).

Record-selection-expressions based on currency enable the records whose identifiers are currently saved by the DBCS for a run-unit to be retrieved by that run-unit without the specification of any other search arguments.

Currency status indicators are, in effect, place markers kept by the DBCS for a run-unit. At the start of a run-unit its currency status indicators, except for the current of run-unit, may be selectively suppressed by a run-unit.

1.6.3.3 Positional Record-Selection-Expressions - Positional record-selection-expressions are all relative to the current record of set-name, or the current record of area-name. Record-selection-expressions which are relative to the current record of set-name permit, for the occurrence of the set named which is identified by the current record of that set-name:

(1) The member records to be retrieved in the logical order declared for that set, either in the NEXT direction or in the PRIOR direction from the current record of the set.

(2) The OWNER, the FIRST, the LAST, or the nth record of the set to be retrieved.

Record-selection-expressions which are relative to the current record of area-name permit the NEXT, the PRIOR, the FIRST, the LAST, or the nth record in the area named to be retrieved. Sequence, as it relates to areas, is in terms of the unique identifiers of the record occurrences present in the area. That is, sequence is defined by values of database keys, the collating sequence of which is implementor-defined.

Positional record-selection-expressions based on set name are an important tool for traveling the logical relationships established by means of set declarations. Such declarations provide a logical road map of the data base. Record occurrences which participate in more than one set are junction points. Positional record-selection-expressions are the steering mechanism which permit branching off in any direction from such junction points.

1.7 SETS

In order for DBMS-10 to allow data to be structured in the manner most suitable to each application without requiring data redundancy, it is necessary to represent the relationships between records by methods other than physical juxtaposition of these records. The Schema DDL provides the facility to declare structures through the medium of the set. The set is, in effect, a building block which allows various data structures to be built. DBMS-10 provides three data structure representations--

(1) Sequential structures (Figure 1-1)
(2) Tree structures (Figure 1-2)
(3) Network structures (Figure 1-3)

The DBMS-10 user is able not only to treat the case of one-to-many relationships, but he can now handle the more complex many-to-many relationships arising out of real-life situations. In addition, the absence of structuring may be represented by declaring records in the schema which do not participate in sets.

It will be assumed that the Data Base Administrator has a general working knowledge of data structuring and linking. This is essential if he is going to create efficient and realistic data bases.

1.7.1 Characteristics of Sets

The following characteristics are relevant to developing an understanding of the concept of a set.

(1) A set is named collection of record types.

(2) An arbitrary number of sets may be declared in a schema.

(3) Each set must be named and must have one owner record type and can have zero, one, or many member records declared for it in the schema.

(4) Each set must have a SET ORDER specified for it in the schema (see Section 1.7.2).

(5) Any record type may be declared in the schema as the owner record type of one or more sets.

(6) Any record type may be declared in the schema as a member record type of one or more sets.

(7) Any record may be specified as both an owner record in one or more sets and a member record in one or more different sets.

(8) The capability for a record to participate as both owner and member in the same set is not supported by DBMS-10.

(9) A record occurrence cannot appear in more than one occurrence of the same set.

(10) A set occurrence is a collection of one or more logically related record occurrences.

(11) Each occurrence of a set includes one occurrence of its owner record. In fact, the existence of the owner record in the data base is a condition of the existence of the set occurrence and distinguishes that set occurrence from all other occurrences of that set-name.

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

(13) In addition to the occurrence of its owner record, a set occurrence may have an arbitrary number of occurrences of each of the member records declared for it in the schema.

(14) A special type of set which has exactly one occurrence and for which DBMS-10 is the owner may be declared. For convenience, this is known as a singular set.

1.7.2 Ordering of Sets

Each set named in the schema must have a SET ORDER specified for it. The effect of this is to cause DBMS-10 to control, in accordance with the set order specified, the logical order of the member record occurrences within each set occurrence. The logical order of the member records of a set is completely independent of the physical placement of the records themselves. Thus, the same member record occurrences could participate in occurrences of two different sets and be ordered differently in each of those sets.

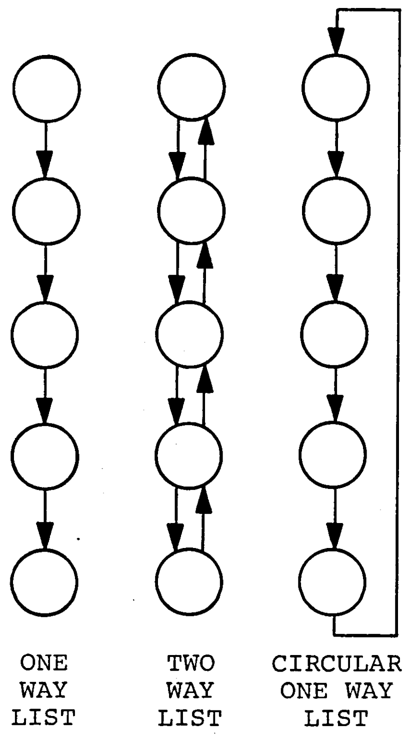


Figure 1-1 Sequential Structures

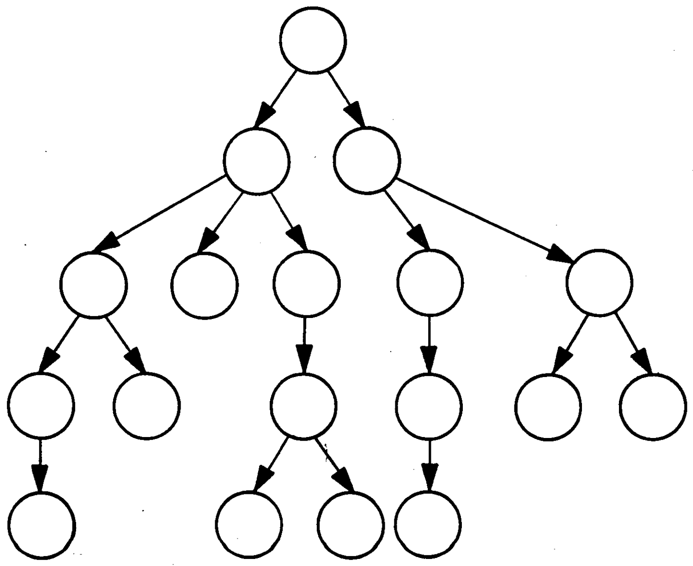


Figure 1-2 Tree Structure

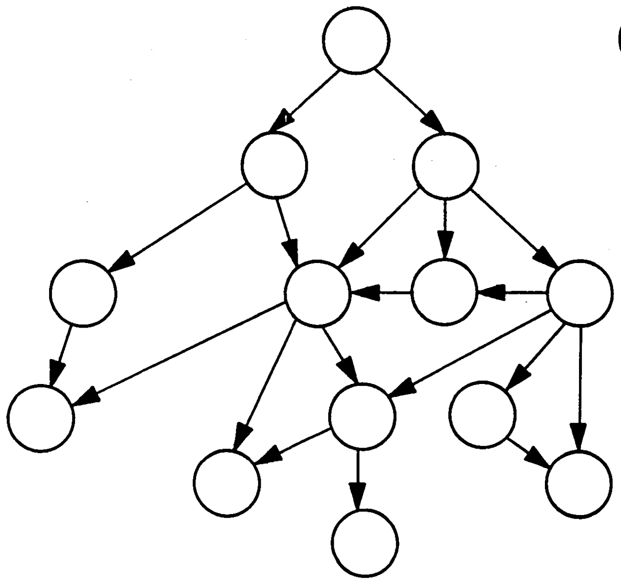


Figure 1-3 Network Structure

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

(1) SORTED 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 unique identifiers, or any combination of these.

(2) In the order resulting from inserting new member record occurrences into the set:

    (a) FIRST, that is, as the immediate successor to the owner record occurrence.

    (b) LAST, that is, as the immediate predecessor to the owner record occurrence.

    (c) NEXT, PRIOR, that is, after or before another record occurrence which is selected by the user program storing or inserting the record in the set.

1.7.3 Types of Membership In Sets

The membership of a record type in a set is declared in the schema to be AUTOMATIC or MANUAL and MANDATORY or OPTIONAL. A record may have different types of membership in different sets.

AUTOMATIC means that membership in the set is established by the DBMS-10 when a record occurrence is stored. That is, whenever an occurrence of a record declared to be an automatic member of a set is added to the data base, it will be logically inserted into (that is, made a current member of) the appropriate occurrences of all the sets in which it has been declared as an automatic member.

The addition to the data base of a record occurrence declared to be a MANUAL member of a set will not cause it to be made a current member of any occurrence of the sets in which it has been declared as a manual member. Manual means that membership in the set is established by a run-unit by means of an INSERT command.

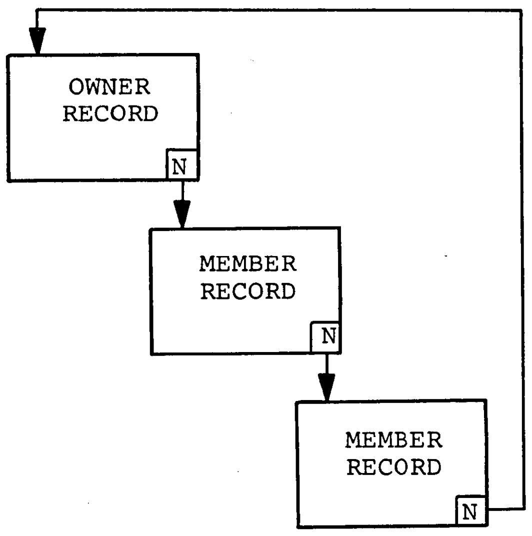
MANDATORY means that, once the membership of a record occurrence in a set is established, the membership is permanent. Its set occurrence may be changed by a MODIFY command, but the record occurrence cannot be removed from the set. If an owner record is deleted, so are all its mandatory members.

OPTIONAL means that the membership of a record occurrence in a set is not necessarily permanent. Its set occurrence may be changed by a MODIFY command or by a REMOVE command followed by an INSERT command. Its membership may be cancelled by a REMOVE or a DELETE command of its owner. A record which is logically removed from any sets in which it participates remains in the data base and is still accessible, though not through any set in which it is no longer a current member.

1.7.4 Set Mode

The current implementation of DBMS-10 provides for one set mode - CHAIN - which is declared in the Schema DDL. For each occurrence of a set declared to have a mode of CHAIN, a chain of pointers is created which can be followed and which provides for serial access to all records in the set occurrence. The pointers are embedded in the records themselves. An illustration of an embedded pointer chain is shown in Figure 1-4. It represents a set occurrence with two member records. The owner record of the set occurrence contains a pointer to the first member record in the set which in turn contains a pointer to the second member which points back to the owner. If the set occurrence contained n member records the chain of pointers would pass through the n member records.

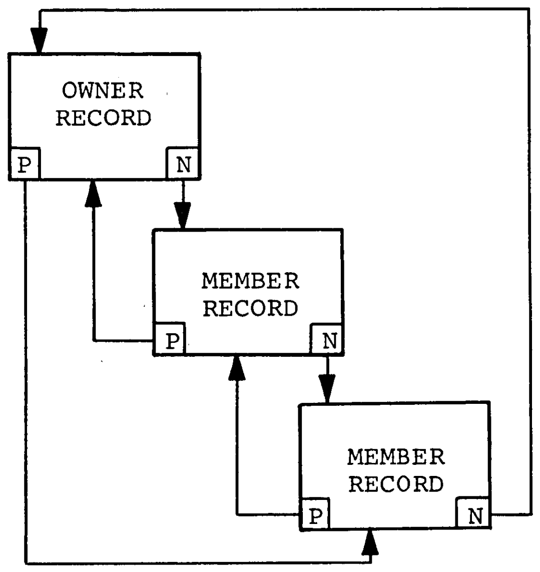
Since Figure 1-4 is a representation of a chain with embedded pointers it is assumed that the records shown contain data as well as pointers. Of course, records may be members in multiple sets. To the extent that the mode of such sets is CHAIN there would then be multiple chains passing through these records (embedded pointers). As a result, when any given record is accessed, a pointer for each chain in which the record participates is available and may be followed. A chain is thus a routing device or junction box. The essential aspect of a chain is that there are pointers, directly linking one record in the chain to the next record in the chain.



N = NEXT POINTER

Figure 1-4 Chain with NEXT Pointers

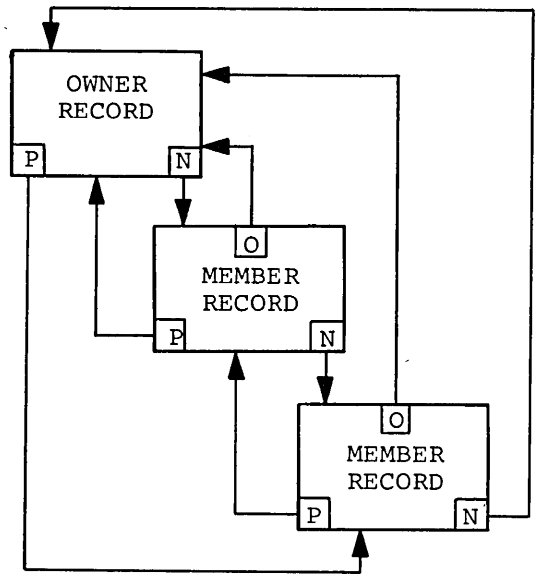
Chains are always processable in either direction from any given record in the chain. However, the linkage provided between the records in a chain is only in the NEXT direction unless the optional clause LINKED TO PRIOR is used. When this clause is used, additional links in the reverse (that is, the PRIOR) direction are also provided. Figure 1-5 is a representation of an embedded chain which is LINKED TO PRIOR.



N = NEXT POINTER
P = PRIOR POINTER

Figure 1-5 Chain with NEXT and PRIOR Pointers

In addition, the occurrences of any of the member record types specified for a set may be declared to be LINKED TO OWNER. This causes the owner record of the set occurrence to be accessible directly from each of the member record occurrences. Figure 1-6 illustrates this.



N = NEXT POINTER
P = PRIOR POINTER
O = OWNER POINTER

Figure 1-6 Chain with NEXT, PRIOR and OWNER Pointers

The unique identifiers (i.e., database keys) assigned by the DBCS for every record occurrence in the data base are used as pointers. Space for a minimum of one pointer (the NEXT pointer) is required for each record and must be assigned by the DBCS for each chain in which a record participates as owner or member. Additional pointers and space are required if the chain is declared to be LINKED TO PRIOR or its members are declared to be LINKED TO OWNER.

1.7.5 Maintenance Of Set Relationships

The establishment and maintenance of relationships between records specified by means of declaring sets in the schema, is a responsibility of DBMS-10. Such maintenance is required whenever:

(1) A record which has been declared as an owner or member in one or more sets is added to or deleted from the data base.

(2) A record is explicitly inserted or removed from a set.

(3) A record is modified in a way which changes its logical position in the set.

Programmers are not involved in the mechanics of this process but must initialize with appropriate values those data-items which are required by DBMS-10 to perform its functions. Such data-items are declared in the schema. Programmers must also ensure that all areas containing any record occurrences affected by a maintenance operation are open for update.

CHAPTER 2

RESPONSIBILITY OF THE DATA BASE ADMINISTRATOR

Because DBMS-10 presupposes an environment where a data base includes data that is shared by many user programs, it is necessary for the schema and the sub-schema to be developed centrally. The data base is, in a sense, a compromise between the needs of the various user programs. Therefore, a means of mediating conflicting needs is required for the data base. This mediation is the prime responsibility of the Data Base Administrator, although it takes on many different aspects.

2.1 DEFINITION AND ORGANIZATION OF THE DATA BASE

2.1.1 Understanding User Requirements

The timeliness, accuracy, and efficiency of the support to be given to users must be explicitly agreed upon between the Data Base Administrator and the users. Users must be educated in the difference between access to specific existing data, information dynamically derived from existing data, and the extensions of the data base to include new data.

The Data Base Administrator must be aware of his organization's long-range plans as well as of long-range needs of the users. For instance, several groups of users might be formulating plans for data bases using interrelated data. The Data Base Administrator should be able to provide common access for these users. Understanding short-range user plans and needs as they pertain to specific application requirements is also necessary. A user, for instance, might require data from an existing data base which could possibly cause conflicts with the current users' interests. These differences must be reconciled. The Data Base Administrator has the responsibility to see that the data base is an effective, efficient tool for all users.

2.1.2 Establishing Data Availability

One of the functions of the Data Base Administrator is to assist users in their search for data to satisfy their application requirements. He should maintain a Data Base Directory (DBD) in which are recorded the record types and set types currently available to users. The DBD will then be the initial source for information relative to data availability. Should some data elements be not available within the existing data base, the Data Base Administrator will arrange the interface with the necessary data sources to satisfy the demands of the user.

Additional factors to be included when considering data availability are:

(1) present form and location of data;

(2) access techniques to be used;

(3) intended use of data in relation to its present accuracy, completeness, and timeliness;

(4) need for modification of data;

(5) present authorizing agent for use of data;

(6) cost of providing the data.

2.1.3 Organizing the Data Base

The establishment and definition of data interrelationships is a vital function of the Data Base Administrator. Users will generally imply, in their requests, their logical data needs. Explicitly defined proposals should be given by the Data Base Administrator, and these ought to reflect his knowledge of fore-seeable developments--including the needs of probable related users. The physical structure of the data base must be designed in such a way so as to effectively meet the logical data needs of the users. Efficient physical structuring requires expertise on the part of the Data Base Administrator in translating and effecting logical data relationships. As part of the establishment and definition process, the Data Base Administrator must weigh the advantages of a given set of linkages against its cost in terms of additional space required and in the degradation of data base access performance. A variety of factors must be taken into consideration; among these are:

(1) logical data formats and relationships;

(2) physical data formats and relationships;

(3) access methods;

(4) frequencies of access;

(5) physical storage media requirements;

(6) search strategies.

Sophisticated application specifications thus might require complex data structures involving many interrelationships. Redundant data should be reduced to a minimum, and only permitted where it may be required for the sake of performance.

During the organization phase, the Data Base Administrator builds the schema and the sub-schemas necessary to describe the data base as a whole and subsets of the data base, respectively. Accordingly, he

(1) Employs data structures that model the business or problem. This is provided in the Schema Data Description Language in terms of areas, records, sets, data-items, and data-aggregates (see Chapter 4).

(2) Assigns names in such a manner as to assure their uniqueness.

(3) Selects search strategies based on the needs of the various users of the data base.

(4) Assigns areas to devices/media based on time/space requirements using the Device Media Control Language (see Chapter 3).

(5) Loads the data base.

Through arbitration with the various application areas, the Data Base Administrator is responsible for the building of the sub-schemas used for application programs. As such he

(6) Assigns names and/or synonyms to protect the uniqueness of the names already assigned to the data base.

(7) Selects and structures the proper subset of the data base that must be available to the application programmer.

Detailed explanation of these seven points will be in the next three chapters.

2.2 PROTECTION OF THE DATA BASE

2.2.1 General Considerations

The general considerations for data base protection are:

(1) data base access and manipulation;

(2) data base integrity;

(3) save/recovery/restart;

(4) violation of rules.

It is important that the Data Base Administrator review these considerations and decide the degree to which they apply to his particular environment. Provision for these considerations will enable the Data Base Administrator to be better equipped to protect the data base(s) under his control.

2.2.1.1 Data Base Access and Manipulation - The Data Base Administrator should control the reading (access) and writing (manipulation) of the data base. Positive control must be exercised if there is to be meaningful protection. Following are some activities which should help establish and maintain this positive control.

(1) The policy statements regarding the data base should be published and circulated among the users. These statements must reflect data base usage and help promote a clear understanding regarding the data base with user and operating personnel.

(2) It must be decided who has the right to know and/or the need to know the content of the data as well as its existence. Authority should be determined as to who can read data from the data base, add new occurrences of data to the data base, update or change existing values of data in the data base, and delete data from the data base. Once the authority has been established, it is important to set up proper controls to insure authority violations of the data base do not occur (see Chapters 4 and 5).

(3) In many data bases, certain elements of data are confidential. Consideration should be given to those protections which can be used to provide the required data security; such protections are software protection and physical protection. In DBMS-10, privacy locks and keys are used to restrict and/or control software access to the data base (see Section 1.5, Chapters 4 and 5). Physical protection can take on numerous forms, including (but not limited to) use of dedicated

lease lines, physical separation of data elements, and limiting the publication of information regarding stored data to the appropriate users of the data base.

