UNIVAC 1100 SERIES

DATA MANAGEMENT SYSTEM (DMS 1100)

SCHEMA DEFINITION

DATA ADMINISTRATOR REFERENCE

SPERRY UNIVAC

This document contains the latest information available at the time of publication. However, the Univac Division reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Univac Representative.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP–7907 Rev. 1

Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	Orig.
Acknowledgment	1	Orig.
Preface	1	Orig.
Contents	1 thru 5	Orig.
1	1 and 2	Orig.
2	1 thru 5	Orig.
3	1 thru 80	Orig.
4	1 thru 17	Orig.
5	1 thru 17	Orig.
6	1 thru 6	Orig.
7	1 thru 5	Orig.
Appendix A	1 and 2	Orig.
Appendix B	1 thru 4	Orig.
Appendix C	1 and 2	Orig.
Appendix D	1 thru 10	Orig.
Appendix E	1 thru 3	Orig.
Appendix F	1 thru 6	Orig.
Appendix G	1 and 2	Orig.
Appendix H	1 thru 3	Orig.
Appendix I	1 thru 13	Orig.
Appendix J	1	Orig.
User Comment Sheet		Orig.
Total	Covers and 188 pages	

Section	Page Number	Update Level

Section	Page Number	Update Level

ACKNOWLEDGMENT

The Univac Division wishes to acknowledge the efforts of the CODASYL Programming Language Committee (PLC) Data Base Task Group (DBTG). The DBTG produced two reports containing specifications for a standardized Data Management facility containing a DATA DESCRIPTION LANGUAGE and a DATA MANIPULATION LANGUAGE. The DBTG issued its first report in October 1969, the second, revised and expanded, in April 1971. Univac is a member of the DBTG and participated in the development of the Data Management specifications. Univac's implementation of the 1100 Series Data Management System (DMS 1100) is based upon the DBTG specifications.

PREFACE

This document is the Data Administrator Reference Manual for the UNIVAC 1100 Series Data Management System (DMS 1100) SCHEMA DEFINITION. It is one of a series of manuals covering DMS 1100. It is intended that this Data Administrator Reference Manual, because it contains the DATA DEFINITION LANGUAGE, should be a prerequisite to the use of other Reference Manuals in the series covering DMS 1100.

CONTENTS

FIGURES

1. INTRODUCTION

1.1. GENERAL

This document describes the Data Definition Language (DDL) for the UNIVAC 1100 Series Data Management System (DMS). Therefore this document is the basic information source for the Data Administrator(s) involved in data-base design.

These sections explain the following:

- Operation of the DDL Translator.
- Mechanics of the DDL.
- The syntax details and functional results.
- Specialized areas of database design such as Record Placement Strategy.

1.2. DDL TRANSLATOR

Database design is the first step in using the DMS 1100 system. The vehicle for articulating the database design, that is, describing it in a form useful to DMS 1100 is the Data Definiton Language (DDL). The DDL definition of the database is called a "schema." It is important to realize that a schema is the definition of the database and not the database itself.

1.2.1. Schema Definitions

The schema, as written using the DDL, is known as a "source schema." To be useful to the Data Management Routine (DMR) and Data Manipulation Language (DML) preprocessor, it must first be processed by the DDL Translator. The DDL Translator is a processor which accepts source schema input and translates it into a series of interpretative tables known as an "object schema" for later use by other DMS 1100 programs. This object schema is treated as an absolute program element, named SCHEMA. The file in which the SCHEMA is placed is commonly referred to as the "schema file" and may also be used for storing data-base-procedures and other program elements.

A source schema is accessed by the DDL Translator in the same manner as a source program is accessed by any standard language processor; viz., the FORTRAN Processor, the COBOL Processor, etc. This means that all the facilities of ELT, ED, CULL, SIR, etc., are available to use with a DDL source schema. Changes and additions to a source schema are made in the same way for @FOR, @COB, etc.

Sample run streams which produce an object schema from a symbolic schema are available in Appendix G, SAMPLE DDL RUN STREAMS.

A complete list of Data Definition Language reserved words is available in Appendix A, DDL RESERVED WORD LIST.

1.2.2. DDL Translator Errors

During the processing of the source schema by the DDL Translator, all errors encountered are noted with the respective diagnostic (see Appendix D, DDL TRANSLATOR ERROR MESSAGES). If the errors encountered are "fatal," the object schema is not produced. If only non-fatal errors are encountered, the object schema is produced.

2. DDL SYNTAX COMPONENTS AND NOTATION

2.1. GENERAL

The DDL syntax consists of basic syntax clauses and combinations of these clauses. These combinations, through definition and convention, are used to communicate meaning or content.

(1) This section introduces the components of these clauses.
(2) It introduces the notation used to indicate allowable syntactical combinations of these clauses.

The clauses themselves are introduced in Section 3.

2.2. DDL SYNTAX COMPONENTS

This section describes the basic component of a *clause* (the character set) and continues the description through the *clause* to the complex structure (the source schema).

2.2.1. Character Set

The DDL character set contains 44 characters that consist of letters, digits, symbols, punctuation marks, and the space. The following list itemizes the 44-character set according to its subsets:

0, 1,...,9	digits
A, B,...,Z	letters
b	space
+	plus sign
–	minus sign or hyphen
,	comma
;	semicolon
.	period
(	left parenthesis
)	right parenthesis

The symbol b is sometimes used in discussions to emphasize that one or more contiguous spaces occur. A set of single quotes, for example 'A', is used in discussions to emphasize a character as opposed to the specific letter or symbol used; for example 'A' is to be read as the DDL character A, not the letter A. These two symbols, b and ' must not appear in DDL source language statements.

2.2.2. Word

A DDL word is a contiguous (no embedded spaces) sequence of not more than 30 characters. Each character in a word must be from the set 'A', 'B', ..., 'Z', '0', ..., '9', and '–', and neither the first nor last character in a word can be '–'. The following are words: 'HERE-AFTER', '22', 'A9C8F7J', and 'A'. The following are not words: 'A,BC', '-AT-THE', '␣', and 'ABC␣'.

2.2.3. Name

A DDL name is a word that names or identifies an entity in a source schema. The use of a name and its meaning is a function of its syntactical position. DDL has seven types of names which are as follows:

(1) schema-name
(2) record-name
(3) area-name
(4) set-name
(5) data-base-procedure
(6) data-base-data-name
(7) data-base-identifier

Certain types of names are restricted to fewer than 30 characters and not all types of characters are allowed. These restrictions are pointed out as a name type is used.

2.2.4. Reserved Word

Any DDL word whose syntactical usage is restricted and cannot be supplied by the Data Administrator is a DDL reserved word. A complete list of DDL reserved words is given in Appendix A.

2.2.5. Literal

A DDL literal is a word, supplied by the Data Administrator whose purpose is to convey explicit information, as opposed to identification information such as names. Thus, a literal is a string of characters whose value is implied by the ordered set of characters, of which the literal is composed. DDL literals are used to specify numeric values in DDL syntax.

Example:

'2' represents the value two.

2.2.6. Clause

A DDL clause is an ordered sequence of words and characters; the last character is an optional punctuation mark, semicolon or period. Clauses are the basic units of DDL syntax that communicate functional pieces of information. Clauses may contain subordinate clauses.

2.2.7. Subentry

One or more clauses form a DDL subentry when placed between period punctuation marks.

2.2.8. Entry

A DDL entry consists of one or more clauses or subentries and constitutes a syntactically complete description of:

- An area,
- A record,
- A set, or
- A schema.

2.2.9. Section

A DDL section is the collection of all entries in the schema pertaining to areas, or records, or sets. Therefore, there are three sections in a DDL source schema:

- The area section,
- Record section, and
- Set section.

2.2.10. Division

A DDL division either identifies the source schema (identification division) or describes the storage structure and characteristics of the database (data division). The data division is comprised of:

- The area,
- Record, and
- Set sections.

2.2.11. Source Schema

The DDL source schema is a syntactically complete definition of the database. It comprises the identification division and data division. The data division in turn comprises the area, record, and set sections. Each section consists of entries. In some instances entries contain subentries. Both entries and subentries are made up of clauses. Clauses or combinations of clauses are the basic units of syntax. They include reserved words, literals, names, and words, all falling within the available character set.

2.3. DDL SYNTAX NOTATION

This section describes the notational symbols and rules that are used to guide the construction of syntactically correct DDL clauses. These conventions apply to all DDL syntax skeletons specified in Section 3.

- The elements which make up a syntax skeleton are upper case words, lower case words, characters (spaces included), notational symbols and discussion comments. Their skeleton order, left to right, and next line, must be stated in DDL source language.

- All upper case words represent DDL *reserved words.* Underlined upper case words are *required* in each specification of the syntax component. *Non-underlined* upper case words are individually optional. Each may be included or omitted in source language specifications.

- Lower case words *(in italics)* are generic terms, names, or literal types, that must be replaced in the actual source specification with a name or literal. Generic terms *always* have integers appended; this avoids confusion in case two generic terms in one skeleton are of the same type; for example, appearance of *data-base-data-name—1* and *data-base-data-name—2* in one skeleton means two *data-base-data-names* must be supplied.

- The use of semicolons and periods in a skeleton is optional in source specifications. When they are used they must be positioned as in the skeleton. The commas and spaces in a skeleton must be used as indicated by a skeleton.

- When one or more syntax components are enclosed by special notational symbols, they should be interpreted as follows:

  [a / b / c] A source specification does not require a syntax in this position, but can include either a, b, or c.

  { a / b / c } A source specification must include exactly one of the components a, b, or c.

  ‖ a / b / c ‖ A source specification must include one of a, b, or c, but can include up to one of each.

  ... Repeated choices may be made and included in a source specification, one after the other.

- When prose surrounded by double quotes (" ") is in a skeleton, the prose contains comments to clarify the skeleton; this quotation can never appear in a source schema.

- All underlined (**LARGE BOLD FACE**) upper case words must begin in a new card, with their first character between column 8 and 11 inclusive. All other words must begin and end between card columns 12 and 72. More than one card can be used to specify the words between two (**LARGE BOLD FACE**) underlined words. These rules must be followed and the indicated word order preserved.

The following syntax skeleton is presented to illustrate notation only. Examples of possible specifications are as follows.

- Syntax Skeleton:

NAME IS { STANDARD / *data-name—1* } [,*data-name—2*] ... ;

HOWEVER ALWAYS ‖ ZIG / ZAG ‖ .

- Syntax Examples:

(a)
```
NAME IS STANDARD, DN-A, DN-B; HOWEVER
        ALWAYS ZIG ZAG
```

(b)
```
NAME DN-C HOWEVER ZIG
```

(c)
```
NAME IS DN-D, DN-E, DN-F;
        HOWEVER ALWAYS ZAG.
```

(d)
```
NAME STANDARD
    ; HOWEVER ZIG ZAG
```

(e)
```
NAME IS
    STANDARD
        DN-G
            DN-1
    HOWEVER     ALWAYS
        ZIG.
```

3. DATA DEFINITION LANGUAGE

3.1. GENERAL

This section discusses the DDL syntax and the rules governing its use. It divides the DDL syntax into four logical categories and discusses each in turn. The four categories are as follows:

- Division and Section Clause Syntax
- Area Entry Syntax
- Record Entry Syntax
- Set Entry Syntax

Each category is first discussed in an introductory manner that is intended to provide a conceptual understanding of what follows. A complete syntax skeleton is also included. Next, the specific DDL syntax clauses in each category are thoroughly discussed on an individual basis.

The complete discussion of each clause is presented under six headings. These headings and their paragraph content are outlined as follows.

Function:

– A brief definition of the functional purpose of the clause.

Syntax Skeleton:

– A notational view of the allowable source syntax combinations which may be constructed from the clause.
– The Syntax Skeleton notation is explained in 2.3.
– Violation of the Syntax Skeleton results in a DDL Translator error message.

Syntactical Constraints:

– The rules which must be followed to produce source clauses from the Syntax Skeleton.
– Rules generally reflect options and restrictions not inherent in the Syntax Skeleton representation.
– Violation of a Syntactical Constraint normally results in a DDL Translator fatal or non-fatal error message.
– A fatal error message prevents the generation of an object schema, but not the completion of the source schema scan.

Data Definition Guides:

- The detailed guidelines for the definition of the database.
- They generally reflect the philosophy and capability of the DMR.
- Violation of the Data Definition Guides normally results in a DMR error return during the run time execution of DML commands.

Data Manipulation Guides:

- The DDL syntax has certain data manipulation ramifications.
- A complete discussion is given under this heading.

Examples and Discussion:

- Examples of clauses and the resulting storage structure are given under his heading.
- Also included is any pertinent information which may aid in database design.

3.2. DIVISION AND SECTION SYNTAX

A source schema is written in two divisions and three sections with area, record, and set entries in the respective three sections. The clauses which comprise the area, record, and set entries are discussed in 3.3, 3.4, and 3.5, respectively.

This section discusses the clauses which comprise the IDENTIFICATION DIVISION, and those clauses of the DATA DIVISION which comprise the framework within which the area, record, and set entries are placed.

The first clause in any source schema is the IDENTIFICATION DIVISION clause. This must be followed by the SCHEMA NAME clause which performs the actual identification function.

The DATA DIVISON clause follows all clauses in the IDENTIFICATION DIVISION (i.e., the SCHEMA NAME clause). The DATA DIVISION clause indicates that the definition of the database storage structure is about to begin. Following the DATA DIVISION clause must be the AREA SECTION clause.

The AREA SECTION contains all area entries defined in the source schema. This section must be present since there must be at least one area for a database. Prior to the first area entry it is possible to control the DMS 1100 internal structure of the database-keys used with the schema. This is accomplished through the optional AREA CONTROL clause. The area entries follow either the AREA SECTION clause or, optionally, the RECOVERY-POINTS clause. After the area entries have been entered, the RECORD SECTION clause must be used to signal the start of the record entries.

The RECORD SECTION clause and at least one record entry must be present. Following the last record entry, the SET SECTION clause is used to signal the start of the set entries. However, a database may be defined without the use of sets; thus, the SET SECTION clause is optional and used *only* when set entries are present. The complete syntax skeleton, minus the area, record, and set entries, is presented in the following section. The individual clauses are then completely discussed.

3.2.1. Complete Syntax Skeleton

Function:

To identify the schema and provide the framework for area, record, and set entries.

Syntax Skeleton:

```
IDENTIFICATION DIVISION.
SCHEMA NAME IS schema-name
[ ;TIP FILE CODE IS integer–0] .
DATA DIVISION.
AREA SECTION.
[AREA CONTROL IS integer–1 AREAS]
[AREA LOOKS INCLUDE { QUICK-BEFORE-LOOKS | BEFORE-LOOKS | AFTER-LOOKS | NO-LOOKS } ]
[RECOVERY-POINTS ARE EVERY integer–2 BLOCKS.]
    { "Area entries follow here" } ...
RECORD SECTION
    { "Record entries follow here" } ...
[ SET SECTION
    { "Set entries follow here" } ... ]
```

Syntactical Constraints:

The clause ordering previously shown must be maintained.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion:

The following listing is an example of a schema, which shows the proper overall layout of a source schema.

```
IDENTIFICATION DIVISION

SCHEMA NAME IS EXAMPLE
    TIP FILE CODE IS 127

DATA DIVISION

AREA SECTION

AREA CONTROL IS 150 AREAS

AREA LOOKS INCLUDE QUICK-BEFORE-LOOKS, BEFORE-LOOKS,
    AFTER-LOOKS

AREA NAME IS FIRST-AREA AREA CODE IS 1
    ALLOCATE 500 PAGES 2 OVERFLOW PAGES EVERY 8 DATA PAGES
    PAGES ARE 448 WORDS
    LOAD IS 50 PERCENT
    CALC USES 2 CHAINS

AREA NAME IS SECOND-AREA AREA CODE IS 2
              :

RECORD SECTION

RECORD NAME IS FIRST-RECORD RECORD CODE IS 1
    LOCATION MODE IS VIA FIRST-SET SET
    WITHIN FIRST-AREA
    02 CITY-NAME PIC IS X(12) USAGE IS DISPLAY
    02 POPULATION PIC IS 9(10)
    02 OFFICE
        03 STREET PIC IS X(12) USAGE IS DISPLAY
        03 NUMBER PIC IS X(4) USAGE IS DISPLAY

RECORD NAME IS SECOND-RECORD RECORD CODE IS 2
              :

SET SECTION

SET NAME IS FIRST-SET SET CODE IS 1
    MODE IS CHAIN
    ORDER IS SORTED
    OWNER IS SOME-RECORD
    MEMBER IS FIRST-RECORD AUTOMATIC
    ASCENDING KEY IS CITY-NAME DUPLICATES ARE NOT ALLOWED
    SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS SECOND-SET SET CODE IS 2
              :
```

3.2.2. IDENTIFICATION DIVISION

Function:

The IDENTIFICATION DIVISION of a source schema provides the means for schema identification.

Syntax Skeleton:

IDENTIFICATION DIVISION

Syntactical Constraints:

The IDENTIFICATION DIVISION statement must be the first statement in the source schema.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.2.3. SCHEMA NAME

Function:

The SCHEMA clause identifies the schema by name to the DMS.

Syntax Skeleton:

SCHEMA NAME IS *schema-name–1*

Syntactical Constraints:

- The SCHEMA clause must immediately follow the IDENTIFICATION DIVISION clause.
- Schema-name–1 must uniquely identify the source schema.
- Schema-name–1 may not exceed 12 characters.

Data Definition Guides:

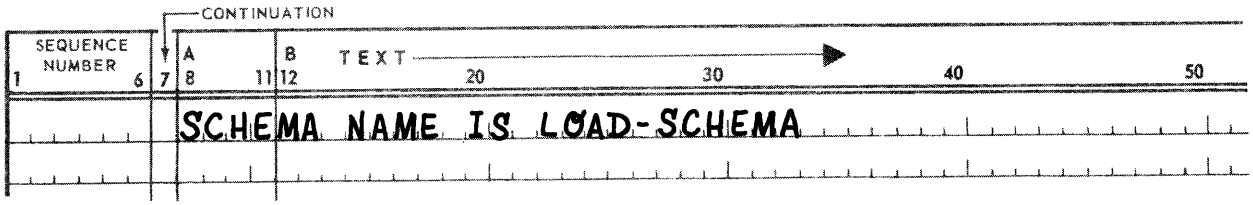
- Prior to the @DDL control card, either an @ASG or a @USE control card must name an Executive file into which the object schema is to be placed.
- Only one object schema may be placed in the file or an automatic deletion of the original object schema will result. The object schema is a non-executable absolute element.

Data Manipulation Guides:

- The INVOKE clause of the SCHEMA SECTION of a DML program must specify "schema-name—1" if the run unit is to manipulate the database described by "schema-name—1". To ensure that the DMLP may access the schema it is necessary to place the object schema, "schema-name—1", in an Executive file named "schema-name—1". This may be accomplished by the appropriate @ASG or @USE control card for "schema-name—1", which must be included in the run stream prior to the @DMLP control card.

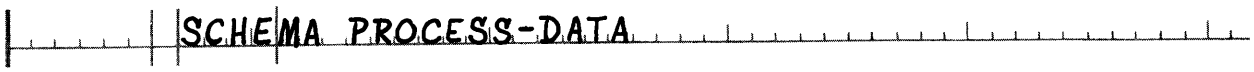
Examples and Discussion:

- **Example 1:**



When a Data Administrator is using different descriptions of the same database for different purposes, the name of a schema may reflect its purpose (see Section 7, DATABASE LOADING). This name may reflect the description of the schema invoked by the run units intending to load the database.

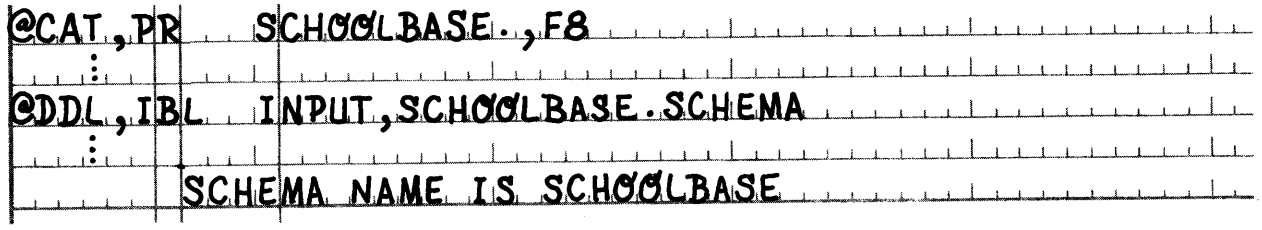
- **Example 2:**



This name may reflect the description of the schema invoked by run units intending to manipulate the database.

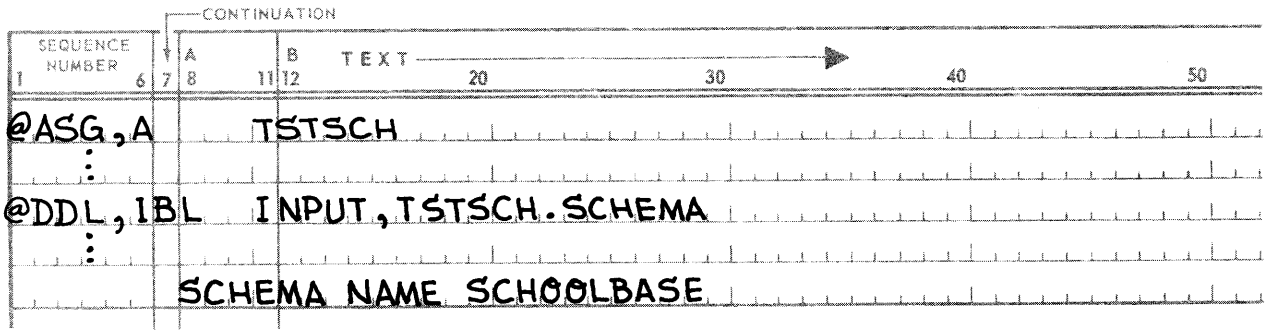
Since it is illegal to use more than one schema simultaneously to control a database, use of a schema of this type assumes prior usage of a schema type as in example 1.

- **Example 3:**



The object schema, whose name is SCHOOLBASE, is placed in the Executive file SCHOOLBASE (named in Field 2 of the DDL call card) as the element SCHEMA.

- **Example 4:**



The object schema, whose name is SCHOOLBASE, is placed in the Executive file TSTSCH (named in Field 2 of the DDL call card) as the element SCHEMA.

3.2.4. TIP FILE CODE

Function:

To specify a numeric value for the TIP file which will contain the absolute SCHEMA.

Syntax Skeleton:

TIP **FILE CODE IS** *integer–0*

Syntactical Constraints:

- The TIP FILE CODE clause is optional; if present, it must immediately follow the SCHEMA NAME clause.
- Integer–0 must be an integer between 1 and 4000, inclusive.

Data Definition Guides:

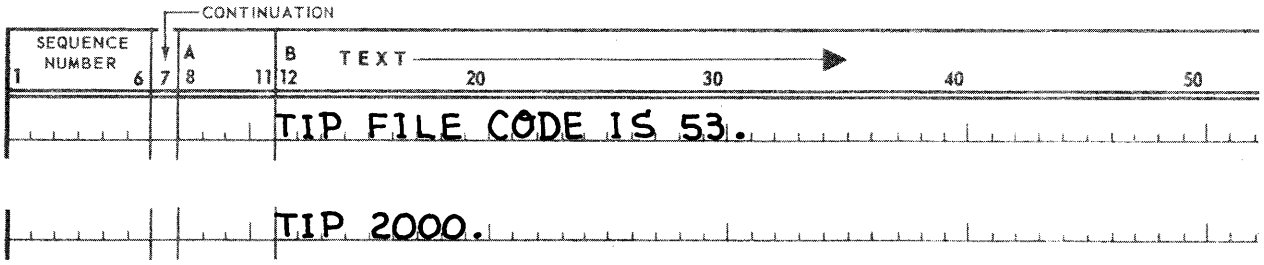
- Integer–0 is used to identify the numeric value of the TIP (Transaction Interface Package) file in which the DMS utility routine, TIPCOPY, will place the absolute schema.
- The TIP file code integer–0 is assigned through a system's generation of TIP. (Refer to the DMS 1100 Release Bulletin for further details.)
- If the TIP FILE CODE clause is specified, and QUICK-LOOK files are to be used, then QUICK-LOOK files must be assigned as TIP files during the system's generation of TIP.

Data Manipulation Guides:

- No @ASG statement is needed for the TIP file containing the absolute SCHEMA or for the TIP QUICK-LOOK files at the time of DML program execution.

- If the optional clause is *not* specified then an @ASG statement is needed for the Executive file named "schema-name–1" prior to DML program execution.

Examples:



3.2.5. DATA DIVISION

Function:

The DATA DIVISION clause serves to logically separate those clauses which identify the database from those which define it.

Syntax Skeleton:

DATA DIVISION

Syntactical Constraints:

The DATA DIVISION clause must be present, must follow the SCHEMA clause and precede the AREA SECTION clause.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.2.6. AREA SECTION

Function:

The AREA SECTION clause signals the start of area entries.

Syntax Skeleton:

AREA SECTION

Syntactical Constraints:

The AREA SECTION clause must be present and must follow the DATA DIVISION clause. It must precede the AREA CONTROL, AREA LOOKS, and/or RECOVERY-POINTS clauses, if specified, or the first area entry.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.2.7. AREA CONTROL

Function:

The AREA CONTROL clause specifies the maximum number of areas in the database.

Syntax Skeleton:

AREA CONTROL **IS** *integer–1* **AREAS**

Syntactical Constraints:

- The AREA CONTROL clause is optional; if used, it must be the first clause following the AREA SECTION clause and must precede all area entries.
- If the AREA CONTROL clause is present, integer–1 must be a value between 127 and 4000, inclusive.

Integer–1	Number of Area Code Bits
1– 127*	7
128– 255	8
256– 511	9
512–1023	10
1024–2047	11
2048–n**	12

- If the AREA CONTROL clause is not present the assumed value is 4000.

Data Definition Guides:

- The AREA CONTROL clause allows subsequent area entries to be added to the schema without reorganizing the database. The total number of area entries must not exceed the bit limit determined by integer–1.
- The AREA CONTROL clause may involve a Data Administrator trade-off between an expansion factor for areas and the number of pages allowed per area and records per page. See Appendix E, DATABASE-KEY FORMAT, for details.

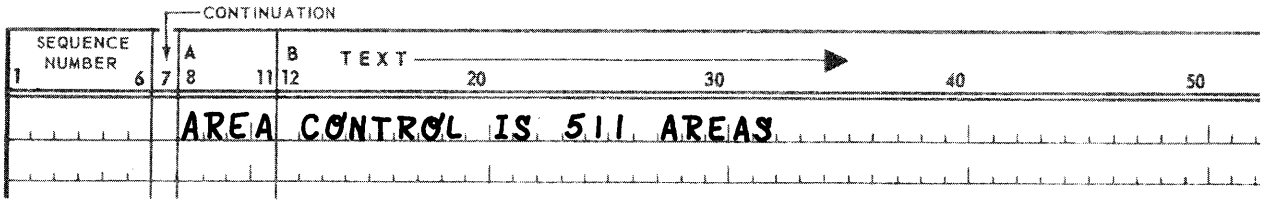
**If integer–1 is less than 127, the DDL translator will flag the integer in error and assign the 7–bit minimum to the area code field of the database key.*

***If integer–1 is greater than 4000, the DDL translator will flag the integer in error and assign the 12–bit maximum to the area code field of the database key.*

Data Manipulation Guides: None

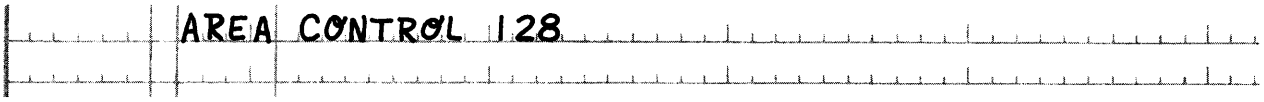
Examples and Discussion:

- **Example 1:**



The DDL translator has assigned a nine-bit maximum to the area code part of the database-key word. Effectively, the Data Administrator has allowed himself three additional bits to be distributed between the number of pages allowed per area and the number of records per page.

- **Example 2:**



Since the DDL translator has assigned an eight-bit maximum to the area code part of the database-key word, the Data Administrator has limited the number of areas which could be specified in his database without reorganization to 255.

3.2.8. AREA LOOKS

Function:

To specify the types of looks to be taken as a default option for those AREA descriptions without individual LOOKS clauses.

Syntax Skeleton:

AREA LOOKS **INCLUDE** { [QUICK-BEFORE-LOOKS] [BEFORE-LOOKS] [AFTER-LOOKS] [NO-LOOKS] }

Syntactical Constraints:

- The QUICK-BEFORE-LOOKS, BEFORE-LOOKS and AFTER-LOOKS phrases can be specified in order.
- See 3.2.1 for the position of this clause.

Data Definition Guides:

- This clause applies to all AREA descriptions without LOOKS clauses.
- This clause is overridden for a particular AREA, by the presence of a LOOKS clause.

- If this optional clause is not present in a Schema, the default value is NO-LOOKS.
- The AREA LOOKS clause specifications must be made in conformance with the system generation parameter settings, indicating whether or not audit trails and/or a quick-look file is present.

Data Manipulation Guide:

- The SAVE DATA clause available in the DML supplies a choice of either command or run unit quick-before-looks to be taken. The SAVE DATA clause is only honored by the DMR if the Schema specifies quick-before-looks for the AREA accessed by the DML program.
- If quick-before-looks are specified in the Schema, but the run unit does not include a SAVE DATA clause, then the run unit quick-before-looks are taken as the default case.

Examples and Discussion: None

3.2.9. RECOVERY-POINT

Function:

To optionally specify the frequency with which recovery points are to be placed on the audit trail tape.

Syntax Skeleton:

RECOVERY-POINTS ARE EVERY *integer–2* **BLOCKS**

Syntactical Constraints:

- See 3.2.1 for the position of this clause.
- If the optional RECOVERY-POINTS clause is present, integer–2 must be a positive integer.

Data Definition Guides:

- The RECOVERY POINTS clause must be established in conjunction with the appropriate systems generation parameters. Both audit trail tape and AUTORP (Automatic Recovery Point) must be set to 1 in DMSSGP in order that the automatic recovery points may be taken by the DMR.

NOTE:

These are specified in the Release Bulletins accompanying the current release of DMS 1100.

Examples and Discussion: None

3.2.10. RECORD SECTION

Function:

The RECORD SECTION clause signals the end of the area entries and beginning of the record entries.

Syntax Skeleton:

RECORD SECTION

Syntactical Constraints:

The RECORD SECTION clause is required and must follow the last area entry and precede the first record entry.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.2.11. SET SECTION

Function:

The SET SECTION clause signals the end of the record entries and the beginning of the set entries.

Syntax Skeleton:

SET SECTION

Syntactical Constraints:

- The SET SECTION clause is optional. If must not be utilized if set entries are not present.
- The SET SECTION clause must be present if there are set entries and must follow the last record entry and precede the first set entry.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.3. AREA ENTRY

An area is a named subdivision of a database. It contains a collection of records. The records can be related in sets wholly contained in one area, or spread across several areas. An area in DMS 1100 has an immediate mass storage connotation by virtue of its one-to-one relationship to an Executive or TIP mass storage file. In Executive terminology, a database is a collection of files. This relating of areas to files is a Data Administrator responsibility accomplished by assigning an Executive file which has the same name (or equated name) as the corresponding area name, or a TIP file which has the same code as the corresponding area code.

This section provides explicit rules for the definition of an area. Each area to be defined must first be identified through the AREA NAME clause. For reasons of data independence, the ability to add areas without impacting programs already written, the area must be secondarily identified by an AREA CODE. An AREA CODE is a one- to four-digit numeric identifier. The amount of mass storage space available in the area depends upon the @ASG specifications for the corresponding file. Specification of the optional TIP clauses enables the use of the

TIP (Transaction Interface Processor) file handler, rather than the standard Executive file handler. (Refer to the DMS 1100 Release Bulletin for further details.)

It is advantageous to group records for I/O purposes. This is accomplished by subdividing the area into pages through the ALLOCATE clause. Each page becomes a unit for I/O and may contain from zero to "n" records. The number of records per page depends upon the page size, which is defined by the PAGES clause. Determination of page size and the grouping of records upon a page for I/O purposes is the subject of Section 6.

An area may be defined to have PRE-INITIALIZED pages; if the pages are defined to be PRE-INITIALIZED, the DMR performs no page initialization; a DMS utility routine may be used to initialize pages. An area whose pages are PRE-INITIALIZED may be opened for initial load. If an existing area whose pages are defined PRE-INITIALIZED is opened for initial load, no existing data is destroyed by page initialization.

Areas are rarely completely static; existing records are often modified or deleted, and new records are often added. To make provisions for these events space may be retained, either in the page or in other pages, for subsequent additions. Thus OVERFLOW pages may be specified as part of the ALLOCATE PAGES clause. OVERFLOW pages may occur following the last data page and/or may be interspersed EVERY so many DATA PAGES. Another means of providing for the addition of records which will keep records more compactly grouped, and thus decrease the amount of I/O, is to use the LOAD clause to limit the number of words in the page in which records are initially placed. The remaining words are filled with records as additions are made over the life of the area. Depending upon the number of records to be added after the initial loading of the area, the LOAD and OVERFLOW clauses may be used in conjunction:

(1) To provide unused space in the page for adding future records, and
(2) To provide additional unused pages for overflow of data pages.

An area may be defined to be EXPANDABLE to a page number greater than the number of allocated pages. When an area is defined to be EXPANDABLE, the area may be procedurally expanded in an operational database without reloading the area.

A calc-chain is a circularly linked group of records. (See 5.3.)

One calc-chain header word is automatically reserved on every data page in the database. The final clause pertaining to areas, the CALC clause, allows reserving "n" total chain header words to identify "n"–1 additional calc-chains on the page. The chain header words reserved through the use of the CALC clause will not be used for any other purpose by the DMR.

The following subsections present the detailed syntax and rules for an area entry. The first subsection presents the complete area syntax skeleton and those rules which govern the area entry as a whole. The remaining subsections respectively present the syntax skeleton and governing rules for the individual area entry clauses. These are presented in the order they appear in the complete area entry skeleton.

3.3.1. Complete Area Entry

Function:

To identify and provide the characteristics of an area within a database.

Syntax Skeleton:

```
AREA NAME IS area-name–1;
    AREA CODE IS integer–1;
    [AREA MAPS TO TIP FILE;]
    ALLOCATE integer–2 [PRE-INITIALIZED] PAGES
    [,integer–10 OVERFLOW PAGES AT END]
    [,integer–3 OVERFLOW PAGES EVERY integer–4 DATA PAGES]
    [,EXPANDABLE TO integer–11 PAGES] ;
    PAGES ARE integer–5 WORDS
    [;LOOKS INCLUDE { QUICK-BEFORE-LOOKS | BEFORE-LOOKS | AFTER-LOOKS | NO-LOOKS }]
    [;LOAD IS { integer–6 WORDS | integer–7 PERCENT }]
    [;CALC USES integer–8 CHAINS [LINKED PRIOR]] .
```

Syntactical Constraints:

- At least one area entry must be defined in the schema. Multiple area entries may be defined up to the limit specified on the AREA CONTROL clause. If only one area entry is specified, then that area contains the entire database.
- All area entries must immediately follow the AREA SECTION (and AREA CONTROL, AREA LOOKS, and/or RECOVERY-POINTS) clause, and must precede the RECORD SECTION clause.
- Area entries may be placed in any order relative to other area entries.
- The clauses specified in an area entry must be ordered as shown in the above skeleton.
- The semicolon and period puncutation is optional, but when used it must separate clauses and end the entry, respectively.

Data Definition Guides:

- All area clauses apply only to the respective area entry. However, the AREA NAME, AREA CODE and AREA TIP clauses have external (i.e., Executive) significance. Each AREA NAME and AREA CODE must be unique within the schema.
- Each area entry defines only one occurence of an area. A separate area entry must be written for each desired area.

Data Manipulation Guides:

- Before an executing run unit may OPEN an area defined by an area entry, the area must be made available to the run unit through the use of an appropriate @ASG, or @USE Control Card unless the area maps to a TIP file.

- For a run unit to access the area, it must INVOKE a schema which contains a definition of that area.

- The first DML command executed against a given area by a run unit must always be an OPEN. The OPEN has three general forms.

  (a) The OPEN for INITIAL LOAD results in the application of the appropriate LOAD factor and permits the initial storage of records, with limited use of OVERFLOW.

  (b) The OPEN for UPDATE causes the appropriate LOAD factor to be ignored and allows full use of Overflow. This OPEN may only be accomplished after the area has been initialized.

  (c) The OPEN for RETRIEVAL permits the use of a limited set of DML commands. This OPEN may only be accomplished after the area has been initialized.