2.2.1.2 Data Base Integrity - Data in the data base must be accurate, complete, current, and timely. Some of the Data Base Administrator's considerations along these lines are as follows:

(1) The Data Base Administrator should monitor the data base for usage, response, and potential reorganization.

(2) When data errors are found, they should be corrected immediately, and the responsible parties should be notified. The timeliness of error correction cannot be overstressed.

(3) The Data Base Administrator can further protect the data base and help insure data integrity by having a complete audit trail of activity against the data base. Such an audit trail would usually consist of the input transaction or message, a copy of the data base record before update, and a copy of the data base record after update. In the event of data base problems, this audit trail will be useful in determining what occurred and in reconstructing data to its correct state.

(4) Application programs will be limited to the access and manipulation of logical data. Only the Data Base Administrator has the need to know the characteristics, formats, organizations, and relationships of the physical data contained in the data base. Limiting users to logical data access will provide increased data base flexibility and a greater amount of data independence from application programs.

(5) The Data Base Administrator should participate in application program testing to help insure that these programs are working correctly and do not inadvertently or incorrectly alter the data base. This participation could include providing test data bases, approving program usage of the data base, and helping prescribe testing procedures.

2.2.1.3 Save/Recovery/Restart - The Data Base Administrator must take proper steps to insure that the data base can be restored to its proper state in the event of destruction or damage. He may design and plan save/recovery/restart procedures which reflect the particular problems and needs of his installation.

The responsibility of the Data Base Administrator does not cease when the data base has been restored after a failure. He must then identify the causes of the failure, and he should assure that appropriate corrections have been made. Those involved in the use of the data base should be notified that a save/restore/restart procedure was necessitated and that some data in the data base might have to be refreshed. As this is part of data integrity, timeliness in carrying out this aspect is most important.

2.2.1.4 Violation Of Rules - In administering the protection of the data base, it is necessary to know whenever violations of rules, standards, and accesses occur. Whenever a violation occurs, the Data Base Administrator should detect the violation, identify the violator, and report the violation. He must then follow up to insure all problems and violations are satisfactorily resolved. Should repeat violations or failure to correct a problem occur, then the violator should be denied future access to the data base.

2.3 DOCUMENTATION OF THE DATA BASE

In addition to being the organizer of and administrator of the data in the data base, the Data Base Administrator is the prime documentor and educator with regards to DBMS-10 at his installation. He should provide for the recording of procedures, standards, guidelines, and data base descriptions necessary for the proper, efficient, and continuing utilization of the data base. Documentation need be structured for and selectively distributed to the data base administration, the computer operations staff, the application programmers, and any other users. The Data Base Administrator has the responsibility for providing and maintaining adequate documentation of all types including, but not limited to: description of the physical data base (i.e., the Data Base Directory); standards; procedures for data base usage; locks, keys, and user identification; DBMS-10 performance and usage measurements; save procedures; recovery and restart procedures; data base testing facilities; training techniques and guidelines.

All of these documentation activities are important in themselves and should not be taken lightly. A description of each of them follows.

2.3.1 Data Base Directory (DBD)

The DBD should contain narratives and/or diagrams illustrating the data structures, as well as the relationships existing between record types--including data base cross-references. Descriptions should also be given of the attributes of physically stored record types--i.e., location mode, area in which stored, format of data-items, etc. (see Chapter 4). In addition, the physical storage medium, including the location, allocator, and utilization of storage space in terms of areas, as well as the location of the data base FAILSAFE tapes and their creation date.

2.3.2 Standards

Standards should be documented as they are established. Each installation will have its own set of user and DBMS-10 standards. Allowable deviations should also be noted, as well as any controls over these deviations.

2.3.3 Procedures For Data Base Usage

These procedures might include, but are not limited to, relationships between transactions and data bases, and responsibilities of the users and data base personnel. In order to fully accomplish all the tasks assigned to him and his staff, the Data Base Administrator should schedule time when the data base is not available to users in order to perform reorganization procedures (such as reassigning areas to different devices/media), FAILSAFE procedures, and other "house-cleaning" chores. This time is in addition to the normal preventive maintainence time for the computer system as a whole, and the users must be made aware of these activities. Likewise, users who merely want to retrieve information from the data base should know when this information is being updated by other users so that fresh copies of data are always available.

2.3.4 Privacy Locks, Keys, And User Identification

In addition to the normal logging and billing facilities of the computer installation, DBMS-10 provides the means of locking out a user from the data base until he supplies the proper key in his program (see Section 1.4). Such a lock-out mechanism can apply to subsets of the data base (see Section 5.2) as well as to specified access to any area (see Section 4.3). This mechanism is only as effective, though, as the attention paid to it by the Data Base Administrator. All privacy locks and keys are assigned by the Data Base Administrator, and the strictest security must surround their use. They should not, for instance, appear in the DBD or any other document which may be viewed by more than one user or multiple personnel.

Should the Data Base Administrator, or any user, suspect that a privacy key has become public knowledge, the Data Base Administrator should immediately issue a new unique lock/key combination to replace the one in question. Timeliness is of the utmost importance if data privacy and integrity should be maintained.

The knowledge of a lock/key combination should only be related to those users who absolutely need it for their particular applications. Whenever a change is made, all affected by the change should be individually notified in order that the respective software may be changed.

A further step which can be taken by the Data Base Administrator in securing positive user identification is to assign users and data bases to specific project-programmer numbers with passwords which are changed frequently. This will keep out non-DBMS-10 users, and should help to eliminate people who could cause damage to the data base. Likewise, lock/key combinations should be changed periodically--especially in sensitive data bases. Whenever a user ceases to participate in the data base, for whatever reason, <u>all</u> locks, keys, passwords, and identifications should be changed immediately.

2.3.5 DBMS Performance and Usage Measurements

As part of the Data Base Administrator's responsibility to monitor the data base for usage, response, and potential reorganization, he should keep a record of the resources used and frequency of use, the users serviced, the effectiveness as related to response time and costs. In addition, he should document how this monitoring is to be done.

2.3.6 Save Procedures

What data is to be backed-up? How much of it is to be backed-up? Which facilities are to be used? What is the schedule and frequency for back-up procedures? Such questions should be answered and documented so that there is no confusion among staff, users, and operators--especially if operators will be depended upon to execute the save procedures.

2.3.7 Recovery And Restart Procedures

In case of crash, or other system failure, the operations and/or Data Base Administration staff need to know what recovery procedures are to be used to restore the data base and what facilities were used as part of the save procedure. These recovery procedures are extremely important in terms of data integrity and reliability, and should be fully documented in detail. Priorities and sequence of data base restoration should also be laid down--this is extremely important for systems with many data bases of varying relative importance and usage.

2.3.8 Data Base Testing Facilities

Users must know what facilities are available to them to aid in the testing of their programs which interface with the data base. The Data Base Administrator should, then, publish what procedure will be used in obtaining a test data base, and what test criteria must be met before a program will be accepted for running with the real data base.

2.3.9 Training Techniques And Guidelines

The Data Base Administrator is the prime educator of the users of the data base under his care. As such, he is responsible for establishing user training methods and facilities. These would include, but are not limited to: seminars and lectures concerning programming using the Data Manipulation Language and interacting with a shared data base, individual consultation with users having specific conceptual and/or operational problems, providing manuals and other DBMS-10 programming aids, establishing training criteria which must be met before a programmer will be permitted to use DBMS to interact with the data base, fostering of optimization techniques in application programs.

CHAPTER 3

THE DEVICE MEDIA CONTROL LANGUAGE (DMCL)

This chapter contains the entire description of the Device Media Control Language (DMCL).

3.1 INTRODUCTION

The DMCL enables the Data Base Administrator to select individual areas, to assign them to files, and to allocate storage space on mass storage devices known to the monitor. Since the DMCL is not a language of its own right, but an extension to the Schema DDL, the DMCL commands must be executed by the DDL processor prior to the processing of the Schema DDL commands. Under DBMS-10, the DMCL entries are submitted to the DDL processor as part of the schema, but before the Schema DDL entries.

There are two types of DMCL ENTRY which serve to:

(1) Specify the maximum number of records per page (Record Entry):

(2) Assign areas to files and specify the physical sizes of these files in terms of pages (Area Entry).

For each area described in the schema, there must be a DMCL Area Entry. There can, though, only be one DMCL Record Entry per schema. When constructing a schema, the DMCL Record Entry must always precede the DMCL Area Entries.

The filenames specified in the DMCL Area Entries may be from one to six letters or digits in length. No filename extension can be given in the entry, although the DDL processor automatically assigns the extension .DBS to these files. Each file so specified for creation by the monitor must be uniquely named under the project and programmer numbers to which the data base(s) belong. Data bases may be created under any number of project and programmer numbers known to the monitor, and there may be duplication of filenames in different project and programmer numbers, but it is advised that all files to which areas are assigned be given unique filenames.

Error messages associated with processing DMCL entries may be found in Appendix A.

3.2 THE CONCEPT OF PAGE

For the purpose of direct access storage, data bases have been described as being divided into areas. Each area is divided into fixed-length, consecutive groups called pages, corresponding in size to the page size specified in the DMCL Area Entry. Pages are further divided into fixed-length, consecutive units called lines. Each line contains an occurrence of a record type, its set linkages, and certain system information for the DBCS. The length of a line is dependent upon:

(1) The number of characters of data composing the record type;

(2) The number of embedded pointers--i.e., set linkages--defined for the record.

The length of a line, then, is determined by the record type with which it is associated. A database key, then, specifies the page and line number on which a record occurrence is to be found (see Section 1.3.1).

Since input/output (I/O) operations between the DBCS and the data base are handled by the monitor, data is transferred between memory and the storage device as a block of words. Block size is dictated by the physical device being used. The physical device normally used by DBMS-10 is the disk, which has a block size of 128 words. The monitor, though, knows nothing about pages and lines--only blocks.

The DBCS, therefore, must translate database keys into relative block numbers within the storage file for I/O access. The size of a page of data storage corresponds to the size of an I/O buffer in the DML preprocessor. The act of transferring a page from the disk into a buffer in core is called a page-in, or the loading of a page. All DML data retrieval and data manipulation operations are performed on that portion of the data base which is in the buffer, and not until the contents of the buffer are returned to the disk have any changes been made permanent. This transferral of a page contained in core to the disk is called page-out.

Page/buffer size is established at area allocation time by the DMCL statements. This size is in multiples of 128 words--since blocks of 128 words are read or written at one time by the monitor from or to the disk. When a page/buffer size is greater than 128 words, the page is sub-divided into sectors of 128 words each. This sub-division has no effect on lines (which may cross over section boundaries), but does increase the amount of I/O time needed to bring an entire page into core. Increased buffer size also expands the need for memory at run-time, but this is justifiable in many instances--e.g., when an occurrence of a record type and its set linkages would take up more than one block of storage space. DBMS-10 does not permit records --i.e., lines--to run across a page boundary because they would not be entirely in the buffer.

3.3 DMCL ENTRIES

This section describes the DMCL entries used to assign areas to mass storage devices.

3.3.1 DMCL Record Entry

FUNCTION	To specify the maximum number of records which can be stored on a page.
FORMAT	<u>RECORDS-PER-PAGE</u> integer-1
NOTES	This entry must precede all other entries in a schema description. Only one DMCL Record Entry can be associated with any single schema. Integer-1 is an unsigned positive decimal integer between 1 and (n/2)-1 where n is the page/buffer size in words. This is the maximum number of lines--i.e., records--permitted on a page.

3.3.2 DMCL Area Entry

FUNCTION	To assign an area to a disk file, to specify the size of this file in terms of pages, to specify the page/buffer size in words.
FORMAT	<u>ASSIGN</u> area-name-1 <u>TO</u> filename-1 <u>FIRST PAGE IS</u> integer-2 <u>LAST PAGE IS</u> integer-3 <u>PAGE SIZE IS</u> integer-4 <u>WORDS</u>.
NOTES	Area-name-1 must be the name of an area defined in the schema. There must be a DMCL Area Entry for each area defined in the schema. Filename-1 may be up to six characters or digits and cannot include a filename extension. The filenames

within a project and programmer number must be unique, and the files so named will be assigned an extension of .DBS by the DDL processor.

Integer-2, integer-3, must be unsigned positive decimal integers. Page numbers within files within a data base cannot overlap--i.e., if one file spans Pages 1 to 120, then no other file in the same data base can contain a page numbered less than 121. The numbering need not be contiguous between areas.

Integer-4 should be an integer multiple of 128, less 1. The reason for "less 1" is that the DDL processor automatically allots the next largest integer multiple of 128 as the page/buffer size--i.e., if the page size is specified as 127 words, then the DDL processor allocates 128 words; if it were specified as 128 words, though, 256 words would be allocated.

A maximum page/buffer size of 32K words is imposed by the system.

The ordering of the statements in this entry is important and must be maintained.

CHAPTER 4

THE SCHEMA DATA DESCRIPTION LANGUAGE (DDL)

This chapter contains the entire description of the Schema Data Description Language (DDL).

4.1 INTRODUCTION

The DDL enables the Data Base Administrator to define and describe the logical and physical mapping of a data base in terms of a schema. A schema written in the Schema DDL consists of four types of entry which serve to:

(1) Identify the schema (Schema Entry);
(2) Define areas (Area Entry);
(3) Define records (Record Entry);
(4) Define sets (Set Entry).

For each area, record type, and set type described in the schema, a separate entry is required. However, one, and only one, Schema Entry can appear in a schema. When a schema is constructed, the following ordering of the entries must be maintained:

(1) The Schema Entry must always be the first entry;
(2) An Area Entry must precede the Record Entries for all record types within that area;
(3) A Record Entry must precede the Set Entries for all record types which participate in those sets as either owners or members.

4.2 SCHEMA ENTRY

FUNCTION	To identify a SCHEMA of a data base.
FORMAT	<u>SCHEMA</u> NAME IS schema-name.
EXAMPLE	SCHEMA NAME IS BARHEX.
NOTES	The schema-name must be unique among the schema-names known to DBMS-10. All schema-names are limited to 1 to 6 characters in length. The schema identified by the specified schema-name consists of the DDL entries that appear after this entry and before the END-SCHEMA indicator.

4.3 AREA ENTRY

FUNCTION To name and give certain characteristics of an area within a data base and to optionally specify privacy locks and/or the temporary nature of the area.

FORMAT

```
AREA NAME IS area-name-1
     [; AREA IS TEMPORARY]
                             (       RETRIEVAL          )
     [; PRIVACY LOCK [FOR   {                            } |
                             ( {EXCLUSIVE} UPDATE       )
                             ( {PROTECTED}              )

        IS literal-1] .
```

$$\underline{\text{AREA}}\ \text{NAME IS area-name-1}\ [;\ \text{AREA IS}\ \underline{\text{TEMPORARY}}]\ [;\ \underline{\text{PRIVACY}}\ \text{LOCK [FOR} \left\{ \begin{array}{c} \underline{\text{RETRIEVAL}} \\ \left\{ \begin{array}{c} \underline{\text{EXCLUSIVE}} \\ \underline{\text{PROTECTED}} \end{array} \right\} \underline{\text{UPDATE}} \end{array} \right\} \ \text{IS literal-1]}.$$

EXAMPLE

```
AREA NAME IS PERSONNEL-AREA
     PRIVACY LOCK EXCLUSIVE UPDATE IS PAEXUP
     PRIVACY FOR RETRIEVAL IS PARER.
```

NOTES All area names must be unique among area-names within the schema, and may be from 1 to 30 characters in length.

At least one area-name must be specified in a schema. If only one area-name is specified, then that area and the data base are equivalent.

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. When a close is executed on a temporary area or the run-unit terminates, record and set occurrences in the area are no longer accessible and the space occupied by the temporary area may be made available for re-use by the DBCS.

PRIVACY LOCK specifies the privacy lock which applies to the use of an area. A separate PRIVACY clause may be stated for each usage-mode. However, the same usage-mode must not be specified in more than one PRIVACY clause.

The literals are privacy locks, to be matched with the pertinent privacy key. These literals must conform to the data characteristics of privacy locks, and are limited to six characters in length.

The same literal may be specified for one or more options included in the PRIVACY clause. If the optional FOR clause is omitted, the literal applies to any use of the area.

If a PRIVACY clause is not specified for a given usage-mode, then the use of that usage-mode on the described area is without restriction.

The privacy locks associated with the three usage-modes must be satisfied by the run-unit in order to enable it to open the area with the corresponding usage-mode.

4.4 RECORD ENTRY

The record Entry will be presented in terms of two sub-entries - Record Sub-Entry and Data Sub-Entry.

4.4.1 Record Sub-Entry

FUNCTION To name a record type in the schema - i.e., to specify a generic name for all occurrences of the record type in the data base. Also, to give certain characteristics of record occurrences within a data base.

FORMAT

```
RECORD NAME IS record-name-1
                                ┌        ┌ data-name-1  ┐                 ┐
                                │ DIRECT └ identifier-1 ┘                 │
                                │                                         │
     ; LOCATION MODE IS        <  CALC USING data-name-2                   >
                                │           [, data-name-3]...            │
                                │         [DUPLICATES ARE [NOT] ALLOWED]  │
                                │                                         │
                                │ VIA  set-name-1                         │
                                └                                         ┘
     ; WITHIN area-name-1 [  ,area-name-2  ... AREA-ID IS
                                   data-name-4] .
```

EXAMPLE

```
RECORD NAME IS CUSTOMER-RECORD
     LOCATION MODE IS CALC USING ACCOUNT
     DUPLICATES ARE NOT ALLOWED
     WITHIN MARKETING-AREA
```

NOTES Record-names must be unique within the records of a schema, and may be 1 to 30 characters in length.

The LOCATION MODE clause is used to define the criteria for selecting a record occurrence and to advise the DBCS of the desired placement for a record occurrence in an area.

Identifier-1 is implicitly declared as a database key and is not part of a record. Data-name-1 must be qualified with a record name and must refer to a data-item defined as a database key. Data-name-2 and data-name-3 must refer to data-items included in the record being described. Set-name-1 must be a set in which the record is defined as being a member.

If the DIRECT option is specified, the specified data item must be initialized prior to the execution of a command that selects the record on the basis of its location mode.

If CALC is specified, then data-name-2,3 ... must be initialized prior to the execution of a command that stores a record or selects it on the basis of its location mode.

If VIA set-name-1 is specified, then all data items used to select a unique set occurrence as specified in the set occurrence selection clause for the set named must be initialized prior to the execution of a command that stores a record or selects it on the basis of its location mode.

The LOCATION MODE clause controls placement of records according to the options selected. Specification of DIRECT or CALC causes placement to be controlled by data-name-1 or identifier-1 or by data-name-2,3 ... respectively. Specification of VIA set-name-1 SET causes placement to be as close as possible to the logical insert point of the record in set-name-1, thus permitting clustering of data.

The DUPLICATES clause refers to the CALC keys. If the DUPLICATES ARE NOT ALLOWED clause is specified, no additional occurrence of this record type will be allowed to exist if all of its CALC key values are identical to those of an occurrence already in the data base. When this clause is omitted, the default is DUPLICATES ARE ALLOWED.

The WITHIN clause tells the DBCS into which area an occurrence of the record is to be placed.

Area-name-1, area-name-2 must be the names of areas for which an area entry is included in the schema prior to this entry. By its appearance in an AREA-ID clause, data-name-4 is implicitly defined to be a character-string that conforms to the rules for the formation of area-names. If the LOCATION MODE IS VIA set-name clause is specified in this Record Entry, the WITHIN clause for this Record Entry must include all areas named in the WITHIN clause of the OWNER record of the set named.

When only one area-name is specified, the contents of data-name-4 determine the area into which a record occurrence is placed.

Data-name-4 must be initialized with an appropriate area-name or with NULL-VALUE prior to the execution of a command which stores a record or one which selects a record on the basis of its LOCATION MODE clause.

If the LOCATION MODE clause specifies DIRECT or CALC, data-name-4 must be initialized with an area-name included in the WITHIN clause; otherwise an error will occur.

If the LOCATION MODE clause specifies VIA set-name SET, data-name-4 must be initialized with an area-name included in the WITHIN clause or with NULL-VALUE. If it is initialized with a legal area-name the record occurrence is placed in the named-area. If it is initialized with NULL-VALUE the record occurrence is placed as close as possible, within the constraints of the areas named in its WITHIN clause, to its logical insert point in the set named.

4.4.2 Data Sub-Entry

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

```
FORMAT 1  02 data-name-1 { PICTURE } IS picture-string.
                         { PIC     }

EXAMPLE 1 02 ACCOUNT   PIC X(6).

FORMAT 2  02 data-name-2  SIZE IS integer-1.

EXAMPLE 2 02 ADDRESS   SIZE IS 50.

FORMAT 3
                                          [{ PICTURE } IS picture-string ]
          {level-number}  data-name-3     [{ PIC     }                   ]
                                          [                              ]
                                          [ OCCURS integer-2 TIMES       ]

EXAMPLE 3 03 SALES   OCCURS 150 TIMES.
```

NOTES

A data-name is a name that is not identical to any reserved word, and may be from 1 to 30 characters in length. A data-name must be unique within a record.

A data sub-entry names and describes a data-item, vector or repeating group. Additional sub-entries are required to name and describe the components of a repeating group.

Format 1 is used to describe elementary data-items which are not part of a data-aggregate. The 02 level number is mandatory. Format 2 is used to specify a data-aggregate, which may only be defined at the 02 level.

Format 3 is used to describe data-items which make up a data-aggregate, and may be any COBOL accepted format.

A picture-string must conform to the rules for ANSI COBOL picture-strings.

A level number in Format 3 is an unsigned decimal integer greater than 2 and less than 100, and it must be specified.

In Format 2, integer-1 must be equal to the sum of the number of character positions specified for each of the subordinate sub-entries for that data-aggregate.

The OCCURS clause in Format 3 must conform to the rules of ANSI COBOL for such a group.

If data-name-1 is used as a database key using DIRECT, then it must have a picture X(10).

4.5 SET ENTRY

FUNCTION To name a set type in the schema - i.e., to specify a generic name for all occurrences of the set type in the data base; to specify the mechanism to be utilized to support the manipulation of a set; to specify the insertion point of a member record occurrence within a set occurrence and thereby define the order of sequential progression. Also to specify the name of a record, each occurrence of which establishes the existence of an occurrence of the set named in this entry; to name the records which may be members of the set named; and to specify the type of membership in that set. In addition, to specify the sort control keys for the member records of a sorted set; to optionally check and reject the insertion within the same set occurrence of member record occurrences that contain duplicate values for the specified sort control keys; to define the rules governing the selection of the appropriate occurrence of a set for the purpose of inserting an occurrence of a member record or accessing a desired member record.

FORMAT

```
SET NAME IS set-name-1

     ;MODE IS CHAIN [LINKED TO PRIOR]

               ⎧        ⎧FIRST⎫                                        ⎫
               ⎪ ALWAYS ⎨LAST ⎬                                        ⎪
               ⎪        ⎪NEXT ⎪                                        ⎪
               ⎪        ⎩PRIOR⎭                                        ⎪
     ;ORDER IS ⎨                                                       ⎬
               ⎪        ⎡WITHIN RECORD-NAME                      ⎤     ⎪
               ⎪ SORTED ⎢BY DATABASE-KEY                         ⎥     ⎪
               ⎪        ⎢                 ⎡FIRST⎤                ⎥     ⎪
               ⎪        ⎢DUPLICATES ARE   ⎢LAST ⎥  ALLOWED       ⎥     ⎪
               ⎩        ⎣                 ⎣NOT  ⎦                ⎦     ⎭

     ;OWNER IS      ⎧record-name-1⎫
                    ⎩SYSTEM       ⎭

     ;MEMBER IS record-name-2 ⎧MANDATORY⎫⎧AUTOMATIC⎫
                              ⎩OPTIONAL ⎭⎩MANUAL   ⎭

       [LINKED TO OWNER]

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

     ⎡⎧ASCENDING ⎫[RANGE]  KEY IS identifier-3 ...
     ⎢⎩DESCENDING⎭
     ⎣                        ⎡FIRST⎤
          [DUPLICATES ARE     ⎢LAST ⎥  ALLOWED]]
                              ⎣NOT  ⎦
```

```
[; SET OCCURRENCE SELECTION IS THRU

     {CURRENT OF SET                                          ]
     {LOCATION MODE OF OWNER                                  |
        [USING identifier-4 [ ,identifier-5] ...        ]}    |
        [ALIAS FOR identifier-6  IS data-name-1 ...     ]}    ]
```

EXAMPLE

```
SET NAME IS ORDER-SET
     MODE IS CHAIN LINKED TO PRIOR
     ORDER IS ALWAYS LAST
     OWNER IS CUSTOMER-RECORD
     MEMBER IS ORDER-RECORD OPTIONAL MANUAL OWNER
        DUPLICATES ARE NOT ALLOWED FOR ORDERNUM
     SET SELECTION CURRENT.
```

NOTES

SET NAME Clause

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

MODE Clause

Each set within a schema may have only one MODE clause. All participating records of a set defined with MODE IS CHAIN clause are linked to the next record. The optional LINKED TO PRIOR clause causes the DBCS to generate an additional pointer in the prior direction for the owner record and for each member record of each occurrence of the set.

ORDER Clause

If the ORDER IS SORTED clause is specified with the DUPLICATES option, an ASCENDING/DESCENDING clause without a DUPLICATES clause must be stated for each member sub-entry for this set.

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

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

When both the ORDER IS LAST and MODE IS CHAIN clauses are specified in a set entry, a pointer to the last member record of each set occurrence will be associated with the owner record of that set occurrence.

ORDER PRIOR/NEXT refers to insertion points relative to the current record of the set. Whenever the current record of a given set-name is not known to the DBCS no new member records can be

inserted into an occurrence of that set-name
without first establishing a current record for it.

The ORDER IS SORTED clause allows specification
of a set order based on the record-names or the
database keys of the member records of the set,
or on the values of the sort control data items
specified in the ASCENDING/DESCENDING clauses
for the member records of the set.

The optional WITHIN RECORD-NAME clause 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 key(s). If the ASCENDING/DESCENDING
clause is not used for any member record-type, the
database keys of the occurrences of that record
type are used as ascending sort keys.

The optional BY DATABASE-KEY clause specifies that
the member records of a set occurrence are kept in
ascending sequence by their database keys.

The optional DUPLICATES clause specifies the action
to be taken when a new record occurrence is to be
added to the set and the values of its sort control
data items are duplicates of sort control data items
of record occurrences which currently participate
as members of the set occurrence. Use of the
DUPLICATES clause also specifies that the member
records in a set occurrence are to be maintained
in a single sequence regardless of the number of
different member record-types specified in the
set entry. The common sort key(s) are specified
in the ASCENDING/DESCENDING clauses for each member
record type and must agree in size, mode and physical
position within the record.

If the ORDER IS SORTED, and the WITHIN RECORD-NAME,
BY DATABASE-KEY, or DUPLICATES clause is not used,
the record-name of the member record is used as the
major sort key. Minor sort keys are specified by
the ASCENDING/DESCENDING clauses for each member
record-type. If the ASCENDING/DESCENDING clause is
not used for any member record-type, the database
keys of the occurrence of that record type are used
as the sort keys for that record type.

OWNER Clause

Record-name-1 must be previously defined in a record
entry.

If the OWNER IS SYSTEM clause is used in a set entry,
none of its member sub-entries can include a set
occurrence selection clause.

MEMBER Clause

Record-name-2 must be previously defined in a record entry.

Identifier-1, identifier-2... must refer to data items included in record-name-2.

Record-name-2 cannot be the name of the record specified in the OWNER clause of this set entry.

If the optional word AUTOMATIC is used occurrences of record-name-2 are inserted unconditionally into the selected occurrence of the set at the time the record occurrence is stored.

If the word OPTIONAL is used occurrences of record-name-2 may be inserted into, or removed from the appropriate set occurrences by the execution of the INSERT and REMOVE commands respectively.

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

The optional LINKED TO OWNER clause causes each member record occurrence of this type to be directly associated with the database key of its OWNER. The LINKED TO OWNER clause cannot be applied to member record of singular sets.

A MEMBER clause must be specified for each record type that can participate as a member in the set being described. More than one record type can be declared as a member of any given set.

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

The DUPLICATES NOT ALLOWED clause must be repeated for each data item or concatenation of data items for which duplicate values are not allowed. The identifiers included in any single DUPLICATES NOT ALLOWED clause will be concatenated.

Each occurrence of a member record participates in only one occurrence of a set. That is, within a set, each member record occurrence may have one and only one owner.

A record may not be defined as both an owner and a member of the sets such that a cycle is formed in which all records participate as automatic members in the sets included in the cycle.

Identifier-3, identifier-4, ... must refer to data items specified in the Record Entry for the record named in the MEMBER clause of this sub-entry.

ASCENDING/DESCENDING Clause

The ASCENDING/DESCENDING clause must be included in all member sub-entries of any set-entry which includes an ORDER IS SORTED clause specifying the optional DUPLICATES clause. This clause must not be used if the set-entry does not include the ORDER IS SORTED clause or if the ORDER IS SORTED BY DATABASE-KEY clause is used. For all other forms of the ORDER IS SORTED clause, specification of the ASCENDING/ DESCENDING clause is optional.

The DUPLICATES clause must be stated if and only if the ASCENDING/DESCENDING clause is used and an ORDER IS SORTED clause does not include a DUPLICATES clause.

The order in which the keys are specified will define the major to minor sequence for sorting.

Within a member record some data items can be defined as ascending keys and some can be defined as descending keys. That is, they can be inter-mixed.

If there are multiple member record types defined for a set, the ASCENDING/DESCENDING KEY clauses can be intermixed between record types. That is, one record type can be defined with an ASCENDING KEY clause, while another can have a DESCENDING KEY clause.

If the ORDER IS SORTED clause is used in a set entry and no ASCENDING/DESCENDING clause is specified in one or more of its member sub-entries, the database key of such a member record is used as an ascending key.

If the ORDER IS SORTED clause includes the DUPLICATES clause, the member records in a set occurrence are maintained in a single sequence regardless of the number of different member record-types. The corres-ponding sort keys specified for each member record type must have identical data characteristics and must also match in terms of whether they are ascend-ing or descending.

If the optional word RANGE is used, the ordering of the member record occurrences is in accordance with the above rules.

Where one or more data items declared with an ASCENDING or DESCENDING RANGE KEY clause are speci-fied as arguments in a SET OCCURRENCE SELECTION clause the use of the optional word RANGE causes each occurrence of such a data item to represent a range of values and has the following implications for the process of set occurrence selection.

An equality match between the range key (which is in the record to be selected) and the input argument value in the user working area is not required for a record to be selected as being the owner of the sought set occurrence.

A match will occur regardless of whether the range key has been specified as ascending or descending as follows:

(a) If the input argument value in the User Working Area equals the value of any specific range key.

(b) If the input argument value in the User Working Area equals the value of any specific range key; then a match will occur on the range key with the lowest value.

(c) If the input argument value in the User Working Area lies between two adjacent range key values; the match will occur with the larger range key value.

A match will not occur if the input value in the UWA is greater than the largest value of any RANGE KEY. When this is the case an Error Status Condition will occur.

If the DUPLICATES ARE NOT ALLOWED clause is used the DBCS will reject the insertion into any given set occurrence of member record occurrences with duplicate values for the specified ascending/descending keys. This may occur during an attempt to store a new record occurrence in the data base, or insert an existing record occurrence into a set, modify the value of a data item specified in an ASCENDING or DESCENDING KEY clause.

If the DUPLICATES ARE FIRST or the DUPLICATES ARE LAST clause is used, member record occurrences with duplicate values for the specified ascending/descending keys will be inserted by the DBCS before or after, as specified, any existing member occurrences with such duplicate values.

If the DUPLICATES clause does not include any of the optional words FIRST, LAST or NOT, the insertion point of duplicate member record occurrences relative to existing duplicates is unpredictable.

SET OCCURRENCE SELECTION Clause

A USING clause may qualify the LOCATION MODE OF OWNER clause only when the LOCATION MODE clause in the Record Entry for the owner record of the set in which this clause appears is VIA set-name. When this is the case either an ALIAS or a USING clause must be specified.

All identifiers must refer to declared data items of the owner record of the set(s) referenced.

Identifier-6 ...must, if the LOCATION MODE clause in the Record Entry for the OWNER record of the set is DIRECT or CALC, refer to data items specified in that LOCATION MODE clause. If, however, the LOCATION MODE is VIA set-name, identifier-6 ...must refer to data-items specified in the USING clause of a SET OCCURRENCE SELECTION clause for another set with the same defined owner record type.

By their appearance in an ALIAS clause, all data-names are implicitly defined as having the same characteristics as their corresponding identifiers. A data-name specified in an ALIAS clause defines a data-name in the run-unit's User Working Area.

The LOCATION MODE OF OWNER option cannot be used if the owner being referenced does not have a LOCATION MODE clause specified for it in its Record Entry.

If the LOCATION MODE OF OWNER option is used the Record Entry for the owner record type being referenced must have a DUPLICATES NOT ALLOWED clause if its LOCATION MODE IS CALC.

All other identifiers explicitly named in a SET OCCURRENCE clause must have a DUPLICATES NOT ALLOWED clause in the appropriate Set Entries. The DUPLICATES NOT ALLOWED clause may be specified in the ASCENDING/ DESCENDING KEY clause, the SEARCH KEY clause or the MEMBER clause of the relevant Set Entries.

The SET OCCURRENCE SELECTION clauses for the appropriate member record and set combinations will govern the selection of specific set occurrences whenever:

A STORE command is executed and the object record is an automatic member of one or more sets.

A MODIFY command is executed which changes the value of a data item specified in a SET OCCURRENCE SELECTION clause.

Prior to the execution of any command involving set occurrence selection, the data items specified in this clause must be initialized. The data items specified by this clause are those which are explicitly named in it and/or those which are not explicitly named but which are implied by the LOCATION MODE option of this clause.

This clause applies where:

The owner record of the set occurrence to be selected is either procedurally pre-selected (the CURRENT OF SET option) or can be uniquely identified on the basis of its LOCATION MODE clause alone (the LOCATION MODE OF OWNER option is used in this clause and the Record Entry for the owner record of the set specifies that its LOCATION MODE IS DIRECT or CALC).

The owner record of the set occurrence to be selected cannot be determined except in terms of its membership in some other set and its associated SET OCCURRENCE SELECTION clause (the LOCATION MODE OF OWNER option is used in this clause and the Record Entry for the owner record of the set specifies that its LOCATION MODE IS VIA set-name).

The CURRENT OF SET option causes the DBCS to select the current set occurrence as defined by the current of the appropriate set-name. This is the set-name of which this clause is a part.

The LOCATION MODE OF OWNER option causes the DBCS to select a set occurrence on the basis of the LOCATION MODE clause specified in the Record Entry for the owner of the appropriate set name. This is the set name of which this clause is a part. Unless the ALIAS option is used in the SET OCCURRENCE SELECTION clause, or the LOCATION MODE clause in the Record Entry for the owner record is VIA set-name, the arguments used for selection of the owner record are the arguments specified in the LOCATION MODE clause of the Record Entry. These arguments must therefore be initialized with the actual argument values prior to each execution of any command controlled by the SET OCCURRENCE SELECTION clause.

If the LOCATION MODE OF OWNER option is used and the LOCATION MODE clause in the Record Entry for the owner record is VIA set-name, the owner record to be selected must be located in terms of its membership in another set. This selection is governed by the SET OCCURRENCE SELECTION clause for the set named in the LOCATION MODE clause of the owner record, and by the USING option of the SET OCCURRENCE SELECTION clause. The data items specified in the USING clause must uniquely identify a specific record occurrence within an occurrence of the set named in the LOCATION MODE clause of the owner record. The SET OCCURRENCE SELECTION clause for the set named in the LOCATION MODE clause may, in turn, specify LOCATION MODE OF OWNER and the LOCATION MODE may again be VIA set-name.

This condition may occur to an arbitrary number of levels, but must eventually terminate with a SET OCCURRENCE SELECTION clause that does not specify LOCATION MODE OF OWNER where the LOCATION MODE is VIA set-name. At each level other than the first, the arguments specified in a USING clause are used to select an owner record in its capacity as a member of another set. The arguments specified may be any data items in the records to be selected.

The optional ALIAS clause provides for the situation where a given record is defined as a member in more than one set type, and each such set type has the same owner record type. In this situation, more than one argument value may be required for the data item named as an argument. The ALIAS clause provides the User Working Area locations for such values.

For sets with more than one record type named as member record - i.e., multiple MEMBER clauses - a SELECTION clause is needed for each MEMBER Clause and precedes the next MEMBER Clause. SELECTION CURRENT need not always be specified, as this is the default.

CHAPTER 5

THE COBOL SUB-SCHEMA DATA DESCRIPTION LANGUAGE (DDL)

This chapter contains the entire description of the DBMS-10 COBOL Sub-Schema Data Description Language (DDL).

5.1 INTRODUCTION

The COBOL Sub-Schema DDL enables the Data Base Administrator to describe the subset of a data base known to one or more COBOL DML programs in terms of a COBOL sub-schema. A sub-schema description written in the COBOL Sub-Schema DDL consists of the Sub-Schema Identification and three sections:

AREA SECTION
RECORD SECTION
SET SECTION

All sections must appear in the above order.

The Area, Record, and Set Sections consist of an entry for each area, record, or set to be included in the sub-schema being defined. Each such entry completely describes an area, record, or set.

Sub-schema descriptions follow the schema description, but before the END-SCHEMA indicator, and are passed through the DDL processor along with the DMCL and Schema DDL statements.

5.2 SUB-SCHEMA IDENTIFICATION

FUNCTION	To define and name a sub-schema within a schema, and to specify the privacy lock for the use of a sub-schema.
FORMAT	<u>SUB-SCHEMA</u> NAME IS sub-schema-name <u>PRIVACY</u> LOCK IS literal-1.
EXAMPLE	SUB-SCHEMA NAME IS SUB01 PRIVACY SALEX.
NOTES	Sub-schema-name must be unique among the sub-schema-names associated with the specific schema. Literal-1 must conform to the data characteristics of privacy locks to be matched with the pertinent privacy keys.

5.3 AREA SECTION

FUNCTION

To enumerate the areas of the schema that are included in the sub-schema and, by implication, to remove from view all other areas of the schema.

FORMAT 1

```
AREA SECTION.
     COPY area-name-1 [,area-name-2] . . .
```

FORMAT 2

```
AREA SECTION.
     COPY ALL AREAS.
```

FORMAT 3

```
AREA SECTION.
     COPY TEMPORARY area-name-3 [,area-name-4] . . .
```

EXAMPLE

```
AREA SECTION.
     COPY MARKETING-AREA, INVENTORY-AREA.
```

NOTES

Area-name-1, area-name-2, . . . must refer to areas defined in the schema.

If Format 2 is used, Formats 1 and 3 entries are not allowed. Otherwise 1 and 3 may be repeated as needed.

Format 1 causes the entries for the referenced areas in the schema to be included in the sub-schema.

Format 2 causes all areas for which entries are included in the schema to be included in the sub-schema.

Format 3 designates the named areas to be sub-schema temporary. A sub-schema temporary area is a private unique occurrence of a normal area which is always opened as if protected update had been specified. However, any changes made to this area will be discarded when the run-unit closes the area or terminates. This is used to permit program testing on "live" data without loss of data base integrity.

5.4 RECORD SECTION

FUNCTION

To enumerate and define the records of the schema that are to be included in the sub-schema. By implication, to remove from view all other records of the schema.

FORMAT

```
RECORD SECTION.
01 record-name-1.
01 record-name-2.
```

EXAMPLE

```
RECORD SECTION.
01 CUSTOMER-RECORD.
01 ORDER-RECORD.
01 STOCK-RECORD.
01 SUPPLIER-RECORD.
```

NOTES

All record-names must be unique within the record-names used in the sub-schema. Since each record-name must be the name of a record declared in the schema, this should be no problem. No changes are allowed.

Each record named must be located within an area named in the Area Section. If a record is located in more than one area, and not all of these areas are included in the sub-schema, then only the record occurrences located in the included areas are made part of the sub-schema.

5.5 SET SECTION

FUNCTION

To enumerate and define the sets of the schema that are to be included in the sub-schema, and, by implication, to remove from view all other sets of the schema.

FORMAT 1

```
SET SECTION.
    COPY set-name-1 [,set-name-2] . . .
```

FORMAT 2

```
SET SECTION.
    COPY ALL SETS.
```

EXAMPLE

```
SET SECTION.
    COPY ORDER-SET, INVENTORY-SET, SUPPLIER-SET.
```

NOTES

All set names must refer to sets defined in the schema.

Format 1 entries may be repeated as required. If Format 2 is used, no Format 1 entries are allowed.

Format 1 causes the entries for the referenced sets in the schema to be included in the sub-schema. No changes are allowed.