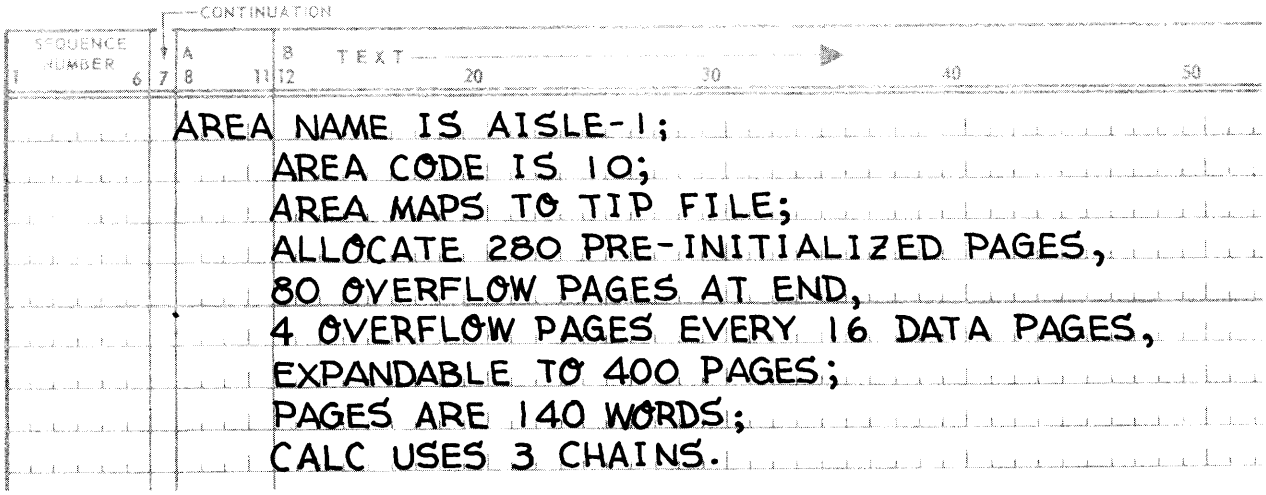
- When a run unit has finished accessing an area, it should CLOSE the area.

Examples and Discussion:

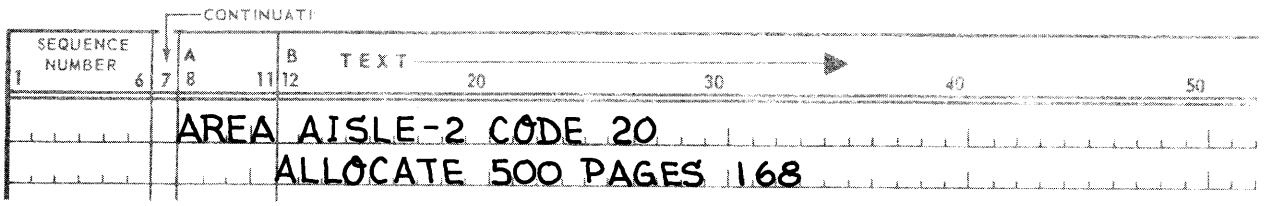
- **Example 1:**

  The Executive file required for the Area Entry need not be available (@ASGed) until it is to be OPENED by an executing run unit. The DDL Translator (and DML preprocessor) do not attempt to reference the file. If the area maps to a TIP file, no @ASG statement is necessary for the file during the time of DML program execution.

- **Example 2:**



- **Example 3:**



3.3.2. AREA NAME Clause

Function:

To name an area within the database.

Syntax Skeleton:

AREA NAME IS *area-name–1*

Syntactical Constraints:

- The name supplied for area-name–1 must be unique among all area-names within the source schema.
- Area-name–1 cannot exceed 12 characters.

Data Definition Guides: None

Data Manipulation Guides:

- The Applications Programmer must be aware of area-name–1 in four instances if he intends to:

  (1) OPEN/CLOSE area-name–1 explicitly, he must supply area-name–1 on the OPEN/CLOSE command.

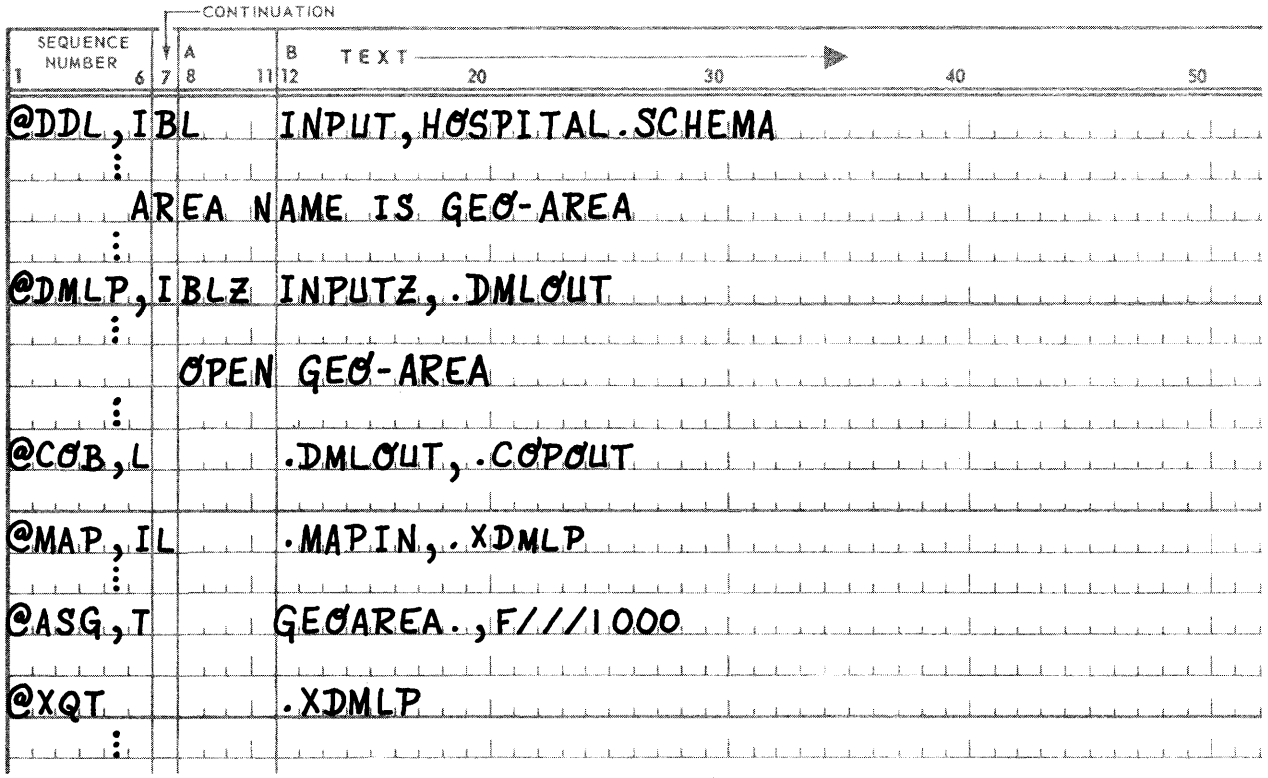
  (2) Perform an operation that directly or indirectly results in a FIND/FETCH or STORE of a record having a LOCATION MODE of DIRECT, he must know the area-name of the area in which the record was/is to be placed.

  (3) Perform an operation that directly or indirectly results in a FIND/FETCH or STORE of a record having a LOCATION MODE of CALC, he must know the area-name, unless the area-name is being supplied by the CALC procedure.

  (4) Use FIND/FETCH format 3 or 4 with the WITHIN area-name–n AREA option.

- When area-name–1 is used as a part of a database-key, it must be initialized in a data-base-data-name. The Data Management Communications Area (DMCA) reserved location AREA-NAME, or a 77-level data-base-data-name defined with a USAGE IS AREA-NAME must be used for this purpose. Such data-base-data-names must be defined in the WORKING or COMMON-STORAGE sections of the Data Manipulation Language (CML) source run unit.

Examples and Discussion:



3.3.3. AREA CODE Clause

Function:

To specify a numeric value for the area-name to be used by DMS 1100 as the internal name of the area, and/or to be used as the external file code for TIP file control.

Syntax Skeleton:

AREA CODE IS *integer–1*

Syntactical Constraints:

- Integer–1 must be greater than or equal to 1. It must also be less than or equal to the bit limit determined by integer–1 of the AREA CONTROL clause, or 4000, if that clause is omitted from the source schema.
- The value supplied for integer–1 must be unique among all area-codes within the source schema.

Data Definition Guides:

- The area-code assigned becomes the area's identification in an all DMS 1100 internally formatted database-keys, on pages, and in internal tables. The Data Administrator should think in terms of only area-names and not area-codes.

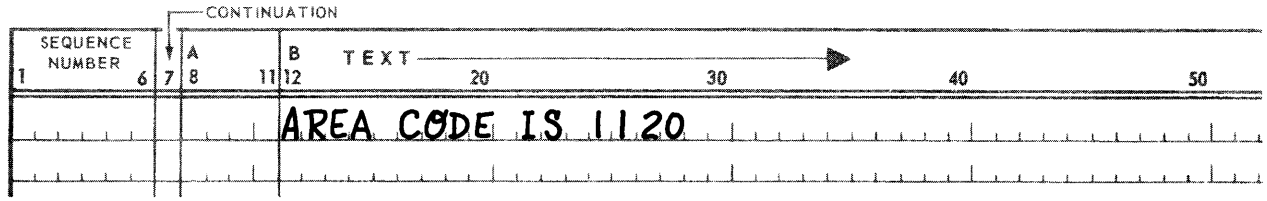
- An assigned area-code cannot be changed without performing a new initial load on the area and all areas containing records and database-keys which include the original area-code.

Data Manipulation Guides:

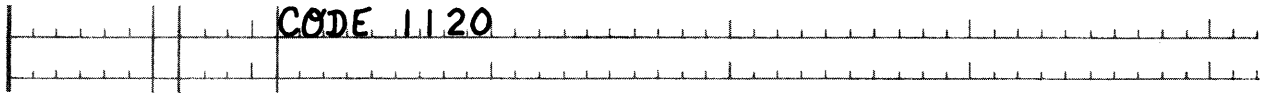
All source DML references to areas are based on the area-name. The existence of an area-code should not concern the Application Programmer. All conversions are strictly a DMS 1100 activity.

Examples and Discussion:

- **Example 1:**



- **Example 2:**



The code 1120 is valid if no AREA CONTROL clause is specified, or if integer–1 in the AREA CONTROL clause specified is ⩾ 1024. (See Syntactical Constraints, 3.2.7, second item, AREA CONTROL clause.)

3.3.4. AREA TIP Clause

Function:

To specify that an area is defined and accessed through the TIP (Transaction Interface Package) file control.

Syntax Skeleton:

AREA MAPS TO TIP FILE

Syntactical Constraints:

- The AREA TIP clause is optional; if not present, an Executive file is assumed.

Data Definition Guides:

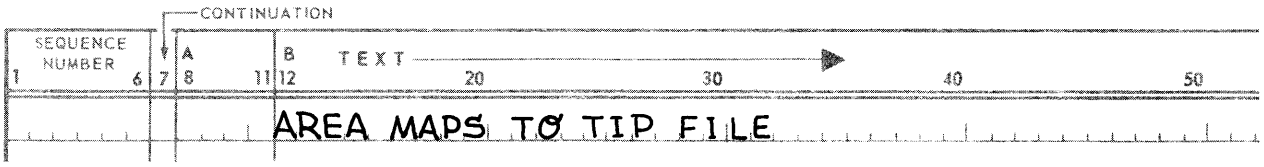
- The AREA TIP clause specifies that the TIP file handler is to be used to access pages within the area.
- The TIP file must be defined and registered with TIP during the system's generation of TIP. (Refer to the DMS 1100 Release Bulletin for further details.) The area code functions as the external file code for TIP file control.
- If the AREA TIP clause is not specified, pages in the area will be accessed by the Executive file handler.

Data Manipulation Guides:

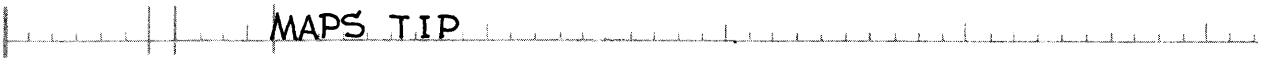
- If the AREA TIP clause is specified, no @ASG statement is needed for accessing the area during execution of a DML program.
- If the AREA TIP clause is not specified, an @ASG statement is needed for accessing the area during execution of a DML program.

Examples:

- **Example 1:**



- **Example 2:**



3.3.5. ALLOCATE Clause

Function:

- To divide an area into pages for efficient input/output handling.
- To specify if the pages in an area are pre-initialized.
- To define the number and sequence of data and overflow pages.
- To provide for future expansion of the area.

Syntax Skeleton:

ALLOCATE *integer–2* [PRE-INITIALIZED] **PAGES**
[,*integer–10* OVERFLOW **PAGES AT END**]
[,*integer–3* OVERFLOW **PAGES** EVERY *integer–4* **DATA PAGES**]
[,EXPANDABLE **TO** *integer–11* **PAGES**]

Syntactical Constraints:

- Integer–2 must be between 1 and 99999 inclusive.
- If integer–2 is specified as 1, no overflow or spaced data pages may be specified.

- Integer–10, integer–3, and integer–4 must each be greater than or equal to 1.

- Integer–11 must be greater than integer–2.

NOTE:

The presence of integer–11 does not affect the syntactical constraints on integers–10, –3, and –4.

- If only the optional integer–10 is specified, integer–10 must be less than integer–2.

- If only the optional integer–3 and integer–4 are specified, the following conditions must be met:

  (integer–3 plus integer–4) $\leqslant$ integer–2

  (n(integer–3 plus integer–4))/integer–2 = 1

- If integer–10, integer–3, and integer–4 are specified, the above conditions become:

  (integer–3 plus integer–4) $\leqslant$ (integer–2 minus integer–10)

  (n(integer–3 plus integer–4))/(integer–2 minus integer–10) = 1

- The following conditions must be met with respect to integer–2 of the INTERVAL subclause of the LOCATION MODE clause (see 3.4.4):

  (1) If only integer–2 of the ALLOCATE clause is specified,

  n(integer–2 of the INTERVAL clause) = (integer–2 of the ALLOCATE clause)

  (2) If only integer–2 and integer–10 of the ALLOCATE clause are specified,

  (integer–2 of the ALLOCATE clause minus integer–10 of the ALLOCATE clause) = n(integer–2 of the INTERVAL clause)

  (3) If integer–4 of the ALLOCATE clause is specified, the condition to be met reduces to

  (integer–4 of the ALLOCATE clause) = n(integer–2 of the INTERVAL clause)

- The following condition must be met with respect to integer–4, the upper page limit, of the WITHIN clause:

  (integer–4 of the WITHIN clause) $\leqslant$ (integer–2 of the ALLOCATE clause)

Data Definition Guides:

- See Section 7, DATABASE LOADING for a discussion of the implications of the optional PRE-INITIALIZED clause.

- Integer–2 specifies the number of pages into which the area is divided. If integer–10, integer–3 and integer–4, are not specified, all pages in the area are data pages.

- If integer–2 and integer–10 are specified, but not integer–3 and integer–4, the last integer–10 pages are overflow pages. These pages are termed global overflow pages. Integer–2 minus integer–10 data pages precede the overflow pages.

- If only integer–2, integer–3, and integer–4 are specified, pages in the area are divided into segments of integer–3 plus integer–4 pages each. The first integer–4 pages in each segment are data pages; the following integer–3 pages are overflow pages. When an area is segmented in this manner, a record to be placed within one segment of integer–4 consecutive data pages must find space within those data pages or must find space within the following integer–3 overflow pages. If space is not available the record is rejected.

- If integer–2, integer–10, integer–3, and integer–4 are specified (integer–2 minus integer–10), pages in the area are divided into segments of integer–3 plus integer–4 pages each. The first integer–4 pages in each segment are data pages; the following integer–3 pages are overflow pages. At the end of the group of segments are integer–10 overflow pages termed global overflow. If space cannot be found for a record within one segment of integer–4 consecutive data pages, nor can be found within the following integer–3 overflow pages, the record is placed in global overflow if space is available. If space is not available the record is rejected.

- All pages in an area are numbered sequentially by the DMR. This numbering disregards differences between data and overflow pages. The page numbers range from 1 to integer–2.

- Integer–2 is the determining factor in the amount of mass storage to be assigned to this area. This is discussed further under the PAGES clause.

- Page numbers are contained in the second field of internally formatted database-keys. Integer–2 is used to determine the number of bits to be assigned to this field. Because integer–2 will normally vary from area to area, the internal format of database-keys will also vary. Note that integer–2, in conjunction with the AREA CONTROL clause in effect determines the number of record numbers available to pages in this area. (See Appendix E to determine the number of record numbers available.)

- When and how overflow pages are used is discussed in detail in Section 5.

- Integer–11 specifies the number of pages to which the area is expandable. If integer–n is specified, integer–11, rather than integer–2, determines the number of bits to be assigned to the page field of the database-keys as described above for integer–2. An area may, if the EXPANDABLE clause is specified, be expanded at some future date. The area may be expanded procedurally in an operational database without reloading the area. The source schema must be changed and processed to create a new object schema.

  When expansion of the area is desired, the following are guides with respect to changing the source schema reflecting the area expansion:

  (1) If the area being expanded has not previously been defined to have PRE-INITIALIZED pages, the area entry in the source schema being changed to reflect expansion must specify PRE-INITIALIZED pages. Pages in the area which already contain data are thus protected.

  (2) The ALLOCATE clause must be altered to reflect the expansion. The general rule which must be adhered to in changing the ALLOCATE clause for the area being expanded is:

      Pages which were previously data pages must remain data pages; pages which were previously overflow pages must remain overflow pages. (Refer to the examples which follow, and also to Section 7, DATABASE LOADING.)

(3) The ALLOCATE clause is the only clause which may be altered in an area entry when an area is expanded.

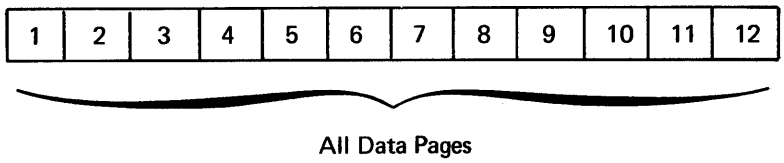
(4) WITHIN clauses referencing the area being expanded may require altering.

Data Manipulation Guides: None

Examples and Discussion:

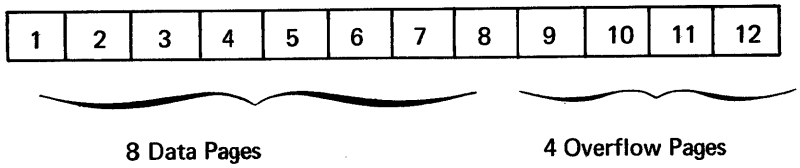
- **Example 1:**

ALLOCATE 12 PAGES



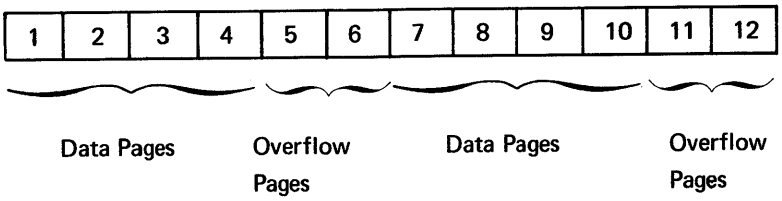
- **Example 2:**

ALLOCATE 12 4 OVERFLOW



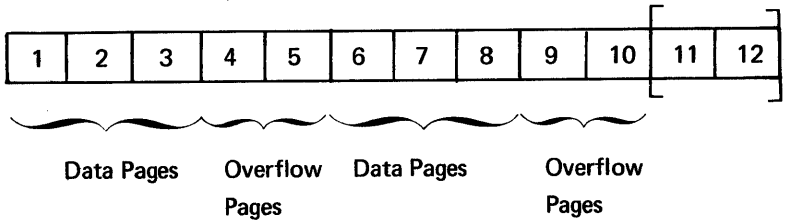
- **Example 3:**

ALLOCATE 12 2 OVERFLOW EVERY 4 DATA



- **Example 4:**

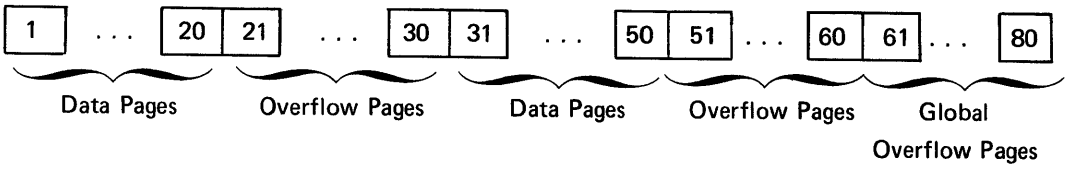
ALLOCATE 12 PAGES, 2 OVERFLOW PAGES EVERY 3 DATA PAGES



Pages 11 and 12 would be made global overflow pages since 12 is not divisible by (2 + 3).

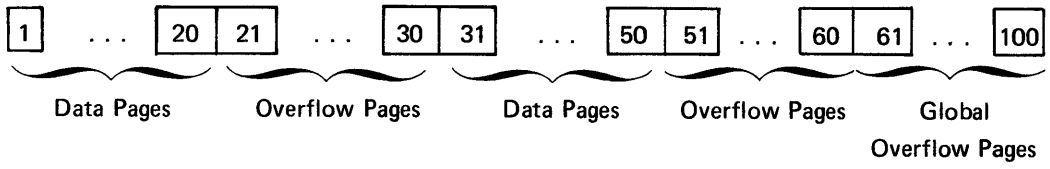
- **Example 5:**

ALLOCATE 80 PAGES
20 OVERFLOW AT END
10 OVERFLOW EVERY 20 DATA
EXPANDABLE TO 100 PAGES



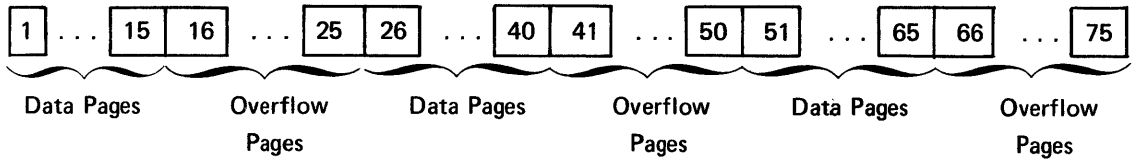
The only possible expansion of the area would be 20 additional global overflow pages. The ALLOCATE clause would appear as:

ALLOCATE 100 PRE-INITIALIZED PAGES
40 OVERFLOW AT END
10 OVERFLOW EVERY 20 DATA



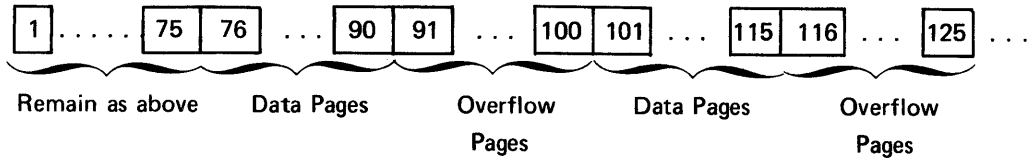
- **Example 6:**

ALLOCATE 75 PRE-INITIALIZED PAGES
10 OVERFLOW EVERY 15 DATA
EXPANDABLE TO 150 PAGES

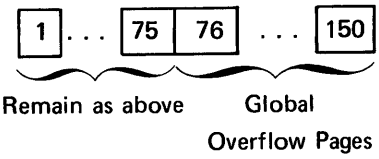


The area whose entry contains the above ALLOCATE clause could be expanded in the following ways:

ALLOCATE 150 PRE-INITIALIZED PAGES
10 OVERFLOW EVERY 15 DATA



ALLOCATE 150 PRE-INITIALIZED PAGES
75 OVERFLOW AT END
10 OVERFLOW EVERY 15 DATA



3.3.6. PAGES Clause

Function:

To define the number of words per page.

Syntax Skeleton:

PAGES **ARE** *integer–5* **WORDS**

Syntactical Constraints:

- Integer–5 must be between 28 and 32760 inclusive.
- Integer–5 must be evenly divisible by 28.

Data Definition Guides:

- Integer–5 applies to both data and overflow pages.
- Selection of page size must be based upon some average number of records per page, taking into account the average record length and the expected variance in both records and record length. Section 6, PAGE AND RECORD STRUCTURE, provides the information about internal page and record structure required to compute page length.

Data Manipulation Guides:

- Initialization of the pages in a non-expanded area may be accomplished in one of two ways.

  (1) If the area is not defined to have PRE-INITIALIZED pages, then an OPEN for INITIAL LOAD will cause the required initialization.

  (2) If the area currently is defined to have PRE-INITIALIZED pages, then either:

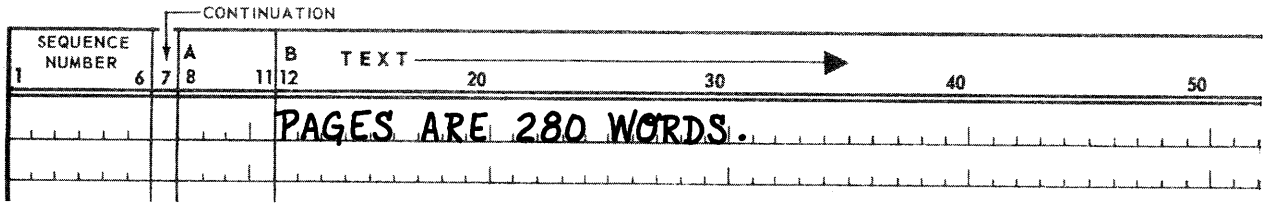
  (a) the area was previously without this clause and was opened for INITIAL LOAD, or

  (b) the Utility for initializing pages was run against this area.

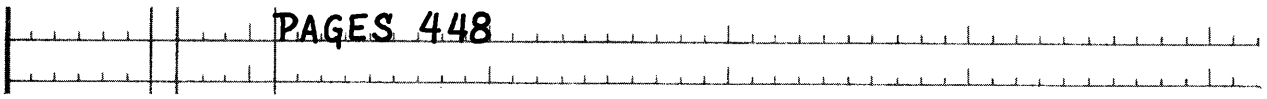
- An expanded area must have the additional pages initialized before it is OPENed. This can only be accomplished by use of the Utility for initializing pages.

Examples and Discussion:

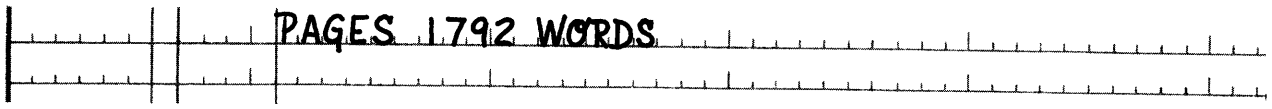
- **Example 1:**



- **Example 2:**



- **Example 3:**



In each of the preceding examples, the page size is a multiple of a sector, and is less than or equal to the number of words in a FASTRAND track. Consideration of the physical characteristics of the mass storage device used is of importance to the Data Administrator when page size is determined.

3.3.7. LOOKS INCLUDE Clause

Function:

To specify the types of data saving techniques to be applied to a particular AREA. This clause over-rides the general Area Looks clause.

Syntax Skeleton:

LOOKS **INCLUDE** { [QUICK-BEFORE-LOOKS] [BEFORE-LOOKS] [AFTER-LOOKS] NO-LOOKS }

Data Definition Guides:

- If this optional clause is omitted from an AREA entry, the default value is the specification in the AREA LOOKS clauses.
- If this specification in the LOOKS clause is different than the specification of the AREA LOOKS clause, then the LOOKS clause is the overriding selection.
- Specification for this clause must be made in conformance with the DMS system generation parameter, namely that an audit trail and/or quick-look file is present.

Data Manipulation Guide:

The SAVE DATA clause available in the DML supplied a choice of either command or run unit quick-before-looks to be taken. This selection will only be honored by DMS 1100 if the looks specified in the SCHEMA for an AREA include QUICK-BEFORE-LOOKS.

Example and Discussion: None

3.3.8. LOAD Clause

Function:

To specify the maximum number of words per page to be used during an initial load of the area.

Syntax Skeleton:

$$\underline{\text{LOAD}}\ \text{IS}\ \left\{\begin{matrix}\textit{integer–6}\ \underline{\text{WORDS}}\\ \textit{integer–7}\ \underline{\text{PERCENT}}\end{matrix}\right\}$$

Syntactical Constraints:

- The LOAD clause is optional; if not used, a load of 100% is assumed.
- If integer–6 is used it must be less than or equal to the number of words per page (integer–5 of the PAGES clause) minus 10, the number of page control words.
- If integer–7 is used, it must be less than or equal to 100.

Data Definition Guides:

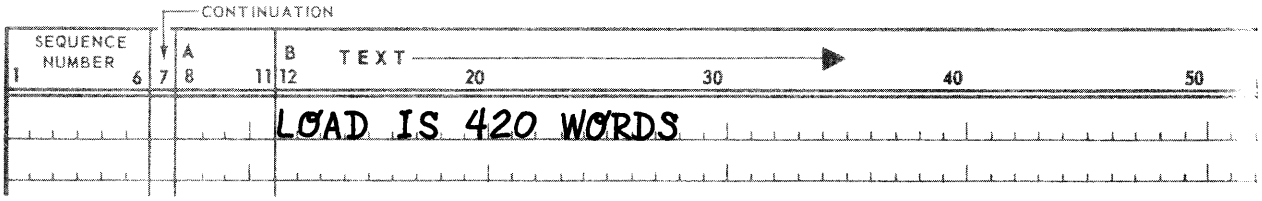
- The LOAD clause is intended for use with a volatile area in which new records are added subsequent to the initial load of the area. The LOAD clause, however, functions only during an initial load of the area. It prevents the pages from completely filling thus leaving space for additional records.
- The LOAD clause is optional; if records are not stored into the area subsequent to the initial load, it is not required. In this situation its presence would have a negative effect because it would prevent a complete filling of each page and thus require additional pages (and mass storage).

- If records are to be added to the area subsequent to the initial load, the LOAD clause is still not required. These records may be stored in unused data pages or overflow pages available for this purpose. However, it is often more efficient and desirable to place new records on the same page as the owner or prior record of the set. In this situation, the LOAD clause may be used to ensure space within the page is available. The page may eventually become full and then placement strategy (see Section 5, RECORD PLACEMENT STRATEGY) must select another data or overflow page.

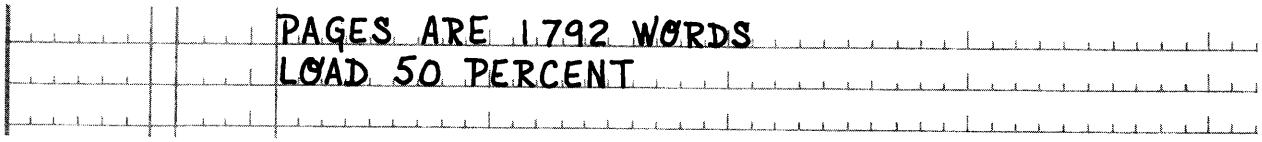
Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**



- **Example 2:**



During the initial load of the area, 891 words are the maximum filled per page in the area. The Data Administrator has saved 891 words on each page in the area for additions after the initial load.

3.3.9. CALC Clause

Function:

To specify the number of calc-chains on a page and to specify whether the calc-chains are singly or doubly linked.

Syntax Skeleton:

CALC **USES** *integer–8* **CHAINS** [**LINKED** PRIOR]

Syntactical Constraints:

- The CALC clause is optional; if not specified, 1 calc-chain is assumed per page.
- Integer–8 must be greater than or equal to 1 and less than or equal to the number of words available per page. (Page size minus number of words in page header, plus 1 to account for calc-chain header word in page header number of words available.)
- Integer–8 must also be less than or equal to the maximum number which can be stored in the smaller of these bit values:

  (a) 15 bits
  (b) 36 bits – page bits – area bits (see Appendix E.)

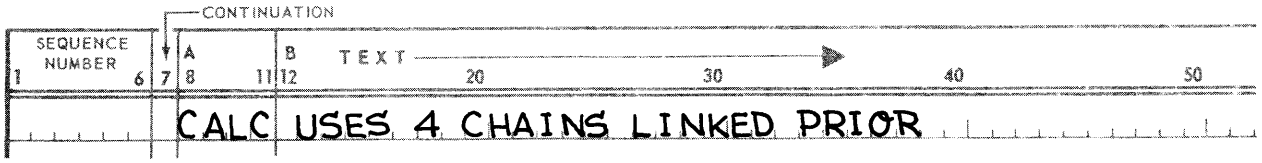
Data Definition Guides:

- Integer–8 is the total number of calc-chains which may be built on each data page in the area.
- The CALC clause specified an output range of (0 to (integer–8)–1) calc-chain numbers for a CALC procedure. The range may be incorporated into the CALC procedure. The calc-chain header words reserved via the CALC clause will not be used for other purposes by the DMR.
- If the optional LINKED PRIOR phrase is specified, each record on a calc-chain contains in its header two calc pointers – one next and one prior. The calc-chain is said to be doubly-linked.

Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**



In addition to the calc-chain header word contained in every page header, three additional calc-chain header words are reserved per page by the above clause.

- **Example 2:**

```
CALC 4 PRIOR
```

3.4. RECORD ENTRY

A record occurrence is the physical unit of transfer between the run units and the DMR. It is a named physical collection of items. This section provides explicit rules for the definition of a record type. All occurrences of a record type are built and maintained according to this definition. Each record types to be defined must be identified first with a RECORD NAME and associated RECORD CODE.

The DMR must have a means of determining where records of a given type are to be stored or found. This is specified through the LOCATION MODE clause. The LOCATION MODE may be DIRECT, in which case the run unit will supply the database-key of the record to be stored or found. The LOCATION MODE may be CALC, in which case a Data Administrator supplied CALC procedure (ASM subroutine) will be passed the search arguments specified by the USING clause and the data-base-data-name–3 specified by the IN clause. The CALC procedure then must determine the page number and the calc-chain number of the record being stored or found. Additionally if a store operation is in progress, the record being stored is subjected to the DUPLICATES ARE [NOT] ALLOWED clause. Finally, the LOCATION MODE may be VIA set-name SET in which case the database-key of the record being stored or found is determined by the DMR based upon the criteria specified for the record's participation in the set.

The WITHIN clause provides a means of constraining the LOCATION MODE clause, thus providing additional criteria for record placement. It is required if the LOCATION MODE is DIRECT, CALC, or VIA set-name SET with an INTERVAL clause specified and serves to ensure that records will not be inadvertently stored outside their intended areas. It is optional if the LOCATION MODE is VIA set-name SET and the INTERVAL clause is not used, and if specified serves as a record placement strategy constraint; if not specified, the "place near" logic is used.

The RESERVE clause is optional, but must be used if the record is to participate as a manual member of one or more sets. It specifies the total number of pointer locations to be allocated per record occurrence to link the record in its manual sets. This allows a record participating in mutually exclusive manual sets to share the same pointer locations, and thus results in a corresponding reduction in the number of pointer locations required in each record occurrence.

The RECORD MODE clause specifies whether items within the record are described according to the rules of FIELDATA COBOL or according to the rules of ASCII COBOL. FIELDATA record mode specifies FIELDATA COBOL item description, and is the default mode of a record in which the optional clause does not appear. ASCII record mode specified ASCII item description. Record mode is used by the DMLP to ensure consistent data description within a DML program.

The remaining clauses define the items within the record. Note that the RECORD NAME clause is an implied 01 level item; thus it may have a PICTURE clause. Each item in the record is in turn provided with an item description.

This description includes:

- A 01 through 49 level number and database identifier (item name) or FILLER;
- A PICTURE clause specifying the size, calc (alphabetic, numeric, or alphanumeric), and editing requirements for the item;
- A USAGE clause specifying the dominate usage of the item (the usage may be for COMPUTATIONAL, DISPLAY, or LOCK purposes or for containing an AREA-NAME, AREA-KEY, or DATABASE-KEY); and data representation in the case of ASCII record mode (DISPLAY and COMPUTATIONAL specify 9-bit characters while DISPLAY–1 and COMPUTATIONAL–4 specify 6-bit characters); and
- An OCCURS clause specifying a fixed or variable number of occurrences of an item.

The standard COBOL rules apply for specifying the PICTURE, USAGE, AND OCCURS clause. Editing symbols are limited to those characters within the DDL character set.

3.4.1. Complete Record Entry

Function:

To identify and provide the characteristics of a record.

Syntax Skeleton:

```
RECORD NAME IS record-name–1 ;

    RECORD CODE IS integer–1 ;

                                  ⎧ DIRECT data-base-data-name–1, data-base-data-name–2
                                  ⎪ CALC   data-base-procedure-name–1
                                  ⎪          IN data-base-data-name–3
                                  ⎪          USING data-base-identifier–1
    LOCATION MODE IS              ⎪                [, data-base-identifier–2] ...
                                  ⎪          DUPLICATES ARE [NOT] ALLOWED
                                  ⎪ VIA    set-name–1 SET
                                  ⎩          [INTERVAL IS integer–2 PAGES]

    [ { ; WITHIN area-name–1 [ { integer–3          } { THRU    } ]
                             [ { data-base-data-name–4 } { THROUGH } ]
                             [ { integer–4          } ]
                             [ { data-base-data-name–5 } ] } ... ]

    [ ; RESERVE integer–5 POINTERS]

    [ ; RECORD MODE IS { FIELDDATA } ]
                       { ASCII     }

    [ { ; { PIC     } IS character-string–1 }
      { ; { PICTURE }                       } ]
      { ; [ { "Item description" } ... ]    }
```

- **Item Description Syntax Skeleton:**

Level-number–1 { *data-base-indentifier–3* / FILLER }

[; { PIC / PICTURE } **IS** *character-string–2*]

[USAGE **IS** { LOCK / DISPLAY / DISP / DISPLAY–1 / DISP–1 / COMPUTATIONAL / COMP / COMPUTATIONAL–4 / COMP–4 / AREA-KEY / AREA-NAME / DATABASE-KEY }]

[; OCCURS { *integer–6* **TIMES** / *integer–7* TO *integer–8* **TIMES** / DEPENDING **ON** *data-base-identifier–4* }]

Syntactical Constraints:

- The clause ordering previously shown must be maintained in all record entries.
- The semicolon and period punctuation is optional, but when used, these must separate clauses and end the entire record entry, respectively.

Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion: None

3.4.2. RECORD NAME Clause

Function:

To name a record type in the schema, by specifying a generic name for all occurrences of the record type in the database.

Syntax Skeleton:

RECORD **NAME IS** *record-name–1*

Syntactical Constraints:

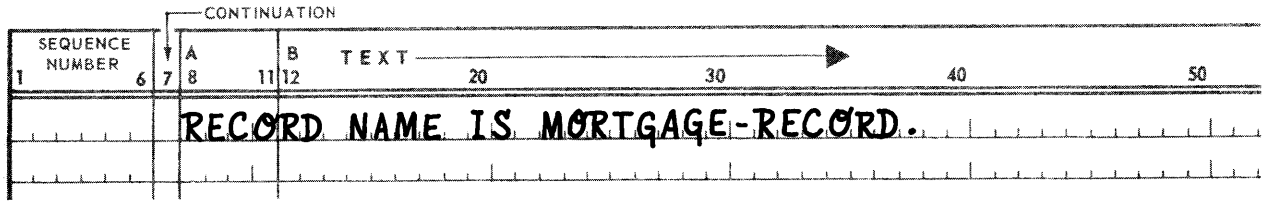
- A record-name must be unique within all record-names applied to a schema.
- Record-name–1 cannot exceed 30 characters.

Data Definition Guides: None

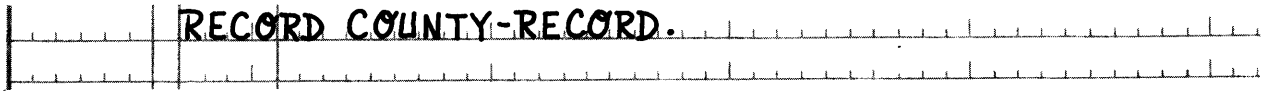
Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**



- **Example 2:**



3.4.3. RECORD CODE Clause

Function:

To specify a value for the record-name to be used by DMS 1100 as an internal shorthand notation for the record-name.

Syntax Skeleton:

RECORD CODE *is integer–1*

Syntactical Constraints:

- Integer–1 must be between 1 and 4000 inclusive.
- Integer–1 must be unique among all record codes assigned within a schema.

Data Definition Guides:

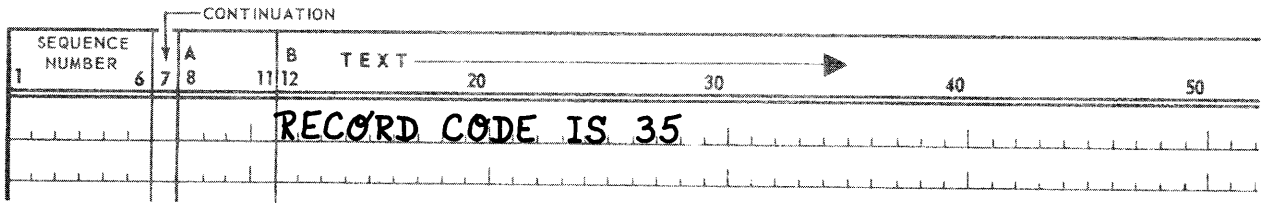
- The record-code assigned to a record becomes the record's identification in DMS 1100 record headers and internal tables. The Data Administrator should think in terms of only record-names, and record-codes.
- An assigned record-code cannot be changed without performing a new initial load on all areas containing records of the original record-code.

Data Manipulation Guides:

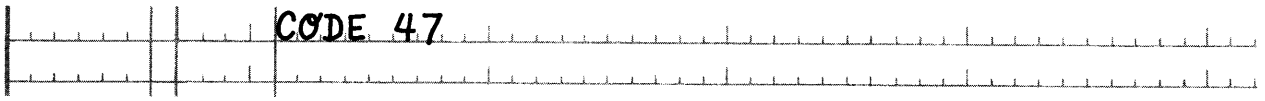
All source DML references to records are based upon the record-name. The Applications Programmer should not be concerned with the existence of a record-code. All conversions are strictly a DMS 1100 activity.

Examples and Discussion:

- **Example 1:**



- **Example 2:**



3.4.4. LOCATION MODE Clause

Function:

To define the criteria for placement and selection of record occurrences.

Syntax Skeleton:

LOCATION **MODE IS** { DIRECT *data-base-data-name–1, data-base-data-name–2* | CALC *data-base-procedure-name–1* [IN *data-base-data-name–3*] [USING *data-base-identifier–1* [,*data-base-identifier–2*] ...] [DUPLICATES **ARE** [NOT] **ALLOWED**] | VIA *set-name–1* **SET** [INTERVAL **IS** *integer–2* **PAGES**] }

Syntactical Constraints:

- Data-base-identifier–1, data-base-identifier–2... must refer to data items within the record being defined. The specification is limited to 10 or fewer elementary data items.
- Set-name–1 must be a set in which the record is defined as being a member.

- If the INTERVAL clause is used, integer–2 must be evenly divisible into the number of data pages in the area or areas where the record is stored. Specifically, if the ALLOCATE clause contains only integer–2, then integer–2 of the ALLOCATE clause must be evenly divisible by integer–2 of the INTERVAL clause. If the ALLOCATE clause contains both integer–2 and integer–10 but not integer–4, then (integer–2 minus integer–10) of the ALLOCATE clause must be evenly divisible by integer–2 of the INTERVAL clause. If integer–4 is present in the ALLOCATE clause, then integer–4 must be evenly divisible by integer–2 of the INTERVAL clause.

- If LOCATION MODE IS DIRECT, CALC, or VIA SET with an INTERVAL clause, the WITHIN clause is required.

Data Definition Guides:

- The LOCATION MODE clause controls placement of records according to the option selected (see Section 5).

  (1) If the DIRECT clause is used, placement is controlled by data-base-data-name–1 and data-base-data-name–2.

  (2) If the CALC clause is used, placement is controlled by data-base-procedure–1 using data-base-identifier–1, data-base-identifier–2... as inputs to select a page and a calc-chain number within the area-name contained in data-base-data-name–3.

  (3) If the VIA set-name SET is used, placement is as close as possible to the logical insert point of the record in set-name–1, as constrained by the WITHIN clause or INTERVAL clause.

- The DUPLICATES clause must be stated if the LOCATION MODE IS CALC. It refers to the input arguments (data-base-identifier–1...) to the CALC routine. It does not refer to the CALC routine output page number and calc-chain numbers. If the DUPLICATES ARE NOT ALLOWED option is specified, additional record occurrences of this record type are rejected if all of the input values are identical to those of previously stored occurrences.

- If the INTERVAL clause is specified, each record occurrence of this type is stored somewhere within a range of integer–2 pages. No other record occurrence of this type may be stored within the same integer–2 pages. The INTERVAL clause is typically used to space owner records to allow them to serve as "cluster points" for the member records in the set occurrences.

Data Manipulation Guides:

- If LOCATION MODE IS DIRECT is specified, data-base-data-name–1 must be defined as a 01-level working or common storage item with a USAGE IS AREA-KEY clause, but no PICTURE clause. Data-base-data-name–2 must be defined as a 77-level working or common storage item with a USAGE IS AREA-NAME clause, but no PICTURE clause. Data-base-data-name–1 must be initialized with a page and record number, and data-base-data-name–2 with an area-name prior to the execution of a command that stores the record or selects it on the basis of its LOCATION MODE clause.

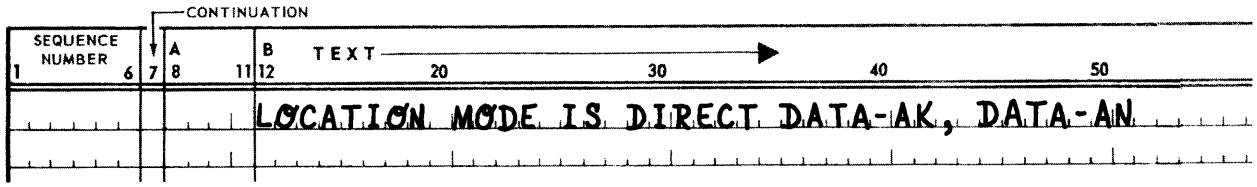
- If LOCATION MODE IS CALC is specified, data-base-procedure-name–1 must be collected into the executing run unit by the MAP processor (see Appendix D., *UNIVAC 1100 Series Data Management System (DMS 1100) COBOL (Fieldata) Data Manipulation Language Programmer Reference, UP–7908* (current version)). Data-base-data-name–3 must be defined as a 77-level working or common storage item with a USAGE IS AREA-NAME clause, but no PICTURE clause. Data-base-identifier–1, data-base-identifier–2... refer to 02 through 49 level elementary data items included in the record. Data-base-data-name–3 must be initialized with an area-name and data-base-identifier–1, data-base-identifier–2... with the appropriate input values prior to the execution of a command that stores a record or selects it on the basis of its LOCATION MODE clause.

The Data Administrator is not required to write his own CALC procedure, but may use the System-supplied CALC procedure, DMSCALC. (Appendix H, SYSTEM CALC PROCEDURE, discusses the use of DMSCALC.)

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

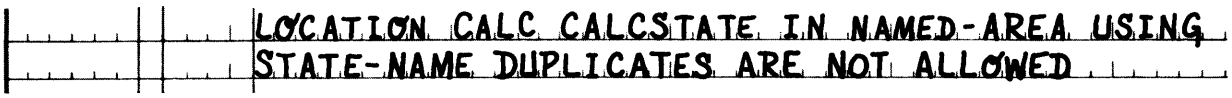
Examples and Discussion:

- **Example 1:**



The data names DATA-AK, DATA-AN are the names of locations through which an area-key and an area-name will be passed during a run unit that stores the record or selects it on the basis of the above LOCATION MODE clause.

- **Example 2:**



The Data Administrator has specified a procedure, CALCSTATE, through which he intends to calculate a page number and a calc-chain number for purposes of storing or finding a record during a run unit. NAMED-AREA is the location through which an area-name will be passed, and STATE-NAME a location through which the input value, used by the CALC procedure to generate a page number and a calc-chain number, will be passed. Additionally, the Data Administrator has imposed the restriction that during a store operation a value passed via STATE-NAME to the calculation procedure must not be identical to a value already passed via STATE-NAME. (Assume the area-name the same.) He has imposed the latter restriction by specifying DUPLICATES ARE NOT ALLOWED. The identifier STATE-NAME must be defined as an item within the record in which the above LOCATION MODE clause appears. STATE-NAME might appear as follows:

■ **Example 3:**

```
SEQUENCE NUMBER | CONTINUATION | A | B TEXT
RECORD EMPLOYEE-EDUCATION
            :
        LOCATION VIA EMPLOYEE-HISTORY
            :
SET     NAME IS EMPLOYEE-HISTORY
            :
        MEMBER IS EMPLOYEE-EDUCATION
            :
```

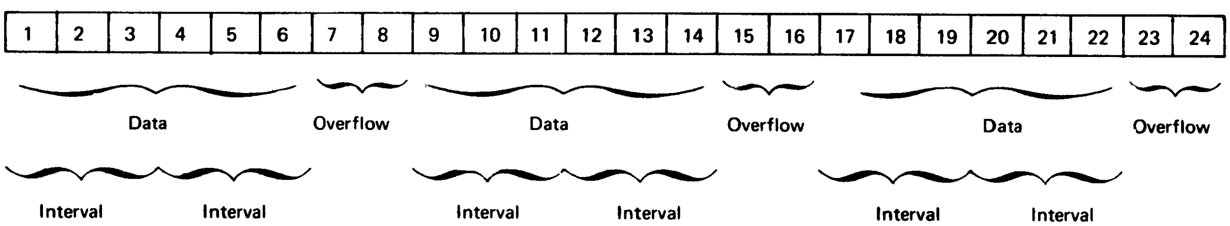
This example illustrates the syntactical relationship which must exist when a record has a LOCATION MODE IS VIA set-name SET clause. The EMPLOYEE-EDUCATION record is located via the EMPLOYEE-HISTORY set; therefore, in the EMPLOYEE-HISTORY set description, the EMPLOYEE-EDUCATION record must be specified in a member subentry.

■ **Example 4:**

```
RECORD R1
        LOCATION VIA S1 INTERVAL 3
```

Suppose A1 is the area in which R1 type records will be stored. Assume the ALLOCATE clause for A1 is the following:

ALLOCATE 24 PAGES 2 OVERFLOW EVERY 6 DATA



Only one record occurrence of type R1 could be stored in each of the intervals.

■ **Example 5:**

```
        LOCATION CALC NAMECALC IN NAMED-AREA USING
        STUDENT-NAME DUPLICATES ARE NOT ALLOWED
```

Suppose that the value passed to the CALC procedure NAMECALC via the location named STUDENT-NAME is always the full name of a student. Suppose further that the CALC procedure employs a simple algorithm which generates a page number and a calc-chain number in the following manner:

(1) The procedure assigns a numeric value (A=1 through Z=26) to each letter in the alphabet and calculates the numeric value of the student name to generate a page number.

(2) The procedure calculates the remainder in the division of the number of letters in the student name by integer–8 of the CALC clause for the area passed in NAMED-AREA to generate the calc-chain number.

Assume NAMED-AREA is HISTORY-AREA, and HISTORY-AREA's definition in the source schema contains the clause:

CALC USES 10 CHAINS

For the student names MARTIN H. JOHNSON and JOHN H. MARTINSON, the CALC procedure would generate identical page (178) and calc-chain (4) numbers. Both records would be placed, space permitting, on page 178, and linked into the calc-chain identified by number 4. (See Section 5, "RECORD PLACEMENT STRATEGY" for a description of the use of overflow pages by calc-chains.)

3.4.5. WITHIN Clause

Function:

To define the area(s) and page limits within which the record occurrence is to be placed.

Syntax Skeleton:

$$\left\{ \underline{\text{WITHIN}} \;\; \textit{area-name–1} \left[\left\{ \begin{matrix} \textit{integer–3} \\ \textit{data-base-data-name–4} \end{matrix} \right\} \;\; \left\{ \begin{matrix} \underline{\text{THRU}} \\ \underline{\text{THROUGH}} \end{matrix} \right\} \;\; \left\{ \begin{matrix} \textit{integer–4} \\ \textit{data-base-data-name–5} \end{matrix} \right\} \right] \right\} \ldots$$

Syntactical Constraints:

- The WITHIN clause must be specified if the LOCATION MODE is either DIRECT or CALC. The WITHIN clause is optional if the LOCATION MODE is VIA SET unless the INTERVAL clause is also specified. If the INTERVAL clause is specified, the WITHIN clause must also be specified.
- The WITHIN clause, if repeated, cannot specify the same area-name more than once.
- Only one range of page numbers may be specified for a given area. This range includes the specified page numbers. The upper limit must not exceed the number of pages specified in the ALLOCATE clause of the respective area entry.
- If integer–3 and integer–4 are specified, then integer–3 must be less than or equal to integer–4.

Data Definition Guides:

- The WITHIN clause constrains the normal DMR placement strategy. If the LOCATION MODE is VIA SET, the WITHIN clause overrides the "place near" logic by storing records only within the area(s) and page(s) specified. Therefore, the intent is to separate owners and members into different areas.

- If the LOCATION MODE is either DIRECT or CALC, the WITHIN clause prevents the run unit or CALC procedure from assigning page numbers outside the limits specified by the Data Administrator.

- The WITHIN clause is a part of the DMR placement strategy. (See Section 5 for a further discussion in which it may be used.)

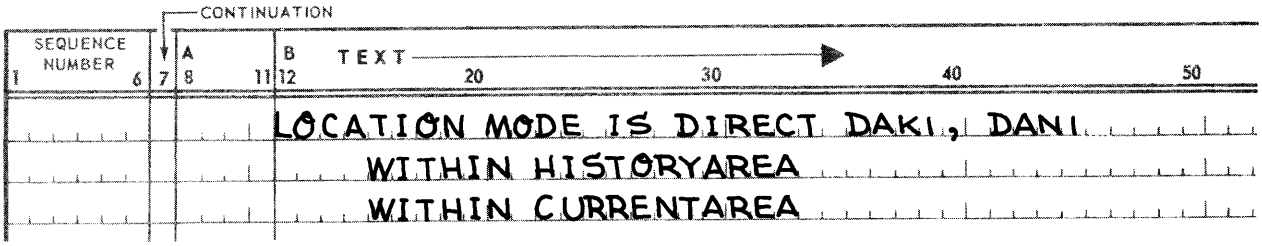
Data Manipulation Guides:

- If a run unit is storing a record whose LOCATION MODE is either DIRECT or CALC, the database-key or page number, respectively, supplied to the DMR must be within the area(s) and page(s) limits imposed by the WITHIN clause. An error status is returned if the limits are exceeded.

- If either data-base-data-name—4 or data-base-data-name—5 are specified, these data-base-data-names must be defined as 77-level data names in the WORKING STORAGE SECTION of the DML source program with a PICTURE IS 9(10) clause and a USAGE OF COMPUTATIONAL clause. Prior to a STORE of a record occurrence the run unit must initialize these 77-level data names with appropriate page numbers. An error status is returned if these page numbers do not meet the same constraints imposed by the Syntactical Constraints 3 and 4.

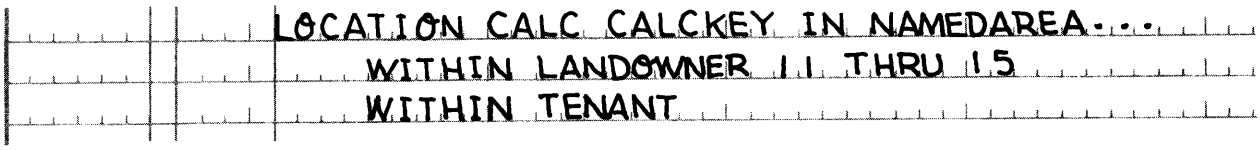
Examples and Discussion:

- **Example 1:**



The Data Administrator has restricted the placement of the record to areas HISTORYAREA and CURRENTAREA. DAN1 must be initialized with the value HISTORYAREA or CURRENTAREA prior to the execution of a command that either stores or selects the record on the basis of its LOCATION MODE clause.

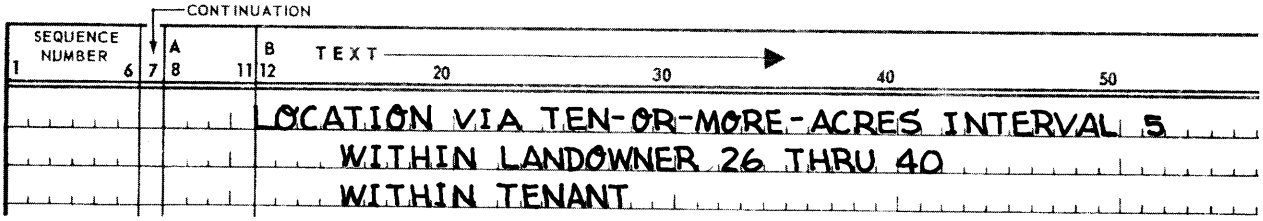
- **Example 2:**



NAMEDAREA must be initialized with the value LANDOWNER or TENANT prior to the execution of a command that either stores or selects the record on the basis of its LOCATION MODE clause. If the CALC clause for both area LANDOWNER and area TENANT were:

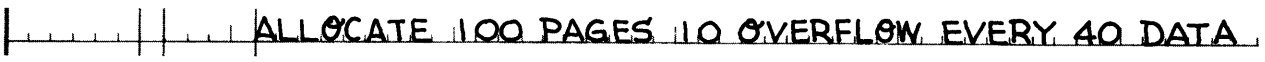
then the calculation routine CALCKEY could be written to generate calc-chain numbers 0–9 in the data pages contained in the page interval 11 through 15 in the area LANDOWNER, and using calc-chain numbers 0–9 on any data page in area TENANT. (See Section 5.)

- **Example 3:**

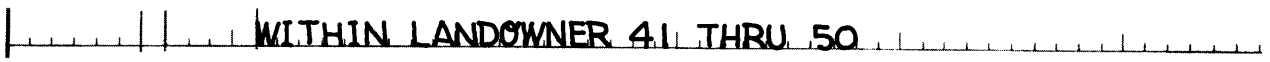


In the areas LANDOWNER and TENANT, the number of data pages before or between overflow pages (if overflow pages are specified) must be an even multiple of five, the INTERVAL value in the LOCATION clause above.

Suppose the ALLOCATE clause for area LANDOWNER is:

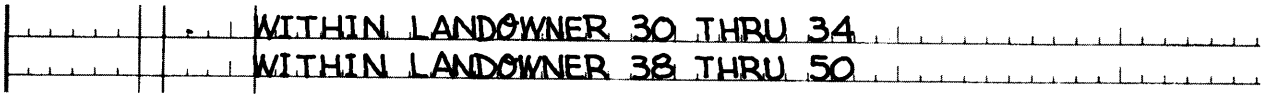


Pages 1–40 and 51–90 in area LANDOWNER are data pages. (Pages 41–50 and 91–100 are overflow pages.) The data pages can be divided into five-page intervals: 1–5, 6–10, 11–15, 16–20, 21–25, 26–30, 31–35, 36–40, 51–55, 56–60, 61–65, etc. If a record of the type containing the above clauses is to be stored in area LANDOWNER, it must be stored in only one of three intervals: 26–30, 31–35, or 36–40. The page limit 26–40 in the WITHIN clause specifying LANDOWNER has imposed the latter restriction. An example of an invalid page limit in the WITHIN clause specifying LANDOWNER would be:



Overflow pages are not used in the INTERVAL specification.

Other types of invalid page limits for the same example are:



The first is invalid because 30–34 contains no interval, i.e., intervals 26–30 and 31–35 span 30–34, but neither is contained in 30–34.

Pages 38–50 contain 3 data pages followed by 10 overflow pages. Therefore, the second is invalid for the same reason.

3.4.6. RESERVE Clause

Function:

To define the number of pointer locations reserved within a record occurrence for its participation in manual set memberships.

Syntax Skeleton:

RESERVE *integer–5* **POINTERS**

Syntactical Constraints:

- The RESERVE clause must be specified if the record being defined participates as a MANUAL MEMBER of any sets.
- Integer–5 must be greater than or equal to 1 and less than or equal to 63.

Data Definition Guides:

- A record occurrence may be a potential manual member of n sets, and an actual member of no more than m sets (where $m \leqslant n$) at any time during its existence in a database. The RESERVE clause is provided in order to conserve on the usage of pointers in each record occurrence of this record type.

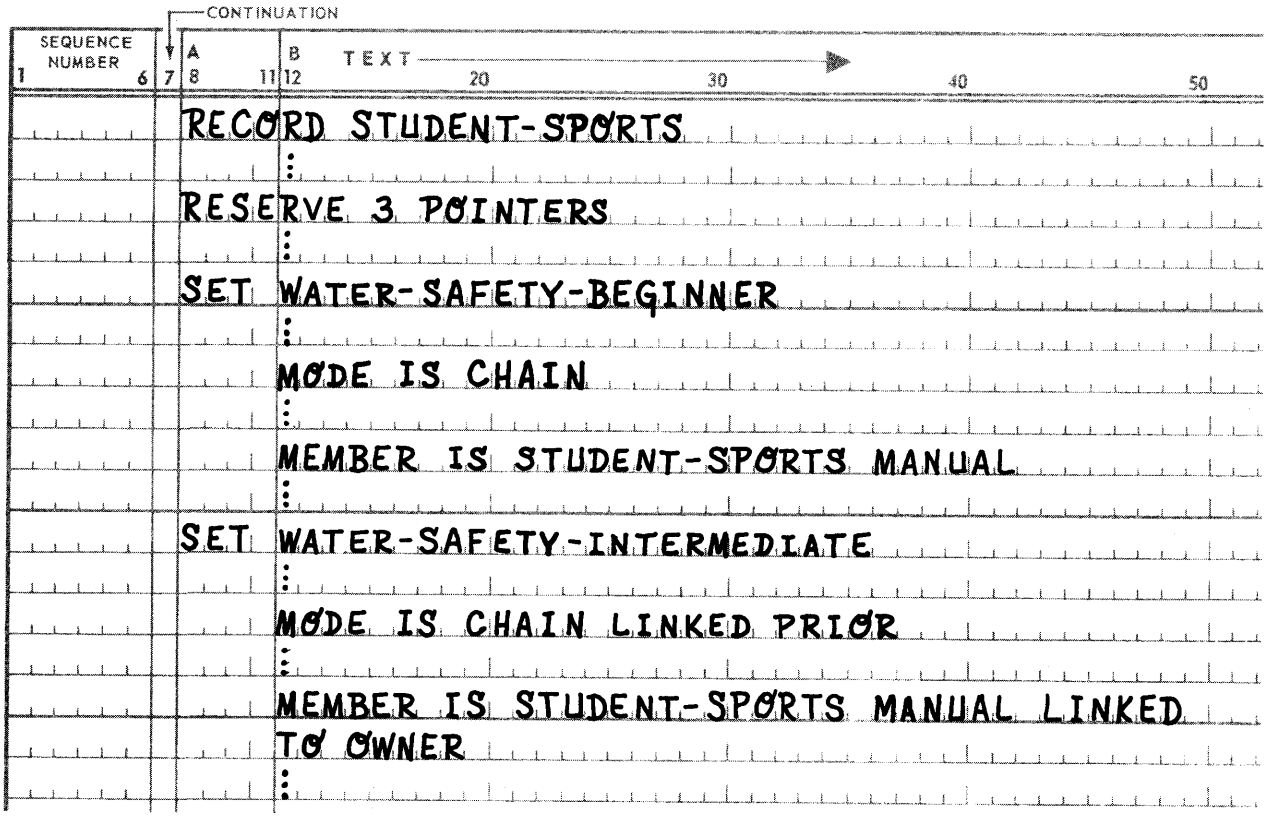
  As a record occurrence is REMOVED from one set and INSERTED in a second, the same pointer location in the record occurrence can be used. Therefore, only m-pointer locations need be established for the record.

- Each pointer location requires one 1100 Series computer word and begins and ends on word boundaries. Thus membership in a set requires 1 to 3 words for pointer storage (always 1 word for the implied NEXT pointers and 1 or 2 additional words depending upon selection of PRIOR and/or OWNER pointers).

- This clause applies only to pointers required for manual set participation. The pointers required for automatic set participation are automatically allocated by the DDL Translator.

Data Manipulation Guides: None

Examples and Discussion:



Membership of a record of type STUDENT-SPORTS in set WATER-SAFETY-BEGINNER requires one pointer, the implied NEXT pointer; membership of a record of the same type in set WATER-SAFETY-INTERMEDIATE requires three pointers which are as follows:

- For the implied NEXT pointer,
- For the PRIOR link, and
- For the OWNER links.

Three pointers have been reserved and will be sufficient if a record of type STUDENT-SPORTS is not inserted in one of the preceding two sets before its occurrence is removed from the other. If a record of type STUDENT-SPORTS could be, simultaneously, a member of WATER-SAFETY-BEGINNER and WATER-SAFETY-INTERMEDIATE, then four pointers should be reserved in the definition.

3.4.7. RECORD MODE Clause

Function:

To specify whether items within the record are described according to the rules of FIELDATA COBOL or according to the rules of ASCII COBOL. The clause is optional with the default mode being FIELDATA.

Syntax Skeleton:

RECORD MODE IS { FIELDATA / ASCII }

Syntactical Constraints: None

Data Definition Guides:

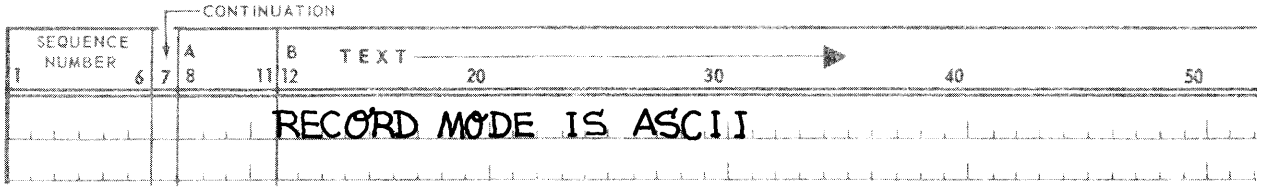
- The RECORD MODE clause is optional; if not used FIELDATA mode is assumed.
- If record mode is FIELDATA then data item descriptions are in accordance with FIELDATA COBOL. (See *UNIVAC 1100 Series American National Standard COBOL (Fieldata) Programmer Reference, UP–7845* (current version).)
- If record mode is ASCII then data item descriptions are in accordance with ASCII COBOL. (See *UNIVAC 1100 Series American National Standard COBOL (ASCII) Programmer Reference, UP–7923* (current version).)

Data Manipulation Guides:

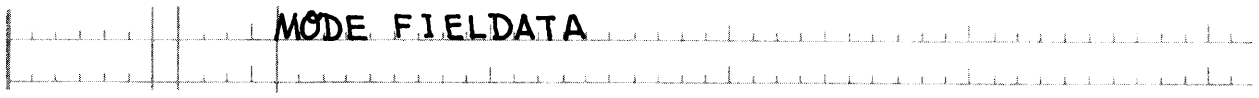
- The FIELDATA version of the DMLP will accept FIELDATA records only.
- The ASCII version of the DMLP will accept FIELDATA records if the T option is specified as a COBOL option, and ASCII records if it is not. All records referenced in a single DML program must be of the same mode.

Examples and Discussion:

- **Example 1:**



- **Example 2:**



3.4.8. LEVEL-NUMBER Clause

Function:

To identify with a data-base-identifier a data item or unused portion of a record and to establish its position relative to the hierarchical data item structure within a record.

Syntax Skeleton:

Level-number { *data-base-identifier–3* / FILLER }

Syntactical Constraints:

- The level-number clause is the required first clause in each item description subentry.
- Item description subentries may only have level-numbers with the values 02 through 49.
- If data-base-identifier–3 is specified, it must not be qualified.

Data Definition Guides:

- The data items within a record are identified by their respective data-base-identifier. The data-base-identifier must be unique within a record.

Data Manipulation Guides:

- When required, qualification must be used to achieve uniqueness.

Examples and Discussion:

- **Example 1:**

CONTINUATION

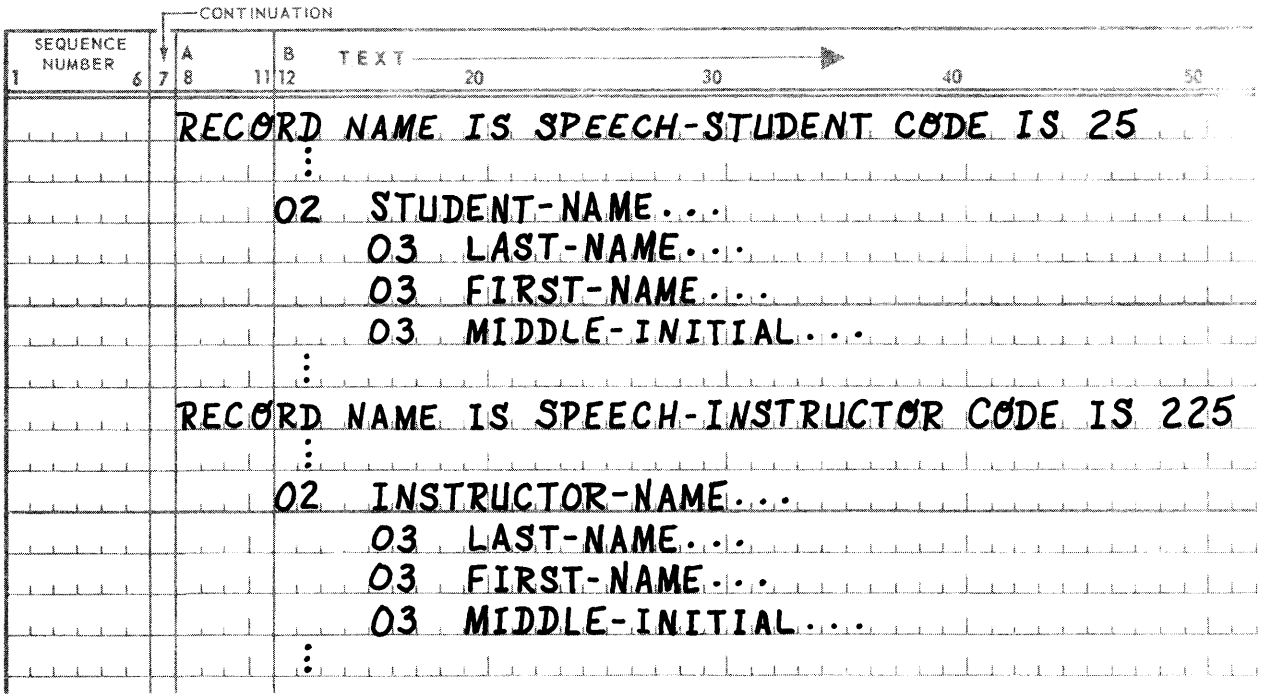
SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT 20 30 40 50

```
RECORD NAME IS BIOLOGY-STUDENT CODE IS 20
    :
    02  STUDENT-NAME...
        10  LAST-NAME...
        10  FIRST-NAME...
        10  MIDDLE-INITIAL...
    02  ACCUMULATIVE-GRADES...
        03  FALL-QUARTER...
            05  FALL-QUIZ-TOTAL...
            05  FALL-TEST-TOTAL...
        03  WINTER-QUARTER...
            05  WINTER-QUIZ-TOTAL...
            05  WINTER-TEST-TOTAL...
        03  SPRING-QUARTER...
            05  SPRING-QUIZ-TOTAL...
            05  SPRING-TEST-TOTAL...
    :

RECORD NAME IS SPEECH-STUDENT CODE IS 25
    :
    02  STUDENT-NAME...
        10  LAST-NAME...
        10  FIRST-NAME...
        10  MIDDLE-INITIAL...
    02  ACCUMULATIVE-GRADES...
        03  FALL-QUARTER...
            05  FALL-QUIZ-TOTAL...
            05  FALL-TEST-TOTAL...
        03  FILLER...
    :
```

When a run unit references an item in one of the preceding records which has the same name as an item in another record, the record name must be used as a qualifier to achieve uniqueness.

- **Example 2:**



When a run unit references one of the identifiers LAST-NAME, FIRST-NAME, or MIDDLE-INITIAL, the identifier must be qualified. STUDENT-NAME or INSTRUCTOR-NAME may be used to make a reference to LAST-NAME, FIRST-NAME, or MIDDLE-INITIAL unique.

3.4.9. PICTURE Clause

Function:

To define the general characteristics (size and type) of an elementary item and its editing requirements.

Syntax Skeleton:

$$\left\{\begin{matrix}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{matrix}\right\} \textbf{ IS } \textit{character-string}$$

Syntactical Constraints:

- The PICTURE clause must be specified for each elementary item, except for a data item defined with a USAGE IS AREA-KEY, USAGE IS AREA-NAME, or USAGE IS DATABASE-KEY, in which case the use of the PICTURE clause is prohibited.
- PIC is an abbreviation for PICTURE.
- The character string specified on the PICTURE clause cannot exceed 30 descriptive characters in length. However, this syntactical constraint does not apply to the length of the data item defined by the character string, which may exceed 30 characters.

Data Definition Guides:

The rules governing the formation of a PICTURE character string in a Fieldata record are found in *UNIVAC 1100 Series American National Standard COBOL (Fieldata) Programmer Reference, UP–7845* (current version). The rules governing the formation of a PICTURE character string in an ASCII record are found in the *UNIVAC 1100 Series American National Standard COBOL (ASCII) Programmer Reference, UP–7923* (current version). All of the *American National Standard* COBOL data character symbols and operational symbols, except '1', are acceptable PICTURE characters to the DDL translator. However, only one editing symbol, 'B', is acceptable to the DDL translator.

Data Manipulation Rules: None

Examples and Discussion:

- **Example 1:**

CONTINUATION | SEQUENCE NUMBER 1–6 | 7 | A 8–11 | B 12 TEXT → 20 30 40 50

```
RECORD NAME SPEECH-STUDENT
    :
    02 STUDENT-NAME
          10 LAST-NAME PICTURE X(15)
          10 FIRST-NAME PIC X(9)
          10 MIDDLE-INITIAL PICTURE IS X
    02 ACCUMULATIVE-GRADES
          03 FALL-QUARTER
                05 FALL-QUIZ-TOTAL PIC H9(2) V99
                05 FALL-TEST-TOTAL PIC H9(3)
          03 FILLER PICTURE X(6)
    :
```

- **Example 2:**

```
RECORD NAME IS MATERIAL
    :
    02 DATE-DUE PICTURE IS 99B99B99
    02 MATERIAL-DATA
          05 MATERIAL-NAME PIC IS X(12)
          05 PART-CODE          PIC IS 99AA9
          05 QUANTITY-OF UNITS  PICTURE 999P(3)V
          05 COST-PERT-UNIT  PIC V999
          05 SHIPMENT-COST-TOTAL  PICTURE  9(10)V99
          05 MATERIAL-DESCRIPTION  PICTURE X(30)
          05 COST-VARIANCE  PIC IS SV999
    :
```

3.4.10. USAGE Clause

Function:

To define the format of a data item.