Format 2 causes all sets for which entries are included in the schema to be included in the sub-schema.

An entry must be included in the Record Section for the owner record of each set included in a sub-schema.

If a set included in a sub-schema is SORTED on one or more keys, all the records containing those keys are to be included in the sub-schema. Otherwise, no record modification will be permitted for members of that set.

CHAPTER 6

DATA ORGANIZATION AND ACCESS

Data organization refers to the manner in which data is arranged in a data base. Data access refers to the way in which data from the data base is read and/or written (I/O).

6.1 DATA ORGANIZATION

As discussed in previous chapters, a data base must be equated to DECsystem-10 monitor files for storage on disks. The DDL processor accepts the DMCL entries, which precede the DDL entries in the Schema-Sub-Schema DDL file, and writes files based on the DMCL entries. The assigment of physical space for these files (hereafter called DBS files, because .DBS is the filename extension given these files by the DDL processor) is performed automatically by the monitor when the DBS files are written by the DDL processor. The DBS files may be any length, and each data base may be divided into as many files as its Data Base Administrator wishes, as long as disk space is available. Under the DECsystem-10 monitor, files may be concurrently read by more than one user at a time--thus permitting concurrent retrieval of data from areas of a data base. Update of a file by more than one person at the same time, though, is not possible, although an updated version of an area of a data base may be written by one user while other users continue to read the old version--i.e., protected update.

Pages are the basic partitions of the DBS files to which a data base is assigned (see Chapter 3). By dividing the DBS files into pages, and storing selected records on these pages, and using a page as the basic I/O buffer size, DBCS operations that affect the records on only one page can be handled with a single disk access. The DBS files are random access files--i.e., they can be accessed by the DBCS either sequentially or randomly (see Section 6.2).

Record locations on a page (hereafter referred to as lines) are consecutively numbered starting with one. When combined with the page number to form a database key (see Section 6.1.4), this line number becomes the address of the corresponding record occurrence in the data base.

Before considering data access, several aspects of data organization should be discussed:

(1) identification of a DBS file;

(2) identification of a page--the page header;

(3) format and identification of a line;

(4) forming a database key.

6.1.1 Identification Of A DBS File

All files created by the DDL processor for storage of areas of a data base are given the filename extension of .DBS when created. Unique to each data base is a schema directory file--also created by the DDL processor--in which the structure of that data base is described (see Chapter 7). As part of each area description in this directory, the corresponding DBS file appears. Further explanation of this will be given in Section 6.2 and in Chapter 7.

6.1.2 Identification Of A Page

When a DBS file is created by the DDL processor, it is also segmented into its appropriate number of pages, and each page is given a page header which serves to specify:

(1) the page number;

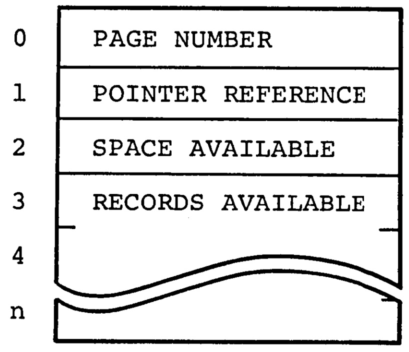
(2) the pointer reference for the page;

(3) the space available on the page in terms of words;

(4) the record space available on the page.

The DECsystem-10 records data as 36-bit words, which can also be viewed as 12 octal bytes. DBMS-10 is therefore designed to fit into these 36-bit words of portions thereof.

A page header is thus a word block which has the following format.



where:

> PAGE NUMBER is the number of the page in octal, right justified, and bit 0 is always zero.
>
> POINTER REFERENCE is an octal number, right justified. The pointer reference is a function of the page number and the records-per-page specified in the DMCL Record Entry.
>
> SPACE AVAILABLE is the number of words left on the page which can be used for lines; it is initially calculated by subtracting the number of words used for the page header from the page size, and is updated.
>
> RECORDS AVAILABLE represents the maximum number of records which can be assigned to a page as determined from the DMCL Record Entry. The number of words necessary to make this representation is dependent on the number of records-per-page specified since one bit is allotted for each record.

The following examples should be helpful in understanding how the page header is formed.

EXAMPLE 6.1

Consider the following DMCL entries:

```
RECORDS-PER-PAGE 63.

ASSIGN EXAMPLE-AREA-1 TO XFILE1
FIRST PAGE IS 1
LAST PAGE IS 43
PAGE SIZE IS 127 WORDS.
```

The page headers in XFILE1.DBS would be five words long. This size is calculated in the following manner. The first word contains the page number; the second, the pointer reference; and the third, the space available. Since the records-per-page is given as 63, 63 bits are needed to represent the 63 lines which the page could at the most hold. Two words contain 72 bits, so the leftmost 64 bits of the combined fourth and fifth words are used for this indication. Bit 0 of the fourth word is left zero, but the following 63 bits are turned on. Later, whenever a record is stored on a particular page, it is assigned a number corresponding to the left-most "on" bit.

The space available now on any page in XFILE1.DBS is 123 words. This figure is derived in the following way. The page size was given as 127 words. The DDL processor always allocates page size as the next higher multiple of 128 words (200_8 words). Since 127<1*128, 128 words is the page size used. If 128 words had been specified, 256 (400_8 words) would have been allocated. Thus, since five words were needed for the page header, only 123 words (173_8 words) remain for data storage.

The pointer reference is calculated as follows:

(1) convert the page number to octal and place it in the second word, right justified;

(2) take the records-per-page value and determine the next highest power of 2 (e.g., the next highest power of 2 with respect to 63 is 6);

(3) shift the number in the second word left as many bits as this power of 2 (thus in this example, it would be shifted 6 bits).

The resulting <u>octal</u> number is the pointer reference. For this example, the pointer reference for Page 1 is 100, while for the last page, the octal page number is 53 and its reference is 5300.

Thus, in an octal dump of XFILE1.DBS, the page header for the first page would look like:

```
ØØØØØØØØØØØ1    ØØØØØØØØØ1ØØ    ØØØØØØØØØ173    377777777777
777777774ØØ
```

EXAMPLE 6.2

Now consider these DMCL entries:

```
RECORDS-PER-PAGE 127.

ASSIGN EXAMPLE-AREA-2 TO XFILE2
FIRST PAGE IS 100
LAST PAGE IS 1201
PAGE SIZE IS 200 WORDS.
```

The page header for Page 823 (1467_8) from an octal dump of XFILE2.DBS is shown below.

```
ØØØØØØØØ1467    ØØØØØØ3156ØØ    ØØØØØØØØØ371    377777777777
777777777777    777777777777    777776ØØØØØØ
```

Note the page number in the first word is in octal; the pointer reference entailed shifting 7 bits left; four words were needed to accommodate records available, leaving 249 (371_8) words for data storage. Note further, even though 200 words were specified for page size, 256 (400_8) were allocated.

6.1.3 Format And Identification Of A Line

When the DDL processor creates a DBS file, the pages are empty, except for their page headers which are always present. The lines on a page are written by the DBCS during execution of a run-unit (see Section 6.2). The format and identification of a line, though, really belongs to data organization, and will be discussed here.

Each line begins on a word boundary, and covers as many words as is necessary, terminating on a word boundary. Lines are confined to the page on which they begin, and can never cross page boundaries. A line has the following format:

LINE HEADER	POINTERS	DATA

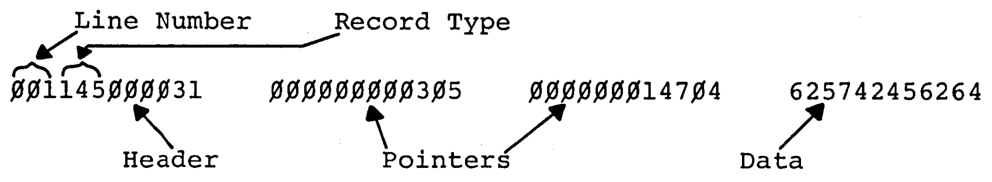
The line header is a single word and contains the following:

Bits 0 through 8	the line number in octal.
Bits 9 through 17	the record type to which this record occurrence belongs (see Chapter 7).

Bits 18 through 19	DELETE flag.
Bits 20 through 28	Must be zero (MBZ).
Bits 29 through 35	the length of the line in words (includes header).

For each set link associated with the record occurrence belonging to a line, there is an embedded set pointer as part of the line. Every pointer is given one complete word, the contents of which is the database key of the line to which it points. A further discussion of pointers follows in Section 6.2.

A line from an octal dump of a DBS file appears below.



```
ØØ1145ØØØØ31   ØØØØØØØØØ3Ø5   ØØØØØØØ147Ø4   625742456264
```

6.1.4 Forming A Database Key

The construction of a database key is essentially the combination of the line number and the number of the page on which it resides. This is not an additive combination, but a merging of the pointer reference in the page header (see Section 6.1.2) and the line number. Thus, considering Example 6.1 again, the third line on the fourth page would have 403_8 as its database key. In Example 6.2, the database key for the ninth line on Page 823 would be 315611_8.

6.2 DATA ACCESS

When a run-unit wishes access to a data base, the DBCS first verifies the acceptability of the INVOKE statement of the run-unit. For a run-unit which is permitted access, a User Working Area is established for the run-unit, and an in-core map of the data base is created. This in-core representation is actually a data base itself--complete with record types and set types. Figure 6.1 shows the set structures of this in-core data base.

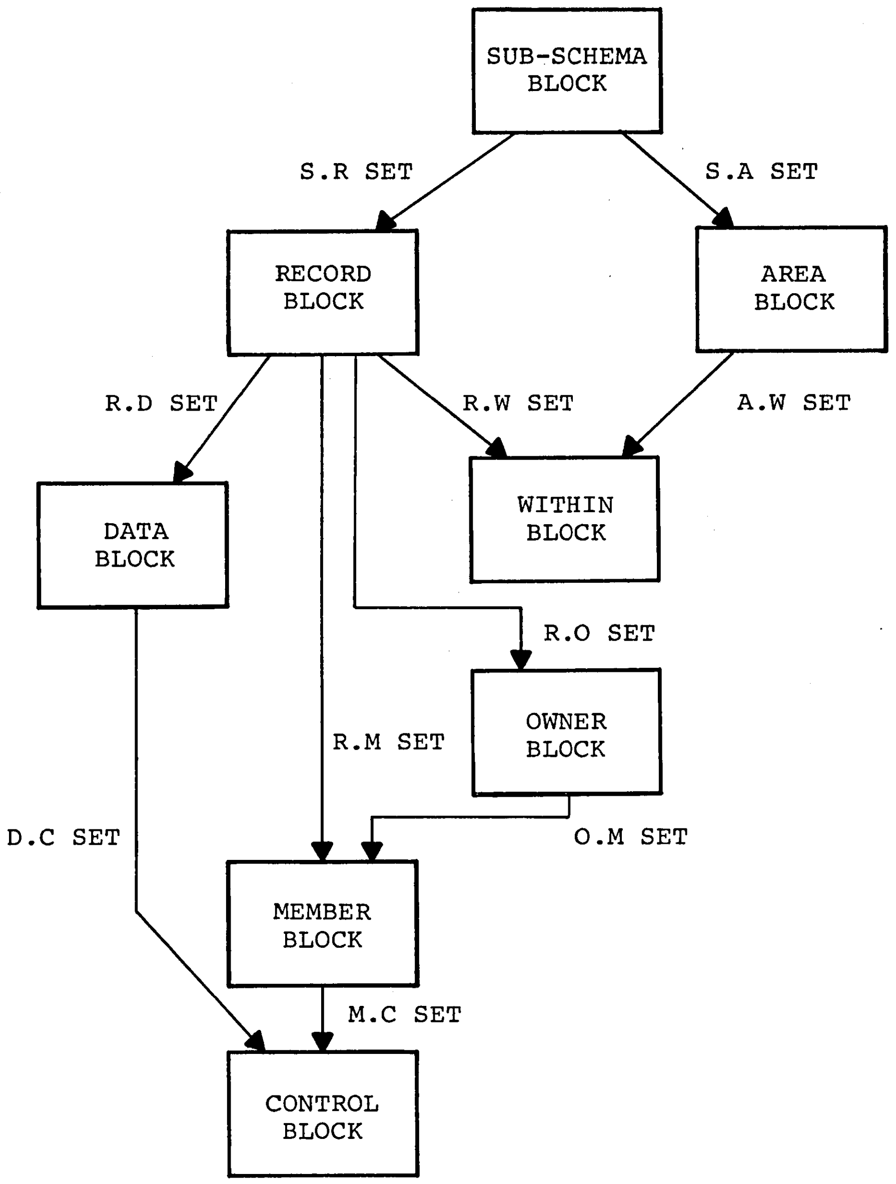


Figure 6-1 In-Core Representation

The DBCS uses this in-core representation to govern the interactions with the data base. All data base I/O operations are dependent on this map. The first point of this section to be covered, then, will be the in-core representation of the sub-schema to which a run-unit has access. Following this, the I/O buffering schema will be discussed, and then the algorithms used to STORE and FIND record occurrences.

6.2.1 The In-Core Representation

The in-core representation is a network structure of eight record types (also called blocks) which serve to map the sub-schema being accessed by the run-unit. It is created in core from the schema directory (see Chapter 7) for the data base to which the sub-schema belongs. A complete analysis will not be given, but a description of each block is provided for those who are interested.

6.2.1.1 Sub-Schema Block - The sub-schema block is the root of the in-core representation and has only one occurrence. The word format for this block is shown in Figure 6-2. BITS/RECORD IN CORE in the rightmost 30 bits of the first word is a function of the records-per-page specified when the data base was created, and the number which appears here is equal to (records-per-page/36) rounded up. For all the blocks shown in this section, a word where NEXT, OWNER, or PRIOR appears is a pointer to the appropriate block in the in-core representation (refer to Figure 6-1 for set types used).

BITS/RECORD IN FILE is identical with the RECORDS AVAILABLE words in a page header (see Section 6.1.2). Thus, the size of the sub-schema block, in words, depends on the records-per-page specified in the DMCL Record Entry.

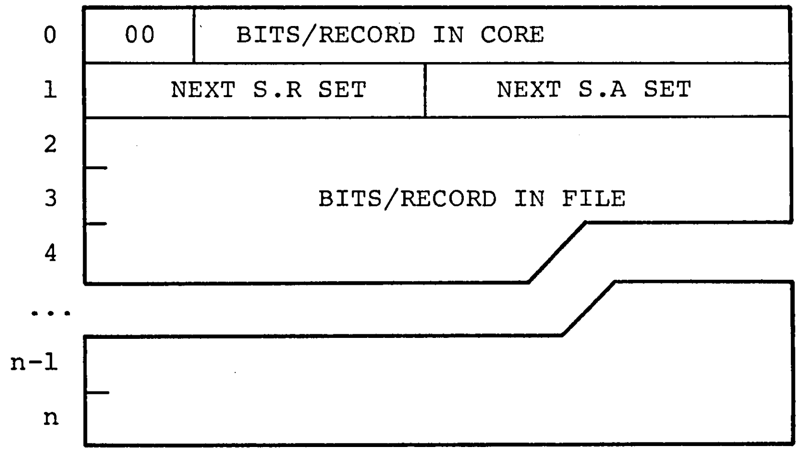


Figure 6-2 In-Core Sub-Schema Block

6.2.1.2 Record Block - For each record type belonging to the sub-schema invoked, there is set up a record block in core. The format for this 11-word block is given in Figure 6-3.

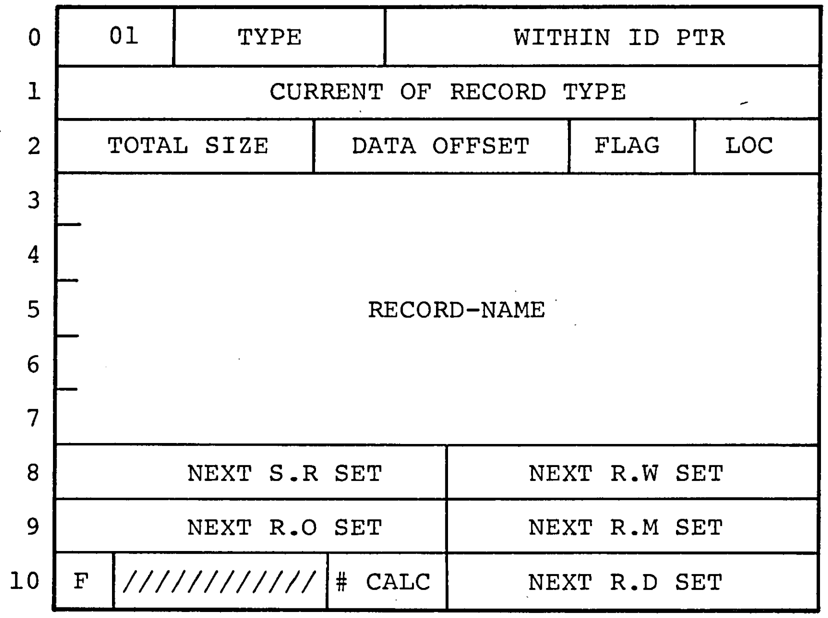


Figure 6-3 In-Core Record Block

TYPE refers to the octal number which denotes the record type.

WITHIN ID PTR is a pointer to the area block for the area in which the record is located.

CURRENT OF RECORD TYPE is the database key of the current occurrence of this record (see Section 1.6).

TOTAL SIZE is the total number of words on the lines on which occurrences of this record are stored.

DATA OFFSET is the number of words on these lines which are used for the header and for pointers.

The FLAG bits in word 2 contain the following:

Bits 25-30	Must be zero (MBZ).
Bit 31 = 1	Delete not permitted.
Bit 32 = 1	Store not permitted.

LOC is the location mode for this record, and is given as follows:

0	None - no way to locate using this sub-schema.
1	DIRECT
2	CALC
3	VIA

RECORD-NAME is the name of the record type in SIXBIT.

The F(lag) bit in word 10 is the CALC duplicate flag. If duplicates are allowed, the bit is on; otherwise, it is off.

# CALC is the number of CALC fields.

6.2.1.3 Area Block - An area block is written in core for each area known to the invoked sub-schema. Each of these blocks contains 18 words, and the format appears in Figure 6-4.

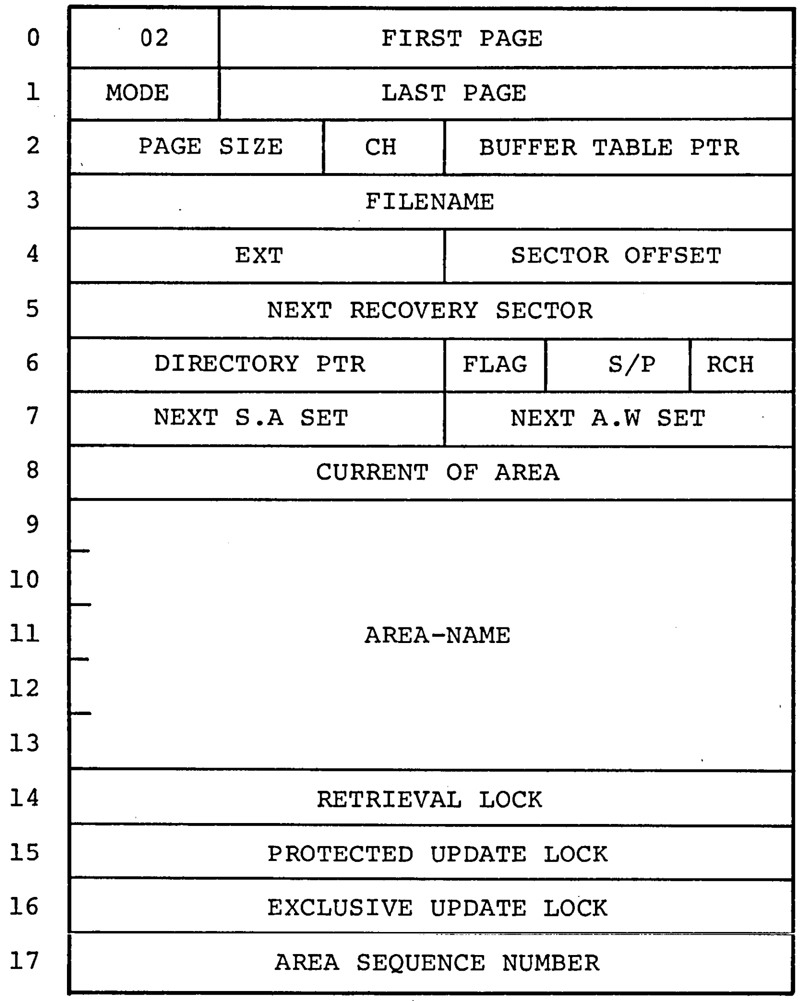


Figure 6-4 In-Core Area Block

MODE refers to the USAGE MODE specified for the area in the OPEN statement. The octal values, and the modes they represent are:

0	CLOSED - not yet open
1	RETRIEVAL
2	EXCLUSIVE UPDATE
4	PROTECTED UPDATE
10	SCHEMA TEMPORARY
20	SUB-SCHEMA TEMPORARY

The way the DBCS works, if two areas are to be bridged using sets, both areas must have the same usage mode; otherwise, the DBCS will disallow STOREs and INSERTs in these sets. Likewise, if a record is located within more than one area, the DBCS ANDs the modes for the areas and will not permit operations on that record if the result is zero.

CH is the software channel on which I/O for this area is performed.

BUFFER TABLE PTR points to in-core buffer table (see Section 6.2.2).

FILENAME and EXT refer to the monitor file in which this area is stored. Both are given in SIXBIT.

SECTOR OFFSET is the number of 128 (200_8) word sectors at the front of this file, which are used for recovery purposes (see Chapter 8), and precede the first page.

NEXT RECOVERY SECTOR is the next available sector in the recovery file for this area (see Chapter 8).

DIRECTORY PTR points to a directory buffer which is used to page in from the recovery file.

FLAG in word 6 has the following configuration:

Bit 18 = 1	Schema Temporary
Bit 19 = 1	Sub-Schema Temporary

S/P is the number of 128 (200_8) word sectors per page.

RCH is the software channel on which I/O for the recovery file for this area is done.

CURRENT OF AREA is the database key of the current record occurrence of this area (see Section 1.6).

AREA-NAME is the name of the area in SIXBIT.

The LOCKS are all given in SIXBIT.

AREA SEQUENCE NUMBER is an access reference number for the area which is advanced by the DBCS each time an area is opened. The primary use for this number is to assure the retrieval user that he is always reading the data that existed in an area when his run-unit began, and that updated data will not be made part of the permanent data base until he closes the area.

6.2.1.4 Within Block - According to the WITHIN CLAUSE specified in the Schema Record Entry for records defined for the invoked sub-schema, there will be one or more in-core within blocks for a record. These within blocks use two words, and the format is given in Figure 6-5.

0	03	////////////////	OWNER A.W SET
1	NEXT R.W SET		NEXT A.W SET

Figure 6-5 In-Core Within Block

6.2.1.5 Control Block - A control block (Figure 6-6) exists for all the data-items and data aggregates in the data base which have been specified as either SORT keys or USING/ALIAS identifiers in the SET SELECTION Clause.

0	06	/////	AD	/	U	S	ALIAS PTR
1	OWNER M.C SET						NEXT M.C SET
2	OWNER D.C SET						NEXT D.C SET

Figure 6-6 In-Core Control Block

AD in word 0 is the ascending/descending flag.

U is the using flag and is on if the block is for using control.

S is the sort-type flag. When off, match sort is used; when on, range sort is performed.

The ALIAS PTR points to a core location in which is stored the ALIAS (see Section 4.5) option used in the SET SELECTION Clause.

6.2.1.6 Owner Block - The DBCS uses a unique owner block to identify and define each set type known to the sub-schema invoked by the run-unit. Figure 6-7 shows the format for this seventeen-word block.

0	04	CHAIN IND	PRI OFFSET	NEXT OFFSET
1	OWNER R.O SET		NEXT R.O SET	
2	///////////////////	FL	NEXT O.M SET	
3	DBKEY OF OWNER OF SET			
4	DBKEY OF PRIOR OF SET			
5	DBKEY OF CURRENT OF SET			
6	DBKEY OF NEXT OF SET			
7	SET-NAME			
8				
9				
10				
11				
12	DBKEY OF NEW MEMBER OF SET			
13	DBKEY OF NEW PRIOR OF SET			
14	DBKEY OF NEW NEXT OF SET			
15	RELINK POINT IF BACKUP IS NEEDED			
16	OWNER OF SET IF BACKUP IS NEEDED			

Figure 6-7 In-Core Owner Block

CHAIN IND denotes MODE IS CHAIN.

PRI OFFSET is the number of words to skip in the line containing an occurrence of an owner record belonging to this set in order to find the prior pointer.

NEXT OFFSET is the offset to the next pointer in that line.

FLag in word 2 has the following format:

Bits 0 - 16	MBZ
Bit 17 = 1	Suppress update after FIND or STORE

The DBKEYs in words 3-6 are for those record occurrences which are indicated.

SET-NAME is the name of the set in SIXBIT.

Words 12-16 are temporary storage words which are used to hold update and backup information during a FIND or STORE. Should a FIND or STORE require successive examination of record occurrences, and one of the later examinations fail, then these words are used to return the data base to its former condition.

6.2.1.7 Member Block - A member block exists for each member record in every set. Figure 6-8 shows this five word block.

0	05	OWNER OFFSET	PRIOR OFFSET	NEXT OFFSET
1	//////////////////////////////			FLAGS
2	OWNER O.M SET		NEXT O.M SET	
3	OWNER R.M SET		NEXT R.M SET	
4	LAST M.C SET		NEXT M.C SET	

Figure 6-8 In-Core Member Block

The OFFSETs in word 0 refer to pointers in the lines on which occurrences of this record are located.

The rightmost 13 bits of word 1 are FLAGS which denote the following:

Bit 23 = 1	Record may be inserted into set		
Bit 24 = 1	Location mode is VIA this set		
Bit 25 = 1	DBKEY sort		
= 0	A/D sort		
Bits 26,27	DUPLICATES flag		
	0	allowed	
	1	not allowed	
	2	are first	
	3	are last	
Bit 28 = 0	Sort by record-name		
= 1	Sort entire set on identical keys		
Bits 29,32	ORDER flag		
	0	ALWAYS FIRST	
	1	ALWAYS NEXT	
	2	ALWAYS LAST	
	3	ALWAYS PRIOR	
	4	SORTED	
	5	SORTED BY DBKEY	
	6	SORTED WITHIN RECORD	
	7	DUPLICATES ALLOWED	
	10	DUPLICATES NOT ALLOWED	SORTED
	11	DUPLICATES FIRST	
	12	DUPLICATES LAST	
Bit 33 = 0	SET SELECTION CURRENT		
= 1	LOCATION OF OWNER		
Bit 34 = 0	AUTOMATIC		
= 1	MANUAL		
Bit 35 = 0	MANDATORY		
= 1	OPTIONAL		

The other pointers refer to Figure 6-1.

6.2.1.8 Data Block - For each data item and data aggregate belonging to the records defined for the invoked sub-schema, there exists a data block which describes its location on the lines on which it appears, its attributes, and MODIFY status. The format for such a block is given in Figure 6-9.

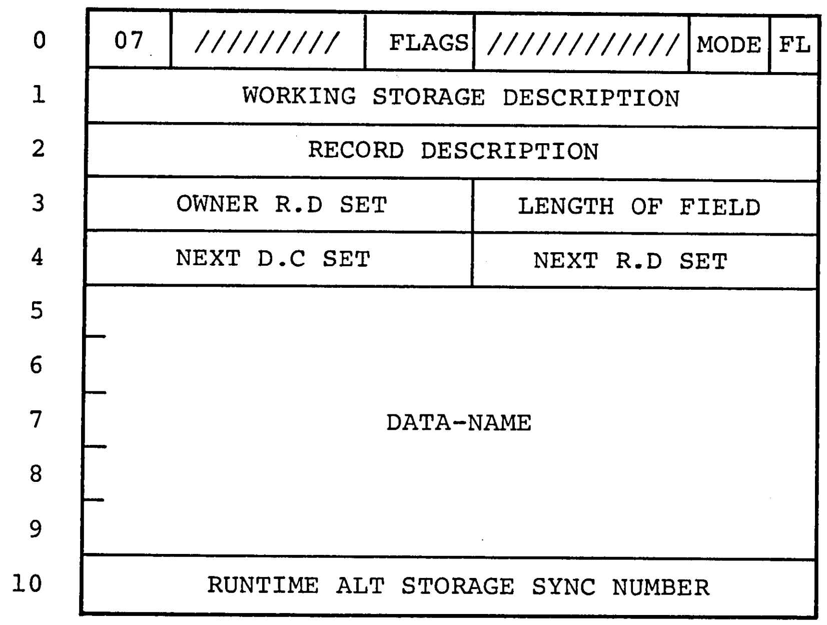


Figure 6-9 In-Core Data Block

The following flag bits are used in word 0:

Bit 14 = 1	Data may be modified
Bit 15 = 1	Data has been modified
Bit 16 = 1	Data is also CONTROL field
Bit 34 = 1	Data is CALC field
Bit 35 = 1	Data is DIRECT field

MODE refers to the data storage mode:

0	Binary-1	(USAGE COMP)
1	Binary-2	(USAGE COMP) (2 words)
2	Float-1	(USAGE COMP-1)
3	SIXBIT	(USAGE DISPLAY-6)
4	ASCII	(USAGE DISPLAY-7)
5	DBKEY	

The WORKING STORAGE DESCRIPTION is a core pointer used to reference the data item, aggregate, or DBKEY.

RECORD DESCRIPTION is a bit offset to the data.

LENGTH OF FIELD is length of data field in octal.

DATA-NAME is the name in SIXBIT.

RUNTIME ALT STORAGE SYNC# is a count of the number of times the record occurrence, to which this data belongs, was modified and this data was not.

6.2.2 I/O Buffering Scheme

When an area is successfully opened by a run-unit--regardless of the USAGE MODE specified--the DBCS allocates three (3) I/O buffers to be used for I/O access for that area. The size of each of these three buffers is identical with the page size specified for the area in its DMCL Area Entry. Thus, the page is the fundamental unit for I/O operations concerning the data base.

The run-unit cannot alter the number of page buffers held within the program. This number is preset in the DBCS, although it may be changed by altering the run-time package. These buffers remain allocated for the duration of the run-unit, and cannot be selectively discarded using CLOSE statements. However, if an OPEN-CLOSE-OPEN sequence is executed for the same area, the DBCS will reuse the same buffers.

The DBCS maintains in core a page-buffer table which serves to:

(1) specify the number of buffers currently allocated;

(2) give the core location which is the starting address for the buffer.

The format for the page-buffer table is found in Figure 6-10.

0	n = NUMBER OF BUFFERS ALLOCATED	
1	//////////////////////	BUFFER LOCATION 1
2	//////////////////////	BUFFER LOCATION 2
3	//////////////////////	BUFFER LOCATION 3
...		
n-2	//////////////////////	BUFFER LOCATION n-2
n-1	//////////////////////	BUFFER LOCATION n-1
n	//////////////////////	BUFFER LOCATION n

Figure 6-10 Page-Buffer Table

Bit 0 of words 1 through n is the MUST WRITE flag--i.e., it designates that a change has been made to the page in this buffer, and this page must be written to the DBS file. In the area block, there is a PAGE-BUFFER TABLE POINTER (see Section 6.2.1). This pointer indicates the location on the page-buffer table which has the starting address of the current page in core.

The current page when three are in core at the same time is determined in the following manner. Consider the three buffers schematically shown in Figure 6-11.

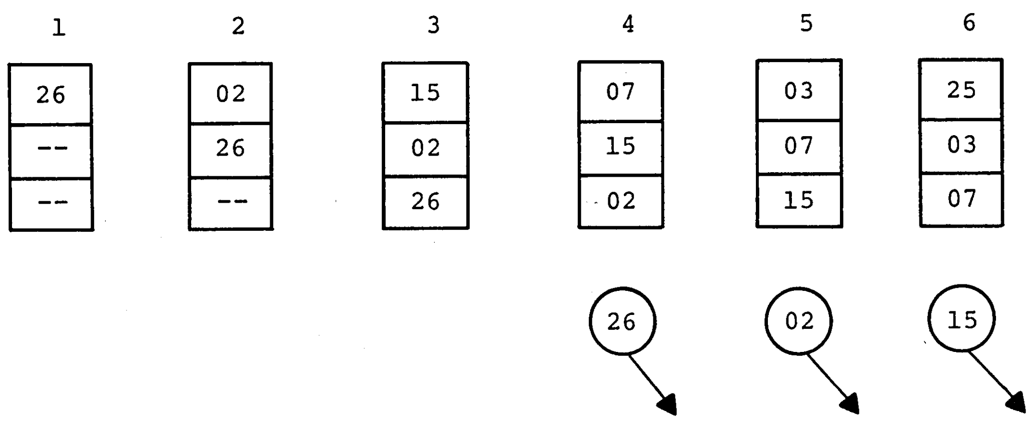


Figure 6-11 I/O Buffering Example

The number in a buffer square indicates the page in that buffer after an I/O operation; the numbers in circles indicate a page which was returned to the data base during the I/O operation. Six I/O accesses are shown. After the first one, only Page 26 is in core; after the second, both Pages 2 and 26 are in core; after the third, all three buffers have a page in them. Note that core has been expanded three times in order to accommodate all three buffers. The current page in core refers to that page in core on which the current record of area is located. Thus, if the current of area were on Page 2, Page 2 would be the current page. If the current of area should change and be located on Page 15, then that page would then be the current page.

Note, so long as Pages 15, 2, and 26 are in core, no change has been made to the permanent data base. When the buffer in which a page resides is needed for another page--or when the area is closed--only then does any physical alteration of data occur in the main storage. Which page is returned to the data base when buffer space is needed? The one at the bottom of the buffer lot. This is shown in I/O operations 4, 5, and 6 in Figure 6-11.

Four points should be remembered concerning the I/O buffering scheme:

(1) Three buffers are allocated for each OPEN area; this number is fixed.

(2) The buffer size is equal to the page size.

(3) The physical alteration of data in the DBS file does not occur until the page is returned to the disk because either its buffer space is needed for another page or the area is closed.

(4) Buffers remain allocated for the life of the run-unit.

6.2.3 STORE Algorithms

The algorithm used during a STORE operation depends on the location mode of the record to be stored.

6.2.3.1 DIRECT - As explained in Section 6.1.4, a database key (DBKEY) is a combination of the page number and the line number on which a record occurrence is stored. When a new occurrence is to be stored, and the record's location mode is DIRECT, one of the following algorithms is used:

(1) If the run-unit specifies a DBKEY, the DBCS ignores the record portion of the DBKEY and accesses the page specified. Once the page is in core, it is checked for space to store the new occurrence. If the room is there, a new line is added to the page; if not, the next page is checked for room, and so on until room is found for the new line. Should the last page of the area be reached, then the DBCS folds back to the first page of the area and continues the search for room.

(2) If no DBKEY is given, the page number of the current of area is used to begin the above process.

(3) If there is no current of area, the first page of the area is used as the starting point.

6.2.3.2 CALC - If the location mode is CALC, then all fields defined as the CALC fields for the record are hashed. The resulting number is folded over the pages of the area until this number can be matched with a page number. The CALC chain on that page is then searched for the next available line in which to locate this record occurrence. Each page has a CALC chain which links all the CALC records on that page, with ordering in the chain being according to record type. The first word after the line header for CALC records contains the pointer to the next CALC record.

If duplicates are allowed, and positioning is not requested, then the DBCS will place the record occurrence where it chooses. If positioning is requested--e.g., LAST--the DBCS will comply.

6.2.3.3 VIA - When the location mode is VIA set-name, the DBCS attempts to find the logical insert point for the record occurrence in that set, and will use that as the starting point to find room for the new line. If this logical point cannot be determined, the current page of the area is used for the starting point. If the current page of the area is not known, the first page of the area is used. VIA can be used to place records where the member records reside in a different area than the owner record.

6.2.4 FIND Algorithms

Again, these algorithms depend on the location mode of the record to be found. A FIND DIRECT will always work, regardless of mode, so long as the DBKEY is given. The DBCS will use the DBKEY to access the correct page and line directly. FIND CALC can only be used if the mode is CALC; in which case, the DBCS calculates, or randomizes, to the appropriate page, and follows the CALC chain on that page until it finds the desired record occurrence.

6.3 OVERHEAD

As has been seen in the previous sections, a certain amount of overhead is required by the DBMS-10 to hold the linking information, page and line headers, etc., in the main storage files used for the data base. This section describes how the overhead can be determined, although rough estimates of overhead are normally sufficient. The overhead can be defined in terms of record (line), page, and file overhead. Overhead in run-units is proportional to the storage overhead for that portion of the data base known to it and will not be discussed separately.

6.3.1 Record Overhead

For each record type defined for the data base, the following can be used to determine the amount of overhead in words:

(1)	location mode is CALC	1 word
(2)	owner of set types	1 word per set
(3)	member of set types	1 word per set
(4)	LINKED TO OWNER in set types	1 word per set
(5)	LINKED TO PRIOR in set types	1 word per set

Add to the calculation 1 word for the line header, and the overhead for each occurrence of this record type is found. Consider the following examples:

EXAMPLE 6.3

The record type INVENTORY-RECORD is a CALC record, is the owner of 4 sets, participates as a member in 5 sets, is LINKED TO OWNER in 3 of these, and is LINKED TO PRIOR in 1 of them. Using the above overhead determination, the record overhead would be calculated as:

```
 1    (for CALC chain)
 4    (for owner of 4 sets)
 5    (for member of 5 sets)
 3    (for OWNER links)
 1    (for PRIOR link)
--
14
```

Adding the word for the line header gives a total of 15 words of overhead for each occurrence of this record type.

EXAMPLE 6.4

The record type SUPPLIER-RECORD is a DIRECT record, and owns only one set. The line (record) overhead for this record type would be one word for the line header and one word for being an owner. Thus each line on which this record type appeared would have 2 words of overhead.

A stand-alone record--i.e., one with no set linkages--will have a minimum of one word of overhead if its location mode is DIRECT, and two words if it is a CALC record.

There will be times when the Data Base Administrator will have to decide whether to repeat a certain item of data in more than one record or to create a set to eliminate the data redundancy. If the redundant data would take up less than one 36-bit word in storage, the justification for using a set would be minimal. This is one of the judgments which Data Base Administrators have to make every time a data base is organized or reorganized--justification of set link overhead.

6.3.2 Page Overhead

Page overhead consists of the number of words which compose the page header. In general, this overhead can be calculated using the formaula:

Page Overhead = 3 + RND (R + 0.5) words

where R is the records-per-page divided by 36. In Example 6.1, the page overhead would be 5 words; in Example 6.2, it would be 7 words.

6.3.3 File Overhead

In addition to page and line overhead, every DBS file has a 128 (200_8) word sector at the front of the file used to store recovery file (see Chapter 8) information. This overhead is inherent to the system and cannot be altered by the Data Base Administrator or a user.

CHAPTER 7

THE SCHEMA DIRECTORY FILE

Unique to every data base is a file created by the DDL processor and named schema-name.SCH. This file reflects the attributes of the data base, which was described using the DMCL and the DDLs, in terms of a data base. In other words, the schema directory file is itself a data base used to map another data base.

7.1 INTRODUCTION

The schema directory file (hereafter called the SCH file because of its extension) is physically structured exactly like a DBS file. It has a 128 (200_8) word recovery sector at the beginning, is divided into 32 pages of 512 (1000_8) words each, and each page is further divided into lines. The page headers are 7 words long - as if a records-per-page of 127 had been specified - and the lines are complete with line headers and set pointers. Each line on the page contains an occurrence of a record type - the same as in the DBS files. The lowest record type code allocated by the DDL processor to a record type submitted in a DDL entry is 101 (145_8). The reason for this is that codes 001 through 100 (144_8) are reserved by the DDL processor for assignment to record types in the SCH file. Figure 7-1 shows the record types, which make up the data base in the SCH file, and the corresponding set structure. The codes for the record types are given below in octal.