Syntax Skeleton:

```
                  ⎧ LOCK                  ⎫
                  ⎪ DISPLAY               ⎪
                  ⎪ DISP                  ⎪
                  ⎪ DISPLAY–1             ⎪
                  ⎪ DISP–1                ⎪
                  ⎪ COMPUTATIONAL         ⎪
USAGE IS          ⎪ COMP                  ⎪
                  ⎪ COMPUTATIONAL–4       ⎪
                  ⎪ COMP–4                ⎪
                  ⎪ AREA-KEY              ⎪
                  ⎪ AREA-NAME             ⎪
                  ⎩ DATABASE-KEY          ⎭
```

Syntactical Constraints:

- If USAGE IS AREA-KEY, AREA-NAME or DATABASE-KEY is specified, the PICTURE clause must be omitted and the item must be an elementary item.
- If USAGE IS AREA-KEY is specified, a level-number of 49 must not be specified.
- USAGE IS LOCK, DISPLAY–1, DISP–1, COMPUTATIONAL–4 or COMP–4 must not be specified if record mode is FIELDATA.

Data Definition Guides:

- For Fieldata or ASCII record modes the rules of governing the specification of the USAGE clause are found in the appropriate *UNIVAC 1100 Series American National Standard COBOL (Fieldata UP–7845 or ASCII UP–7923) Programmer Reference* (current version). The following apply only to a USAGE IS AREA-KEY, AREA-NAME or DATABASE-KEY.
- If the USAGE IS AREA-KEY clause is specified and the record mode is FIELDATA (ASCII) it will be replaced by the DDL translator with a USAGE is COMPUTATIONAL (COMPUTATIONAL–4) clause. The DDL translator treats the item as a group item by adding two elementary items to it, each with a 1 higher level-number and each with a PICTURE of 9(5). The two elementary items have the data-base-identifiers of PAGE-NUM and RECORD-NUM, respectively. If an OCCURS clause is specified, it refers to the resultant group item.
- If the USAGE IS AREA-NAME clause is specified and the record mode is FIELDATA (ASCII) it is replaced by the DDL translator with a PICTURE IS X(12) clause (PICTURE IS X(12) USAGE IS DISPLAY–1 clauses).
- If the USAGE IS DATABASE-KEY clause is specified and the record mode is FIELDATA (ASCII) it will be replaced by the DDL Translator with a PICTURE IS 9(10) clause (PICTURE IS 9(10) USAGE IS DISPLAY–1 clauses).

Data Manipulation Guides:

- PAGE-NUM and RECORD-NUM will appear as elementary items each time the original USAGE was specified as AREA-KEY; therefore, all DML references to them must be qualified by the respective data-base-identifier of the newly formed group item.

Examples and Discussion:

- **Example 1: (Assumes FIELDATA record mode.)**

```
03  KEY-ITEM USAGE IS AREA-KEY
```

The DDL translator will produce the following:

```
03  KEY-ITEM USAGE IS COMPUTATIONAL
    04  PAGE-NUM  PICTURE IS 9(5)
    04  RECORD-NUM  PICTURE IS 9(5)
```

- **Example 2: (Assumes FIELDATA record mode.)**

```
07  NAMED-AREA USAGE IS AREA-NAME
```

The DDL translator will produce the following:

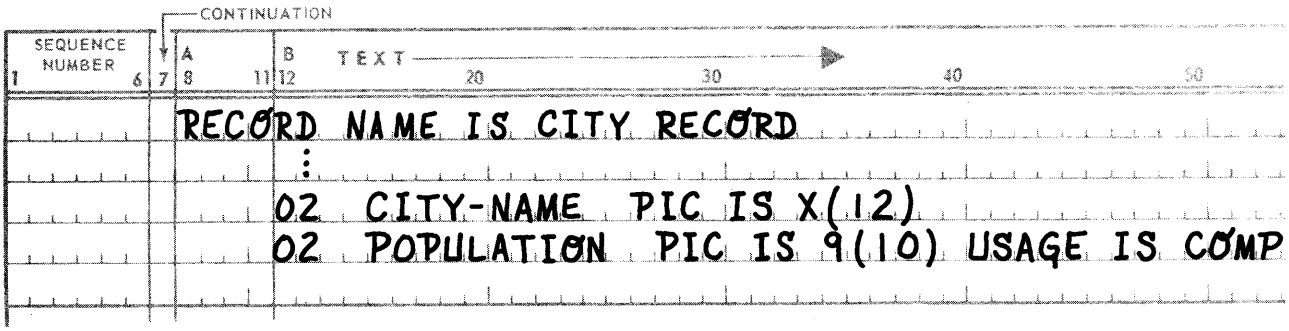
```
07  NAMED-AREA PICTURE IS X(12)
```

- **Example 3:**

```
04  STUDENT-DATA USAGE IS DISPLAY
    06  STUDENT-NAME  PIC X(15)
    06  TUITION-PAID  PIC 9999V99
```

The USAGE applies to both 06-level identifiers. TUITION-PAID could not have a picture H9999V99. The "H" implicitly specifies computational usage, which would conflict with the group usage.

- **Example 4:**



- **Example 5:**

  05 DBK USAGE IS DATABASE-KEY

  The DDL translator will produce the following:

  05 DBK PICTURE IS 9(10)

3.4.11. OCCURS Clause

Function:

To supply the information required for the application of subscripts for repeated data and for the elimination of the need for separate entries.

Syntax Skeleton:

Format 1:

OCCURS *integer–6* **TIMES**

Format 2:

OCCURS *integer–7* TO *integer–8* **TIMES**
DEPENDING **ON** *data-base-identifier–4*

Syntactical Constraints:

- Integer–6, integer–7 and integer–8 must be unsigned positive integers. Integer–7 must be less than integer–8. The value of integer–7 may be zero, but integer–8 cannot be zero.

- Data-base-identifier–4 must be previously defined as an 02 through 49-level item within this record. Data-base-identifier–4 must be defined with a USAGE IS COMP.

- If a data item is defined with an OCCURS clause having the DEPENDING option, it must follow the definition of all data items which might require reference by the DMR; e.g., those items defined as sort items, CALC items, search items, or items controlling the length of repeated data.

Data Definition Guides:

- The use of an OCCURS clause, with a DEPENDING option, results in a variable length record on the database. Each record occurrence is compacted during either a STORE or MODIFY operation and expanded during either a FETCH or GET operation. The main storage allocated for the record is of maximum size.

- A group item which has an OCCURS clause and which contains a mixture of 6-bit and 9-bit data must have a length which is a multiple of halfwords. If the length is not a multiple of halfwords, alignment of elementary items within the group is unpredictable from one occurrence of the group to the next occurrence of the group; the latter alignment inconsistency will cause problems when elementary items are referenced.

- The use of variable length records is a considerable savings of mass storage. This savings must be balanced against a slight increase in expansion-compaction procedural overhead. However, overhead will be further increased, if the number of data items in use is increased (on a MODIFY operation) and space is not available to return the expanded record to the page. The record will be stored on an overflow page, the space on the original page designated vacant and a linkage established between the original and overflow page. The record may still be accessed on either the basis of set participation or its original database key; however, a second I/O operation is now required. Further implications resulting from the use of variable length records are discussed in Section 5, RECORD PLACEMENT STRATEGY.

Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**

CONTINUATION | SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT 20 30 40 50

```
       02 PAYMENT-STATUS OCCURS 36 TIMES
          04 DATE PIC X(6)
          04 PAYMENT PIC 9(10)
```

- **Example 2:**

```
   RECORD NAME IS CAR-PAYMENT-RECORD
       :
       02 INDIVIDUAL PIC X(12)
       :
       02 PAYMENT-PERIOD PIC 9(2) USAGE IS COMP
       02 PAYMENT-STATUS OCCURS 12 TO 36 TIMES
          DEPENDING ON PAYMENT
          04 DATE PIC X(6)
          04 PAYMENT PIC 9(10)
```

3.5. SET ENTRY

A set is a named collection of record types. These record types have a hierarchical relationship to each other, since one must be defined as an owner and the remainder as members. A given record type may appear in more than one collection of records, that is, in more than one set, either as an owner or a member of that set.

This section provides explicit rules for the definition of a set. Each set to be defined must first be externally identified through the SET NAME clause and associated with a unique internal code through the SET CODE clause. The MODE clause indicates the type of set; it must be CHAIN, i.e., each record occurrence starting with the owner is linked to the next record occurrence and the last linked to the owner. If desired, the record occurrences may also be LINKED PRIOR.

Where a given record occurrence is inserted within the chain is governed by the ORDER clause. The ORDER may be FIRST, LAST, NEXT, PRIOR, or SORTED.

The OWNER clause specifies the owner record type. In addition, one or more member record types must be specified, each defined by a MEMBER clause and its associated clauses. The member record must be AUTOMATIC or MANUAL and may be LINKED TO OWNER. If the ORDER IS SORTED, ASCENDING or DESCENDING KEY(s) are required. These key items determine the logical sequence of records within a set occurrence. The RANGE option allows a "less than or equal to" or "greater than or equal to" sort criteria to be used. For all key items, DUPLICATES ARE FIRST, LAST, or NOT ALLOWED.

Each owner record occurrence stored on the database determines a separate set occurrence. During a STORE or FIND/FETCH operation affecting this set, the correct set occurrence (OWNER record occurrence) is determined by the SET OCCURRENCE SELECTION'S THRU CURRENT OF SET clause or LOCATION MODE OF OWNER clause. If the latter, search items for the OWNER record may be specified with a USING clause, or substitute search items may be specified with an ALIAS clause.

3.5.1. COMPLETE SET ENTRY

Function:

To identify and provide the characteristics of a set.

Syntax Skeleton:

```
SET NAME IS set-name–1;
    SET CODE IS integer–1;
    MODE IS CHAIN [LINKED PRIOR] ;
                 ( FIRST                                  )
                 ( LAST                                   )
    ORDER IS     ( NEXT                                   )
                 ( PRIOR                                  )
                 ( SORTED [WITHIN RECORD-NAME]            )
    OWNER IS record-name–1;
        {"Member sub-entry"} ...
```

Syntax Format of Member Sub-Entry:

```
MEMBER IS record-name-2 { AUTOMATIC } [LINKED TO OWNER]
                        { MANUAL    }
[{{ASCENDING } [RANGE] KEY IS || RECORD-NAME                        || } ...
 {{DESCENDING}                || data-base-identifier-1             || }
                              || [ ,data-base-identifier-2] ...     || }

              DUPLICATES ARE { FIRST       } ] ;
                             { LAST        } ]
                             { NOT ALLOWED } ]

{ "Set occurrence selection sub-entry" }
```

Syntax Format of Set Occurrence Selection Sub-Entry:

Format 1:

```
;SET OCCURRENCE SELECTION IS THRU
{CURRENT OF SET            }
{LOCATION MODE OF OWNER    }
  [{USING data-base-identifier-3[ ,data-base-identifier-4] ...                          }]
  [{{ALIAS{FOR data-base-identifier-5} IS data-base-data-name-2} ...                    }]
  [{       {OF  data-base-data-name-1}                                                  }]
```

Format 2:

```
;SET OCCURRENCE SELECTION IS THRU set-name-2 USING
{CURRENT OF SET            }
{LOCATION MODE OF OWNER    }
  [{ USING data-base-identifier-6 [ ,data-base-identifier-7] ...                        }]
  [{{ALIAS {FOR data-base-identifier-8 } IS data-base-data-name-2} ...                  }]
  [{       {OF  data-base-data-name-1  }                                                }]
{ set-name-3                                                                            }
{ { USING data-base-identifier- 9 [ ,data-base-identifier-10] ..                    }   }
{ {{ALIAS FOR data-base-identifier-11 IS data-base-data-name-3} ...                 }   } ...
```

Syntactical Constraints:

- The clause ordering shown preceding must be maintained in all set entries.
- The MEMBER subentry must be repeated for each member record of the set. The SET OCCURRENCE SELECTION clause (Format 1 or Format 2) is part of the MEMBER subentry.
- The semicolon and period punctuation is optional, but when used, these must separate clauses and end the set entry, respectively.

Data Definition Guides:

The set entry is optional. It is required only if set relationships are to exist between record types.

Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**

```
SET NAME IS COUNTY-CITY
    SET CODE IS 35
    MODE IS CHAIN LINKED PRIOR
    ORDER IS SORTED
    OWNER IS COUNTY-RECORD
    MEMBER IS CITY-RECORD AUTOMATIC LINKED TO
    OWNER
          ASCENDING RANGE KEY IS POPULATION
          DUPLICATES ARE FIRST
          SET OCCURRENCE SELECTION IS THRU LOCATION
                MODE OF OWNER USING COUNTY-NAME
```

- **Example 2:**

```
SET COUNTY-CITY CODE 35
    MODE CHAIN PRIOR ORDER SORTED
    OWNER COUNTY-RECORD
    MEMBER CITY-RECORD AUTOMATIC OWNER
          ASCENDING RANGE POPULATION DUPLICATES FIRST
          SELECTION LOCATION USING COUNTY-NAME
```

- **Example 3:**

CONTINUATION

SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT → 20 30 40 50

```
SET NAME IS CURRICULUM CODE IS 405
    MODE IS CHAIN
    ORDER IS SORTED WITHIN RECORD-NAME
    OWNER IS SCHOOL
    MEMBER IS HISTORY AUTOMATIC
        ASCENDING KEY IS COURSE-TITLE
        DUPLICATES ARE LAST
        SET OCCURRENCE SELECTION IS THRU LOCATION
            MODE OF OWNER
    MEMBER IS BIOLOGY AUTOMATIC
        ASCENDING KEY IS ENROLLMENT
        DUPLICATES ARE LAST
        SET OCCURRENCE SELECTION IS THRU LOCATION
            MODE OF OWNER
    MEMBER IS HEALTH MANUAL
        ASCENDING KEY IS GRADE-LEVEL
        DUPLICATES ARE LAST
        SET OCCURRENCE SELECTION IS THRU
            LOCATION MODE OF OWNER
```

3.5.2. SET NAME Clause

Function:

To identify a set type in the schema and in all of the set type's occurrences in the database.

Syntax Skeleton:

<u>**SET**</u> **NAME IS** *set-name–1*

Syntactical Constraints:

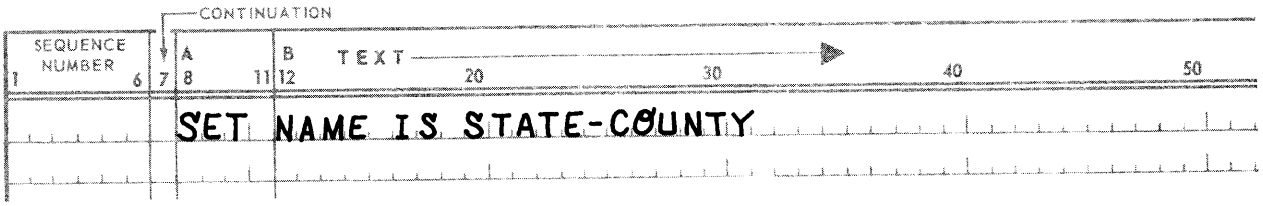
- The set-name must be unique among all set-names within the schema.
- Set-name–1 cannot exceed 30 characters.

Data Definition Guides: None

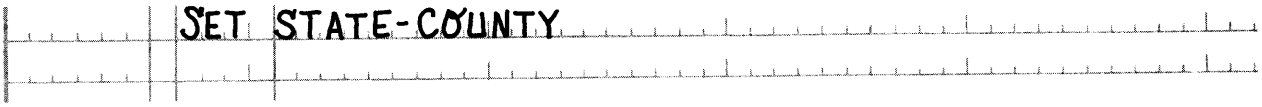
Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**



- **Example 2:**



3.5.3. SET CODE Clause

Function:

To specify a value for the set-name to be used by DMS 1100 as an internal shorthand notation for the set-name.

Syntax Skeleton:

SET CODE **IS** *integer–1*

Syntactical Constraints:

Integer–1 must be between 1 and 4000 inclusive.

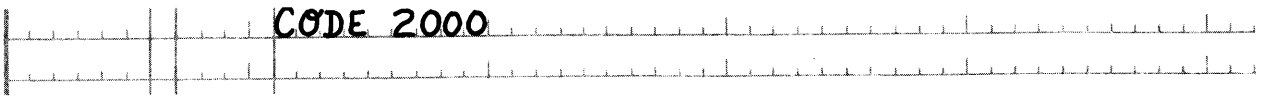
Data Definition Guides: None

Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**

- **Example 2:**



3.5.4. MODE Clause

Function:

To specify the mechanism to be utilized for supporting the manipulation of a set.

Syntax Skeleton:

MODE IS CHAIN [LINKED PRIOR]

Syntactical Constraints: None

Data Definition Guides:

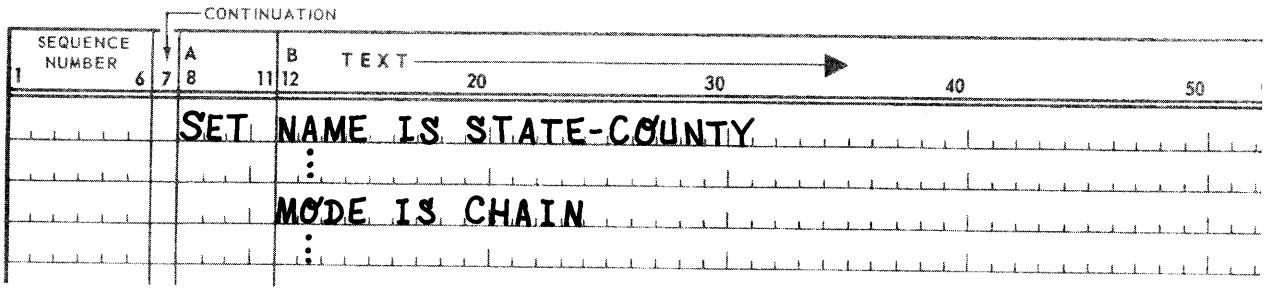
- MODE IS CHAIN results in the formation of a ring structure. Each owner record occurrence contains a forward pointer to its respective first member record occurrence, which in turn points to the second member record occurrence, etc.; the last member record occurrence contains a pointer to its respective owner record.

- The optional LINKED PRIOR clause adds reserve pointers to the ring structure, but does not otherwise alter it. Each owner record occurrence contains a pointer to its respective last member, which in turn contains a pointer to the second to the last member record occurrence, etc.; the first member record occurrence contains a pointer to its respective owner record.

Data Manipulation Guides: None

Examples and Discussion:

- **Example 1:**



Unbroken lines in the following STATE-COUNTY set occurrence (Figure 3–1) show the forward links which are always present.

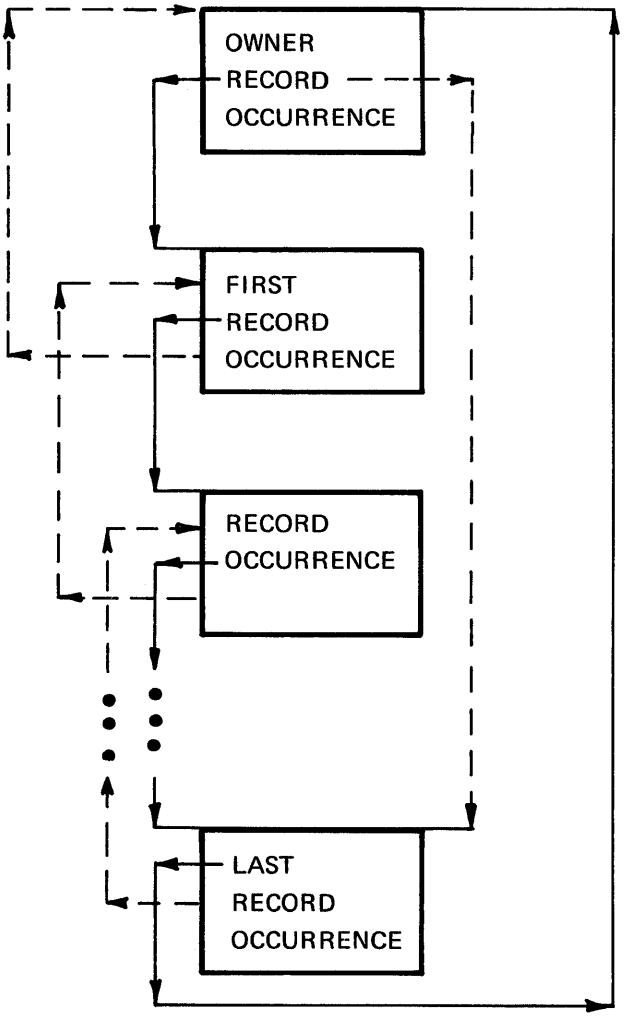
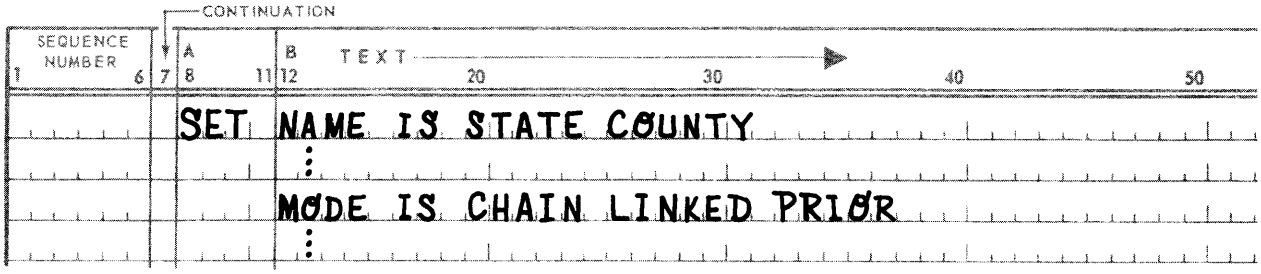


Figure 3–1. A Set Occurrence of State-County

- **Example 2:**



Broken lines in Figure 3–1 show additional pointers for prior links.

3.5.5. ORDER Clause

Function:

To specify the insertion point of a member record occurrence within a set occurrence, thereby defining the order of sequential progression.

Syntax Skeleton:

```
ORDER IS { FIRST                              }
         { LAST                               }
         { NEXT                               }
         { PRIOR                              }
         { SORTED [WITHIN RECORD-NAME]        }
```

Syntactical Constraints:

- The ORDER clause must be specified.
- If the ORDER is NEXT or PRIOR, the SET OCCURRENCE SELECTION clause for all member subentries must specify the CURRENT OF SET option.
- If the ORDER IS SORTED, an ASCENDING/DESCENDING KEY clause must be specified for all member subentries.

Data Definition Guides: (See Figure 3–2.)

- ORDER FIRST refers to the position within the set occurrence that immediately follows the OWNER record occurrence. This is a reversed chronological sequencing; the next MEMBER record 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 new MEMBER record occurrence becomes the last member of the set occurrence.

- ORDER PRIOR/NEXT refers to insertion points relative to the current record of the set. If the current record of a given set-name is not know, no new member records can be inserted into an occurrence of that set-name. If the CURRENT record of the set is the OWNER record, ORDER PRIOR is equivalent to ORDER LAST and ORDER NEXT is equivalent to ORDER FIRST.

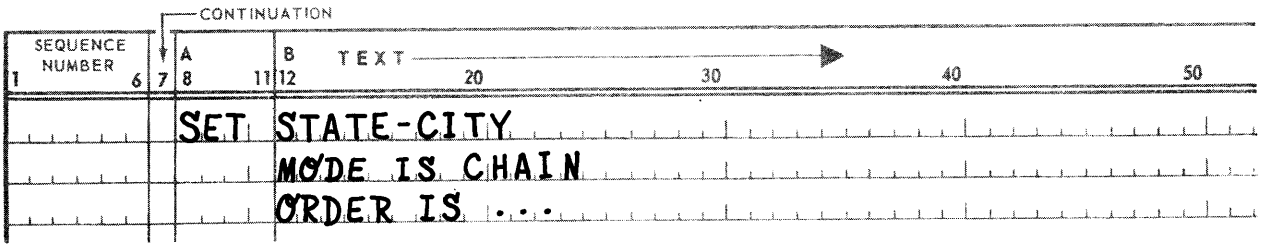
- ORDER SORTED allows specification of a set order based on the values of the sort control data-items, specified in the ASCENDING/DESCENDING KEY clauses, for the member records of the set.

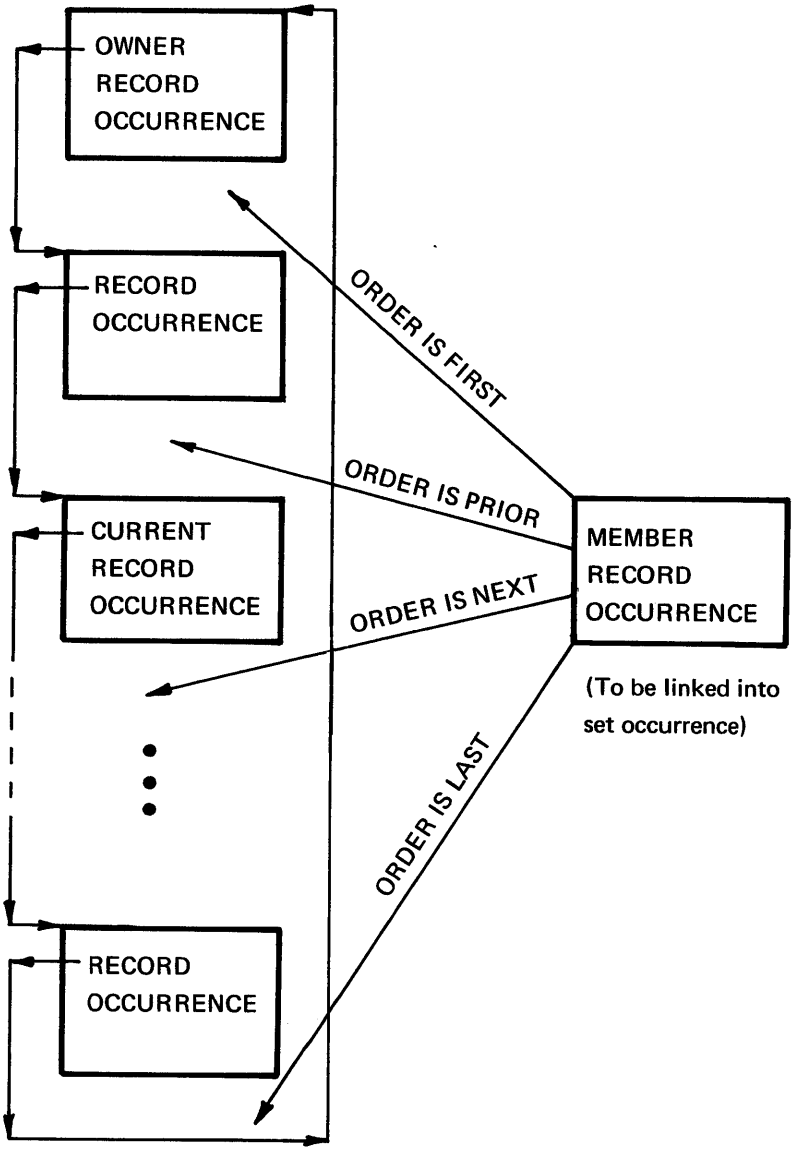
- The optional WITHIN RECORD-NAME clause allows records to be sorted without regard to the ASCENDING/DESCENDING KEY sort control data-items of other member record types in the set. A major sort by record type does not occur; thus record occurrences of different record types may be intermixed. However, any given member record type is considered independently of all other member record types; it is in sequence by its own sort key(s).

- If the ORDER IS LAST and the MODE IS CHAIN, each OWNER record occurrence automatically contains a pointer to the last member record of its respective set occurrence.

Data Manipulation Guides: None

Example and Discussions:





NOTE:

Diagrams for ORDER IS SORTED and ORDER IS SORTED WITHIN RECORD-NAME are included after the ASCENDING/DESCENDING KEY clause.

Figure 3–2. Order Record Occurrence

3.5.6. OWNER Clause

Function:

To specify the name of a record, each occurrence of which establishes the existence of a set occurrence of the set named in this set entry.

Syntax Skeleton:

<u>OWNER</u> IS *record-name–1*

Syntactical Constraints:

Record-name–1 must have been previously defined by a record entry.

Data Definition Guides:

A record type may be defined as an owner in more than one set entry, and as a member in one or more set entries.

Data Manipulation Guides:

When a record is stored, an occurrence of all sets of which the record is defined as an owner is established. The record is linked into an occurrence of any sets in which the record has been defined as an AUTOMATIC member. However, a record must be INSERTed into an occurrence of a set in which it has been defined as a MANUAL member. Storage of a record does not link a record into an occurrence of a set in which the record's membership is MANUAL.

Examples and Discussion:

- **Example 1:**

```
SET NAME IS S1
      :
    OWNER IS R1
    MEMBER IS R2  AUTOMATIC
      :
SET NAME IS S2
      :
    OWNER IS R2
    MEMBER IS R3  AUTOMATIC
      :
SET NAME IS S3
      :
    OWNER IS R3
    MEMBER IS R4  AUTOMATIC
      :
```

- **Example 2:**

```
SET NAME IS S1
  :
    OWNER IS R1
    MEMBER IS R2
  :
SET NAME IS S2
  :
    OWNER IS R1
    MEMBER IS R2
  :
SET NAME IS S3
  :
    OWNER IS R3
    MEMBER IS R1
  :
SET NAME IS S4
  :
    OWNER IS R4
    MEMBER IS R1...
  :
```

3.5.7. MEMBER Clause

Function:

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

Syntax Skeleton:

<u>MEMBER</u> IS *record-name–2* { <u>AUTOMATIC</u> / <u>MANUAL</u> } [LINKED TO <u>OWNER</u>]

Syntactical Constraints:

- Record-name–2 must be previously defined in a Record Entry.
- Record-name–2 cannot be the name of the record specified in the OWNER clause of this Set Entry.

Data Definition Guides: (See Figure 3–3.)

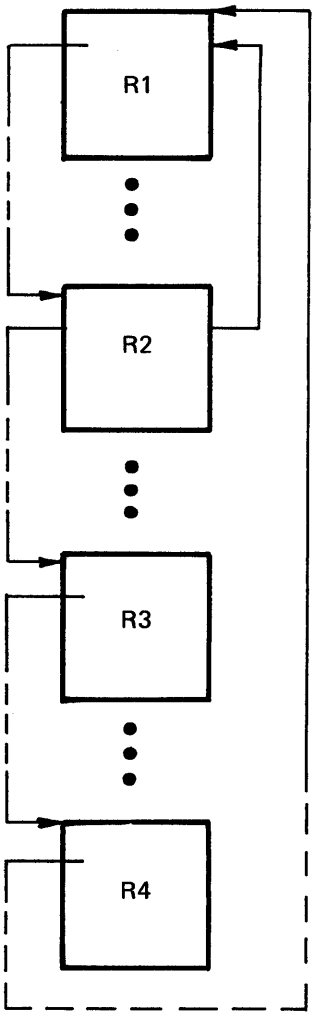
- The optional LINKED TO OWNER clause will force each member record occurrence of this type to contain the database-key of its OWNER.
- More than one record type can be declared as a member of any given set.
- The maximum number of member record types for a given set type is 63.
- The maximum number of set types in which a record type can be defined as a manual member is 36.
- A MEMBER clause must be specified for each record type that can participate as a member in the set being described.
- A record may be defined as a member in more than one set. It may also be defined as an owner in one or more sets.
- Each occurrence of a member record type participates in only one occurrence of a given set type. That is, for one set type, 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 sets, viz, such that a cycle is formed in which all records participate as automatic members in the sets included in the cycle.

Data Manipulation Guides:

- If the optional word AUTOMATIC is used, an occurrence of record-name–2 is inserted unconditionally into the selected occurrence of the set when an occurrence of record-name–2 is stored.
- If the optional word MANUAL is used, occurrences of record-name–2 may be either inserted into, or removed from the appropriate set occurrences by the execution of the INSERT and REMOVE commands, respectively.

Examples and Discussion:

```
SET NAME IS S1 CODE IS 40
    MODE IS CHAIN
       :
    OWNER IS R1
    MEMBER IS R2 MANUAL LINKED TO OWNER
       :
    MEMBER IS R3 AUTOMATIC
       :
    MEMBER IS R4 MANUAL
       :
```



R2 is the only member record type, other than the last record occurrence in S1, to contain a pointer to the owner, R1.

Figure 3–3. Set Occurrence of S1

3.5.8. ASCENDING/DESCENDING Clause

Function:

- 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.

Syntax Skeleton:

```
[ { ASCENDING  } [RANGE] KEY IS { RECORD-NAME                          } ... ]
  { DESCENDING }                { data-base-identifier–1                }
                                { [ ,data-base-identifier–2] ...        }

DUPLICATES ARE { FIRST           }
               { LAST            }
               { NOT ALLOWED     }
```

Syntactical Constraints:

- All data-base-identifiers must refer to data items specified in the Record Entry for the record named in the MEMBER clause of this subentry.
- This clause must be used only when the set ORDER is SORTED and then must be used for all member record subentries in the set.
- RECORD-NAME can be used only in one repetition of this clause.

Data Definition Guides:

- The order in which the keys are specified defines the major to minor sequence for sorting.
- The key collating sequence and key comparision is according to COBOL comparison rules.
- 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 intermixed.
- If the set ORDER is SORTED, and the optional WITHIN RECORD-NAME clause is not used, sort keys for all member records in the set must be comparable. Therefore, the ASCENDING/DESCENDING clause(s) must be identical for all member record types, except that the data-base-identifiers specified for each record type are unique to that record type. However, each data-base-identifier for a member record must have the identical PICTURE and USAGE clause describing it as its corresponding data-base-identifier in the other member record types.
- If the set ORDER is SORTED WITHIN RECORD-NAME, then the ascending/descending keys can be defined independently for each member record type. That is, the PICTURE and USAGE clauses need not agree.
- If the DUPLICATES ARE NOT ALLOWED clause is used, the insertion into any given set occurrence of member record occurrences with duplicate values for the specified ascending/descending keys will be rejected. 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 a data item specified in an ASCENDING or DESCENDING KEY clause.
- The DUPLICATES ARE NOT ALLOWED clause must be specified if this member record type has a LOCATION MODE of via set-name and is an OWNER of a set, the members of which have a SET OCCURRENCE SELECTION clause of LOCATION MODE OF OWNER and no USING or ALIAS clause. This allows the DMR to select a unique set occurrence at intermediate levels of a hierarchy.

- If either 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 before or after (as specified) any existing member occurrences with such duplicate values.

- If the optional word RANGE is used, the ordering of the member record occurrences is in accordance with the previous (8) Data Definition Guides.

- The use of the RANGE option causes each occurrence of a RANGE KEY to represent a range of values in accordance with the following rules.

  (a) 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.

  (b) A match will occur regardless of whether the RANGE KEY has been specified as ASCENDING or DESCENDING, as follows:

  - If the input argument value in working storage equals the value of any specific RANGE KEY value.

  - If the input argument value in working storage is stored between two adjacent RANGE KEY values, the match will occur with the RANGE KEY value which is the larger value.

  - If the input argument value in working storage is less than the lowest value of any RANGE KEY, then a match will occur on the lowest-valued RANGE KEY.

  (c) A match will not occur if the input value in working storage is greater than the largest value of any RANGE KEY.

- If RECORD-NAME is specified as the key in either an ASCENDING or DESCENDING clause, the record code of the member record is used as a sort key. (The record code is assigned by the CODE clause in a record definition.)

Examples and Discussion:

■ **Example 1:**

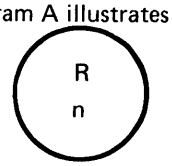
```
SET NAME S1 CODE 1
    MODE IS CHAIN
    ORDER IS "see example"
    OWNER IS RO
      :
    MEMBER IS R1···
        ASCENDING KEY IS R1-KEY
        DUPLICATES···
      :
    MEMBER IS R2···
        ASCENDING KEY IS R2-KEY
        DUPLICATES···
      :
    MEMBER IS R3···
        ASCENDING KEY IS R3-KEY
        DUPLICATES···
      :
```

Assume record occurrences were submitted for linkage into an occurrence of set S1 in the following order:

R0, owner, establishes the set occurrence
R3, R3–KEY = 4
R2, R2–KEY = 7
R1, R1–KEY = 2
R2, R2–KEY = 9
R2, R2–KEY = 6
R1, R1–KEY = 5
R1, R1–KEY = 1
R3, R3–KEY = 8

Diagram A illustrates set S1 when ORDER IS SORTED.

Key:



where R is the record type, and n is the value of the key.