Table 7-1

Codes for Record Types

RECORD TYPE CODE	RECORD TYPE
ØØ1	Schema Block
ØØ2	Record Block
ØØ3	Data Block
ØØ4	Control Block
ØØ5	Member Block
ØØ6	Owner Block
ØØ7	Within Block
Ø1Ø	Area Block
Ø11	Text Block
Ø12	Sub-Schema Block

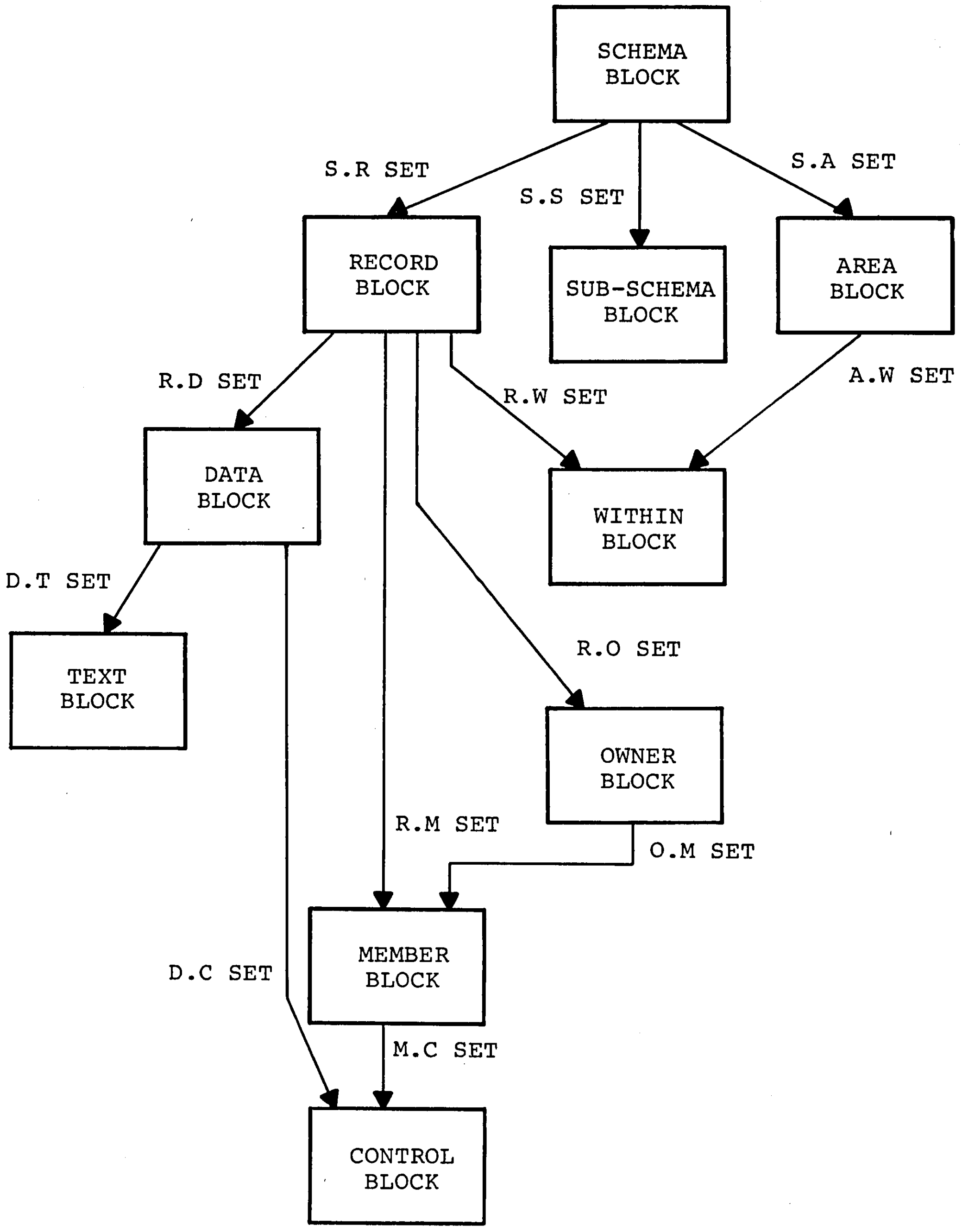


Figure 7-1 Schema Directory Representation

Currently codes 013_8 through 144_8 are reserved for future use by the DDL processor.

The lines on the pages are divided again into header, pointers, and data. Data, in the case of the SCH file, is in the form of the word blocks which specify different attributes of the data base being described. These word blocks are similar to the word blocks used by the DBCS for the in-core representation (see Section 6.2.1). In fact, the in-core blocks are initially written from the blocks in the SCH file. The next section will describe how the blocks in the SCH file appear. Section 7.3 will discuss how the DML preprocessor used this schema directory to write an executable COBOL program.

7.2 WORD BLOCKS IN SCH FILE

Ten word blocks are used in the SCH file. Eight of them are used by the DBCS to create its in-core representation; the other two are unique to the SCH file.

7.2.1 Schema Block

The Schema block is the data portion of the schema line in the SCH file. This one-word block (see Figure 7-2) provides information on how many sub-schemas have been specified for the schema and how many bits per record are used in the DBKEYS. This block is used to write the in-core sub-schema block.

0	////////	LAST S-S BIT	BITS PER RECORD

Figure 7-2 SCH Schema Block

LAST S-S BIT indicates the last bit used to designate sub-schema membership. This is checked by the DDL processor when new sub-schemas are added.

BITS PER RECORD denotes how many rightmost bits in the DBKEY are allocated for the line number. See Section 6.2 for how this is calculated.

7.2.2 Sub-schema Block

As shown in Figure 7-3, the sub-schema block is a block of 7 words which serves to identify a sub-schema. There is one for each sub-schema in the schema.

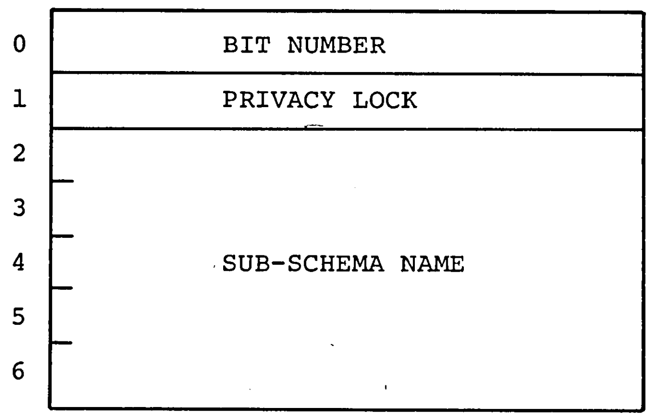


Figure 7-3 SCH Sub-schema Block

BIT NUMBER indicates which bit is used in the other blocks to indicate membership in this sub-schema.

PRIVACY LOCK is the six character privacy lock, stored in SIXBIT.

SUB-SCHEMA NAME is stored in SIXBIT.

7.2.3 Record Block

The record block is a 13-word block (see Figure 7-4, and identifies a record type according to its

(1) type
(2) name
(3) location

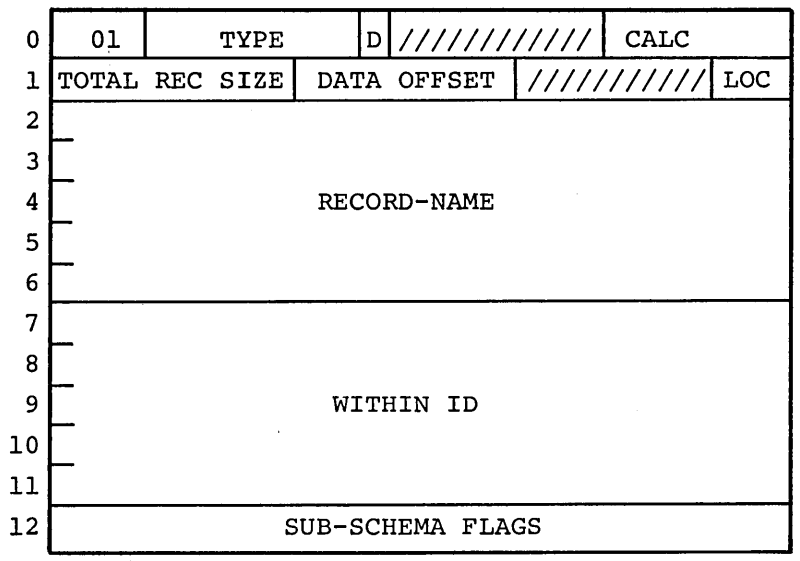


Figure 7-4 SCH Record Block

Each record defined in the data base has a record block in the SCH file, and this block serves as the basis for the in-core record block.

TYPE is the record type code (see Section 7.1).

CALC is the number of fields on which this record is CALCed.
D in bit 18 of word 0 is the duplicates allowed flag - when on, they are permitted.

TOT REC SIZE is the total line size in words for an occurrence of this record type in the data base.

DATA OFFSET is the number of words in the line used for header and pointers.

LOC is the same as for the in-core record block (see Section 6.2.1).

RECORD-NAME is the name of the record in SIXBIT.

WITHIN ID is the AREA-ID (if specified) in SIXBIT (see Section 4.4.1).

SUB-SCHEMA FLAGS denote for which sub-schema this record type is defined.

7.2.4 Area Block

Each area defined for the data base has an entry in the SCH file, and its area block comprises 15 words, (see Figure 7-5). These blocks are used to create the area blocks in core by the DBCS.

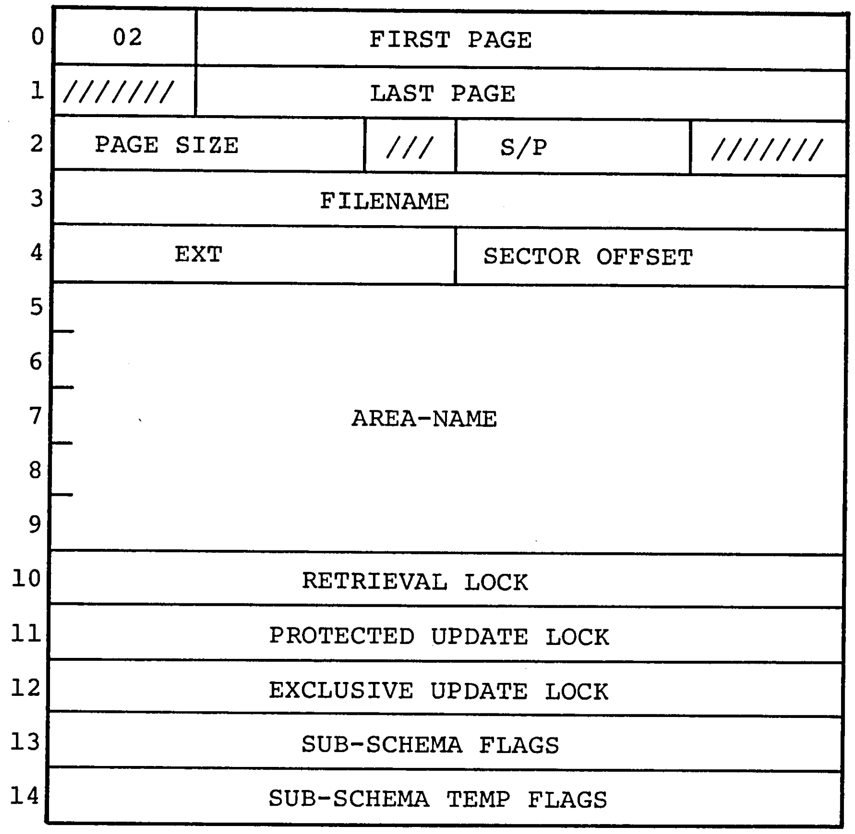


Figure 7-5 SCH Area Block

SUB-SCHEMA FLAGS are the same as for the record block.

SUB-SCHEMA TEMP FLAGS denotes for which sub-schema this area is defined as sub-schema temporary.

For all other notes see "Area Block" in Section 6.2.1.

7.2.5 Within Block

The within block merely provides the link between a record type and the areas within which it may be located. RECORD TYPE in this single-word block (see Figure 7-6) is the record type code for the record type whose WITHIN this is. The in-core within block is written from this block.

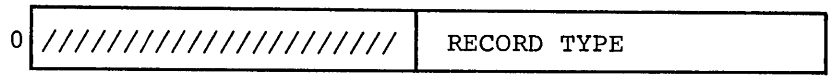


Figure 7-6 SCH Within Block

7.2.6 Owner Block

An owner block is written in the SCH file for each set type defined for the schema. The block of seven words used to specify the set is shown in Figure 7-7, and is used as the basis for the in-core owner block.

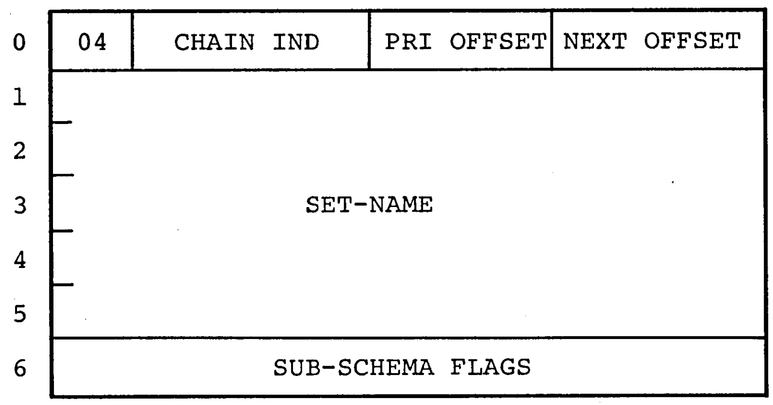


Figure 7-7 SCH Owner Block

Word 0 is the same as for the in-core block (see Section 6.2.1).

SET-NAME is the name of the set in SIXBIT.

The SUB-SCHEMA FLAGS are the same as described for the record block.

7.2.7 Member Block

Each record that participates as a member of a set has associated with it a member block that serves to identify it as a member of that set. The SCH member block is three words long (Figure 7-8) and is used to form the in-core member block. Words 0 and 1 are the same as for the in-core block (see Section 6.2.1). TYPE R.M OWNER refers to the record type code for the record type that is the owner of the set in which this record is a member.

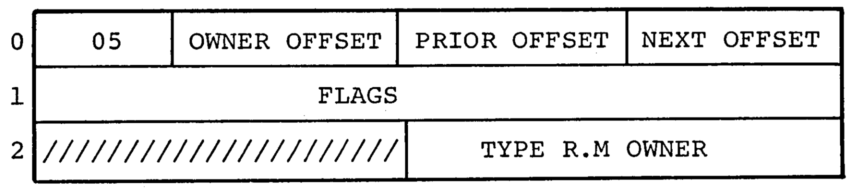


Figure 7-8 SCH Member Block

7.2.8 Control Block

The control block appears for the same reasons as the in-core control block - to specify SORT keys or range keys, or to specify set selection invoking a USING clause. The control line carries the pointers, and the block itself (see Figure 7-9) only contains the FLAGS (same as for the in-core block) and the ALIAS NAME (see Section 4.5) in SIXBIT if it is used.

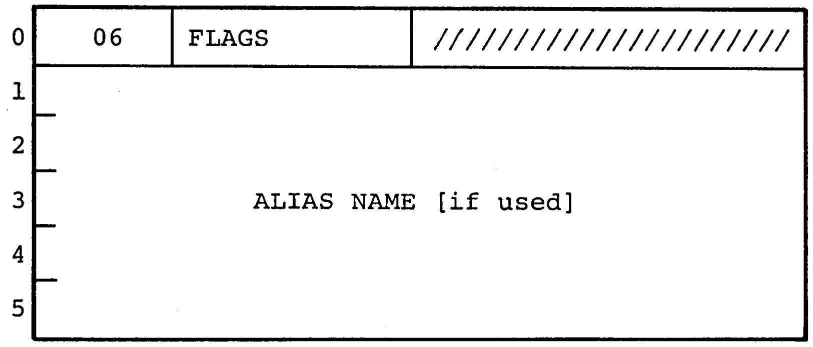


Figure 7-9 SCH Control Block

7.2.9 Data Block

An occurrence of the line for data is written for each data-item and data-aggregate defined for the schema. This 17-word block (Figure 7-10) is used to generate the in-core data block.

Word 0 is identical for both data blocks (see Section 6.2.1), as are words 2 and 3.

RECORD TYPE in word 4 is the record type code of the record type of which this data is part.

DATA-NAME is the data-name in SIXBIT.

PICTURE is the picture string for data-items in ASCII. For data-aggregates this is null.

SUB-SCHEMA FLAGS are the same as for the record block.

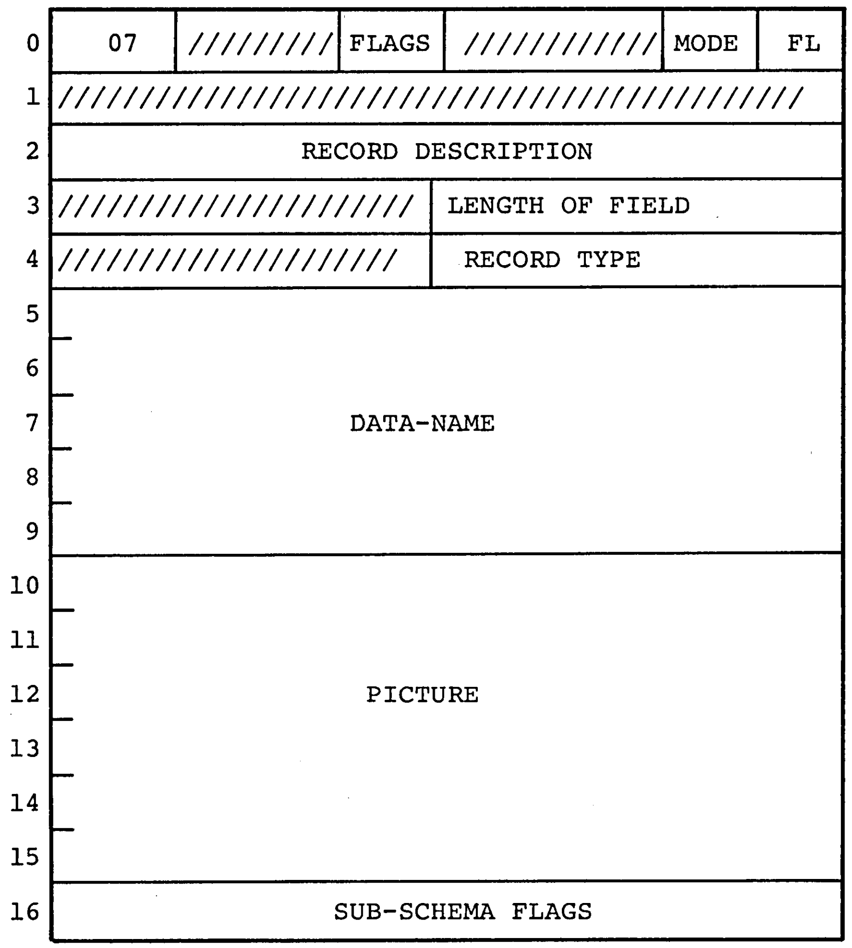


Figure 7-10 SCH Data Block

7.2.10 Text Block

The text block is unique to the SCH file. The text block itself is written for every data-aggregate specified in the schema description. It is 22 words long, with the 21 words after the character count being used for 105 7-bit ASCII characters to describe the data sub-entries below the 02 level (see Section 4.4.2). Figure 7-11 illustrates the format of the text block.

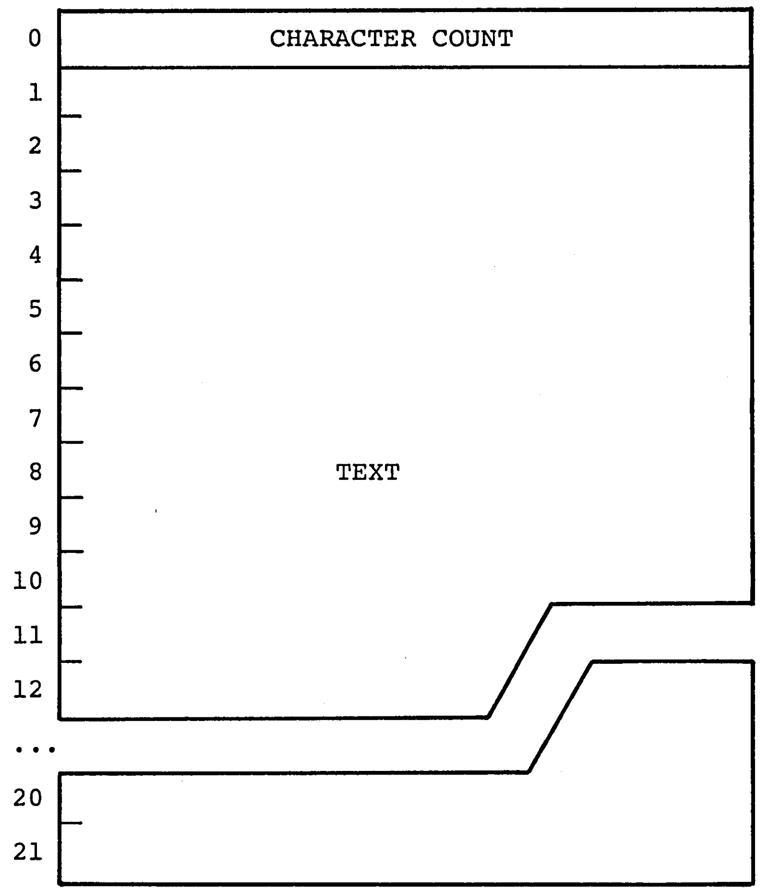


Figure 7-11 SCH Text Block

7.3 COBOL DML PREPROCESSOR

Until such time that the COBOL DML imperatives are interfaced with COBOL in the COBOL compiler, it will be necessary to pass programs containing the COBOL DML through the COBOL DML preprocessor in order to obtain an executable COBOL program called LAFCOB.CBL. The workings of the preprocessor are as follows. First, the preprocessor scans the specified source code for any COBOL DML statements. If none are found, it aborts the job; if these statements exist, it begins its task of writing the COBOL program.

Next, the preprocessor checks for an INVOKE statement. When this is found, the SCH file for the schema specified is opened and the privacy key FOR COMPILE is compared with the privacy lock for the sub-schema invoked. Should an invalid sub-schema-name be given, or a key/lock match fail, no further processing is done. Otherwise, the WORKING-STORAGE SECTION of the program is modified to include the System Communication Locations (SCL), a register called DBMS-NULL which is used in MACRO arguments, and those records and associated data-items and data-aggregates that are defined as belonging to the sub-schema invoked. Figure 7-12 shows an INVOKE statement from a COBOL DML program, and gives the WORKING-STORAGE SECTION changes made by the preprocessor from the information read from the Schema Directory File.

INVOKE Statement

```
IDENTIFICATION DIVISION.
PROGRAM-ID. STKSPL.

DATA DIVISION.
WORKING-STORAGE SECTION.
*       DBMS
        INVOKE SUPPL-INVENT OF SCHEMA JTAB01 PRIVACY KEY COMPILE JTCC01.
```

Resulting Code

```
IDENTIFICATION DIVISION.
PROGRAM-ID. STKSPL.

DATA DIVISION.
WORKING-STORAGE SECTION.
77      AREA-NAME, PIC X(30).
77      RECORD-NAME, PIC X(30).
77      ERROR-STATUS, PIC 9(6).
77      ERROR-SET, PIC X(30).
77      ERROR-RECORD, PIC X(30).
77      ERROR-AREA, PIC X(30).
77      ERROR-COUNT, PIC 99, USAGE COMP.
77      DBMS-NULL PIC 99 USAGE COMP.
01    INVENTORY-RECORD.
     02  PART-NUM PIC X(8) USAGE DISPLAY.
     02  PART-DESCRIP PIC X(20) USAGE DISPLAY.
     02  QUANTITY-ON-HND PIC 9(6) USAGE DISPLAY.
     02  REORDER-AT PIC 9(4) USAGE DISPLAY.
     02  QUANTITY-ON-ORD PIC 9(6) USAGE DISPLAY.
     02  QUANTITY-TO-SHP PIC 9(6) USAGE DISPLAY.
     02  UNIT-COST PIC 9(5)V99 USAGE DISPLAY.
     02  UNIT-LIST PIC 9(5)V99 USAGE DISPLAY.
     02  UNIT-RETAIL PIC 9(5)V99 USAGE DISPLAY.
     02  UNIT-QUANTITY PIC 9(4) USAGE DISPLAY.
01   SUPPLIER-RECORD.
     02  SUPPL-CODE PIC X(6) USAGE DISPLAY.
     02  SUPPLIER PIC X(15) USAGE DISPLAY.
     02  SUPPL-ADDRESS PIC X(20) USAGE DISPLAY.
     02  SUPPL-CITY PIC X(10) USAGE DISPLAY.
     02  SUPPL-STATE PIC X(2) USAGE DISPLAY.
     02  SUPPL-ZIP PIC 9(5) USAGE DISPLAY.
     02  SUPPL-PHONE PIC 9(12) USAGE DISPLAY.
```

Figure 7-12 INVOKE Statement Example

In addition, a DBMS SECTION is added to the PROCEDURE DIVISION. This section provides the arguments for a MACRO subroutine, which will later be loaded with the COBOL program, that binds the system parameters. This is shown for the same INVOKE statement in Figure 7-13.

```
PROCEDURE DIVISION.
DBMS-SECTION.
        ENTER MACRO SBIND USING "SUPPL-INVENT","JTABØ1",ERROR-COUNT
        ENTER MACRO BIND USING "INVENTORY-RECORD"
-       ,PART-NUM
-       ,PART-DESCRIP
-       ,QUANTITY-ON-HND
-       ,REORDER-AT
-       ,QUANTITY-ON-ORD
-       ,QUANTITY-TO-SHP
-       ,UNIT-COST
-       ,UNIT-LIST
-       ,UNIT-RETAIL
-       ,UNIT-QUANTITY
-       .
        ENTER MACRO BIND USING "SUPPLIER-RECORD"
-       ,SUPPL-CODE
-       ,SUPPLIER
-       ,SUPPL-ADDRESS
-       ,SUPPL-CITY
-       ,SUPPL-STATE
-       ,SUPPL-ZIP
-       ,SUPPL-PHONE
-       .
```

Figure 7-13 DBMS Section Example

The remaining COBOL DML statements are translated into arguments for the ENTER MACRO verb. The ENTER MACRO verb is used for linkage to MACRO subroutines which are external to the resultant COBOL program. These MACRO subroutines are later loaded with the compiled COBOL program to form a single, unified program. The COBOL DML preprocessor during this phase does not check if the arguments given in the COBOL DML code are valid for the sub-schema invoked - this is done during run-time by the DBCS. However, syntax checking is done, as is checking for key words being used as arguments.

Three examples are given below which illustrate the conversion of COBOL DML statements into MACRO calls.

EXAMPLE 7.1

```
     FIND LAST STKHLD RECORD OF AREA1 AREA.
```

The preceding code will generate:

```
     ENTER MACRO FIND3 USING "LAST", "STKHLD", "AREA1", "AREA".
```

EXAMPLE 7.2

```
     MODIFY DVQTR3.
```

This code will generate:

```
     ENTER MACRO MODIFY USING "DVQTR3".
```

EXAMPLE 7.3

```
     OPEN ALL USAGE MODE IS EXCLUSIVE UPDATE.
```

This code will generate:

```
     ENTER MACRO OPEND USING "UPDATE", "EXCLUSIVE", DBMS-NULL,
-    , "ALL".
```

CHAPTER 8

RECOVERY FILES

Unlike other files, DBMS-10 mass storage files are susceptible to damage if a terminal, system, or run-unit problem occurs while one of these files is open for update. DBMS-10, therefore, provides a system of recovery files which are used to maintain a backup to data in the data base. This chapter describes these files and their use.

8.1 RECOVERY FILE OPERATION

Every file created or used by DBMS-10 - regardless whether it is a mass storage (DBS) file, a schema directory (SCH) file, or a recovery (RCV) file - has a recovery file written for it every time it is opened for protected update. Basically, the recovery file operates as follows. When a file is changed by execution of the DML commands, the page on which the change was made is written into the recovery file created for that main file. If any problems occur during the execution of the run-unit, the recovery file may be damaged but the main file is unaffected. It will be necessary to run the program again to perform the update a second time, but changes made by previous run-units are safe in the main file and need not be restored. Thus, data integrity is assured during update.

When a DBMS-10 file is opened, the DBCS will check for the filename and file extension of a recovery file in the main file if the DBCS finds a CURRENT STATUS of RECOV or UPDAT (see Section 8.2). In the case of the latter, a single reader will have the pages from the RCV file written into the main file. If there are other readers, a new RCV file will be written which is a merge of the older RCV file and any new changes made to the data in the data base. This new RCV file would then be recovered later. When RECOV is encountered, and there are no other readers, the main file is updated in place from the RCV file.

If a DBS file or a SCH file has an ABORT STATUS, the data within it is questionable and should be refreshed from the latest backup source. The DBCS will not process an aborted file. The fourth possibility is that the STATUS is READY, in which case the data in the main file is in good order and is up to date.

The size of an RCV file depends on:

(1) the number of pages in the main file;

(2) the number of pages needing recovery.

For every multiple (or fractional multiple) of 128 (200_8) pages in the main file, a 128 (200_8) word sector is created in the RCV file to contain page pointers to the recovery pages. The rest of the file (except the prefix sector described in the next section) grows as each new page is added to the file. As each page is added, a pointer is created for it. The recovery file is closed when the area is closed. Thus the recovery file size can be smaller than, equal to, or greater than the size of the file for which it was created.

During a merge of RCV files, the prefix sector of the newer file supersedes all other prefix sectors. The old recovery pages are made part of the new RCV file, and new pages are added as needed. Any change made to a recovery page which is already in a RCV file supersedes the older page.

8.2 PREFIX SECTOR

All DBMS-10 files have a 128 (200_8) 36-bit word sector preceding all other contents which is known as the prefix sector and is part of the file overhead. This sector serves to specify:

(1) the current status of the file;

(2) the filename and file extension of the latest recovery file written for the file;

(3) certain access information.

A schematic of the 7-word recovery block of this prefix (or recovery) sector is shown in Figure 8-1; a description of the block follows.

CURRENT STATUS is a 5-character ASCII word which denotes the present status of the file, and is used by the DBCS to determine if recovery is needed - or even possible (see Section 8.1). The four conditions are:

READY	- no recovery file exists and the data in the file is good.
ABORT	- for some reason the file was not closed successfully.
UPDAT	- the file was aborted in the middle of a recovery procedure to update the main file.
RECOV	- a recovery file exists for this file and the data in the file is processable.

0	CURRENT STATUS	
1	RECOVERY FILENAME	
2	EXT	//////////////////////
3	AREA SEQUENCE #	
4	NEXT AVAILABLE SECTOR	
5	NUMBER OF READERS	
6	FIRST RECOVERY SECTOR	

Figure 8-1 Prefix Sector

RECOVERY FILENAME is the filename of the recovery file in SIXBIT. These filenames are randomly generated internally by the DBCS as a function of time of day when needed.

EXT is the file extension for the recovery file in SIXBIT.

AREA SEQUENCE # is a binary number, and is the same as the one used by the DBCS in the in-core area block (see Section 6.1.2). This number is used by the DBCS in order to assign a reader to the proper version of the data base.

NEXT AVAILABLE SECTOR is the number of the next available sector in the recovery file in which a page can be begun (only used in a recovery file prefix sector).

NUMBER OF READERS is the number of run-units currently accessing the file.

FIRST RECOVERY SECTOR is the number of the first sector in the recovery file that is used for recovery pages (only used in a recovery file prefix sector).

Every time a file is opened, for whatever reason, its prefix sector is updated. Thus, the name of the recovery file for any given file is always the latest one available.

APPENDIX A

ERROR MESSAGES

The error messages issued by the DDL processor, the DML pre-processor, and the DBCS are listed in the following sections. Lower case words are generic terms; the actual name or number is given when the message is issued.

A.1 DDL PROCESSOR MESSAGES

Listed below are the error messages that can be received during the processing of the DMCL, Schema DDL, and COBOL Sub-Schema DDL entries by the DDL processor. Those messages that are marked D are associated with DMCL entries; those marked S with schema entries; those marked C with sub-schema entries. Those messages marked with E indicate messages that are output at the end of processing, and those with A are associated with the processing of all entries. Any message beginning with the word WARNING is a warning message and does not affect the processing. All other messages indicate fatal errors, and processing of the description file cannot be done. Most messages are accompanied by the phrase

```
ON LINE # n
```

where n is the line of the description file on which the error occurs.

If the message

```
PROGRAM ERROR
```

should be received, a DDL processor error has occurred, and the processor has aborted. The Software Specialist should be immediately notified, as this indicates a processor failure.

S	data-name UNDEF IN RECORD
A	END-OF-FILE BEFORE END-SCHEMA
S	ERROR - A/D INVALID WITH DBKEY SORT
S	ERROR - A/D MUST BE FOR SORTED SET
S	ERROR - AREA-ID MISSING
A	ERROR - AREA-NAME MISSING

S	ERROR - CANNOT BE BOTH OWNER AND MEMBER OF SET
S	ERROR - CANNOT HAVE BOTH DISPLAY 6 AND DISPLAY 7
S	ERROR - DATA-NAME HAS CONFLICTING MODES
DS	ERROR - DEVICE-MEDIA CONTROL MISSING FOR AREA
S	ERROR - DIRECT FIELD MISSING
A	ERROR - DUPLICATE AREA-NAME
D	ERROR - DUPLICATE CLAUSE
S	ERROR - DUPLICATE DATA-NAME
SC	ERROR - DUPLICATE PRIVACY LOCK
SC	ERROR - DUPLICATE RECORD-NAME
D	ERROR - DUPLICATE RECORDS-PER-PAGE FOUND
SC	ERROR - DUPLICATE SET-NAME
C	ERROR - DUPLICATE SUB-SCHEMA NAME
D	ERROR - FILENAME MUST BE UNIQUE
D	ERROR - FIRST PAGE MISSING
DS	ERROR - INVALID AREA-NAME
S	ERROR - INVALID CALC FIELD
S	ERROR - INVALID DATA-NAME
D	ERROR - INVALID DUPLICATE AREA-NAMES
D	ERROR - INVALID FILENAME
C	ERROR - INVALID NAME
D	ERROR - INVALID NUMERIC VALUE
S	ERROR - INVALID OR MISSING-DATA-NAME
D	ERROR - INVALID PAGE RANGE
S	ERROR - INVALID PICTURE CLAUSE
SC	ERROR - INVALID SET-NAME
A	ERROR - KEYWORD MISSING
D	ERROR - LAST PAGE MISSING
S	ERROR - LOCATION MODE MISSING
S	ERROR - MODE CLAUSE MUST PRECEDE MEMBER
S	ERROR - MULTIPLE SCHEMA ENTRIES

D	ERROR - NUMERIC EXPECTED
S	ERROR - OWNER CLAUSE MUST PRECEDE MEMBER CLAUSE
C	ERROR - OWNER OF SET NOT IN SUB-SCHEMA
D	ERROR - PAGE SIZE TOO LARGE
D	ERROR - PAGES CANNOT OVERLAP
SC	ERROR - RECORD-NAME MISSING
S	ERROR - SCHEMA NAME MISSING
S	ERROR - SIZE CANNOT BE ZERO
S	ERROR - SORT FIELDS DO NOT AGREE
D	ERROR - TOO MANY FILES
S	ERROR - TOO MANY SORT FIELDS DEFINED
C	ERROR - TOO MANY SUB-SCHEMAS DEFINED
SC	ERROR - UNDEFINED RECORD-NAME
A	ERROR - UNEXPECTED END-OF-STATEMENT
S	ERROR - VIA SET set-name INVALID FOR RECORD record-name
S	ERROR - WITHIN CLAUSE MISSING
S	ERRORS FOR SET set-name *********
C	INVALID OR MISSING RECORD-NAME
C	MISSING OR INVALID PRIVACY LOCK
S	MODE CLAUSE MISSING
C	NO AREA DEFINED IN SUB-SCHEMA FOR RECORD
S	ORDER CLAUSE MISSING
S	OWNER CLAUSE MISSING
S	SORT FIELD(S) MISSING
S	SORT FIELDS DO NOT AGREE IN NUMBER MODE SIZE
A	WARNING - PERIOD MISSING - ASSUMED
A	WARNING - UNEXPECTED PERIOD - IGNORED
E	***** n ERRORS FOUND *****

A.2 DML PREPROCESSOR MESSAGES

The error messages from the DML preprocessor are listed below. All preprocessor errors are fatal. Except for the two messages indicated by *, the error messages are followed by

```
ON LINE # n
```

where n is the line on which the error occurs.

```
     ERROR - INVALID AREA-NAME
     ERROR - INVALID DATA-NAME
     ERROR - INVALID RECORD-NAME
     ERROR - INVALID SET-NAME
     ERROR - KEYWORD EXPECTED
     ERROR - UNEXPECTED END-OF-STATEMENT
*    FATAL ERROR - NO PROCEDURE DIVISION FOUND
     INVALID INVOKE STATEMENT
*    NO DBMS STATEMENTS FOUND
```

A.3 DBCS MESSAGES

The error messages from the object-time system, DBCS, are listed below. All of these messages are fatal and denote serious failures of DBMS-10. The Data Base Administrator should determine the cause for this failure, as the correction needed may be as simple as adding more storage space for the data base. If the Data Base Administrator cannot correct the situation which caused the error, it should be reported immediately to the Software Specialist.

```
ALLOCR PAGE FAILURE
BLKERR IN GETBLK
CANNOT OPEN SCHEMA DEFINITION FILE
DYNAMIC STORAGE EXHAUSTED
FATAL ERROR IN DIRECTORY I/O
FATAL SET MUST WRITE ERROR
PAGE REQUESTED IS NOT PAGE READ
RECOVERY PAGE I-O ERROR
```

APPENDIX B

ERROR STATUS CONDITION CODES

The Error Status Condition codes which are made available in the special register ERROR-STATUS are actually the combination of a major code--designating which imperative was attempted--and a minor code--denoting the condition which caused the failure of execution. Thus, the contents of ERROR-STATUS would read as MMmm, where MM is the major code and mm is the minor code.

The following table is a summary of the DML commands and their associated major codes.

Table B-1

Major Error Codes

DML COMMAND	MAJOR CODE
CLOSE	01
DELETE	02
FIND	03
GET	05
INSERT	07
MODIFY	08
OPEN	09
REMOVE	11
STORE	12

The minor codes representing the existing Error Status Condition appear in this second table.

Table B-2

Minor Error Codes

CONDITION	MINOR CODE
Area not open	01
Database key inconsistent with area-name	02
Violation of DUPLICATES NOT ALLOWED clause	05
Current of set, area, or record-name not known	06
End of set, area, or record	07

Table B-2 (Cont.)

CONDITION	MINOR CODE
Referenced area, record, or set-name not in sub-schema	08
Incorrect usage mode for area	09
Privacy breach attempted	10
Physical space not available	11
Database key not available	12
No current record of run-unit	13
Object record is mandatory automatic in named set	14
Object record is mandatory in named set	15
Record already a member of named set	16
Current record of run-unit not of record-name	20
Record not currently member of named or implied set	22
No set occurrence satisfies argument values	25
No record satisfies rse specified	26
Area already opened	28
Unqualified DELETE attempted on non-empty set	30
Removed record involved	50
Deleted record involved	51
Function requested not permitted	78
Access requested conflicts with existing access	87
No channels available	88
Area in abort status	89
Usage mode conflict	99

APPENDIX C

DBMS-10 PROCESSES

This appendix gives a brief outline of the processes involved in using the DBMS-10.

C.1 DATA DEFINITION PROCESS

To create a data base, the following steps should be taken:

(1) Draw a schematic of the data structure to be built, deciding which record types are to be owner records, which are to be member records, which should stand alone, what location mode should be used for a given record, what data relationships are to be maintained, etc. Also decide which record types should be located in different areas.

(2) Using the Schema DDL, write the schema description of each area, record type, and set type (see Chapter 4).

(3) Decide in what files the areas are to be stored. Define the records-per-page for the schema and establish the page size and page range for each area. Suppose a data base is to be designed which will hold data on a company's inventory and its customers. Primary concern would center around inventory information and sales information. Then set up an area to hold the data essentially belonging to the former, and one for the data inherent to the latter.

(4) Define as many sub-schemas as might be necessary.

(5) Complete the description of the data base and save it as a permanent file having either no extension or the extension .DDL. This file will be passed through the DDL processor which will create the schema file.

An example of a data base description file is given in Appendix E. Figure C-1 illustrates the Data Definition Process.

To run the DDL processor, the following sequence is observed:

```
.R DDL
INPUT DDL FILENAME
*?
```

at which point the name of the file containing the data base description is typed followed by a carriage return. If any errors are encountered during processing (see Appendix A), the DDL processor will not create the SCH or the DBS files. Fix the errors and reprocess.