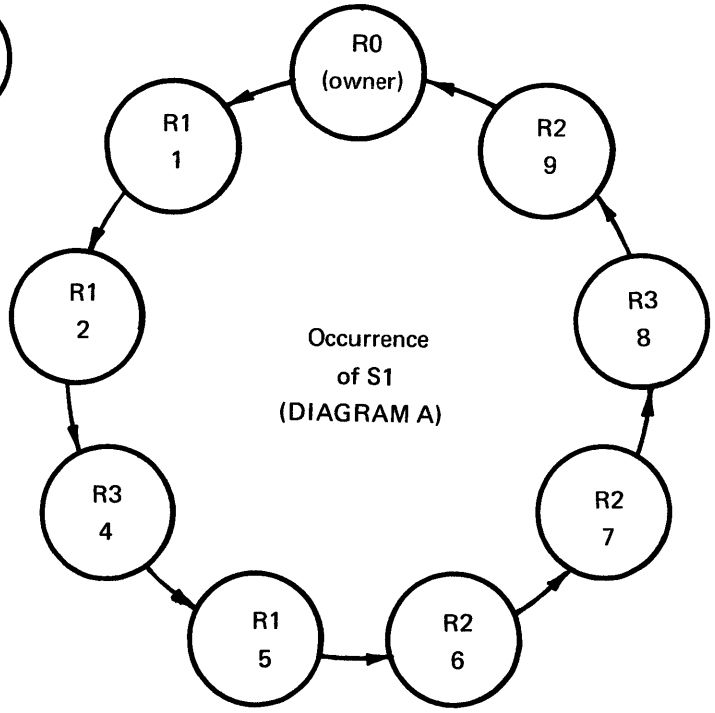


Diagram B illustrates set S1 when ORDER IS SORTED WITHIN RECORD-NAME.

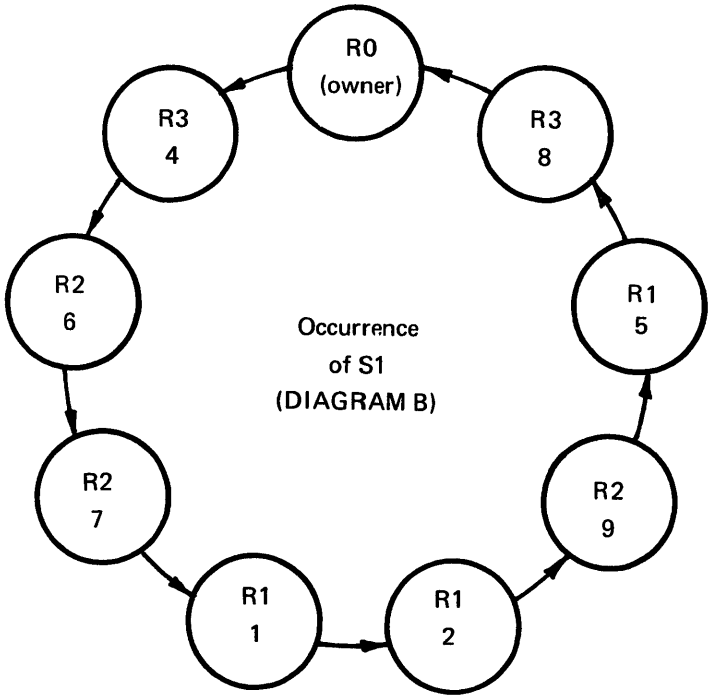
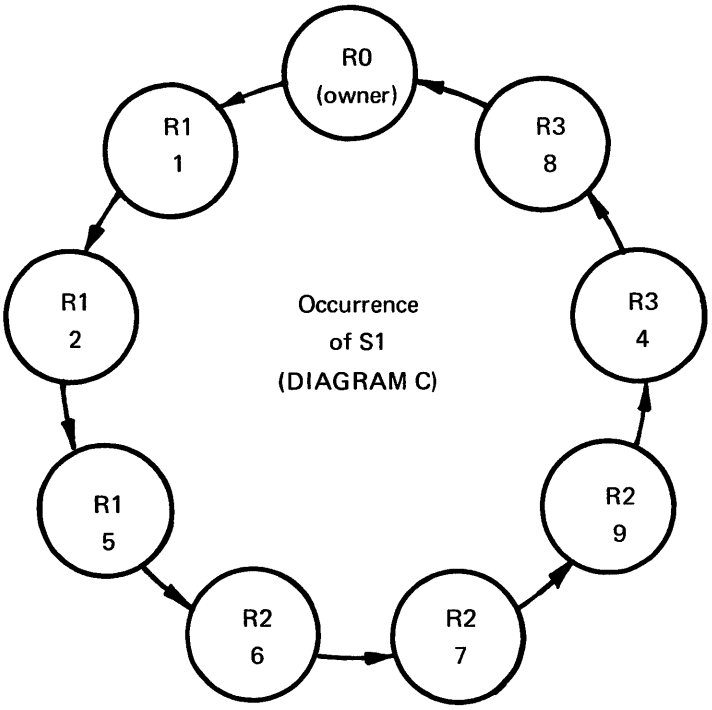


Diagram C illustrates set S1 when ORDER IS SORTED, and each member entry in S1 contains the clause ASCENDING KEY IS RECORD-NAME immediately preceding the ASCENDING clause in each member entry in Diagram B. Assume the record codes are 1, 2, and 3 for R1, R2, and R3, respectively. R1–KEY, R2–KEY, and R3–KEY must have identical PICTURE and USAGE clauses.



- **Example 2:**

```
SET NAME S1 CODE
        :
        ORDER IS SORTED
        :
        MEMBER IS R1
            ASCENDING KEY IS R1-AKEY, R1-BKEY
            DESCENDING KEY IS R1-CKEY
            ASCENDING KEY IS R1-DKEY
            DUPLICATES...
        :
        MEMBER IS R2
            ASCENDING KEY IS R2-AKEY, R2-BKEY
            DESCENDING KEY IS R2-CKEY
            ASCENDING KEY IS R2-DKEY
            DUPLICATES...
        :
```

R1—AKEY and R2—AKEY must have identical PICTURE and USAGE clauses, as must R1—BKEY and R2—BKEY, R1—CKEY and R2—CKEY, R1—DKEY and R2—DKEY.

- **Example 3:**

The following example illustrates the use of the RANGE option on the ASCENDING/DESCENDING clause to simulate indexed sequential record processing.

To anticipate facility and manpower in advance, a certain warehouse manager wishes to store and select material shipment records on the basis of the shipment due-in date. The warehouse manager wants records saved over a ten-year period, and desires the use of only one key to access a record.

The Data Administrator wants to simulate an indexed sequential system using the Data Management System. He must enable an application programmer to store or select a material record using one key — the shipment due date. (With respect to the remainder of the discussion, refer to Figure 3—4 and record and set definition on the following pages.)

Using the concept of sets, the Data Administrator decides to build a three-level index by first dividing the ten-year period into ten one-year periods. A CALC procedure will be used to store each of the ten occurrences of record type YEAR-INDEX. Each of the one-year periods is divided into four quarters. Each of the ten occurrences of set type YEAR-QUARTER will have four member occurrences of record type QUARTER-INDEX. The proper occurrence of set type YEAR-QUARTER in which a specific occurrence of record type QUARTER-INDEX is to be linked will be determined by locating the proper YEAR-INDEX via the CALC procedure DBKROU, using the year portion of the QUARTER-INDEX date.

Each quarter of a one-year period is divided into thirds of a month. Each of the forty occurrences of set type QUARTER-THIRD-MONTH has nine member occurrences of record type THIRD-MONTH-INDEX. The proper occurrence of set type QUARTER-THIRD-MONTH into which an occurrence of record type THIRD-MONTH-INDEX is to be linked is determined in the following manner:

- The year portion of the record occurrence THIRD-MONTH-INDEX is used by the CALC procedure DBKROU to determine the proper occurrence of record type YEAR-INDEX, the owner type of set type YEAR-QUARTER set. The entire date of the record occurrence of THIRD-MONTH-INDEX can be used to uniquely identify the proper occurrence of record type QUARTER-INDEX, since occurrences of QUARTER-INDEX are sorted by date within an occurrence of set type YEAR-QUARTER.

- The RANGE option is specified since an equality match between the RANGE KEY in the QUARTER-INDEX record to be selected and the input argument value in the USER WORKING AREA (the date of record occurrence THIRD-MONTH-INDEX) is not required for the proper QUARTER-INDEX record to be selected.

NOTE:

The OVERLAY statement of the Data Management Language could be used to overlay DATE-DUE, DATE–1, DATE–2, and DATE–3. Use of the OVERLAY statement effectively causes the input arguments for the RANGE keys of all record types in the example to be initialized when the input argument for the RANGE key of one record type is initialized. Since the Data Administrator has constructed an index of three levels, he may want to store each level of the index in a different area, or two levels of the index in one area with the third level and the MATERIAL-DUE records being stored in a second area, etc.

Assume that the index and all MATERIAL records currently available are stored. The warehouse manager asks the application programmer to determine if any shipments are due on March 17, 1972. DATE–3 must be initialized with 031772. Assuming the OVERLAY capability is used, DATE–2, DATE–1, and DATE-DUE address the same date, 031772. The YEAR portion of DATE-DUE is used by the CALC procedure DBKROU to obtain the database-key for YEAR-INDEX 123172.

- Within the set occurrence of YEAR-QUARTER (determined by YEAR-INDEX 123172), RANGE key DATE–1, initialized with 031772, is used to obtain the 033172 occurrence of record type QUARTER-INDEX.

- Within the set occurrence of QUARTER-THIRD-MONTH (determined by QUARTER-INDEX 033172), RANGE key DATE–2, initialized with 031772, is used to obtain the 032172 occurrence of record type THIRD-MONTH-INDEX.

- Within the set occurrence of MATERIAL-DUE (determined by THIRD-MONTH-INDEX 033172), RANGE key DATE–3, initialized with 031772, is used to determine an occurrence of MATERIAL record type with a shipment due date of March 17, 1972. If no MATERIAL records due date 031772 exists, a match will be made on the MATERIAL record with a RANGE key value greater than 031772.

  If there are no MATERIAL records in the set occurrence of MATERIAL-DUE for which DATE–3 in the record occurrence has a value greater than or equal to DATE–3 in WORKING-STORAGE, no match is made. If an exact match is made on 031772, the application programmer can easily obtain the next MATERIAL record, etc., until all MATERIAL records with due date 031772 are obtained.

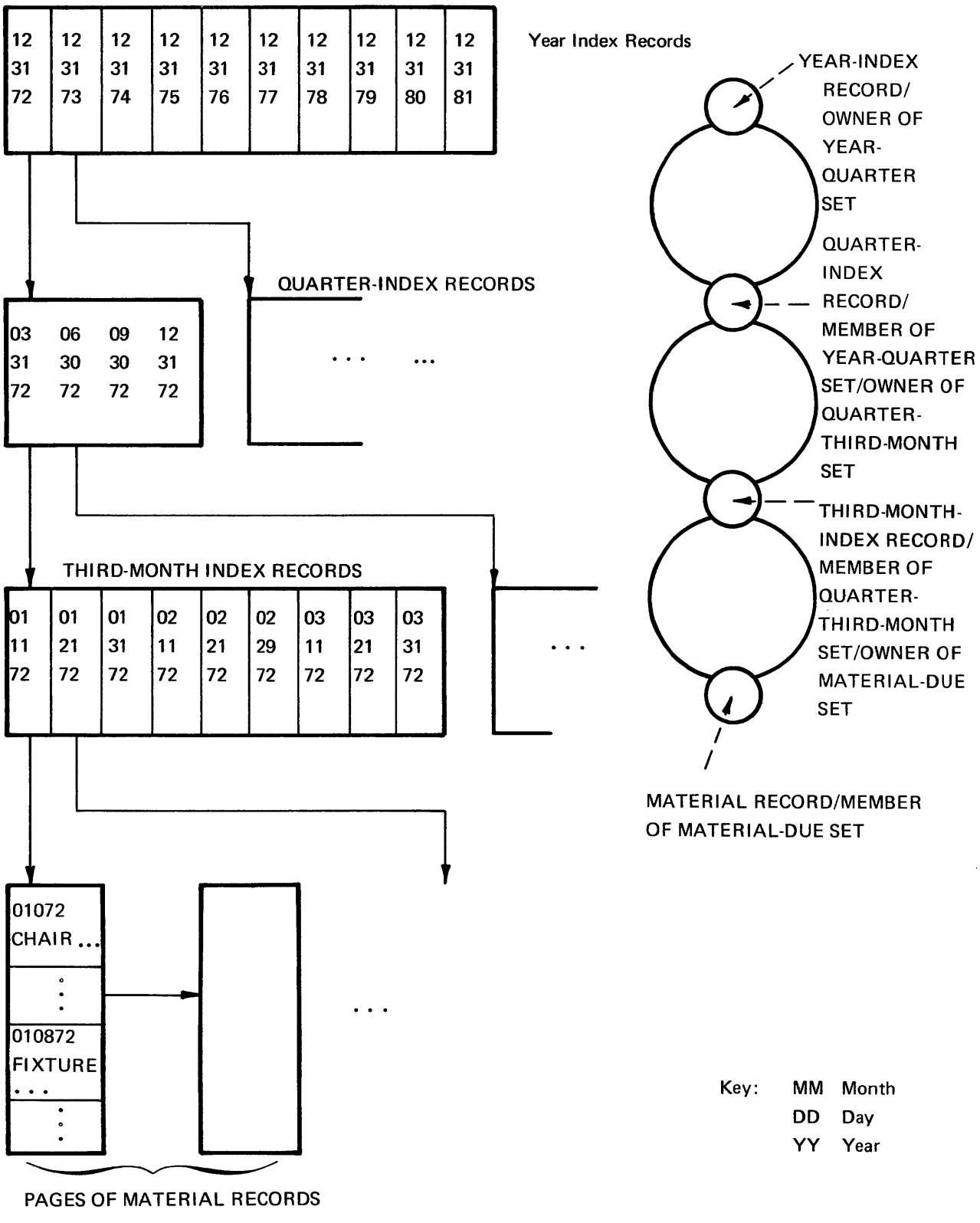


Figure 3–4. Index Records

The record and set descriptions could appear as follows:

```
RECORD NAME IS YEAR-INDEX CODE IS 1
    LOCATION MODE IS CALC DBKROU IN AREA1D
        USING YEAR DUPLICATES ARE NOT ALLOWED
    WITHIN AREA1
    02  DATE-DUE
        03  MONTH    PIC X(2)
        03  DAY      PIC X(2)
        03  YEAR     PIC X(2)
RECORD NAME IS QUARTER-INDEX CODE IS 2
    LOCATION MODE IS VIA YEAR-QUARTER SET
    WITHIN AREA1
    02  DATE-1
        03  MONTH1   PIC X(2)
        03  DAY1     PIC X(2)
        03  YEAR1    PIC X(2)
RECORD NAME IS THIRD-MONTH-INDEX CODE IS 3
    LOCATION MODE IS VIA QUARTER-THIRD-MONTH SET
    WITHIN AREA1
    02  DATE-2
        03  MONTH2   PIC X(2)
        03  DAY2     PIC X(2)
        03  YEAR2    PIC X(2)
RECORD NAME IS MATERIAL CODE IS 4
    LOCATION MODE IS VIA MATERIAL-DUE SET
    WITHIN AREA2
    02  DATE-3
        03  MONTH3   PIC X(2)
        03  DAY3     PIC X(2)
        03  YEAR3    PIC X(2)
    02  MATERIAL-DATA
        03  MATERIAL-NAME    PIC X(12)
        03  PART-NUMBER      PIC X(6)
        03  QUANTITY         PIC 9(10)
        03  DESCRIPTION      PIC X(30)
SET NAME IS YEAR-QUARTER SET CODE IS 1
    MODE IS CHAIN
    ORDER IS SORTED
    OWNER IS YEAR-INDEX
    MEMBER IS QUARTER-INDEX  AUTOMATIC
    ASCENDING RANGE KEY IS DATE-1
        DUPLICATES ARE NOT ALLOWED
    SET OCCURRENCE SELECTION IS THRU LOCATION
    MODE OF OWNER
SET NAME IS QUARTER-THIRD-MONTH SET CODE IS 2
    MODE IS CHAIN
    ORDER IS SORTED
    OWNER IS QUARTER-INDEX
    MEMBER IS THIRD-MONTH-INDEX  AUTOMATIC
    ASCENDING RANGE KEY IS DATE-2
        DUPLICATES ARE NOT ALLOWED
    SET OCCURRENCE SELECTION IS THRU LOCATION
    MODE OF OWNER
SET NAME IS MATERIAL-DUE SET CODE IS 3
    MODE IS CHAIN
    ORDER IS SORTED
    OWNER IS THIRD-MONTH-INDEX
    MEMBER IS MATERIAL  AUTOMATIC
    ASCENDING RANGE KEY IS DATE-3
        DUPLICATES ARE ALLOWED
    SET OCCURRENCE SELECTION IS THRU LOCATION
    MODE OF OWNER
```

3.5.9. SET OCCURRENCE SELECTION Clause

Function:

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.

Syntax Skeleton:

Format—1:

```
SET OCCURRENCE SELECTION IS THRU
{CURRENT OF SET                                                                    }
{LOCATION MODE OF OWNER                                                            }
 [{ USING data-base-identifier—3[,data-base-identifier—4]...                       }]
 [{{ALIAS{FOR data-base-identifier—5} IS data-base-data-name—2 }...                }]
 [{      {OF  data-base-data-name—1 }                                              }]
```

Format—2:

```
;SET OCCURRENCE SELECTION IS THRU set-name—2 USING
{CURRENT OF SET                                                                    }
{LOCATION MODE OF OWNER                                                            }
 [{ USING data-base-identifier—6[,data-base-identifier—7]...                       }]
 [{{ALIAS{FOR data-base-identifier—8 } IS data-base-data-name—2 }...               }]
 [{      {OF  data-base-data-name—1 }                                              }]
  { set-name—3                                                                     }
  {{ USING data-base-identifier—9 [,data-base-identifier—10]...                    }
  {{{ALIAS FOR data-base-identifier—11 IS data-base-data-name—3}...                }}...
```

Syntactical Constraints:

- All data-base-identifiers must refer to declared data items of the OWNER record of the set(s) referenced. All data-base-data-names refer to 77- or 01-level entries.
- 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 SET.
- An ALIAS clause may qualify the LOCATION MODE OF OWNER clause regardless of the LOCATION MODE clause in the Record Entry for the owner record of the set in which this clause appears.

  (1) If the LOCATION MODE IS DIRECT the data-base-data-names being substituted for must refer to data-base-data-names specified on the LOCATION MODE IS DIRECT clause.

  (2) If the LOCATION MODE IS CALC, the data-base-data-name being substituted for (if specified) must be the data-base-data-name specified on the IN clause. The data-base-identifier being substituted for (if specified) must be the data-base-identifiers specified on the USING clause and a substitution must be made for each. The data-base-identifiers being substituted for must be defined as elementary items in the Record Entry and their substitutions must be 77-level entries in working storage.

(3) If the LOCATION MODE IS VIA set-name SET, the data-base-identifiers specified must refer to:

(a) Data items specified in the USING clause of a SET OCCURRENCE SELECTION clause for another set with the same defined owner record type or

(b) To the ASCENDING/DESCENDING key items specified for the set in which the owner record is a member (if the USING clause is omitted).

- Set-name–2, set-name–3... must form a continuous path, where the owner of set-name–3 is a member of set-name–2..., with set-name–2 as a start point or root. The same set-name cannot appear more than once. The last set named must be the subject of the Set Entry of which this clause is a part.

- 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. If the USING clause is omitted, and its LOCATION MODE IS VIA set-name SET and the ORDER is SORTED, then the ASCENDING/DESCENDING KEY items in the Set Entry must also have a DUPLICATES NOT ALLOWED clause.

Data Definition Guides:

- *Format–1* 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 determined on the basis of its LOCATION MODE clause (the LOCATION MODE OF OWNER option). If the owner record LOCATION MODE clause specifies either DIRECT or CALC, the owner record is uniquely identified. If the owner record LOCATION MODE clause specifies VIA set-name SET, the owner record must be determined in terms of its membership in some other set.

- If *Format–1* is specified, and the CURRENT OF SET option is used, the set occurrence selected is determined by the current record of the set-name of which this clause is a part. This record must be procedurally pre-selected.

- If *Format–1* is specified and the LOCATION MODE OF OWNER option is used, the set occurrence selected is determined on the basis of whether a USING or ALIAS option is also present and on the basis of the LOCATION MODE clause specified in the Record Entry for the owner of the set-name of which this SET OCCURRENCE SELECTION clause is a part.

(1) If the LOCATION MODE of the owner record is DIRECT, the owner record occurrence can be uniquely identified.

– If either USING or ALIAS option is not present, the owner record occurrence is uniquely identified by the contents of the data items specified on the DIRECT option.

– If a USING option is present, it is ignored, since the only data items which could be meaningfully specified are those already specified on the DIRECT option.

– If an ALIAS option is present, the original working storage data items specified on the DIRECT option are to be replaced by substitute working storage data items. Substitution may be used for one or both of the original data items.

(2) If the LOCATION MODE of the owner record is CALC, the owner record can be uniquely identified.

– If either USING or ALIAS option is not present, the owner record is uniquely identified by the CALC routine based upon the contents of the data items specified on the CALC option.

- If a USING option is present, it is ignored, since the only data items which could meaningfully be specified are those already specified on the CALC option.

- If an ALIAS option is present, the original data items specified on the CALC option are to be replaced by substitute working storage data items. Substitution may be made for the working storage data item designated to contain the area-name and/or all of the data items input to the CALC routine.

(3) If the LOCATION MODE of the owner record is VIA set-name SET, the owner record can be uniquely identified only in terms of its membership in another set. This selection is governed by the SET OCCURRENCE SELECTION clause (either *Format–1* or *Format–2*) for 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 SET. This condition may occur to an arbitrary number of upward levels, but must eventually terminate with a SET OCCURRENCE SELECTION clause (either *Format–1* or *Format–2*) that does not specify LOCATION MODE OF OWNER where the LOCATION MODE IS VIA set-name SET.

- If either USING or ALIAS option is not present, the owner record occurrence of that level is uniquely identified by the contents of the ASCENDING/DESCENDING KEY data items if specified for the owner record in its capacity as a member subentry of the Set Entry specified on The VIA set-name SET option.

- If a USING option is present, the owner record occurrence of that level is uniquely identified by the contents of the data items specified.

- If an ALIAS option is present, the owner record occurrence of that level is uniquely identified by the contents of the working storage data items specified as substitutes for the control data items of the owner record.

- *Format–2* applies where the immediate owner record of the set occurrence to be selected cannot itself be uniquely selected, except in terms of its membership in another set, and, the criteria for selecting that set are included in the SET OCCURRENCE SELECTION clause. This condition may occur to an arbitrary number of levels forming a continuous path from a starting point of the specific owner of set-name–2 to the specific owner of the set-name of which the SET OCCURRENCE SELECTION clause is a part.

- If *Format–2* is specified and the CURRENT OF SET option is used, the set-name–2 occurrence selected is determined by the current record of set-name–2. This record must be procedurally pre-selected.

- If *Format–2* is specified and the LOCATION MODE OF OWNER option is used, the set occurrence of set-name–2 selected is determined on the basis of whether a USING or ALIAS option is also present, and on the basis of the LOCATION MODE clause specified in the Record Entry for the owner of set-name–2. The selection of the owner record occurrence of set-name–2 for *Format–2* is identical to the selection of the owner record occurrence of this Set Entry for *Format–1* as previously discussed in the Data Definition Guide.

- If *Format–2* is specified and the LOCATION MODE OF OWNER option is used, specific occurrences of each of the sets named must be selected. The occurrence of set-name–1 is selected on the basis of the initialized values for its owner record, for example data-base-identifier–9... as specified on the USING option. The occurrences of all subsequent sets named are also selected on the basis of the initialized values for their owner records. The LOCATION MODE clauses in the respective record entries are ignored. If necessary, the ALIAS option may be used to substitute working storage data items for the data items within the record. However specified, in each case the values of the data items are used to uniquely select the owner record occurrence of

the named set in its capacity as a member of the previously named set.

- 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 set type has the same owner record type. In this situation, more than one value may be required for the data item. The ALIAS clause provides substitute working storage locations for these values.

Data Manipulation Guides:

- All data-base-data-names appearing on ALIAS clause must be defined as working storage data items. They must have the same PICTURE and USAGE characteristics as the data items for which they are substitutes.

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

  (1) *Format–5* of the FIND command is executed and the LOCATION MODE of the object record is VIA set-name SET.

  (2) *Format–6* of the FIND command is executed.

  (3) A STORE command is executed and the LOCATION MODE of the object record is VIA set-name SET or the object record is an automatic member of one or more sets.

  (4) A MODIFY command is executed which causes the object record to be removed from its present set occurrence(s) and inserted into a different set occurrence(s).

Examples and Discussion:

- **Example 1:**

```
RECORD NAME IS COUNTY-RECORD
           .
           LOCATION MODE IS VIA STATE-COUNTY SET
           .
           02  COUNTY-NAME PIC IS X(15)
           .
SET        NAME IS COUNTY-CITY SET
           .
           OWNER IS COUNTY-RECORD
           .
           MEMBER IS CITY-RECORD
           .
           SET OCCURRENCE SELECTION IS THRU LOCATION
           MODE OF OWNER USING COUNTY-NAME
           .
```

In order to either store or select an occurrence of record type CITY-RECORD, the proper occurrence of set COUNTY-CITY must be determined. The Data Administrator has specified that the proper occurrence of COUNTY-CITY SET can be determined through the location mode of the owner record type COUNTY-RECORD, using the key COUNTY-NAME. The location mode of record type COUNTY-RECORD is via STATE-COUNTY set. The location mode of COUNTY-RECORD should not be DIRECT or CALC since the USING option was specified, and DIRECT or CALC assumes all meaningful data has already been specified. Because the location mode of COUNTY-RECORD is via STATE-COUNTY set, the proper occurrence of STATE-COUNTY set must be determined. As progression is made upward in the storage hierarchy, a level must be reached in which the SET OCCURRENCE clause encountered specifies CURRENT of SET, or LOCATION MODE OF OWNER where the location mode of the owner is DIRECT or CALC.

For example, suppose the owner record type of STATE-COUNTY SET is a record type STATE, whose location mode is DIRECT. Suppose a record of type CITY-RECORD is to be stored or selected for Ramsey County in the state of Minnesota. The specific occurrence of owner record type STATE for Minnesota can be located directly. Thus the set occurrence of STATE-COUNTY set, in which the record occurrence for RAMSEY County is a member, can be determined. Using COUNTY-NAME, whose value has been initialized to RAMSEY, the proper occurrence of record type COUNTY-RECORD can be determined. Therefore, the set occurrence of type COUNTY-CITY, in which CITY-RECORD type is a member, can be determined. The desired occurrence of CITY-RECORD can now be determined through conventional means.

- **Example 2:**

```
SEQUENCE NUMBER 1-6 | CONTINUATION 7 | A 8-11 | B 12  TEXT  20  30  50

       RECORD NAME IS DEPARTMENT
           :
           LOCATION MODE IS DIRECT DATA-KEY1 , DATA-KEY2
           :
       SET NAME IS DEPARTMENT-EMPLOYEES
           :
           OWNER IS DEPARTMENT
           :
           MEMBER IS EMPLOYEE  AUTOMATIC
           :
           SET OCCURRENCE SELECTION IS THRU LOCATION
           MODE OF OWNER
           :
       SET NAME IS DEPARTMENT-TEMPORARY-EMPLOYEES
           :
           OWNER IS DEPARTMENT
           :
           MEMBER IS EMPLOYEE AUTOMATIC
           :
           SET OCCURRENCE SELECTION IS THRU LOCATION
           MODE OF OWNER
                 ALIAS OF DATA-KEY1 IS DATA-KEY3
                 ALIAS OF DATA-KEY2 IS DATA-KEY4
           :
```

Suppose that the EMPLOYEE, John Smith, is an engineer, and that he is on loan to the DEPARTMENT, Systems Programming. Suppose a record occurrence of type EMPLOYEE is to be stored for John Smith. The latter occurrence of type EMPLOYEE for John Smith will be a member of an occurrence of set DEPARTMENT-EMPLOYEES whose owner record type is DEPARTMENT, the specific occurrence of type DEPARTMENT being Engineering. The occurrence of record type EMPLOYEE for John Smith will be a member of a set occurrence of type DEPARTMENT-TEMPORARY-EMPLOYEES (whose owner record type is also DEPARTMENT), the specific occurrence, however, of type DEPARTMENT, being Systems Programming. DATA-KEY1 and DATA-KEY2, properly initialized before the storing of record type EMPLOYEE for John Smith, specify the information necessary to locate directly the proper occurrence of owner record type DEPARTMENT of set DEPARTMENT-EMPLOYEES, into which EMPLOYEE record type for John Smith is to be linked. DATA-KEY1 and DATA-KEY2 are used to locate the owner record type DEPARTMENT for the specific occurrence Engineering. However, a different occurrence of owner record type DEPARTMENT, namely the occurrence for Systems Programming, is the owner of the occurrence of set DEPARTMENT-TEMPORARY-EMPLOYEES into which record type EMPLOYEE for John Smith is to be linked. Since DATA-KEY1 and DATA-KEY2 cannot be initialized to locate directly two occurrences of owner record type DEPARTMENT, simultaneously, DATA-KEY3 and DATA-KEY4 are specified in the ALIAS clause as the locations in working storage which will be initialized to locate directly the proper occurrence of owner record type DEPARTMENT (Systems Programming) of a set occurrence of DEPARTMENT-TEMPORARY-EMPLOYEES.

- **Example 3:**

The following example illustrates the use of *Format–2* SET OCCURRENCE SELECTION. Figure 3–5 shows the relationship of the sets used in the example.

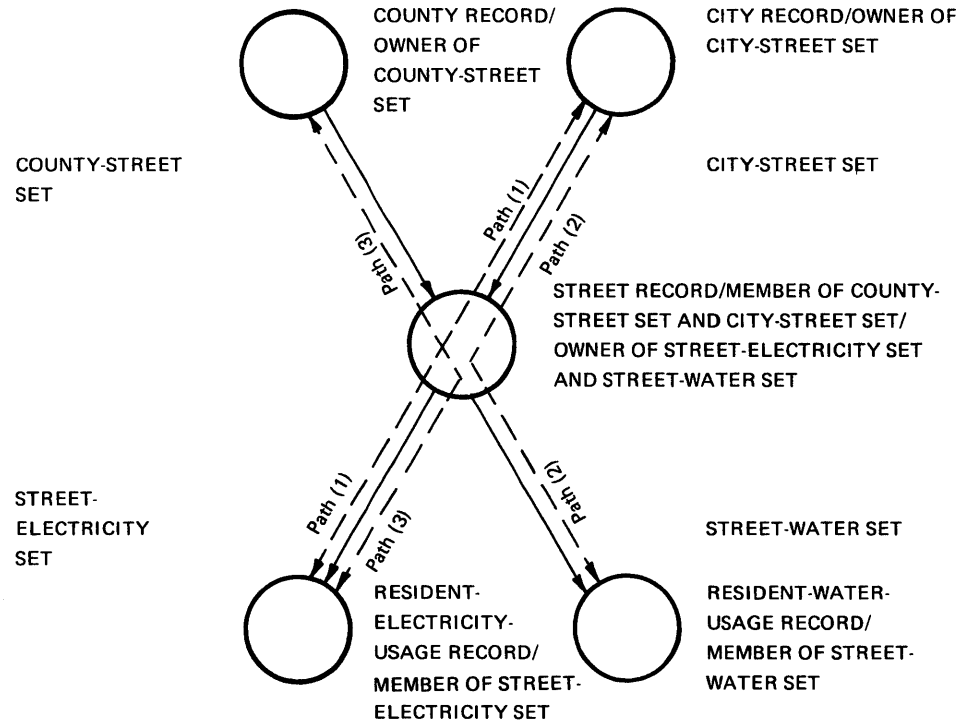


Figure 3–5. Set Occurrence Selection

Assume the location mode of both record types COUNTY and CITY is DIRECT and the location mode of STREET record type is VIA CITY-STREET. A store or select of record type RESIDENT-ELECTRICITY-USAGE in STREET-ELECTRICITY set type, or record type RESIDENT-WATER-USAGE in STREET-WATER set could use paths (1) or (2), respectively. Let us assume that electricity usage is more commonly required on a county basis.

Format–2 enables the Data Administrator to specify a path which will be used to either store or select an occurrence of record type RESIDENT-ELECTRICITY-USAGE such that the LOCATION MODE of STREET record type will not be used. Path (3) is defined below:

CONTINUATION
SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT 20 30 40 50

```
SET STREET-ELECTRICITY
    :
    OWNER IS STREET
    :
    MEMBER IS RESIDENT-ELECTRICITY-USAGE
    AUTOMATIC
    :
    SET OCCURRENCE SELECTION IS THRU COUNTY-
    STREET
        USING LOCATION MODE OF OWNER
        STREET-ELECTRICITY USING STREET-NAME
```

Assuming the previous SET OCCURRENCE SELECTION clause, the following locations must be initialized before path (3) can be used to either store or select an occurrence of record type RESIDENT-ELECTRICITY-USAGE.

The data-names specified on the LOCATION MODE IS DIRECT clause for record type COUNTY must be initialized, so the proper occurrence of COUNTY can be determined.

STREET-NAME, an item which must be defined within the owner record type STREET, must be initialized so the proper occurrence of STREET can be determined within the set occurrence of COUNTY-STREET, which was selected on the basis of owner record type COUNTY.

The proper occurrence of record type RESIDENT-ELECTRICITY-USAGE can then be determined on whatever basis is desired by the application programmer.

4. HIERARCHICAL STORAGE STRUCTURES

4.1. GENERAL

Any schema definition containing set entries contains a hierarchical storage structure; that is, it contains at least one owner-member relationship between two or more record types. These relationships may be as follows:

- A one-to-one relationship,
- A one-to-many relationship (tree structure),
- A many-to-one relationship (also a tree structure, although inverted), or
- A many-to-many relationship (a network structure).

These relationships may co-exist within a schema definition, and different types of relationships may appear at different levels of the hierarchy.

The existence of a hierarchical storage structure should not present special problems to either the Data Administrator or the Application Programmers. Record occurrences are selected on the basis of the values of key data-items contained within them. (Record occurrences may also be found based on currency.) Traditionally the key items might have been a concatenation of last name, house number, street, city, and state, all defined within the record type. In a hierarchical storage structure the key items remain the same; however, they are defined in different record types at different levels of the hierarchy. Thus, the state item might be contained in the record type defined to top the hierarchy; the city item might be contained in the record type defined to be at the first intermediate level; and the street item might be contained in the record type defined to be the second intermediate level. Finally the name and house number items would be defined in the lowest level record type. The point is that the key items and the results are the same. Only the concept of the storage structure is different.

The Data Administrator will become intimately familiar with the definition of hierarchical storage structures; however, the Application Programmer may remain relatively unconcerned. In fact, as long as the Data Administrator has provided the Application Programmer with the details as to which records to INVOKE and which data items to initialize as key data items, the Application Programmer may remain totally unconcerned.

The remainder of this section reviews in detail the definition of hierarchical storage structures, with emphasis on the key data items which must be initialized to ensure successful completion of a command which utilizes that structure. Following are the DDL clauses and DML commands involved. This is a discussion of the utilization of a hierarchical storage structure. The latter begins with a discussion of the use of a LOCATION MODE of DIRECT and CALC, which is included for completeness and because all hierarchies must eventually terminate (at the top) with a record type located either directly or through a CALC procedure. Examples for using several alternate hierarchical storage structures are provided.

The following DDL clauses are used to define a hierarchical storage structure. Note that related clauses, such as the WITHIN and INTERVAL clauses, are used to specify physical characteristics of the hierarchy and are considered in the following section, RECORD PLACEMENT STRATEGY.

```
LOCATION MODE IS { DIRECT data-base-data-name–1, data-base-data-name–2
                 { CALC data-base-procedure-name–1
                 {     IN data-base-data-name–3
                 {     USING data-base-identifier–1 [,data-base-identifier–2] ...
                 {     DUPLICATES ARE [NOT] ALLOWED
                 { VIA set-name–1 SET

ORDER IS { FIRST
         { LAST
         { NEXT
         { PRIOR
         { SORTED [WITHIN RECORD-NAME] ;

[{ {ASCENDING } [RANGE] KEY IS { RECORD-NAME                          } ... }
 { {DESCENDING}                { data-base-identifier–1               }     }
                               { [,data-base-identifier–2] ...        }

      DUPLICATES ARE { FIRST       } ] ;
                     { LAST        }
                     { NOT ALLOWED }
```

Format–1:

```
;SET OCCURRENCE SELECTION IS THRU
{ CURRENT OF SET          }
{ LOCATION MODE OF OWNER  }
   [{ USING data-base-identifier–3 [,data-base-identifier–4] ...                                    }
   [{ ALIAS { FOR data-base-identifier–5 } IS data-base-data-name–2 } ...                           } ]
            { OF  data-base-data-name–1  }
```

Format–2:

```
·SET OCCURRENCE SELECTION IS THRU set-name–2 USING
{ CURRENT OF SET          }
{ LOCATION MODE OF OWNER  }
   [{ USING data-base-identifier–6 [,data-base-identifier–7] ...                                    }
   [{ ALIAS { FOR data-base-identifier–8 } IS data-base-data-name–2 } ...                           } ]
            { OF  data-base-data-name–1  }
{ set-name–3                                                                                        }
{ { USING data-base-identifier–9 [,data-base-identifier–10] ...                                  } }
{ { ALIAS FOR data-base-identifier–11 IS data-base-data-name–3 } ...                             } } ...
```

The following DML commands may traverse a hierarchical storage structure. Note that only *Formats–5* and *–6* of the FIND/FETCH command are included. The MODIFY is included as the alteration may require a change in set occurrences.

```
STORE record-name.
{FIND  } record-name–3 RECORD.
{FETCH }
{FIND  } record-name–4 VIA set-name–5
{FETCH }
    [USING data-base-identifier–1
    [,data-base-identifier–2]...].
MODIFY [data-base-identifier–1 [data-base-identifier–2]...].
```

4.2. OBJECT RECORD SELECTION

If either of the two commands

```
STORE      record-name-n

{FIND  }
{FETCH }   record-name-n RECORD    (Format–5)
```

is executed, the logic followed is dependent upon the LOCATION MODE clause for record-name-n.