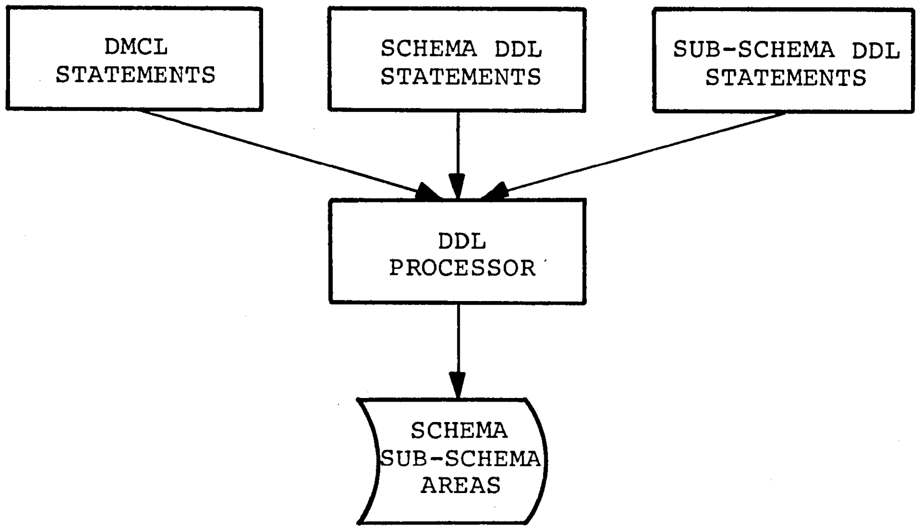


Figure C-1 Data Definition Process

C.2 PROGRAM BUILDING PROCESS

To create an executable COBOL program using the COBOL DML commands, the following steps should be taken:

(1) Create a new program or use an existing one.

(2) Run the preprocessor.

(3) Compile the resultant COBOL program.

(4) Load the relocatable object code with the LIBOL and DBCS object-time systems.

(5) Optionally save the object code.

(6) Execute the run unit.

Figure C-2 shows this process schematically.

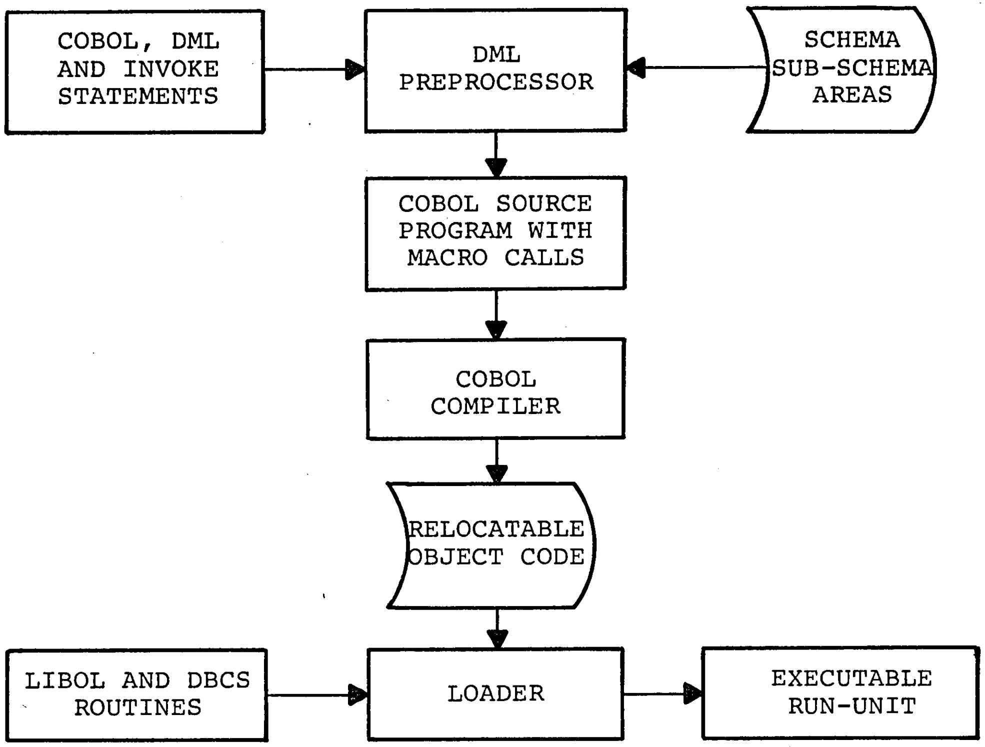


Figure C-2 Program Building Process

The following examples show how this is done at the terminal.

<u>EXAMPLE C.1</u>

Running an old COBOL DML program:

```
.RUN DSK:OLDFIL
```

<u>EXAMPLE C.2</u>

Creating a new program:

```
.R DDL
INPUT DDL FILENAME
*NEWFILE
.COMPILE LAFCOB.CBL
.LOAD LAFCOB, SYS:DBCS.REL
.EXECUTE LAFCOB
```

APPENDIX D

RESERVED WORDS

In addition to the standard COBOL reserved words, the following words are reserved in DBMS-10, along with their abbreviations enclosed in parentheses.

-A-

ACTUAL
ALIAS
ALL
ALLOWED
ALTER
ALWAYS
ARE
AREA
AREA-CODE
AREA-ID
ASCENDING
AUTOMATIC (AUTO)

-B-

BINARY (BIN)
BIT
BY

-C-

CALC
CALL
CHAIN
CHARACTER (CHAR)
CHECK
CLOSE
COMPLEX
CURRENT

-D-

DATABASE-KEY (DBKEY)
DECIMAL (DEC)
DECODING
DELETE
DESCENDING (DESC)
DIRECT
DISPLAY
DUPLICATES (DUP)
DYNAMIC

-E-

ENCODING
EXCLUSIVE (EXCL)

-F-

FIND
FIRST
FIXED
FLOAT
FOR

-G-

GET

-I-

IF
IN
INDEX
INDEXED
INSERT
IS

-K-

KEY

-L-

LAST
LINKED
LOCATION (LOC)
LOCK
LOCKS

-M-

MANDATORY
MANUAL
MEMBER
MEMBERS
MODE
MODIFY

-N-

NON-EXCLUSIVE (NEXCL)
NOT

-O-

OCCURRENCE
OCCURS
OF
ON
ONLY
OPEN
OPTIONAL (OPT)
OR
ORDER
OWNER

-P-

PICTURE (PIC)
POINTER-ARRAY (PTR)
PRIOR
PRIVACY
PROCEDURE (PROC)
PROTECTED (PROT)

-R-

RANGE
REAL
RECORD
RECORD-NAME
REMOVE
RESULT
RETRIEVAL (RETR)

-S-

SCHEMA
SEARCH
SELECTION
SELECTIVE
SET
SORTED
SOURCE
STORE
SYSTEM

-T-

TEMPORARY (TEMP)
THRU
TIMES
TO

-U-

UPDATE
USAGE
USE
USING

-V-

VALUE
VIA
VIRTUAL

-W-

WITHIN

APPENDIX E

A SCHEMA/SUB-SCHEMA EXAMPLE

Consider a research company which is organized into departments. Some of these departments are actively engaged in research; others are support departments--e.g., accounting, payroll, legal, etc. Each research department is broken up into research project groups, although there are some special projects which overlap departments. The research projects are supported by contracts and grants which come from external sources--some projects receiving funds from more than one contract or grant. The employees of the company are grouped according to department, project, and personnel classification. Research personnel--other than project leaders--work only at one project at a time. There are, though, some project leaders who administer more than one project. The example at hand, then, is to create a data base which contains data and data relationships which describe this activity.

The schema needed to describe the company's organization, personnel, and associated research projects is quite simple, but is illustrative as an example. A step-by-step approach in its creation follows.

(1) Sub-divide the data into areas. Two areas might be used for this example--one which encompasses data on the corporate organization, personnel, and employee classifications (call it PERSONNEL-AREA), and another which handles data on the research projects, research contracts, and contract sources (call it RESEARCH-AREA).

(2) Assign these areas to disk files. Just call the files FILE01 and FILE02, respectively, and set the page size at 127 words. Let the records-per-page be 63. How many pages should be allotted to an area? The actual number would depend on the size of the company. For the sake of the example, let PERSONNEL-AREA have 100 pages, and RESEARCH-AREA have 60 pages.

(3) Define record types and associated data-items and data aggregates. The record types chosen for this example reflect the aforementioned attributes of the company. They are

```
DEPARTMENT-RECORD
EMPLOYEE-RECORD
CLASSIFICATION-RECORD
PROJECT-RECORD
CONTRACT-RECORD
SOURCE-RECORD
```

The data-items shown as part of the schema description in Figure E-2 are self-explanatory.

(4) Define set relationships and set types. Eight sets have been defined for this example, and are illustrated in Figure E-1. Note that two of these sets involve situations where owners and members have been reversed to form new sets. In the one case this is necessitated by the fact that some projects cross department lines; in the other, because some employees can be project leader of more than one project.

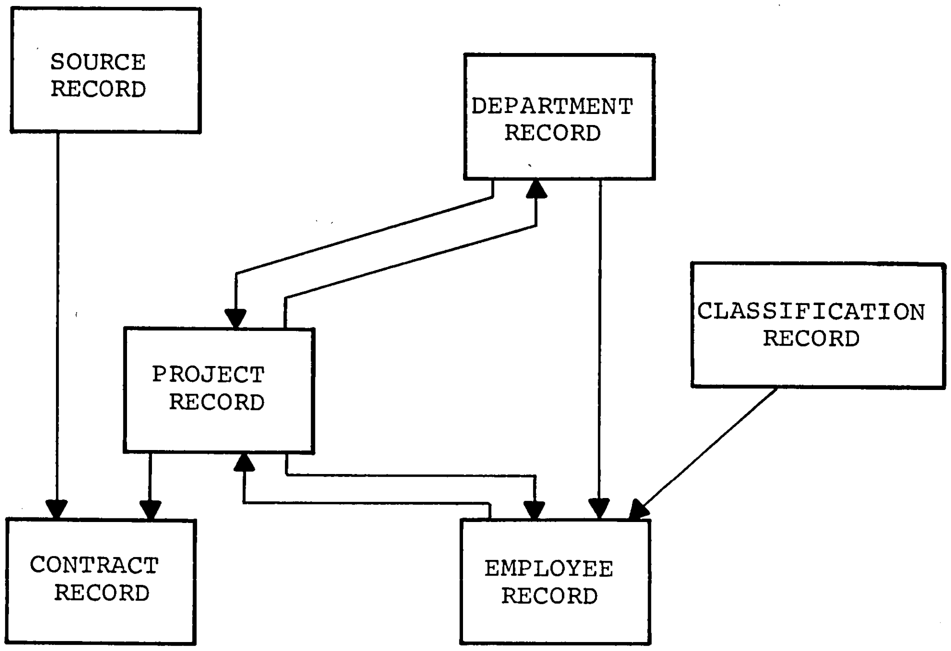


Figure E-1 Examples of Set Relationships

Sub-schema designation is also possible. Three are provided in the example, and they are self-explanatory.

In no way does this example exhaust all the possible ways to establish data bases and describe them. It does, though, illustrate many of the major points discussed in those chapters concerning the DMCL and the DDLs. Note that no comments are permitted in a DDL file; files containing comments would be aborted.

```
RECORDS-PER-PAGE 63.
ASSIGN PERSONNEL-AREA TO FILE01
FIRST PAGE IS 1
LAST PAGE IS 100
PAGE SIZE IS 127 WORDS.

ASSIGN RESEARCH-AREA TO FILE02
FIRST PAGE IS 300
LAST PAGE IS 359
PAGE SIZE IS 127 WORDS.

SCHEMA NAME IS SCHEX.

AREA NAME IS PERSONNEL-AREA
        PRIVACY EXCLUSIVE UPDATE IS PREXUP
        PRIVACY PROTECTED UPDATE IS PRPTUP
        PRIVACY RETRIEVAL IS PRRTLK.

AREA NAME IS RESEARCH-AREA
        PRIVACY LOCK RSCHLK.

RECORD NAME IS DEPARTMENT-RECORD
        LOCATION MODE IS DIRECT DPTKEY
        WITHIN PERSONNEL-AREA.

02      DEPARTMENT      PIC X(20).

RECORD NAME IS EMPLOYEE-RECORD
        LOCATION MODE VIA CLASS-SET
        WITHIN PERSONNEL-AREA.

02      EMPLOYEE        PIC X(25).
02      HOME-ADDRESS    PIC X(40).
02      HOME-PHONE      PIC 9(10).
02      SS-NUMBER       PIC 9(9).
02      MARITAL-STATUS  PIC X.
02      HIRED           PIC 9(8).
02      DEPENDENTS      PIC 99.
02      BASE-SALARY     PIC 9(5)V99.

RECORD NAME IS CLASSIFICATION-RECORD
        LOCATION MODE IS CALC USING CLASSIFICATION
        DUPLICATES ARE NOT ALLOWED
        WITHIN PERSONNEL-AREA.

02      CLASSIFICATION  PIC X(15).
02      LEVEL           PIC 99.
```

Figure E-2 Schema/Sub-schema Examples

```
RECORD NAME IS PROJECT-RECORD
        LOCATION MODE CALC USING PROJECT-ID
        DUPLICATES ARE NOT ALLOWED
        WITHIN RESEARCH-AREA.

Ø2      PROJECT-ID       PIC 99V9999.
Ø2      PROJECT-TITLE    PIC X(3Ø).

RECORD NAME IS CONTRACT-RECORD
        LOCATION MODE VIA FUNDS-SET
        WITHIN RESEARCH-AREA.

Ø2      CONTRACT-NUMBER PIC 9(7).
Ø2      CONTRACT-DATE    PIC 9(1Ø).
Ø2      COMPLETION-DATE PIC 9(1Ø).
Ø2      INITIAL-AMOUNT   PIC 9(7)V99.
Ø2      BALANCE          PIC 9(7)V99.

RECORD NAME IS SOURCE-RECORD
        LOCATION DIRECT SRCKEY
        WITHIN RESEARCH-AREA.

Ø2      SOURCE-NAME      PIC X(25).
Ø2      SOURCE-ADDRESS   PIC X(4Ø).
Ø2      SOURCE-PHONE     PIC 9(1Ø).
Ø2      SOURCE-CONTACT   PIC X(25).
Ø2      SOURCE-TITLE     PIC X(1Ø).

SET NAME IS DEPT-SET
        MODE IS CHAIN LINKED TO PRIOR
        ODER IS SORTED DUPLICATES NOT ALLOWED
        OWNER IS DEPARTMENT-RECORD
        MEMBER IS EMPLOYEE-RECORD OPTIONAL MANUAL LINKED TO OWNER
                ASCENDING KEY IS SS-NUMBER.
        SELECTON CURRENT.

SET NAME IS CLASS-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS SORTED
        OWNER IS CLASSIFICATION-RECORD
        MEMBER IS EMPLOYEE-RECORD OPTIONAL MANUAL LINKED TO OWNER
                ASCENDING KEY IS BASE-SALARY
        SELECTION CURRENT.

SET NAME IS PROJECT-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS ALWAYS LAST
        OWNER IS PROJECT-RECORD
        MEMBER IS EMPLOYEE-RECORD OPTIONAL MANUAL LINKED TO OWNER
        SET SELECTION IS CURRENT.
```

Figure E-2 (cont) Schema/Sub-schema Example

```
SET NAME IS PROJLDR-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS ALWAYS NEXT
        OWNER IS EMPLOYEE-RECORD
        MEMBER IS PROJECT-RECORD OPTIONAL MANUAL LINKED TO OWNER
        SET SELECTION CURRENT.

SET NAME IS DEPT-PROJ-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS SORTED
        OWNER IS DEPARTMENT-RECORD
        MEMBER IS PROJECT-RECORD MAND AUTO LINKED TO OWNER
                ASCENDING KEY IS PROJECT-ID
        SELECTION CURRENT.

SET NAME IS SPECIAL-PROJ-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS SORTED
        OWNER IS PROJECT-RECORD
        MEMBER IS DEPARTMENT-RECORD OPTIONAL MANUAL LINKED TO OWNER
        SET SELECTION IS CURRENT.

SET NAME IS FUNDS-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS ALWAYS LAST
        OWNER IS PROJECT-RECORD
        MEMBER IS CONTRACT-RECORD MAND AUTO LINKED TO OWNER
        SELECTION IS CURRENT.

SET NAME IS SOURCE-SET
        MODE IS CHAIN LINKED TO PRIOR
        ORDER IS ALWAYS LAST
        OWNER IS SOURCE-RECORD
        MEMBER IS CONTRACT-RECORD MAND AUTO LINKED TO OWNER
        SELECTION CURRENT.

SUB-SCHEMA NAME IS FUNDING
        PRIVACY LOCK IS DNUF.
AREA SECTION.
        COPY RESEARCH-AREA.
RECORD SECTION.
Ø1      SOURCE-RECORD.
Ø1      CONTRACT-RECORD.
Ø1      PROJECT-RECORD.
SET SECTION.
        COPY SOURCE-SET, FUNDS-SET.
```

Figure E-2 (cont) Schema/Sub-schema Example

```
SUB-SCHEMA IS DEPT-PROJ
        PRIVACY LOCK IS DEPROJ.
AREA SECTION.
        COPY TEMPORARY PERSONNEL-AREA, RESEARCH-AREA.
RECORD-SECTION.
Ø1      DEPARTMENT-RECORD.
Ø1      PROJECT-RECORD
Ø1      CONTRACT-RECORD.
SET SECTION.
        COPY DEPT-PROJ-SET, SPECIAL-PROJ-SET, FUNDS-SET.

SUB-SCHEMA NAME TOTAL PRIVACY ETHING.
AREA SECTION.
        COPY ALL AREAS.
RECORD SECTION.
Ø1      DEPARTMENT-RECORD.
Ø1      EMPLOYEE-RECORD.
Ø1      CLASSIFICATION-RECORD.
Ø1      PROJECT-RECORD.
Ø1      CONTRACT-RECORD.
Ø1      SOURCE-RECORD.
SET SECTION.
        COPY ALL SETS.

        END-SCHEMA.
```

Figure E-2 (cont) Schema/Sub-schema Example

INDEX

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Repcrt (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

____
____
____
____
____
____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

____
____
____
____
____
____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

____
____
____
____
____
____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______________________________ Date ____________________

Organization ____

Street ____

City ____________________ State ______________ Zip Code or Country ______________

If you do not require a written reply, please check here. ☐

-- Fold Here --

-- Do Not Tear - Fold Here and Staple --

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts, Telephone: (617) 897-5111 • ARIZONA, Phoenix • CALIFORNIA, Sunnyvale, Santa Ana, Los Angeles, Oakland, San Diego and San Francisco (Mountain View) • COLORADO, Denver • CONNECTICUT, Meriden • DISTRICT OF COLUMBIA, Washington (Riverdale, Md.) • FLORIDA, Orlando • GEORGIA, Atlanta • ILLINOIS, Chicago • INDIANA, Indianapolis • LOUISIANA, New Orleans • MASSACHUSETTS, Cambridge and Waltham • MICHIGAN, Ann Arbor and Detroit (Southfield) • MINNESOTA, Minneapolis • MISSOURI, St. Louis • NEW JERSEY, Englewood, Metuchen, Parsippany and Princeton • NEW MEXICO, Albuquerque • NEW YORK, Centereach (L.I.), Manhattan, Syracuse and Rochester • NORTH CAROLINA, Durham/Chapel Hill • OHIO, Cleveland and Dayton • OKLAHOMA, Tulsa • OREGON, Portland • PENNSYLVANIA, Philadelphia and Pittsburgh • TENNESSEE, Knoxville • TEXAS, Dallas and Houston • UTAH, Salt Lake City • WASHINGTON, Seattle • WISCONSIN, Milwaukee • ARGENTINA, Buenos Aires • AUSTRALIA, Adelaide, Brisbane, Melbourne, Perth and Sydney • AUSTRIA, Vienna • BELGIUM, Brussels • BRAZIL, Rio de Janeiro, Sao Paulo and Porto Alegre • CANADA, Calgary, Alberta; Vancouver, British Columbia; Ottawa and Toronto, Ontario; and Montreal, Quebec • CHILE, Santiago • DENMARK, Copenhagen • FINLAND, Helsinki • FRANCE, Grenoble and Paris • GERMANY, Cologne, Hannover, Frankfurt, Munich and Stuttgart • INDIA, Bombay • ITALY, Milan • JAPAN, Tokyo • MEXICO, Mexico City • NETHERLANDS, The Hague • NEW ZEALAND, Auckland • NORWAY, Oslo • PHILIPPINES, Manila • PUERTO RICO, Miramar • SPAIN, Barcelona and Madrid • SWEDEN, Stockholm • SWITZERLAND, Geneva and Zurich • UNITED KINGDOM, Birmingham, Edinburgh, London, Manchester and Reading • VENEZUELA, Caracas