4.2.1. LOCATION MODE IS DIRECT

If the LOCATION MODE IS DIRECT, then prior to the execution of the command, the contents of the following locations specified on the LOCATION MODE clause must be initialized.

- The 77-level entry data-base-data-name–2 must be initialized with the area-name of the area in which the object record is either to be stored or found.

  On a STORE command, the area-name must appear in the list of area-names specified on the WITHIN clause for record-name-n or an Error Status Condition results.

- The 01-level entry data-base-data-name–1 must be initialized with a page number and record number in the DML Preprocessor supplied,

  ```
  PAGE-NUM OF data-base-data-name–1
  RECORD-NUM OF data-base-data-name–1,
  ```

  respectively. The page and record numbers are combined with the area-name by the DMR to form a database-key.

  On a FIND/FETCH command the non-existence of the record occurrence of record-name-n identified by the database-key results in an Error Status Condition. Conversely, on a STORE command, the existence of a record occurrence of any type identified by the database-key results in an Error Status Condition. Also, on a STORE command, if either the page number is beyond the allocated number of pages for the area, or if the page number is outside the page range allocated for record-name-n on its WITHIN clause, then Error Status Conditions result.

4.2.2. LOCATION MODE IS CALC

If the LOCATION MODE IS CALC, then prior to the execution of the command, the contents of the data-base-data-name and the data-base-identifiers specified on the LOCATION MODE clause must be initialized. In addition, data-base-procedure-name–1 must have been included in the absolute element containing the run unit through the use of the @MAP Processor. (See Appendix D, in the *UNIVAC 1100 Series Data Management System (DMS 1100) American National Standard COBOL (Fieldata UP–7908 or ASCII UP–7992) Data Manipulation Language Programmer Reference* (current version) for details.)

- The 77-level entry data-base-data-name–3 must be initialized with the area-name of the area in which the object record is to be either stored or found. The initialization of data-base-data-name–3 may be accomplished either by the run unit or by data-base-procedure–1.

  On a store command the area-name must appear on the list of area-names specified on the WITHIN clause for record-name-n or an Error Status Condition results.

- The input data-items to data-base-procedure-name–1, that is, data-base-identifier–1, data-base-identifier–2... must be initialized with the argument values required to develop the desired page number and calc-chain number. The page and calc-chain number developed by data-base-procedure–1 are supplied to the DMR (see Appendix F for details) and determine placement of the object record.

  On a FIND/FETCH command the non-existence of a record occurrence of record-name-n identified by the page number and calc-chain number, and having the same values for data-base-identifier–1, data-base-identifier–2... as those input to data-base-procedure-name–1, results in an Error Status Condition.

  On a STORE command the existence of a record occurrence of record-name-n, identified by the page number and calc-chain number and having the same values for data-base-identifier–1, data-base-identifier–2... as those input to data-base-procedure-name–1 results in an Error Status Condition, unless the LOCATION MODE IS CALC clause specified that DUPLICATES ARE ALLOWED.

  Other Error Status Conditions may arise:

  (1) If overflow pages are required and are full or not defined;

  (2) If the area-key developed by data-base-procedure-name–1 contains a page number beyond the allocated number of pages for the area;

  (3) If the page number is the number of an overflow page;

  (4) If the page number is outside the page range allocated for record-name-n on its WITHIN clause; and

  (5) If the calc-chain number is outside the range of allowable CALC clause calc-chain numbers.

4.2.3. LOCATION MODE IS VIA SET-NAME SET

If the LOCATION MODE IS VIA set-name-n SET, then depending upon the ORDER of set-name-n, the contents of certain data items in WORKING-STORAGE or COMMON-STORAGE must be initialized. Because record-name-n is defined as a member of set-name-n and is either found or stored based upon that relationship, on a FIND/FETCH

command the same result may be obtained by executing a

$$\left\{\begin{matrix}\underline{\text{FIND}}\\ \underline{\text{FETCH}}\end{matrix}\right\}\ \textit{record-name-n}\ \underline{\text{VIA}}\ \textit{set-name-n} \qquad \textit{(Format–6)}$$

command, thus by-passing the examination of the LOCATION MODE clause. However, specification of the set-name on the FIND/FETCH command is generalized as the set-name specified may be any set in which record-name-n participates as a member.

Note that, if the USING option is present on the FIND/FETCH *Format–6* command, that is:

$$\left\{\begin{matrix}\underline{\text{FIND}}\\ \underline{\text{FETCH}}\end{matrix}\right\}\ \textit{record-name-n}\ \underline{\text{VIA}}\ \textit{set-name-n}\ [\underline{\text{USING}}\ \textit{data-base-identifier-n}\ [\textit{data-base-identifier-n1}]\ldots]$$

then the member record occurrence of record-name-n in the selected occurrence of set-name-n is selected on the basis of the initialized values for data-base-identifier-n, data-base-identifier-n1... Thus the following discussion of set ORDER does not apply in this situation. However, the discussion on set ORDER does apply to the STORE, FIND/FETCH *Format–5,* and *Format–6* without the USING option.

- The area, page number, and record number, that is, the database-key of the object record is determined by the DMR.

- If the ORDER of set-name-n is other than SORTED, then either selection or insertion of the object record, once the object set occurrence has been selected, is based upon the following criteria:

  (1) If the ORDER IS FIRST, on a FIND/FETCH command the record selected is the first member record occurrence which follows (in a forward direction) the owner of the selected set occurrence. On a STORE command (or inter-set MODIFY) the logical insert point is immediately following (in a forward direction) the owner of the selected set occurrence.

  (2) If the ORDER IS LAST, on a FIND/FETCH command the record selected is the last member record occurrence, the one which immediately follows (in a backward direction) the owner of the selected set occurrence. On a STORE command (or inter-set MODIFY) the logical insert point is immediately following (in a backward direction) the owner of the selected set occurrence.

  (3) If the ORDER IS NEXT on a FIND/FETCH command the record selected is the first member record occurrence which follows (in a forward direction) the current of the set type. On a STORE command (or inter-set MODIFY) the logical insert point is following (in a forward direction) the current record of the set type.

  If the current record of the set type is the owner record, the result is effectively the same as ORDER IS FIRST.

  (4) If the ORDER IS PRIOR, on a FIND/FETCH command the record selected is the member record occurrence which immediately precedes (in a forward direction) the current of the set type. On a STORE command (or inter-set MODIFY) the logical insert point is immediately preceding (in a forward direction), the current record of the set type.

  If the current record of the set type is the owner record, the result is effectively the same as ORDER IS LAST.

- If the ORDER of set-name-n is SORTED, then prior to the execution of the command, the ASCENDING/DESCENDING key data items for the object member record type must be initialized. On a FIND/FETCH command the record occurrence selected is that record which meets the ASCENDING/DESCENDING key criteria (which may include the RANGE option). On a STORE command (or inter-set MODIFY) the logical insert point of the object record is determined by using the ASCENDING/DESCENDING key criteria.

  Thus far only the selection of the object record of insert point within the object set occurrence has been discussed. The availability of the object set occurrence has been assumed; the following discusses the selection of the object set occurrence.

4.3. OBJECT SET SELECTION

The selection of the object set occurrence, that is, the set occurrence which contains or is to contain the object record is based upon the SET OCCURRENCE SELECTION clause associated with the object record type as a member of the object set type. The SET OCCURRENCE SELECTION clause does this by specifying the criteria to be used to select the owner record occurrence which, in effect, identifies the object set occurrence.

These are two formats of the SET OCCURRENCE SELECTION clause.

Format–1:

```
SET OCCURRENCE SELECTION IS THRU
{CURRENT OF SET                }
{LOCATION MODE OF OWNER        }
 [{USING data-base-identifier–3 [,data-base-identifier–4]...                                                    }]
 [{ALIAS {FOR data-base-identifier–5} IS data-base-data-name–2 }...                                             }]
 [       {OF  data-base-data-name–1 }                                                                           ]
```

The first format provides the selection of the immediate owner record occurrence (which identifies the object record occurrence). It allows the DMR to advance one level in the hierarchy, that is, from the member to the owner. If the SET OCCURRENCE SELECTION clause specifies the CURRENT OF SET option, it is not necessary to advance further. The current record (referred to by the CURRENT OF SET clause) has been procedurally pre-selected, that is, a prior STORE or FIND/FETCH command has established the current record of the set under discussion. This identifies a unique occurrence of the owner record and the object set occurrence owned by it. The discussion in 4.2, OBJECT RECORD SELECTION, governs the selection of the object member record if it is being found or the prior record (logical insert point) if the member record is being inserted.

The second option available under *Format–1* of the SET OCCURRENCE SELECTION clause is the LOCATION MODE OF OWNER option. It also provides the selection of the immediate owner record occurrence by referring to the LOCATION MODE clause defined for the owner record type. If the LOCATION MODE of the owner IS DIRECT or CALC, the hierarchy has ended. The database can be entered at a uniquely determined point. As with CURRENT OF SET clause above, the discussion in 4.2, OBJECT RECORD SELECTION, applies to the set occurrence owned by the owner record. Note that the items specified on the DIRECT or CALC clause must be reinitialized, as well as those required by the set ordering criteria.

If the LOCATION MODE of the owner record type is VIA set-name SET, then the owner record may only be found based upon its participation in another set. In this situation the owner record is examined in its capacity as a member of the higher level set and the SET OCCURRENCE SELECTION clause specified for it in this capacity is examined to determine how to select the owner record's owner. This process could continue upward for several levels, until at some level the database may be uniquely entered. For this reason the SET OCCURRENCE SELECTION *Format–1* is known as the "step" method, because set occurrence selection proceeds a step at a time through the levels of the hierarchy.

Additional options are available when the LOCATION MODE OF OWNER clause is used. The USING option specifies the data-items in the owner record description to be used as search arguments to identify the proper owner record occurrence. The USING option is now allowed when the LOCATION MODE of the owner record is DIRECT or CALC as the only meaningful criteria for owner record selection have been specified on the LOCATION MODE clause. The use of any other criteria would require a brute force search through all areas specified on WITHIN clauses, which is a prohibitive approach. If the LOCATION MODE of the owner record is VIA set-name SET, then the USING option specifies the criteria for selecting the owner record from the set occurrence in which it participates as a member. Therefore, the higher set occurrence which contains the owner record occurrence (the owner's owner) has been determined (this is the "step" method in operation), data-base-identifier–3, data-base-identifier–4... are used to select the lower level owner from the higher level owner's set occurrence. Note that normally selection of a record occurrence from a set occurrence is based upon the set ordering criteria discussed in 4.2, OBJECT RECORD SELECTION; however, the presence of the USING clause overrides this and selection is solely upon the contents of data-base-identifier–3, data-base-identifier–4. After the lower level owner (the immediate owner in this case) has been selected, the set occurrence, which it owns, is then searched for the object member record, again in accordance with the set ordering criteria discussed in 4.2, OBJECT RECORD SELECTION. Thus, the USING option is a means of selecting the owner record occurrence for the object member, by a criteria other than that used when the owner record occurrence was inserted into the higher set.

The ALIAS option may also be used when the LOCATION MODE OF OWNER clause is used. The ALIAS clause is used in a slightly different manner; however, like the USING option its purpose is to select the owner record occurrence for the object member record. The ALIAS option may be used when the LOCATION MODE of the owner record is DIRECT or CALC, to specify alternate data items for the data items specified on the LOCATION MODE clause. These alternate data items are required, only if the member record happens to be an automatic member of two or more sets having the same owner record type. In this situation, the SET OCCURRENCE SELECTION for the member record is defined in one set without an ALIAS clause, and in all other sets having the same owner, with an ALIAS clause specifying alternate data items unique to that set. Note that the alternate data items are defined as 01- or 77-level items having the same PICTURE and USAGE characteristics as the data items they substitute.

Use of the ALIAS option when the LOCATION MODE clause for the owner record specifies VIA set-name SET has the same substitution of data items effect. The ALIAS clause in this case is an extension of the USING clause. The data-base-identifiers specified on the left-hand side of the ALIAS clause (those being substituted for) are the same data items which would be specified if the "two owners of the same type" condition did not apply and the USING option were to be used, or implied by the owner record's LOCATION MODE set-name SET having ORDER IS SORTED.

The previous discussion of the USING clause applies. The set ordering criteria of the set in which this owner record type is a member type is ignored and the comparison is made between the specified data item (data-base-identifier–*5*) in owner records on the database and an alternate data-item (data-base-data-name–2) in WORKING or COMMON STORAGE.

Format–2:

```
;SET OCCURRENCE SELECTION IS THRU set-name–2 USING
{CURRENT OF SET                  }
{LOCATION MODE OF OWNER          }
   [{USING data-base-identifier–6[ ,data-base-identifier–7]...                                  }]
   [{{ALIAS {FOR data-base-identifier–8 } IS data-base-data-name–2}...                          }]
   [{{      {OF  data-base-data-name–1 }                           }                           }]
{ set-name–3                                                                                    }
{ {USING data-base-identifier–9 [ ,data-base-identifier–10]...                                  }} ...
{ {{ALIAS FOR data-base-identifier–11 IS data-base-name–3 }...                                  }}
```

The second of the SET OCCURRENCE SELECTION formats extends the concept of SET OCCURRENCE SELECTION from the more limited "step" method of advancing through the hierarchy one level at a time, to the concept of the"path." The "path" method allows the specification of the criteria for advancing through multiple levels of the hierarchy to be placed on the SET OCCURRENCE SELECTION clause of the object member record.

The path is specified starting at the top of the hierarchy with the owner of set-name–2 and proceeds downward a level at a time through the hierarchy. Thus the owner of set-name–3 must be a member of set-name–2, the owner of the next set must be a member of set-name–3, etc. The path must be continuous and must eventually end with the set name of the Set Entry of which the SET OCCURRENCE SELECTION clause is a part.

Everything stated previously for selecting the owner of the object set occurrence using the *Format–1* criteria applies to selecting the owner of the set-name–2 occurrence using *Format–2* criteria. That is, it may be procedurally preselected and the SET OCCURRENCE SELECTION clause CURRENT OF SET option used. It may have a LOCATION MODE IS DIRECT or CALC and the SET OCCURRENCE SELECTION clause LOCATION MODE OF OWNER option used, with or without the ALIAS option. It may also have a LOCATION MODE IS VIA set-name SET and the SET OCCURRENCE SELECTION clause LOCATION MODE OF OWNER option, used with or without either the USING or the ALIAS option. In the latter case, the top of the path ends short of the top of the hierarchy and a "step" up is made of one level to either the bottom of the next path or the next step.

Within each intermediate level of the path, either the USING or ALIAS option which follows the set-name (for example, set-name–3) specified on the SET OCCURRENCE SELECTION clause specifies the criteria for selection of the owner record of that set. Thus, assuming the selection of the member of set-name–2 has been made as previously discussed (the member of set-name–2 selected identifies an occurrence of set-name–3), the selection of a member occurrence from the occurrence of set-name–3 is based upon, if the USING option is present, the contents of data-base-identifier–9, data-base-identifier–10... If the ALIAS option is used, selection is based upon the contents of data-base-data-name–3 in its role as a substitute data item for data-base-identifier–11. Note that in either case the member record selected is in turn the owner record (thus identifying an occurrence) of the next lower set.

The selection of record occurrences in their dual role as both a member of one set and owner of the next set proceeds until the object set occurrence is reached. In the object set occurrence the object record occurrence is selected based upon either the set ordering criteria or the USING option of the FIND/FETCH command, as discussed in 4.2.3, LOCATION MODE IS VIA SET-NAME SET.

The following examples illustrate the use of the SET OCCURRENCE clause *Format–1* and *Format–2*. Each of the examples use the skeleton definition of records and sets outlined as follows.

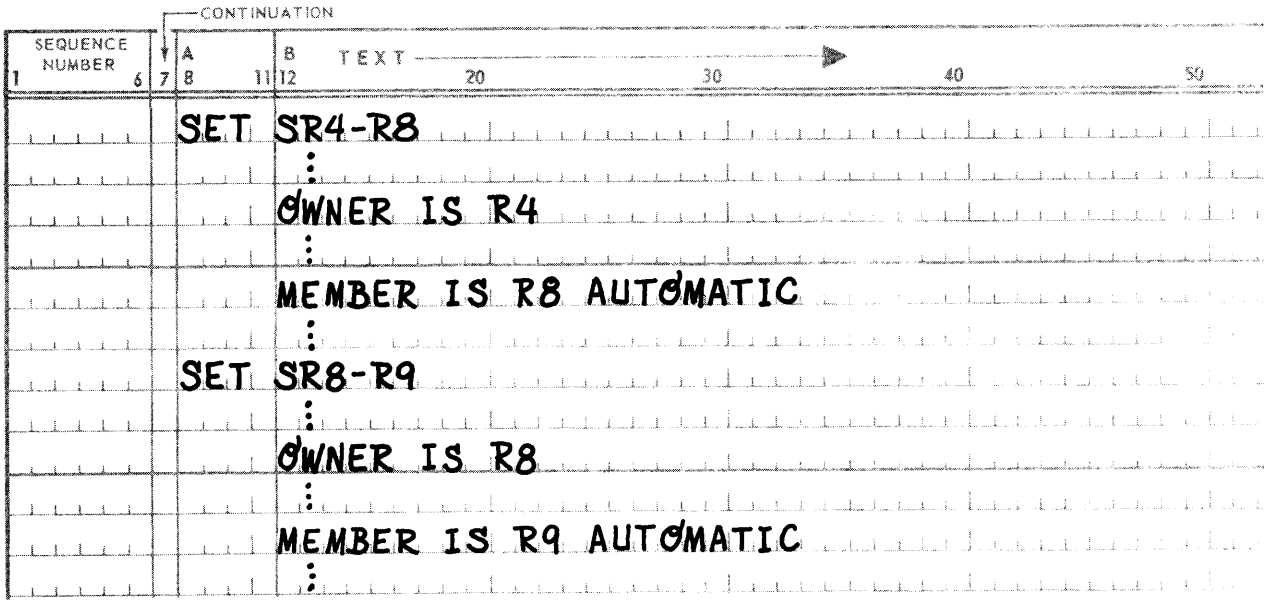
Any additions to the skeleton definition will be specified for a particular example.

```
RECORD R1
          :
          LOCATION MODE IS DIRECT DAN1, DAN2
          :
RECORD R2
          :
          LOCATION MODE IS VIA SR1-R2 SET
          :
          02 R2-DATA....
```

```
       :
RECORD R3
       :
    LOCATION MODE IS VIA SR1-R3 SET
       :
    02 R3-KEY...
       :
RECORD R4
       :
    LOCATION MODE IS VIA SR1-R4 SET
       :
    02 R4-KEY...
    02 ALTKEY4...
       :
RECORD R5
       :
    LOCATION MODE IS VIA SR2-R5 SET
       :
RECORD R6
       :
    LOCATION MODE IS VIA SR3-R6-R7
       :
    02 R6-KEY...
RECORD R7
       :
    LOCATION MODE IS VIA SR1-R7
       :
    02 R7-KEY...
    02 R7-KEY1...
    02 R7-KEY2...
       :
RECORD R8
       :
    LOCATION MODE IS VIA SR4-R8
       :
    02 R8-KEY...
    02 ALTKEY8...
       :
```

```
RECORD R9
    :
    LOCATION MODE IS VIA SR7-R9
    :
SET SR1-R2
    :
    ORDER IS SORTED
    OWNER IS R1
    :
    MEMBER IS R2 AUTOMATIC
    ASCENDING KEY IS R2-DATA
    :
SET SR2-R5
    :
    OWNER IS R2
    :
    MEMBER IS R5 AUTOMATIC
    :
SET SR1-R3
    :
    ORDER IS SORTED
    OWNER IS R1
    :
    MEMBER IS R3 AUTOMATIC
    ASCENDING KEY IS R3-KEY
    :
SET SR3-R6-R7
    :
    ORDER IS SORTED
    OWNER IS R3
    :
    MEMBER IS R6 AUTOMATIC
    ASCENDING KEY IS R6-KEY
    :
    MEMBER IS R7 AUTOMATIC
    ASCENDING KEY IS R7-KEY
    :
```

```
SET SR6-R9
     :
     OWNER IS R6
     :
     MEMBER IS R9 AUTOMATIC
     :
SET SR1-R7
     :
     ORDER IS SORTED
     OWNER IS R1
     :
     MEMBER IS R7 AUTOMATIC
     ASCENDING KEY IS R7-KEY1
     :
SET SR7-R9
     :
     OWNER IS R7
     :
     MEMBER IS R9 AUTOMATIC
     :
SET SR1-R4
     :
     ORDER IS SORTED
     OWNER IS R1
     :
     MEMBER IS R4 AUTOMATIC
     ASCENDING KEY IS R4-KEY
     :
SET SR4-R7-R8
     :
     ORDER IS SORTED
     OWNER IS R4
     :
     MEMBER IS R7 AUTOMATIC
     ASCENDING KEY IS R7-KEY2
     :
     MEMBER IS R8 AUTOMATIC
     ASCENDING KEY IS R8-KEY
     :
```



The relationships between the previously defined records and sets are shown in Figure 4–1.

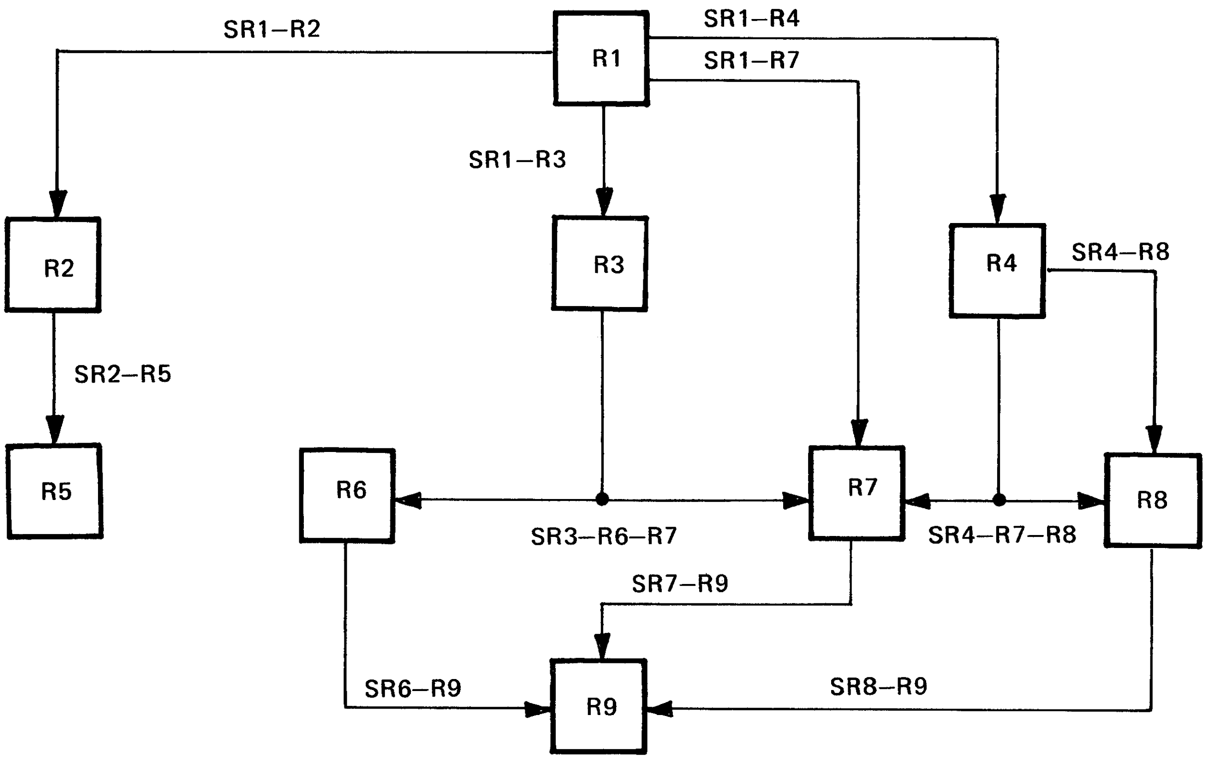


Figure 4–1. Relationship of Records and Sets

Example 1:

This example discusses the use of *Format–1* SET OCCURRENCE clause in the member entry for record type R5 in set type SR2–R5, and for record type R2 in set type SR1–R2.

```
SET  SR2-R5
       :
     MEMBER IS R5 AUTOMATIC
     SET OCCURRENCE SELECTION IS THRU CURRENT
     OF SET
       :
SET  SR1-R2
       :
     MEMBER IS R2 AUTOMATIC
       :
     SET OCCURRENCE SELECTION IS THRU LOCATION
     MODE OF OWNER
```

Suppose that some type of sequential processing is to be performed on record type R5 occurrences in a specific occurrence of set type SR2–R5. Prior to selecting any occurrence of record type R5 in the desired occurrence of set type SR2–R5, statements of the type accomplishing the following would appear in the PROCEDURE DIVISION of the DML Preprocessor program:

```
           MOVE 2 TO PAGE-NUM OF DAN1.
           MOVE 17 TO RECORD-NUM OF DAN1.
           MOVE 'AREA1' TO DAN2.
           MOVE 'AAZ' TO R2-DATA.
           FIND R2 RECORD.
```

The first three statements initialize the values necessary to determine the proper occurrence of record type R1. The fourth statement initializes a value necessary to determine the proper occurrence of record type R2 in the set occurrence SR1–R2 determined by owner record R1. The last statement causes the specific occurrence of set type SR2–R5, determined by owner record R2, to become CURRENT OF SET. Now any selection of record type R5 can be made from the set type SR2–R5 which is CURRENT.

Example 2:

With reference to example 1, assume that occurrences of record type R5 are to be selected in a random manner from *different* occurrences of set type SR2–R5. The SET OCCURRENCE clauses for member types R2 and R5 in their respective set entries could be:

```
       SET SR1-R2
           :
           MEMBER IS R2 AUTOMATIC
           SET OCCURRENCE SELECTION IS THRU LOCATION
           MODE OF OWNER
           :
       SET SR2-R5
           :
           MEMBER IS R5 AUTOMATIC
           SET OCCURRENCE SELECTION IS THRU LOCATION
           MODE OF OWNER
           :
```

Prior to selecting any occurrence of record type R5 in a desired occurrence of set SR2–R5, statements of the type accomplishing the following would appear in the PROCEDURE DIVISION of the DML Preprocessor program:

```
           MOVE 10 TO PAGE-NUM OF DAN1.
           MOVE 22 TO RECORD-NUM OF DAN1.
           MOVE 'AREA2' TO DAN2.
           MOVE 'CBC' TO R2-KEY.
```

The first three statements initialize the value necessary to determine the proper occurrence of record type R1. The fourth statement initializes a value necessary to determine the proper occurrence of record type R2 in the occurrence of set type SR1—R2 determined by owner record R1. Now selection may be made of occurrences of record type R5 in the set of type SR2—R5 whose owner is type R2. The difference between this example and example 1 is that in this example the run unit does not rely on the current occurrence of set type SR2—R5.

Example 3:

Assume for this example that occurrences of record type R9 are to be selected. Occurrences of record type R9 could be determined using a path which includes record type R1, owner type of set type SR1—R7, and record type R7, owner type of set type SR7—R8. The latter path could be utilized in the same manner as the path to record type R5 in examples 1 and 2. Suppose, however, that an alternate path to occurrences of record type R9 is, at times, more advantageous. The alternate path could include record type R1, owner type of set type SR1—R3, record type R3, owner type of set type SR3—R6—R7, and record type R6, owner type of set type SR6—R9.

The SET OCCURRENCE clause could be defined for member record type R9 in its entry in set type SR6—R9 to utilize the latter path:

```
       SET SR6-R9
           :
           MEMBER IS R9 AUTOMATIC
           SET OCCURRENCE SELECTION IS THRU SR1-R3
               USING LOCATION MODE OF OWNER
               SR3-R6-R7 USING R3-KEY
               SR6-R9 USING R6-KEY
           :
```

Prior to selecting any occurrence of record type R9 in an occurrence of set type SR6—R9, statements of the type accomplishing the following would appear in the PROCEDURE DIVISION of the DML Preprocessor programs:

```
           MOVE 18 TO PAGE-NUM OF DAN1.
           MOVE 27 TO RECORD-NUM OF DAN1.
           MOVE 'AREAZ' TO DAN2.
           MOVE 'MAINE' TO R3-KEY.
           MOVE 'SMITH' TO R6-KEY.
```

The first three statements initialize the values necessary to determine the proper occurrence of record type R1, owner of the desired occurrence of set type SR1—R3. The fifth statement initializes a value necessary to determine the proper occurrence of record type R3 in the set of type SR1—R3. The fourth statement initializes a value necessary to determine the proper occurrence of record type R6 in the set occurrence of SR3—R6—R7 determined by the occurrence of record type R3. Since set type SR6—R9 has been determined by the occurrence of R6, R9 type occurrences may be selected using FIND/FETCH *Format–6*, e.g., FIND R9 VIA SR6—R9.

Example 4:

This example is similar to example 3. Another path will be used to locate occurrences of record type R9. Suppose in certain instances of selecting occurrences of record type R9 that the ascending keys R3-KEY and R6-KEY in record types R3 and R6, respectively, are not known. Assume, though, that in the same instances, ALTKEY4 and ALTKEY8, items defined in record types R4 and R8, respectively, are known. The path which includes record type R1, owner type of set type SR1—R4, record type R4, owner of set type SR4—R7—R8, and record type R8, owner type of set type SR8—R9, could be utilized.

The SET OCCURRENCE clause could be defined for member record type R9 in its entry in set type SR8—R9 to utilize the latter path:

CONTINUATION

SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT 20 30 40 50

```
       SET SR8-R9
           :
           MEMBER IS R9 AUTOMATIC
           SET OCCURRENCE SELECTION IS THRU SR1-R4
               USING LOCATION MODE OF OWNER
               SR4-R7-R8 USING ALTKEY4
               SR8-R9 USING ALTKEY8
           :
```

Prior to selecting an occurrence of record type R9, values PAGE-NUM and RECORD-NUM OF DAN1, DAN2, ALTKEY4, and ALTKEY8 must be initialized in the PROCEDURE DIVISION of the DML Preprocessor program similar to the manner described for values PAGE-NUM and RECORD-NUM of DAN1, DAN2, R3-KEY, and R6-KEY in example 3 and, as also in example 3 FIND/FETCH *Format–6* would be used to utilize this SET OCCURRENCE SELECTION clause.

Example 5:

Since record type R8 is a member of set type SR4—R8 whose owner type is R4, and a member of set type SR4—R7—R8 whose owner type is also R4, the following SET OCCURRENCE clauses will be defined for record type R8 in sets SR4—R8 and SR4—R7—R8, respectively:

```
SET SR4-R8
         :
         MEMBER IS R8 AUTOMATIC
         SET OCCURRENCE SELECTION IS THRU LOCATION
         MODE OF OWNER
         :
SET SR4-R7-R8
         :
         MEMBER IS R8 AUTOMATIC
         :
         SET OCCURRENCE SELECTION IS THRU LOCATION
         MODE OF OWNER
             ALIAS FOR R4-KEY IS DAN4
         :
```

Before an occurrence of R8 can be stored or selected, the proper occurrence of type R4, and subsequently R1, must be determined. Assume an occurrence of R8 is to be stored. Data-names DAN1 and DAN2 must be initialized so the proper occurrence of R1 can be determined. When R1 is determined, R4's occurrence is set SR1–R4 must be determined. Suppose the particular occurrence of R8 being stored belongs to set SR4–R8 owned by one occurrence of R4, and belongs to set SR4–R7–R8 owned by a different occurrence of R4. Effectively two different occurrences of R4 in set SR1–R4 must be determined. Therefore, an ALIAS clause was specified on the SET OCCURRENCE clause for member type R8 in set type SR4–R7–R8. DAN4 (which must be defined in the WORKING STORAGE SECTION of the DML Preprocessor program) and R4-KEY must be initialized to determine the occurrence of R4 desired. If the occurrence of R4 were the same for sets SR4–R8 and SR4–R7–R8, DAN4 and R4-DATA would be initialized with the same value.

This example assumes that the two different occurrences of R4 are members of the same SR1–R4 set occurrence. Where this not the case, then *Format–2* of the SET OCCURRENCE SELECTION clause would be used to provide ALIAS data-base-data-names for DAN1 and DAN2.

5. RECORD PLACEMENT STRATEGY

5.1. GENERAL

This section discusses the Record Placement Strategy employed for selection of the page in which the object record occurrence is to be placed as the result of a STORE command.

The remainder of this section discusses the following:

- The DDL clauses involved in constructing a Record Placement Strategy.
- The storage of records having a LOCATION MODE of DIRECT, CALC and VIA SET, and
- The alteration of a variable length record by execution of a MODIFY command.

There are certain DDL clauses which are significant in determining the outcome of the Record Placement Strategy.

(1) Within the DDL Area Entry the following clauses are involved:

ALLOCATE *integer–2* [PRE-INITIALIZED] **PAGES**
[, *integer–10* OVERFLOW **PAGES AT END**]
[, *integer–3* OVERFLOW **PAGES** EVERY *integer–4* **DATA PAGES**]
[, EXPANDABLE **TO** *integer–11* **PAGES**]
PAGES **ARE** *integer–5* **WORDS**
[LOAD **IS** { *integer–6* WORDS / *integer–7* PERCENT }]
[CALC **USES** *integer–8* **CHAINS** [**LINKED** PRIOR]

(2) Within the DDL Record Entry the following clauses are involved:

```
                                  ⎧ DIRECT data-base-data-name–1, data-base-data-name–2
                                  ⎪ CALC data-base-data-name–3
                                  ⎪     IN data-base-data-name–3
LOCATION MODE IS                  ⎪     USING data-base-identifier–1
                                  ⎪         [,data-base-identifier–2]...
                                  ⎪     DUPLICATES ARE [NOT] ALLOWED
                                  ⎪ VIA set-name–1 SET
                                  ⎩   [INTERVAL IS integer–2 PAGES]

[{;WITHIN area-name–1 [{integer–3       }{THRU    }                    ]}... ]
                      {data-base-data-name–4}{THROUGH}{integer–4       }
                                             {data-base-data-name–5}
```

(3) Within the DDL Set Entry the following clauses are involved:

OWNER IS *record-name–1*

MEMBER IS *record-name–2*

5.2. DIRECT RECORD PLACEMENT

Storage of records having a LOCATION MODE IS DIRECT option is the simplest form of record placement. Typically, it is used either to place a single occurrence of an owner record at the top of a hierarchy, or to strategically place owner records to take advantage of the system "place near" logic utilized in storing the member records. Also, it may be used to incrementally store records during initial load, thus relieving the system of the burden of achieving the same result.

A run unit which stores a record having a LOCATION MODE IS DIRECT must supply the following:

- The name of the area in which the record is to be stored.
- The area-key which is to be assigned to the record once stored in the named area. The area-key contains the page number in which the record is to be stored and the record number to be assigned to the record once stored in the specified page.

If possible the record is stored with the specified area-key. However, there are certain conditions which prevent the storage of the record. These conditions are as follows:

- If the area name supplied is not listed as area-names specified on the WITHIN clause, then an Error Status Condition results.
- If sufficient space does not exist in the specified page, then an Error Status Condition results.
- If the record number is already in use on the specified page, then an Error Status Condition results.

■ If the specified page is defined as an overflow page, then an Error Status Condition results.

If an Error Status Condition results, the record is not stored. Note that the previous Error Status Conditions pertain only to Record Placement Strategy and are not an exclusive list. For example, if the area is OPENed with a USAGE-MODE IS RETRIEVAL, and Error Status Condition results. Various Error Status Conditions may also result from violation of the rules specified in the previous section, HIERARCHICAL STORAGE STRUCTURES. For example, if storage of the record violates a DUPLICATES NOT ALLOWED clause, an Error Status Condition results.

The following example serves to illustrate the storage of a record having a LOCATION MODE IS DIRECT.

Assume the schema definition includes the following record entry:

```
RECORD NAME IS STATE
    :
    LOCATION MODE IS DIRECT DAK1, DAN1
    WITHIN AREA1
    WITHIN AREA2
    :
```

The user program must contain the following DML statements in the WORKING STORAGE SECTION:

```
77 DAN1 USAGE IS AREA-NAME.
01 DAK1 USAGE IS AREA-KEY.
```

The COBOL source language which is produced by the DML Preprocessor for the above two statements is:

```
77 DAN1 USAGE IS DISPLAY PIC X(12).
01 DAK1 USAGE IS COMP.
    02 PAGE-NUM PIC 9(5).
    02 RECORD-NUM PIC 9(5)
```

Before an occurrence of record type STATE can be stored by a run unit, DAN1 must be initialized with either area name AREA1 or area name AREA2; PAGE-NUM must be initialized with a page number, and RECORD-NUM must be initialized with a record number. The procedure division of the program input to the DML Preprocessor might contain statements like the following:

```
    MOVE 'AREA1' TO DAN1.
    MOVE 17 TO PAGE-NUM OF DAK1.
    MOVE 2 TO RECORD-NUM OF DAK1.
    STORE STATE.
```

Execution of the program would result in the occurrence of STATE record being stored as the second record on page 17 in area AREA1. If the first WITHIN clause specified pages 20 THRU 30, PAGE-NUM could be initialized only with values within the 20 thru 30 range.

5.3. CALC RECORD PLACEMENT

Storage of records having a LOCATION MODE IS CALC option provides the Data Administrator with additional flexibility. The Data Administrator may use DMSCALC or may provide his own CALC procedure which in turn provides the page number and calc-chain number; however, the Data Management Routine automatically resolves conflicts between records, places records into overflow pages, etc. Records placed through a CALC procedure are generally available in one access. A CALC procedure may be used by itself to access data records, or it may be used in conjunction with a set definition, thus determining owner records, that is, entry points into the respective set occurrences. In the latter case, it is used together with the system "place near" logic to provide quick access to member records.

The CALC procedure may be written to perform as follows:

- A table lookup,
- Perform a key transformation (randomizing technique), or
- Store records incrementally using special counters, etc.

A run unit which stores a record having a LOCATION MODE IS CALC must supply the following:

- The name of the area in which the record is to be stored. This may alternatively be supplied by the CALC procedure.
- The name of the CALC procedure (an entry point) to be used to provide the page number and calc-chain number and, if required, the area-name. A table with this name and the name of the segment the CALC procedure is in must also be provided. See Appendix D, *UNIVAC 1100 Series Data Management System (DMS 1100) American National Standard COBOL (Fieldata UP–7908 or ASCII UP–7992) Data Manipulation Language Programmer Reference* (current version).
- The specified input values to be used by the CALC procedure to determine the page number of the page upon which the record is stored and the calc-chain number on that page.

The following describes the process of storing a record via CALC. The description is in two stages:

- The straightforward case of a single calc-chain per page, and
- The more involved case of additional calc-chains.

When the Data Administrator desires to store records using calc-chain per page he indicates this in the Area Description by including the clause

CALC **USES 1 CHAIN [LINKED** PRIOR]

with or without the optional LINKED PRIOR portion. The implications of linking are discussed below. The entire clause may be omitted if single linking is desired, since the default value is 1 calc-chain per page. The header for this single chain, chain 0, is a word in the page header of each data page.

During the store logic, DMSCALC or the user supplied calc procedure is referenced to obtain the page number and calc-chain number. In this case, the page number is a data page number (H1 of A15) and the calc-chain number must have the value 0 (H2 of A15). In addition the appropriate area name has been defined.

The CALC record placement logic will place the object record on the data page, indicated by the page number as long as sufficient space exists on that page. If the page contains insufficient space, the DMR will store the record on an appropriate overflow page and link the record into the calc-chain for the data page.

There are two kinds of linking which may be specified for calc-chains at the discretion of the Data Administrator.

(1) The default value is single links (pointers) in the next direction only.

(2) Double linking (bi-directional) may be requested by including the LINKED PRIOR portion on the CALC USES clause of the area description.

The DDL reserves one word in each CALC record which is a member of a singly linked calc-chain and two words for doubly linked chains. Single linking is always sufficient. Some efficiencies are possible for bi-directional chains which are updated frequently.

The CALC record placement logic inserts each new record into the page according to three principles:

(1) Construct each chain so that if it does have to overlap several pages a minimum number of I/O's are required to read the entire chain.

(2) Link each new record into the chain so that a subsequent search for space in that chain avoids full pages.

(3) Link each record into the calc-chain as soon after the head as possible (the head is a location in the page which points to the first record on the chain).

As these principles are encoded, four insert points into a chain are possible:

(1) Chain empty – store the record as the first in the chain.

(2) Between head and existing first record.

(3) Between the last record on the data page and the first record on another page.

(4) At the end (just before the header) of an existing chain which extends into other pages.

Which of the insert points is used depends on the three principles above and also upon whether or not the Schema allows duplicates in the chain.

Example 1:

Suppose we consider area AREA1 which consists of one data page and two overflow pages. Each page will hold exactly three of the records being stored CALC. Duplicated are allowed.

- Store record occurrence A. It is the first member of the page and the calc-chain looks like:

  HEAD → A → HEAD (Assume single linking)

- Store record occurrences B and C in that order. All three records are stored on the data page. By the third principle the second insert point is selected each time so that the calc-chain appears:

  HEAD → C → B → A → HEAD

- Store record occurrences D, E and F in that order. The three records go on the first overflow page. To implement the first and second principles the placement logic selects insert points four then three. The resulting calc-chain is:

  HEAD → C → B → A → F → E → D → HEAD

- Store record G. As before insert point three yields:

  HEAD → C → B → A → G → F → E → D → HEAD

  Record G is stored on the second overflow page. Note that, the first link which points off the data page points to the last page where space was found. Subsequent searches for space avoid the full overflow page.

Example 2:

Assume the same conditions as the previous example except that duplicates are not allowed.

- The first three stores result in the calc-chain:

  HEAD → C → B → A → HEAD

- Store record occurrences D, E and F into the page. As before they go on the first overflow page. This time however, as each is stored the entire calc-chain must be searched to ensure that the object record does not duplicate an existing record. Rather than make two passes through the calc-chain it is linked as follows:

  HEAD → C → B → A → D → E → F → HEAD

- Finally record G is stored into the second overflow page. The calc-chain appears:

  HEAD → C → B → A → D → E → F → G → HEAD

It is possible for the Data Administrator to define that more than one calc-chain is to be associated with a data page. The number of calc-chains is defined by the clause:

CALC **USES** *integer–8* **CHAINS [LINKED** PRIOR**]**

where integer–8 is two or greater. For each additional calc-chain thus specified, the DMR reserves a word at the end of each data page in the area to function as the head of the calc-chain. The head of the first calc-chain remains in the page header.

During the store logic the user CALC procedure is referenced to obtain the page and chain number. In this case, the page number is a data page number (H1 of A15) and the calc-chain number, (H2 of A15), must fall between 0 and (integer–8)–1 inclusive.

New records are stored and multiple calc-chains built on a page exactly as described above for a single calc-chain.

The design for calc-chains allows a record to become the first record in the calc-chain, even if it cannot be placed on the data page.

There are certain conditions which prevent storage of the record and are as follows:

- If the area-name is not on the list of area-names specified on the WITHIN clause, then an Error Status Condition results.

- If the specified calc-chain number is greater than the number of calc-chains declared for the area on the CALC USES clause, then an Error Status Condition results.

- If the specified page has been defined as an overflow page, then an Error Status Condition results.

- If the specific entry point to the CALC procedure is not in the table, then an Error Status Condition results (see 4.2.2).

5.4. RECORD PLACEMENT VIA SET

Storage of records having a LOCATION MODE IS VIA set-name SET frees the Data Administrator from determination of area-keys. The Data Management Routine assigns an appropriate page and record number in accordance with its "place near" Record Placement Strategy. However, the Data Administrator may influence this decision through the use of the WITHIN and INTERVAL clauses. In any event, the area-key assigned by the DMR is always available to the run unit in the DMCA reserved location AREA-KEY.

The guiding rule of the Record Placement Strategy, used when the LOCATION MODE IS VIA set-name SET, is that placement is always made as near as possible to the logical insert point.

The logical insert point is defined as the prior record of the selected set occurrence, which may be the owner record or another member record occurrence. The prior record of the set occurrence is determined in accordance with the set ordering criteria.

The place near logic is defined as the placement of each new record in the same page as its prior record, whenever sufficient space exists.

- WITHIN Clause but no INTERVAL Clause – If the object record type being stored has a WITHIN clause, but no INTERVAL clause, then the following rules govern the page selection for each occurrence. Note that this type of record acts to break the place near logic, and controls the placement of all record types below it in the hierarchy, until another record type having a WITHIN clause is encountered.

  (1) If the page containing the prior record (logical insert point) is included in the pages specified on the WITHIN clause, the search for available space begins with that page. If the page containing the prior record is outside the limits of the WITHIN clause, then the search for available space begins with the first allowable page in the area specified on the first WITHIN clause. If sufficient space is available, the record is stored in that page.

  (2) If sufficient space is not available in the starting page, the search proceeds in a forward direction through the remaining allowable data pages in the area. If the starting page is other than the first allowable page, then a wrap-around to the starting page occurs. All overflow pages are skipped. The record is stored in the first page encountered having sufficient space.

  (3) If sufficient space has not yet been found, the search continues through the allowed pages of the areas specified on the remaining WITHIN clauses. If the search did not start with the area specified on the first WITHIN clause, the search wraps-around to include any areas specified ahead of the starting area. Thus the order of the area-names specified on the WITHIN clauses has a bearing upon page selection. Again, all overflow pages are skipped.

(4) If sufficient space is not found on any data pages allowed by the WITHIN clauses, then the record occurrence is stored in the overflow pages associated with the starting page of the search. If no overflow pages exist, or sufficient space is not available, then an Error Status Condition results.

- WITHIN Clause and an INTERVAL Clause – If the object record type being stored has both a WITHIN clause and an INTERVAL clause, then the following rules govern the page selection for each occurrence. Note that occurrences of this record type act as cluster points, i.e., as owner record occurrences strategically positioned in such a manner that member record occurrences may be clustered about the respective owner. The presence of the INTERVAL clause imposes a further constraint upon page selection. The following rules apply.

  (1) If this is the first record occurrence of this record type, it is placed in the first page having sufficient space within the first number of pages specified on the INTERVAL clause, i.e., in the first interval. The first interval starts with the first allowable page in the area specified on the first WITHIN clause.

  (2) Each subsequent record of this type stored is placed in its respective interval, no two record occurrences ever sharing the same interval.

  (3) To facilitate the assignment of intervals, the last interval assigned is retained. The search for an available interval always begins with the next interval. Each interval under consideration is always examined to determine if a record of this type already is stored within it. If so, the search advances to the next interval.

  (4) The wrap-around rules previously specified for searching through the WITHIN clauses apply here, except that the search advances an interval at a time in accordance with the interval rules.

  (5) The first "empty interval" encountered having sufficient space in one of its pages is used. If one is not found, and Error Status Condition is returned.

  (6) Overflow pages may not be included in an INTERVAL specification; therefore, no record occurrence having an INTERVAL clause associated with it can be stored in an overflow page.

- No WITHIN Clause or INTERVAL Clause – If the object record does not have a WITHIN or INTERVAL clause associated with it, then the lowest level record type (higher than the object record in the hierarchy) having a WITHIN or INTERVAL clause constrains the page selection for the object record. Considering first the situation in which a higher level record has a WITHIN clause, but no INTERVAL clause, then the object record must be placed in a page with the limits given by the WITHIN clause. The rules are the same which apply to the record type having the WITHIN clause.

  (1) The page containing the logical insert point (prior record of the object set occurrence) is first examined to see if the page is allowable per the WITHIN clause of the next higher hierarchical record. If it is, and sufficient space is available, the record is stored on that page.

  (2) If sufficient space is not available in the page, or the page containing the prior record lies outside the limits of the next higher hierarchical record, then all other data pages within the same area (within the limits specified on the WITHIN clause) are searched in a wrap-around fashion for available space. All overflow pages are skipped.

  (3) If sufficient space is not found, the search proceeds to the allowable data pages of the next area (if any) specified on the WITHIN clause. All overflow pages are skipped.

  (4) The search for available space continues through the allowed data pages of the remaining areas specified on the WITHIN clause, then wraps-around to include any areas specified before the starting area. Thus the order of areas specified on the WITHIN clause has a bearing on page selection. Again all overflow pages are skipped.

(5) If sufficient space is not found on any data page allowed by the WITHIN clause, then the record occurrence is stored on the overflow pages associated with the page containing the next hierarchical record. If overflow pages do not exist, or sufficient space is not available, then an Error Status Condition results.

If a record type higher in the hierarchy has both a WITHIN and an INTERVAL clause associated with it, the INTERVAL clause imposes a further constraint upon page selection for the new lower level record occurrence. This is because in the absence of the INTERVAL clause, the new member record could go in any data page the higher level record type could go into; now with the presence of the INTERVAL clause, the new member record can go only in those data pages in the interval belonging to the associated higher level record occurrence.

(1) The page containing the logical insert point (prior record) is first examined. If sufficient space is available, the record is stored in that page.

(2) If sufficient space is not available in the page containing the prior record, then all other data pages within the interval containing the associated higher level record occurrence are searched, in a wrap-around fashion, for available space. Note that no overflow pages are included within the interval.

(3) If sufficient space is not found, the search proceeds to the overflow pages associated with the data pages in the interval. The record occurrence is stored in the first overflow page having sufficient space. If overflow pages are not defined or do not contain sufficient space, an Error Status Condition results.

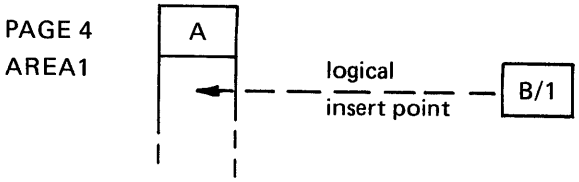
In the previous discussion the terms "sufficient space" and "associated overflow pages" are used repeatedly. It must be remembered that when the areas involved are open for initial load, the LOAD clause, if present, reduces the amount of space available in each page. Secondly, if the areas involved are open for initial load, overflow pages are not available for use; therefore, at any point a search of overflow pages becomes required, an Error Status Condition results. The following examples illustrate the Record Placement Strategy when the LOCATION MODE IS VIA set-name SET.

As an example, assume the following record and set definitions:

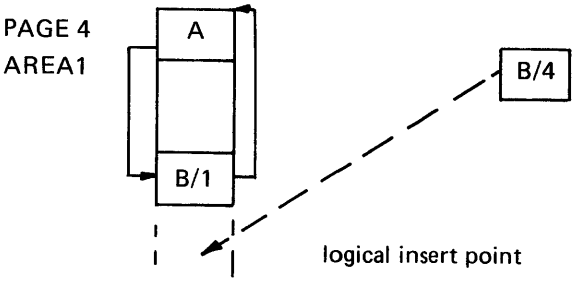
```
RECORD A
    :
    LOCATION MODE IS DIRECT DAK1, DAN1
    WITHIN AREA1
    :
RECORD B
    :
    LOCATION MODE IS VIA A-B
    :
SET A-B
    :
    ORDER IS SORTED
    OWNER IS A
    :
    MEMBER IS B
    ASCENDING KEY IS B-KEY
    :
```

Suppose that an occurrence of record type A is stored DIRECT in page 4 of area AREA1. The following diagram shows the logical insert point of members of set A–B (determined by the occurrence of A just stored) as they are subsequently stored.

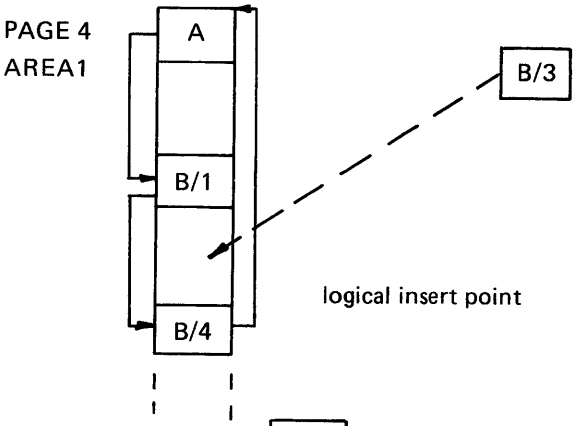
Notation: B/n — An occurrence of B for which n is the value of B–KEY.



The record is placed as near as possible to, and following, the owner record.

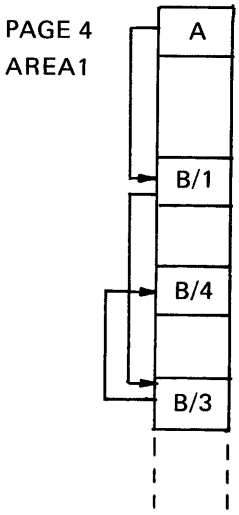


The record is placed as near as possible to, and following, the record which immediately precedes it in the set.



Since the set is sorted ascending on key B–KEY, this record should logically be inserted between the B–KEY of 1 and the B–KEY of 4.

If there were space for record B/3 on page 4:



If the record does not fit on page 4, a search is made for space on page 5, then page 6 if necessary, etc., until the end of the data pages specified for AREA1 is reached. If space has not been found, a "wrap-around" search is made, i.e., space is sought on page 1, then page 2 and page 3 if necessary. If space has not been found, and no overflow was specified for AREA1, the record cannot be stored. If overflow exists, space is sought for this record in the closest overflow interval following page 4.

Assume, for the following discussion, these record and set definitions:

CONTINUATION | SEQUENCE NUMBER 1 6 | 7 | A 8 11 | B 12 TEXT 20 30 40 50

```
RECORD NAME IS P
    :
    LOCATION MODE IS DIRECT DAN1, DAN2
    WITHIN AREA1
    :
RECORD NAME IS Q
    :
    LOCATION MODE IS VIA P-Q SET
    :
RECORD NAME IS R
    :
    LOCATION MODE IS VIA Q-R SET
    :
RECORD NAME IS T
    :
    LOCATION MODE IS VIA R-T SET
    :
SET NAME IS P-Q
    :
    OWNER IS P
    :
    MEMBER IF Q AUTOMATIC
    :
SET NAME IS Q-R
    :
    OWNER IS Q
    :
    MEMBER IS R AUTOMATIC
    :
SET NAME IS R-T
    :
    OWNER IS R
    :
    MEMBER IS T AUTOMATIC
    :
```

Also assume the following notation used in the following diagrams:

Qi – denotes the i*th* occurrence in the set type P–Q, where P is the owner occurrence of P–Q;

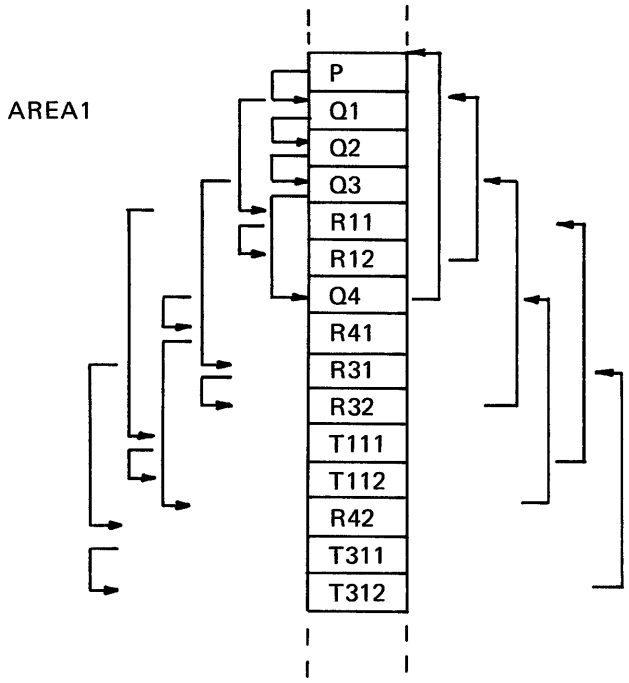
Rij – denotes the j*th* occurrence in the set type Q–R, where Qi is the owner of Q–R;

Tijk – denotes the k*th* occurrence in the set type R–T, where Rij is the owner of R–T.

The example first shows the siimple place-near logic. None of the record types whose location mode is via set will contain INTERVAL or WITHIN clauses in their respective definitions. Following the illustration of the simple place-near logic, the effect of WITH clauses, and then INTERVAL clauses, on the place-near logic will be shown.

Assume that the following records are stored during initial load in this sequence: P, Q1, Q2, Q3, R11, R12, Q4, R41, R31, R32, T111, T112, R42, T311, T312. Their locations in AREA1 would be:

NOTE: Assume P is stored DIRECT in page 1.

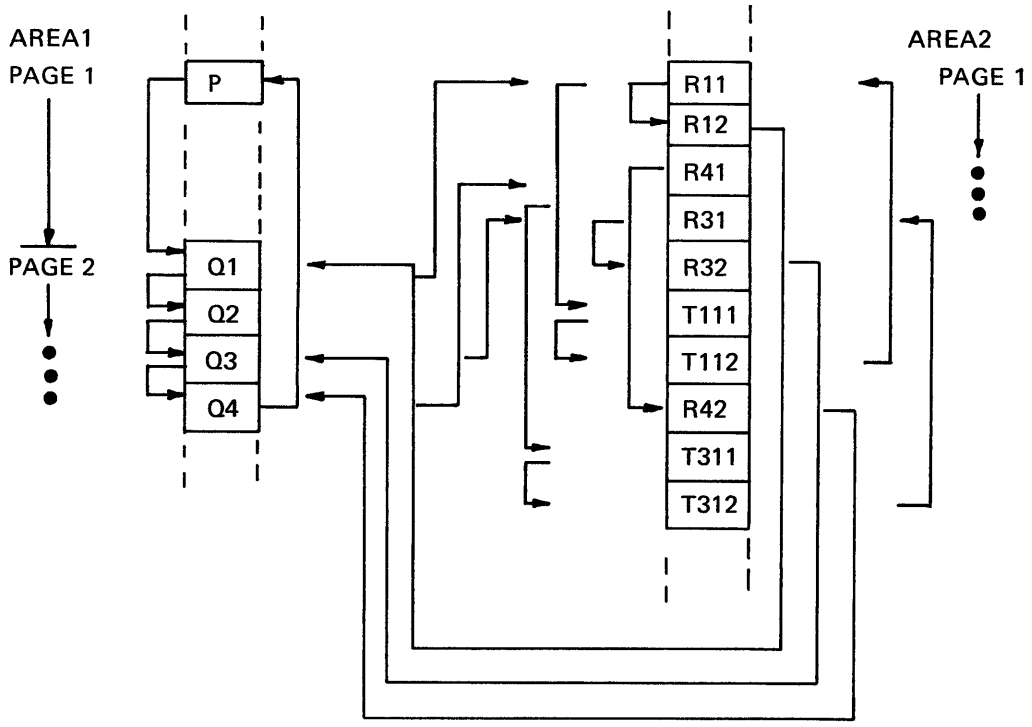


Records are stored as many to a page as possible beginning at the page in which P was stored DIRECT. If the sequence of storing were different during initial load, the physical locations in AREA1 would also be different.

Assume now that the following record types include these WITHIN clauses in their definitions:

Q – WITHIN AREA1 2 THRU 10
R – WITHIN AREA2

For the same sequence of storing as above, the record locations in AREA1 and AREA2 would be:



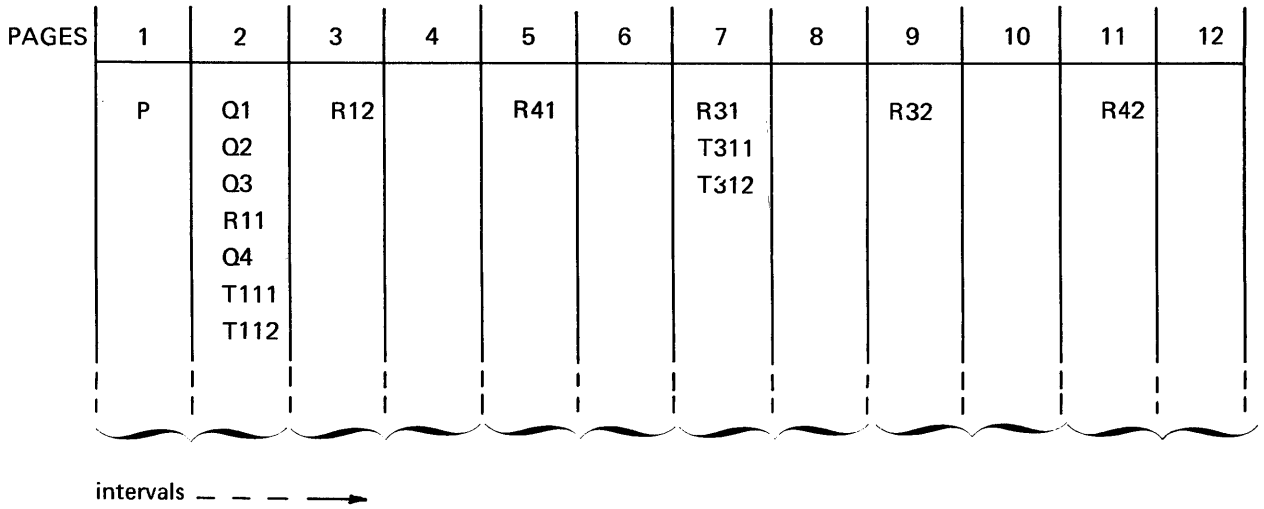
In the previous case, record type Q could not be stored on the same page as its owner, P. The WITHIN clause in the definition of type Q has restricted the placement of type Q to pages 2–10 in AREA1. Likewise, the WITHIN clause in the definition of type R overrides the place near logic. And even though record type T specifies no WITHIN clause, records of type T are placed in AREA2 (using the place near logic) since their owner type is R.

Next, assume that the following record types includes these WITHIN and INTERVAL clauses in their definitions:

Q – WITHIN AREA1 2 THRU 10
R – LOCATION MODE IS VIA Q–R INTERVAL 2 WITHIN AREA1

For the same sequence of storing, the records would be placed on these pages in AREA1:

AREA1



Type Q is restricted to pages 2–10. R11 is placed on the same page as its owner, Q1, but R12 must be placed in the next two-page interval, since no more than one record type R can be placed in one interval. Because type T has no restrictions imposed upon it, the place near logic is used to store type T.

The time required to place records stored via set with no interval clause can be significantly accelerated by turning the switch LDPAGE on in DMSSGP. When this switch is on then for each area, search for available space begins at this last data page saved if it is an allowable page as constrained by the page specification on the associated WITHIN clause. Although no space is ever wasted in the area, the placement of records may be somewhat surprising to the Data Administrator unless he is familiar with the way the last data page is used.

The last data page is used by the store via set routines in the DMR. If the page containing the logical insert point has no available page space or lies outside the limits of the WITHIN clause, the last data page for that area would be checked to see if it is an allowable page of the associated WITHIN clause. If it is not an allowable page, the search would proceed in the normal manner, that is, with the first allowable page in the area specified on the first WITHIN clause. If it is an allowable page the search for page space begins at the last data page and proceeds in the forward direction through the remaining allowable data pages in the area. If the starting page was other than the first allowable page, then a wrap-around to the first page occurrs. If the search continues through other areas specified in associated WITHIN clauses, the last data page for each area would be used to start the search, provided it was an allowable page. The last data page is updated when space has been found in an area.

By adhering to the above logic, it can be seen that if all the records stored via set contain no page specification in their WITHIN clauses, then the records are placed in the normal manner. However, if the records stored via set contain a variety of page specifications for their WITHIN clauses, placement may divert somewhat from the normal manner. For example, suppose two records are stored via set in the same area. The first record has no page specification on its WITHIN clause and the second has a page specification near the end of the area. Now suppose the second record is stored. The last data page for this area then would be a page near the end of the area. Suppose now the first record is stored, and no space is available on the page containing the logical insert point; search for space will then continue at the last data page (since it is an allowable page) and the record will be placed near the end of the area; normally it would have been placed at the beginning of the area.

Situations like this may or may not be serious to the Data Administrator. Since most conflicts of the above nature can usually be minimized by careful use of the INTERVAL specifications on the WITHIN clause, it is suggested to turn LDPAGE on if many records are to be stored via set.

5.5. MODIFIED VARIABLE LENGTH RECORDS

This section applies only to variable length records, that is, those record types which contain a data item defined with an:

OCCURS *integer–7* **TO** *integer–8* **TIMES**
DEPENDING **ON** *data-base-identifier–4*

clause. This section applies regardless of whether the LOCATION MODE clause specifies the DIRECT, CALC or VIA set-name SET options.

All variable length records are stored based upon their respective LOCATION MODE clause in accordance with 5.2, 5.3, and 5.4. The exact length (including system headers and pointers) of the record occurrence at the time of storage is used in the search for sufficient space. After storage, if a variable length record occurrence is the object of a MODIFY command, then the modified record occurrence is returned to the database in accordance with one of the following three criteria.

(1) If the length of the modified variable length record occurrence has not been altered, then the modified copy in the record-delivery-area replaces the copy in the page. Record Placement Strategy does not apply.

(2) If the length of the modified variable length record occurrence is less than the length of the copy in the page, then the modified copy in the record-delivery-area replaces the copy in the page. The unused words from the original copy of the page are surrendered and are therefore available for re-use by Record Placement Strategy.

(3) If the length of the modified variable length record occurrence is greater than the length of the copy in the page, then the amount of space available on the page must be examined. If sufficient space (the difference between the original and new lengths) is available on the page, the modified copy in the record-delivery-area replaces the copy in the page. If sufficient space is not available within the page, then the overflow pages associated with this page are examined.

- If the area is not defined as containing overflow pages on the ALLOCATE clause, then an Error Status Condition results.
- If the associated overflow pages are full, then an Error Status Condition results.

If sufficient space is available on an associated overflow page, the modified copy of the record occurrence in the record-delivery-area is stored in the overflow page. The space assigned to the original copy is surrendered and is therefore available for re-use by Record Placement Strategy. Note that the database-key of the record is not changed. The record remains available through its original database-key.

If a variable length record occurrence, already on an overflow page as a result of a previous MODIFY, is again expanded and sufficient space does not exist on the overflow page, the above rules for expansion apply. However, before the other associated overflow pages are searched for space, the original data page is examined to determine if sufficient space is now available.

Displacement of a modified and expanded record from its original page causes the DMR to add one word to its length for a system pointer to the original page when the record is placed on the overflow page.

5.6. MODIFIED CALC INPUT VALUES

Those records defined with a:

LOCATION **MODE IS** CALC *data-base-procedure-name–1*
IN *data-base-data-name–3*
USING *data-base-identifier–1, data-base-identifier–2 ...*
DUPLICATES **ARE** [NOT] **ALLOWED**

are assigned page numbers and chain numbers (through the CALC procedure) on the basis of the values of the input parameters to the CALC procedure. If these values are subsequently altered by a MODIFY command, the CALC procedure is recalled to determine if the new values of the input parameters result in a new area name, page number and/or chain number (i.e., calc-chain). If they do result in a new calc-chain, the record is unlinked from the calc-chain representing its old membership and linked into the new calc-chain following the same rules as described for initial placement. The record is no longer available through its previous calc-chain. Before the record is actually linked into the new calc-chain it is checked against member records in the new calc-chain if duplicates are not allowed.

If a subsequent MODIFY also alters the CALC input values the above procedure will be repeated.

The relinking is logical only; the record is never physically moved to a new page, and its database-key remains unchanged.

6. PAGE AND RECORD STRUCTURE

6.1. GENERAL

Conceptually, a page is a physical sub-division of an area. Once defined, it has a fixed size for the duration of the database and it contains:

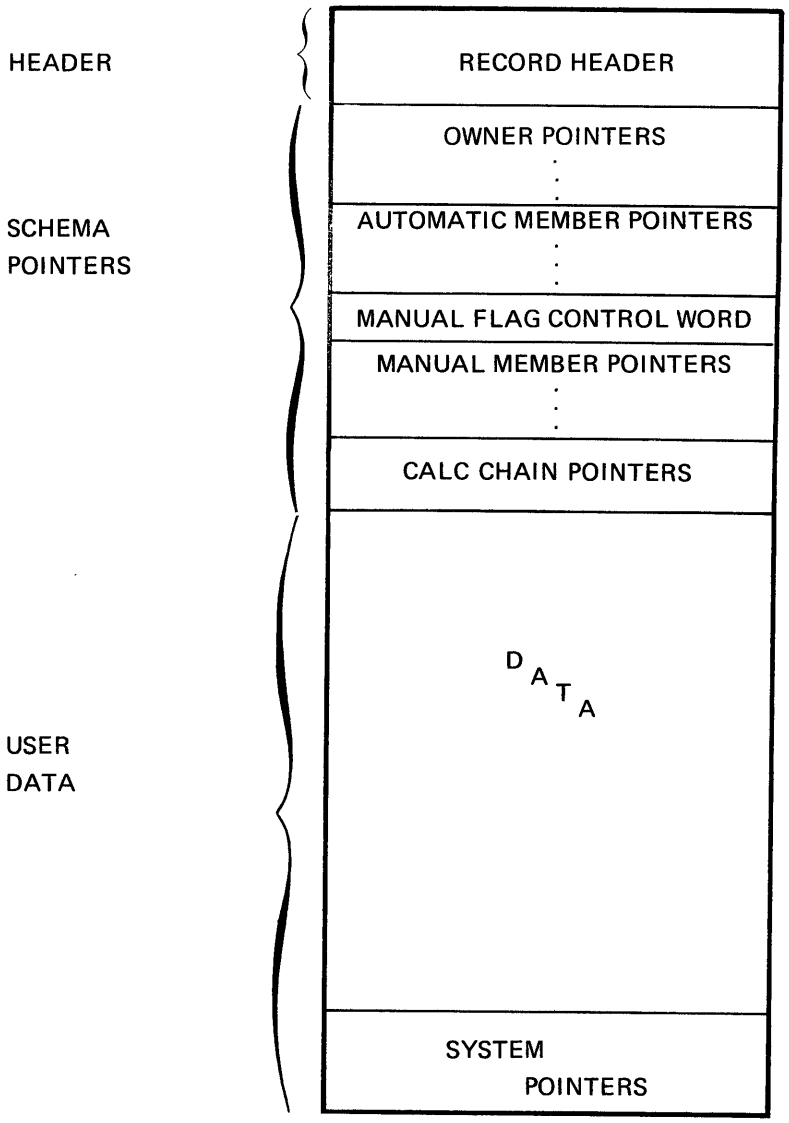
- User supplied data,
- Header information for identification,
- Pointers to overflow pages,
- An internal index, and
- System information for internal storage allocation.

In DMS 1100, where an area is implemented as a mass storage data file, a page functions as the basic unit for input/output transfers. Once retrieved, a page resides in a system main storage buffer. It remains in main storage until the space it occupies is required for another I/O operation. It is then overlaid, or, if altered, first returned to the database, and then overlaid.

During an operation involving storage or retrieval of a record, the Data Management Routine checks to determine if the page involved is presently in main storage. If it is, I/O transfer from mass storage is not required, greatly improving operational efficiency. For this reason, the Data Administrator must carefully consider his page designs and record placement logic.

6.2. DETERMINATION OF RECORD SIZE

Each record occurrence, as internally represented, contains identification information and pointers in addition to user data. The general format of each record is as follows:



HEADER — Indicates record type length, and status.

OWNER POINTERS — Links to the first (and optionally last) member of each set of which the record is an owner.

AUTOMATIC MEMBER POINTER — Links to the next (an optionally prior and/or owner) record of each set in which the record is participating as an automatic member.

MANUAL FLAG CONTROL WORD — Designates in which manual sets the record currently participates as a manual member.

MANUAL MEMBER POINTER — Links to next (and optionally prior and/or owner) record of each set of which the record is currently participating as a manual member.

CALC CHAIN LINKS	–	Links CALC record into calc-chain, one pointer for single linking, two for double.
DATA	–	User-supplied data.
SYSTEM POINTERS	–	Special purpose pointer for DMR use.

Use the following expanded formula to determine the storage space required for a particular record type.

Record Header (Fixed)	1 WORD(s)
Owner Pointers (Variable)	
Owner of _____ sets with Next links	_____
Owner of _____ sets with Prior links	_____
Owner of _____ sets with ORDER LAST	_____
Automatic Member Pointers (Variable)	
Member of _____ sets with Next links	_____
Member of _____ sets with Prior links	_____
Member of _____ sets with Owner links	_____
Manual Flag Control (present only if defined as a Manual member)	1
Manual Member Pointers (Variable)	
Number of pointers reserved	_____
Calc Chain Pointers	
Add 1 for single links, 2 for double	_____
User Data (Variable)	
System Pointers	
Add 1 if record modified and written on overflow page.	_____
TOTAL	WORDS

For example, assume a record named BENCHMARK, defined with picture of X(80), is stored on initial load into the database. The record BENCHMARK is identified in the DDL as OWNER of the ACTIVITY-TIME-COST set whose MODE IS CHAIN LINKED TO PRIOR. It is an automatic member of the JOB-PERFORMANCE set. Last, it is a manual member of 2 mutually-exclusive sets, both LINKED to OWNER, whose MODE IS CHAIN LINKED PRIOR – hence only three pointers are reserved.

Filling in the skeleton formula to calculate storage required is as follows:

Record Header	1 WORD
Owner Pointers	
Owner of 1 set with NEXT Links	1
Owner of 1 set with PRIOR Links	1
Owner of 0 set with ORDER LAST	0
Automatic Member Pointers	
Member of 1 set with NEXT Links	1
Member of 0 sets with PRIOR Links	0
Member of 0 sets with OWNER Links	0
Manual Flag Control	
Manual Member Pointers	
Number of pointers reserved	3
Calc Chain Links	0
User Data	14
System Pointers	0
TOTAL	22 WORDS

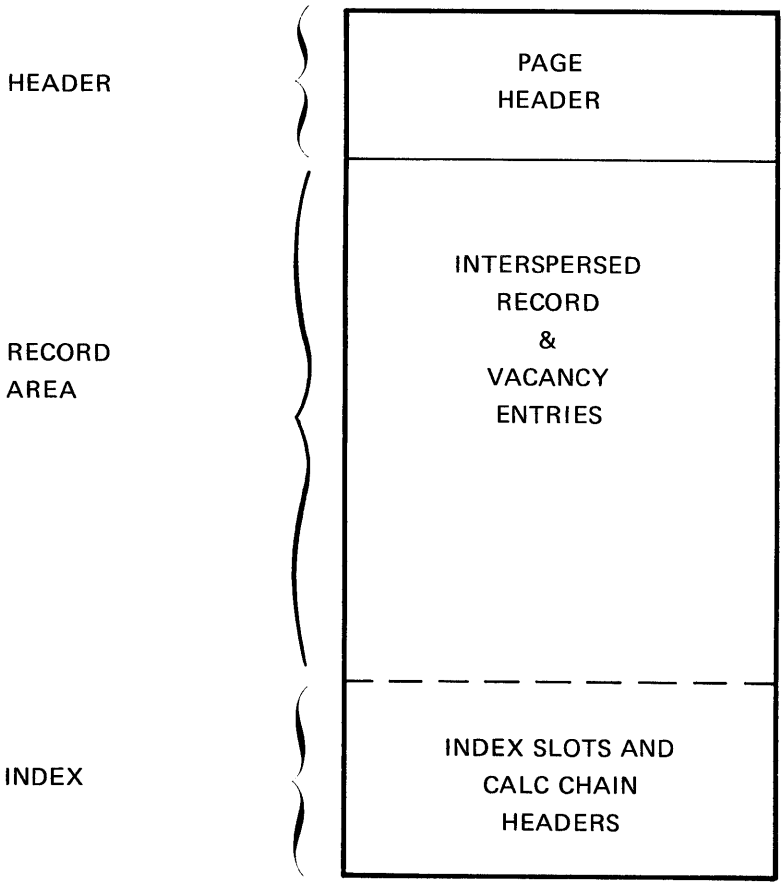
Each fixed length record occurrence of the type BENCHMARK occupies 22 words on the page on which it is stored.

6.3. DETERMINATION OF PAGE SIZE

Each page consists of three principal parts:

- A fixed length header,
- Record occurrences possibly with vacancies, and
- An internal index.

The Data Administrator should be aware of a basic difference between considerations to determine record size and page size. The former is either completely determined by the nature of the data, or variable within fixed limits as in the case of variable length records. Hence, the appropriate record size is essentially static. The latter, appropriate page size, depends not only on static considerations such as the number of records desired per page, but also on dynamic considerations during execution of the DMR. The order in which records are added to the database or removed from it, modifications which lengthen or shorten records, and the mix of record types at a given instant are important factors in determining page size. This is in addition to the role these functions play in operating efficiency.



PAGE HEADER — A ten word header containing identification, pointers and overflow pages, and information about page usage.

RECORD AREA — Storage area for record entries as previously described, 5.2.

INDEX SLOTS — Expandable index of one word entries, one entry for each record on the page and at times for records stored externally.

CALC CHAIN HEADERS — One word for each calc-chain declared on CALC USES clause minus 1. Note that the first calc chain header per page is allocated out of the page header.

Use the following expanded formula to determine the minimum storage required for a particular group of records to be stored on one page. Note that the formulation is valid only for initial load.

NOTE:

Page size must be a multiple of 28 words.

Page Header (Fixed)	10 WORD(s)
Record Area (Variable)	
(Average record size times number of records desired)	_____
Index Slots (variable) (One word for each record)	_____
Calc-chain Headers (one word each)	_____

SUBTOTAL
TOTAL = SUBTOTAL rounded to next higher multiple of 28.

Example:

Assume that because of the nature of his application, the Data Administrator needs three types of records stored on a page. There are 10 type A records of 230 words each and 170 type C records each 28 words long. Type B records are of variable length, from 15 to 25 words each, and at least 100 are to be stored per page. All record sizes are calculated, as shown by 5.2.

Then, the previous format gives:

Page Header	10 WORD(s)
Record Area	
(10 x 230 + 170 x 28 + 100 x 25)	9560
Index Slots	280
Calc-chain Headers	0
SUBTOTAL	9850
TOTAL	= 9856 (352 x 28)

NOTE:

If the Data Administrator has additional information about the distribution of Type B records, he may be able to reduce the preceding figure.

This example does not allow space for expanding record occurrences through future modifications, nor does it allow for duplicates. If these occur, the page size must be increased.

The DMR records all unused space on a page and will reorganize a page to accommodate a new record it at all possible. Ordinarily new records are stored sequentially in the record area of a page. If the DMR finds insufficient space remaining in the record area to write a new record, it will check for a vacancy entry large enough for the new record. If no single vacancy is large enough, the page will be compacted. Records will be shifted in the page and vacancy entries accumulated to make space. Similarly, if an existing record is expanded through modification, records within the page will be shifted to accommodate the expanded record if sufficient space is known to exist on the page.

See Section 5, RECORD PLACEMENT STRATEGY, for a complete discussion of page selection and the treatment of records for which space is not available on the page.

7. DATABASE LOADING

7.1. GENERAL

The purpose of this section is to suggest to the Data Administrator several steps which may be taken to speed the loading of a database. It is not intended to be an exhaustive discussion of loading techniques.

Prior to loading a database, the run unit must either, execute an OPEN for INITIAL LOAD to initialize pages, or specify PRE-INITIALIZATION pages on the schema ALLOCATE clause and run the DMS Utility Routine for initializing pages before executing an OPEN command. Initialization of pages sets all pages in an AREA to contain only a page header with the remainder of the page set to zero. Opening an AREA for INITIAL LOAD is as follows:

(1) gives the run unit the right of exclusive use within the areas named on the OPEN command;

(2) fills data and overflow pages only to the load factor specified in the schema area entry; and

(3) uses overflow pages only when storing CALC, not DIRECT or VIA SET.

If PRE-INITIALIZED pages have not been specified, the pages will be initialized during the OPEN for INITIAL LOAD. In this case, pages are initialized as they are referenced by DML commands; that is, all pages up to the highest page referenced are initialized. All remaining un-initialized pages in an area OPEN for INITIAL LOAD are initialized by a CLOSE or a DEPART command. The FREE command on OPEN for INITIAL LOAD, when before or after LOOKS have been specified in the 'ALLOCATE' clause for a particular area or areas, causes the area code, LOOKS specification, and page number of the last page initialized for each area for INITIAL OPEN LOAD to be written to the System File and to the Audit Trail tape. This information will be used should Recovery become necessary. Recovery will re-establish the database to the state just previous to the last FREE command.

The Data Administrator may save time in loading his database if he makes a differentiation between a Schema used for loading, and one used for processing. In keeping with the discrimination made between Data Administrator and Application Programmer (i.e., that the Data Administrator, because of his responsibilities, must have a knowledge of the database structure whereas the Application Programmer, to a great extent, does not), the load-schema should be used only by the Data Administrator to load his database. The Application Programmer may then operate within the framework of the processing-schema. The load-schema does not (perhaps cannot) have the degree of data independence which the processing-schema must have. It should be kept in mind, however, that the schemas must be compatible (e.g., the set linkages and hierarchical structures must be the same).

The optional EXPANDABLE phrase on the DDL ALLOCATE clause allows the Data Administrator to re-state the ALLOCATE clause parameters, to increase the number of pages in an area up to the limit specified by integer–11 of the ALLOCATE clause. There are two basic rules that apply when expanding Areas:

(1) The DMS Utility Routine for initializing pages must be used to initialize the new pages in the area before the issuance of the OPEN command; and

(2) The re-stated ALLOCATE clause is constrained by the original and subsequent ALLOCATE clauses for an Area, in that pages previously initialized must not have their Data or Overflow status changed.

The following examples illustrate area expansion:

The original clause reads:

ALLOCATE 10 PAGES EXPANDABLE TO 100;

giving 10 data pages and setting the number of page bits in the Data Base Key to allow for 100 pages.

The following are area expansion examples:

(1) ALLOCATE 50 PRE-INITIALIZED EXPANDABLE 100; giving 50 data pages

(2) ALLOCATE 50 PRE-INITIALIZED 40 OVERFLOW AT END EXPANDABLE 100; giving 10 data 40 overflow.

(3) ALLOCATE 50 PRE-INITIALIZED 10 OVERFLOW EVERY 15 DATA EXPANDABLE 100; giving 2 sets of 15 data 10 overflow

(4) ALLOCATE 50 PRE-INITIALIZED 10 OVERFLOW AT END 5 OVERFLOW EVERY 15 DATA PAGES EXPANDABLE 100; giving 2 sets of 15 data 5 overflow, with 10 overflow at end.

In each of the above cases, the DMS Utility Routine for initializing pages would have to be called passing eleven as the page # to start initializing pages. Page initialization will always initialize pages from the starting page # passed (or 1 if none is passed) to the 'integer–2' specification on the ALLOCATE Clause.

If in the above example the Area were to be expanded further to 100 pages, this expansion would be limited again by the constraints of the previous ALLOCATE Clause.

(1) above could be expanded to have either 50 more DATA pages or 50 OVERFLOW pages.

(2) above could be expanded to have 2 sets of 10 DATA 40 OVERFLOW, or 90 OVERFLOW AT END.

(3) above could be expanded by leaving the clause following the ALLOCATE 100 as is, giving 4 sets of 10 OVERFLOW EVERY 15 DATA, or putting in an AT END OVERFLOW value of 50 or 25.

(4) above could be expanded only by changing AT END OVERFLOW to 60.

The following paragraphs will illustrate the techniques which can be used to devise a schema for the most efficient database loading. Clearly, the main advantage of such techniques (time savings) will be more significant when a large database is loaded. This time spent in devising a load-schema can perhaps be better utilized elsewhere if the database is a relatively small one.

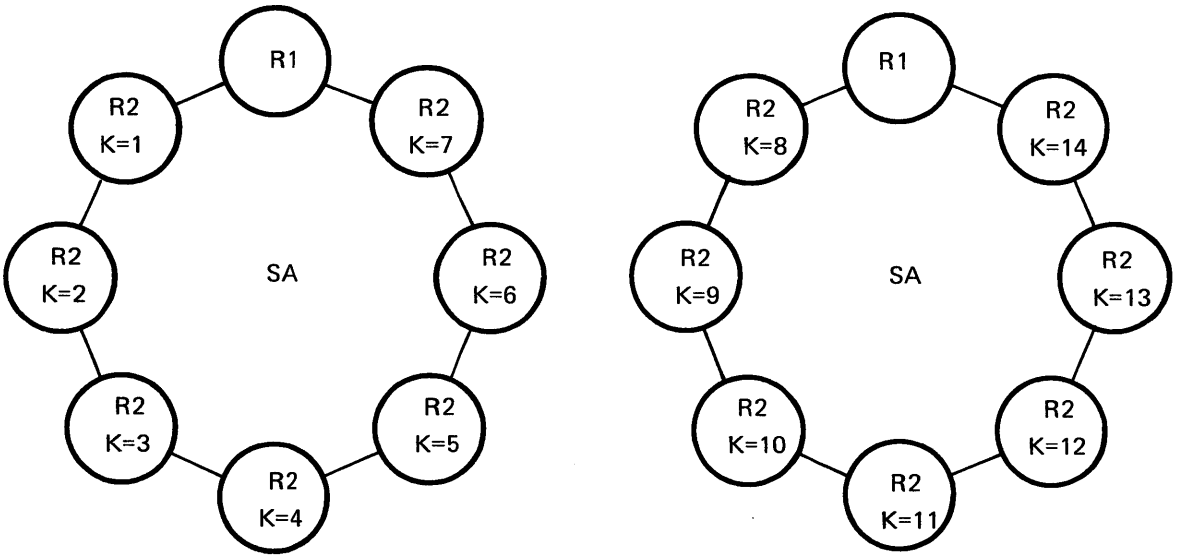
7.2. TECHNIQUES FOR EFFICIENT LOADING

The following example will serve to illustrate two efficiency techniques. Consider this partial schema.

Example:

```
                    :
RECORD NAME IS R1 RECORD CODE IS 1
     LOCATION MODE IS DIRECT ...
                    :
RECORD NAME IS R2 RECORD CODE IS 2
     LOCATION MODE IS VIA SA SET
     02 K PIC IS X(3)
                    :
SET  NAME IS SA SET CODE IS 1
     MODE IS CHAIN
     ORDER IS SORTED
     OWNER IS R1
     MEMBER IS R2 AUTOMATIC
     ASCENDING KEY IS K DUPLICATES ARE NOT
     ALLOWED
     SET OCCURRENCE SELECTION IS THRU LOCATION
     MODE OF OWNER
                    :
```

The layout of the database desired for processing would look like this:

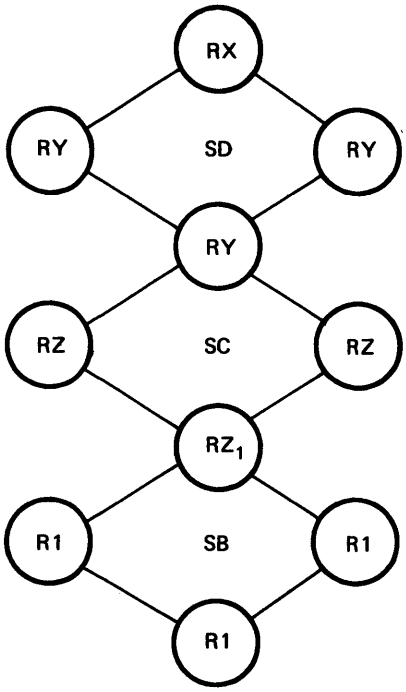


The process of storing these records using the schema described preceding is generally quite efficient. However, any time-saving techniques may make a considerable difference cumulatively when used to store a large database. One of these techniques concerns the finding of the logical insert point (here in set SA) and another concerns the placement strategy involved in the physical storing of records.

- The process of finding the logical insert point for each R2 in SA can be facilitated by pre-sorting the R2 records (in descending order) and changing the set order to SA to first. This technique dispenses with any logical tracking through of SA which may have been necessary.

- The placement strategy (i.e., that logic which is used to determine the physical location at which the record will be stored) involved in the storing of records whose location mode is via set may also be done more expeditiously. In certain circumstances it may be possible to store many of these records direct. For instance, in the preceding example assume the Data Administrator desired to store all the records in the first occurrence of set SA on page one of a given area and the records of the second occurrence of set SA on page two. Assume pages of 112 words in length and records of 10 words. Assume, also, that the R1 records were each stored direct; hence, their database-keys are known. One each page, then, the R2 records may be stored direct in turn by incrementing the record number of that page's R1 record. This technique is especially useful in the case of a multi-leveled data hierarchy because each store via set necessitates stepping down the hierarchy until the object set is reached.

- A third technique may be used in which all members of a set are stored immediately following their owner record. This would allow the SET OCCURRENCE SELECTION clause of this member entry in the particular set description of the load-schema to be changed from Location Mode of Owner (as in the processing-schema) to Current of Set. This technique eliminates path-building and hierarchy-tracking above the owner record which would be necessary in using the processing-schema. In the following example, suppose RZ_1 has just been stored, and the R1 member entry in the RZ set description has a Set Occurrence Selection of Current of Set. Then the R1 records may be stored next, and the hierarchy above RZ_1 may be ignored.


RX
RY
SD
RY
RY
RZ
SC
RZ
RZ1
R1
SB
R1
R1

APPENDIX A. DDL RESERVED WORD LIST

A.1. GENERAL

The following is a list of DDL reserved words. These words must only be used in the manner intended as shown by the respective syntax skeleton. If used in any other manner, a DDL error diagnostic will result.

A.2. RESERVED WORD LIST

AFTER-LOOKS
ALIAS
ALLOCATE
ALLOWED
ARE
AREA
AREA-KEY
AREA-NAME
AREAS
ASCENDING
ASCII
AT
AUTOMATIC
BEFORE-LOOKS
BLOCKS
CALC
CHAIN
CHAINS
CODE
COMP
COMP–4
COMPUTATIONAL
COMPUTATIONAL–4
CONTROL
CURRENT
DATA
DATABASE-KEY
DATABASE-KEYS
DEPENDING
DESCENDING
DIRECT
DISP
DISP–1
DISPLAY
DISPLAY–1
DIVISION
DUPLICATES
END
EVERY
EXPANDABLE
FIELDATA
FILE
FILLER
FIRST
FOR
IDENTIFICATION
IN
INCLUDE
INSERT
INTERVAL
IS
KEY
LAST
LINKED
LOAD
LOCATION
LOCK
LOOKS
MANUAL
MAPS
MEMBER
MODE
NAME
NEXT
NO-LOOKS
NOT
OCCURRENCE
OCCURS
OF
ON
ORDER
OVERFLOW
OWNER
PAGES
PERCENT
PIC
PICTURE
POINTERS
PRE-INITIALIZED
PRIOR
QUICK-BEFORE-LOOKS

RANGE
RECORD
RECORD-NAME
RECOVERY-POINTS
RESERVE
SCHEMA
SEARCH
SECTION
SELECTION
SET
SORTED
TEMPORARY
THROUGH
THRU
TIMES
TIP
TO
USAGE
USES
USING
VIA
WITHIN
WORDS

APPENDIX B. DDL SYNTAX SKELETON

B.1. GENERAL

Following is a complete DDL Syntax Skeleton.

B.2. SYNTAX SKELETON

IDENTIFICATION DIVISION.

SCHEMA NAME IS *schema-name*.

[; TIP FILE CODE IS *integer–0*] .

DATA DIVISION.

AREA SECTION.

[AREA CONTROL IS *integer–1* AREAS.]

[AREA LOOKS INCLUDE { QUICK-BEFORE-LOOKS / BEFORE-LOOKS / AFTER-LOOKS / NO-LOOKS }]

[RECOVERY-POINTS ARE EVERY *integer–2* BLOCKS.]

AREA NAME IS *area-name–1* ;

AREA CODE IS *integer–1* ;

[AREA MAPS TO TIP FILE ;]

ALLOCATE *integer–2* [PRE-INITIALIZED] PAGES

[, *integer–10* OVERFLOW PAGES AT END]

[, *integer–3* OVERFLOW PAGES EVERY *integer–4* DATA PAGES]

[, EXPANDABLE TO *integer–11* PAGES] ;

PAGES ARE *integer–5* WORDS

```
[ ; LOOKS  INCLUDE  { || QUICK-BEFORE-LOOKS || } ]
                    { || BEFORE-LOOKS       || } 
                    { || AFTER-LOOKS        || }
                    {    NO-LOOKS              }

[ ; LOAD  IS  { integer-6  WORDS   } ]
              { integer-7  PERCENT }

[ ; CALC  USES  integer-8  CHAINS  [LINKED  PRIOR] ] .

RECORD  SECTION .

RECORD  NAME  IS  record-name-1 ;

    RECORD  CODE  IS  integer-1 ;

                          { DIRECT  data-base-data-name-1 ,  data-base-data-name-2 }
                          {                                                         }
                          { CALC  data-base-procedure-name-1                        }
                          {     IN  data-base-data-name-3                           }
    LOCATION  MODE  IS    {     USING  data-base-identifier-1                       }
                          {          [ , data-base-identifier-2] ...                }
                          {     DUPLICATES  ARE  [NOT]  ALLOWED                     }
                          {                                                         }
                          { VIA  set-name-1  SET                                    }
                          {     [INTERVAL  IS  integer-2  PAGES]                    }

    [ { ; WITHIN  area-name-1  [ { integer-3             }  { THRU    }
                                 { data-base-data-name-4 }  { THROUGH }

                                 { integer-4             } ] } ... ]
                                 { data-base-data-name-5 }

    [ ; RESERVE  integer-5  POINTERS]

    [ ; RECORD  MODE  IS  { FIELDATA } ]
                          { ASCII    }

  [ { ; { PIC     }  IS  character-string-1 } ]
  [ {   { PICTURE }                          } ]
  [ {                                        } ]
  [ { ; [ { "Item description" }  ... ]      } ]
```

- **Item Description Syntax Skeleton**

```
Level-number-1 { data-base-identifier-3 }
               { FILLER                 }

[ ; { PIC     } IS character-string-2 ]
    { PICTURE }

[                { LOCK              } ]
[                { DISPLAY           } ]
[                { DISP              } ]
[                { DISPLAY-1         } ]
[                { DISP-1            } ]
[                { COMPUTATIONAL     } ]
[ ; USAGE IS     { COMP              } ]
[                { COMPUTATIONAL-4   } ]
[                { COMP-4            } ]
[                { AREA-KEY          } ]
[                { AREA-NAME         } ]
[                { DATABASE-KEY      } ]

[ ; OCCURS { integer-6 TIMES                                  } ]
[          { integer-7 TO integer-8 TIMES                     } ]
[          {     DEPENDING ON data-base-identifier-4          } ]

SET SECTION.

SET  NAME IS set-name-1 ;

    SET CODE IS integer-1 ;

    MODE IS CHAIN [ LINKED PRIOR];

              { FIRST                               }
              { LAST                                }
    ORDER IS  { NEXT                                } ;
              { PRIOR                               }
              { SORTED [WITHIN RECORD-NAME]         }

    OWNER IS record-name-1;

    { "Member sub-entry" } . . .
```

■ **Syntax Format of Member Sub-entry**

```
MEMBER IS record-name–2 { AUTOMATIC } [LINKED TO OWNER]
                        { MANUAL    }

[ { { ASCENDING  } [RANGE] KEY IS || RECORD-NAME                            || } ...
[ { { DESCENDING }                || data-base-identifier–1                 || }
[                                 || [,data-base-identifier–2] ...          || }

        DUPLICATES ARE { FIRST         } ] ;
                       { LAST          } ]
                       { NOT ALLOWED   } ]

{"Set occurrence selection sub-entry"}
```

■ **Syntax Format of Set Occurrence Selection Sub-entry**

Format–1:

```
;SET OCCURRENCE SELECTION IS THRU
{ CURRENT OF SET           }
{ LOCATION MODE OF OWNER   }
  [ { USING data-base-identifier–3 [,data-base-identifier–4] ...                        } ] .
  [ { {ALIAS { FOR data-base-identifier–5 } IS data-base-data-name–2 } ...              } ]
  [          { OF data-base-data-name–1   }                                               ]
```

Format–2:

```
;SET OCCURRENCE SELECTION IS THRU set-name–2 USING
{ CURRENT OF SET           }
{ LOCATION MODE OF OWNER   }
  [ { USING data-base-identifier–6 [,data-base-identifier–7] ...                        } ]
  [ { {ALIAS { FOR data-base-identifier–8 } IS data-base-data-name–2 } ...              } ]
  [          { OF data-base-data-name–1   }                                               ]
{ set-name–3                                                                              }
{   { USING data-base-identifier– 9 [,data-base-identifier–10] ...                   }    } ...
{   { {ALIAS FOR data-base-identifier–11 IS data-base-data-name–3} ...               }    }
```

APPENDIX C. DDL TRANSLATOR CONTROL CARD

C.1. GENERAL

The control card for the DDL Translator follows standard conventions established for 1108 Language Processors. (See *UNIVAC 1100 Series Operating System Programmer Reference, UP–4144* (current version).) If not part of a system library, the DDL Translator may be loaded into a program file from tape using an appropriate FUR/PUR command.

C.2. DDL CONTROL CARD FORMAT

The format of the DDL Translator Control Card is:

@file.DDL,options Element1, Element2, Element3

where:

file – is the name of the program file containing the absolute DDL element. This name and trailing period may be omitted if the program file is TPF$ or LIB$.

DDL – is the name of the absolute element containing the Data Definition Language Translator.

options – specify processing alternatives for the DDL Translator. The options are:

I – source input is from cards or card images in the control stream.

U – source input is to be updated producing new cycle of input element.

Neither I nor U – I option is assumed if Element1 is blank. Otherwise a new source element is created from Element1 and update cards and the updated source is stored in Element3.

L – produce a complete listing of (updated) source input and any diagnostics.

W – produce a separate listing of update cards.

B — ignore sequence numbers in columns 1–6.

R — produce the schema report.

NOTE:

The "I" and "U" options are mutually exclusive; but it is meaningful to omit both. The "W" option is ignored if it appears with the "I" option. In the absence of the "L" option only summary diagnostics will be produced.

Each of the three following fields (ELEMENT1, ELEMENT2, ELEMENT3) specify program file elements in the standard format. (See *UNIVAC 1100 Series Operating System Programmer Reference, UP–4144* (current version).)

- ELEMENT1 — Program file element name of symbolic input. If present, and I option is not present, the lines immediately following the control statement are taken to be corrections to the source language element. If an I option is present, then the lines following the control statement are given to the processor and are inserted into the program file as well. Hence, not required if input is from cards.

- ELEMENT2 — Specifies the object schema produced by the DDL Translator. Must be of the form Schema-file.SCHEMA. The file referred to as the schema-file must be assigned. The element name must be SCHEMA.

- ELEMENT3 — Specifies the source language element produced by updating the input source language element. If this field is void, no updated source language element will be produced, unless a 'U' option is specified. In that case, an updated element is produced with the same name and version as the input element, but with a cycle number one greater. Need not be specified when 'I' option used.

APPENDIX D. DDL TRANSLATOR ERROR MESSAGES

D.1. GENERAL

Following is the list of diagnostics output by the DDL Translator. The diagnostics are classified as ERROR or FATAL ERROR. An ERROR diagnostic occurs when the user has violated the form or content of the symbolic schema; the violation, although serious, does not inhibit the construction of the absolute schema. However, the validity of the absolute schema is questionable. A FATAL ERROR diagnostic occurs when the user has violated the form or content of the symbolic schema to the extent that an absolute schema cannot be constructed.

During the syntax analysis pass by the DDL, both syntactical and logical diagnostics may be produced. A diagnostic produced during the syntax analysis pass appears in the listing directly above the line of symbolic schema to which it refers. An asterisk appears directly above the first character of the clause or word specifically referred to by the diagnostic. If no fatal errors are encountered during the syntax analysis, the DDL performs additional logic verification of the collective symbolic input. All diagnostics produced during the latter logic verification are FATAL and are listed directly following the end of the symbolic schema listing.

A clause or entry is flagged as being misplaced when its occurrence in the symbolic schema is past its required position. When a clause or entry occurs earlier than its required position in the symbolic schema, 'MISSING' diagnostic(s) are generated for each clause which should occur between the clause appearing too early, and the clause previous to it. Fatal errors may result if any such 'MISSING' clause is the schema name, and/or an area entry, and/or a record entry. A clause flagged as misplaced with be flagged missing if it does not appear in its required position.

In the event of an abnormal termination of the DDL Translator, the user should resubmit the run. The DDL Translator terminates abnormally whenever it cannot recover from a hardware or software error encountered by a system routine which it calls.

D.2. ERROR MESSAGES

Error Messages	Classification
ALLOCATE CLAUSE IS IN ERROR. ALLOCATED PAGES EXCEEDS MAX. – 99,999 is stored for integer–2.	Error
ALLOCATE CLAUSE IS IN ERROR. ALLOCATED PAGES IS GREATER THAN OR EQUAL TO EXPANSION – INTEGER–11 is assumed to be zero.	Error

Error Messages	Classification
ALLOCATE CLAUSE IS IN ERROR. ALLOCATED PAGES IS ZERO.	Fatal Error
ALLOCATE CLAUSE IS IN ERROR. (ALLOCATED PAGES – [AT END OVERFLOW])/(SPACED OVERFLOW & DATA) PRODUCES REMAINDER. – Remainder is added to global overflow.	Error
ALLOCATE CLAUSE IS IN ERROR. AT END OVERFLOW EXCEEDS OR EQUALS ALLOCATED PAGES. – Integer–10 is assumed to be zero.	Error
ALLOCATE CLAUSE IS IN ERROR. EXPANSION INTEGER IS ZERO.	Error
ALLOCATE CLAUSE IS IN ERROR. OVERFLOW PAGES IS ZERO.	Error
ALLOCATE CLAUSE IS IN ERROR. SPACED DATA PAGES IS ZERO. – Integer–3 and Integer–4 are assumed to be zero.	Error
ALLOCATE CLAUSE IS IN ERROR. SPACED OVERFLOW + DATA EXCEEDS ALLOCATED PAGES [–AT END OVERFLOW]. – Integer–3 and Integer–4 are assumed to be zero.	Error
AREA CODE IS NOT UNIQUE	Fatal Error
AREA-KEY USAGE NOT ALLOWED IN LEVEL 49 ENTRY.	Fatal Error
AREA NAME IS NOT UNIQUE.	Fatal Error
AREA NAME IS TOO LONG. TRUNCATED TO 12 CHARACTERS.	Error
BADLY FORMED PICTURE. – Character content and/or sequence is illegal.	Error
CALC INPUT IDENTIFIER-NAME IN RECORD TYPE RECORD-NAME IS A VARIABLE LENGTH ITEM AND/OR FOLLOWS A VARIABLE LENGTH ITEM.	Fatal Error
CODE IS ZERO. – Area code is zero. – The record code is zero. – The set code is zero.	 Fatal Error Fatal Error Fatal Error
COMMENTS SHOULD NOT APPEAR FIRST. – The symbolic schema began with a comment.	Error
COMMENTS SHOULD NOT APPEAR LAST. – The symbolic schema ended with a comment.	Error

Error Messages	Classification
CONSTRUCTION IS TOO LONG. – The construction exceeded 130 characters. Syntax analysis resumes at the following A-margin entry.	Fatal Error
CONTINUATION BEGINS IN THE A-MARGIN.	Error
CONTINUATION IMAGE IS BLANK.	Error
CORRECTION SEQUENCE ERROR: card image – A correction to the symbolic schema is out of sequence. The diagnostic will appear before the listing of the symbolic schema.	Error
CORRECTION SEQUENCE NUMBER ERROR.	Error
DATA NAME IS TOO LONG. TRUNCATED TO FIVE WORDS.	Error
DUPLICATE IMAGE. – Two or more consecutive clauses are identical.	Error
#ERROR(S). VALIDITY OF SCHEMA QUESTIONABLE. – Output at the end of the listing of the symbolic schema.	
#FATAL ERROR(S). UNABLE TO BUILD SCHEMA. – Output at the end of the listing of the symbolic schema.	
FILLERS CANNOT BE GROUP ITEMS.	Error
FILLERS CANNOT HAVE AN OCCURS COUNT.	Error
FILLER CANNOT HAVE A USAGE CLAUSE.	Error
ILLEGAL CHARACTER. – An illegal character (@ # = & $ * % : ? ! ' / ≠) was found between columns 8 and 72 inclusive. Syntax analysis resumes at the following A-margin clause.	Fatal Error
ILLEGAL CHARACTER – COMMENT ASSUMED. – A character other than '*' or '/' was found in column 7.	Error
ILLEGAL INTEGER VALUE. – When integer–9 of the 'AREA CONTROL' clause is greater than 4000, the value stored is 4000. – Integer–6 of the 'OCCURS' clause exceeds the maximum, 262143. The value is truncated to 18 bits. – When integer–2 of the 'INTERVAL' clause is greater than the maximum, 99999, the clause is ignored. – Integer–2 of the 'INTERVAL' clause is zero. – Integer–0 of the 'TIP FILE CODE' clause is zero or exceeds 4000.	Error Error Error Error Fatal Error

Error Messages	Classification
– Integer–5 of the 'PAGES' clause is less than 28. 28 is assumed.	Error
– Integer–5 of the 'PAGES' clause is not a multiple of 28 words. The remainder in the division of integer–5 by 28 is discarded.	Error
– Integer–8 of the 'CALC USES' clause is zero. One chain total is assumed per page.	Error
– Integer–8 of the 'CALC USES' clause exceeds the maximum value of (page size minus page header +1). One chain total is assumed per page.	Error
– Integer–8 of the 'CALC USES' clause exceeds the maximum bit value. The value stored is the maximum value which can be stored in the smaller of these bit values: (1) 15 bits (2) 36 bits – page bits – area bits	Error
– Integer–5 of the 'RESERVE' manual pointers clause is zero.	Error
ILLEGAL INTEGER VALUE. LIMITS IGNORED.	
– When integer–4 (upper bound) of the 'WITHIN' clause is greater than the maximum, 99999, both integer–3 and integer–4 in the clause are ignored.	Error
– When integer–3 (the lower bound) of the 'WITHIN' clause is greater than integer–4 (the upper bound), both integer–3 and integer–4 are ignored.	Error
– When integer–4 (upper bound) of the 'WITHIN' clause is zero, both integer–3 and integer–4 in the clause are ignored.	Error
ILLEGAL INTEGER VALUE. SET TO MINIMUM.	
– When integer–9 of the 'AREA CONTROL' clause is less than 127, the value stored is 127.	Error
ILLEGAL INTEGER VALUE. TRUNCATED.	
– Integer–7, the low occurs count, exceeds the maximum, 262142. The value is truncated to 18 bits.	Error
– Integer–8, the high occurs count, exceeds the maximum, 262143. The value is truncated to 18 bits.	Error
– Integer–5 of the 'RESERVE' clause exceeds the maximum, 63. The integer is truncated to 6 bits.	Error
– The set code exceeds the maximum, 4000. The code is truncated to 12 bits.	Error
– The record code exceeds the maximum, 4000 (12-bit maximum). The code is truncated to 12 bits.	Error
– When the area code exceeds the number of bits necessary to contain it, the code is truncated to the bit-maximum. The bit-maximum is determined from integer–9 of the area control clause. If the area control clause is not used, integer–9 is set to 4000 (12 bits maximum).	Error

Error Messages	Classification
ILLEGAL INTEGER VALUE. 1 ASSUMED.	
– When integer–3 (lower bound) in the 'WITHIN' clause is greater than the maximum (99998) the integer is set to 1.	Error
– When integer–3 (lower bound) of the 'WITHIN' clause is zero, the integer is set to one.	Error
ILLEGAL LEVEL NUMBER.	
– The level number of the item is outside the allowable range ($2 \leqslant n \leqslant 49$) Level 2 is assumed for further processing.	Error
IMPROPER CONTINUATION.	
– A continuation line followed a blank line.	Error
IMPROPER LOAD FACTOR; 100 PERCENT ASSUMED.	
– When the load factor in words exceeds the page size, 100 percent is assumed.	Error
– When the load factor is greater than 100 percent, 100 percent is assumed.	Error
IMPROPER OCCURS RANGE	
– Integer–7, the low occurs count, in the 'OCCURS' clause is greater than integer–8, the high occurs count.	Error
INAPPROPRIATE GROUP ITEM	
– The group item has a picture clause.	Error
INCORRECT NUMBER OF DIGITS FOR LEVEL NUMBER.	
– The number of digits exceeds 2 or is 0.	Error
INTERNAL ERROR-DATA NAME CODE LOST.	
– User must resubmit run.	Fatal Error
ITEM identifier-name in RECORD TYPE record-name IS USED AS AN ASCENDING/DESCENDING KEY in set-name SET, BUT IS A VARIABLE LENGTH ITEM AND/OR FOLLOWS A VARIABLE LENGTH ITEM.	Fatal Error
ITEM NAME IS NOT UNIQUE WITHIN THE RECORD.	Fatal Error
LOCATION MODE OF RECORD TYPE record-name IS VIA set-name SET, BUT record-name IS NOT A MEMBER OF set-name.	Fatal Error
MAXIMUM NUMBER OF MEMBERS EXCEEDED FOR SET.	Fatal Error
MEMBERS OF A GROUP SHOULD HAVE THE SAME LEVEL NUMBER.	
– The level of the item is assigned the value of the previous level.	Error
MISPLACED AREA CONTROL ENTRY; 4000 AREAS ALREADY ASSUMED.	Error

Error Messages	Classification
MISPLACED AREA DEFINITION.	Error
MISPLACED AREA SECTION HEADER.	Error
MISPLACED DATA DIVISION HEADER.	Error
MISPLACED IDENTIFICATION HEADER.	Error
MISPLACED RECORD DEFINITION.	Error
MISPLACED RECORD SECTION HEADER.	Error
MISPLACED SCHEMA NAME.	Error
MISPLACED SET DEFINITION.	Error
MISPLACED SET SECTION HEADER.	Error
MISSING AREA DEFINITIONS.	Fatal Error
MISSING AREA SECTION HEADER.	Error
MISSING DATA DIVISION HEADER.	Error
MISSING IDENTIFICATION DIVISION HEADER.	Error
MISSING PICTURE FOR ELEMENTARY ITEM.	Error
MISSING RECORD DEFINITIONS.	Fatal Error
MISSING RECORD SECTION HEADER.	Error
MISSING SCHEMA NAME.	Fatal Error
MORE THAN ONE ALIAS OF CLAUSE WITH ALIAS FOR CLAUSE(S).	Fatal Error
MORE THAN TWO ALIAS OF CLAUSES SPECIFIED.	Fatal Error
NO SORT KEY SPECIFIED FOR SORTED SET.	Fatal Error
N POINTERS HAVE BEEN RESERVED FOR RECORD-NAME RECORD, BUT THE RECORD'S MANUAL MEMBERSHIP IN SET-NAME SET REQUIRES M POINTERS.	Fatal Error
NUMERIC PICTURE SPECIFIES TOO MANY DIGITS. – The number of digits specified is greater than 18. The number is set to 18.	Error

Error Messages	Classification
OVERFLOW PAGES PLUS DATA PAGES GREATER THAN ALLOCATED. – The overflow and data page values are ignored.	Error
PAGE RANGE ILLEGAL WITH CALC PROCEDURE DMSCALC. PHRASE IGNORED.	Error
PICTURE IS TOO LONG. – The character string in the picture clause exceeded 30 characters.	Error
POINTERS SHOULD BE RESERVED FOR MANUAL MEMBERSHIPS – A record was specified as being a manual member in a set, but no pointers were reserved for the record in its entry in the record section.	Fatal Error
PROCEDURE NAME IS TOO LONG. TRUNCATED TO TWO WORDS.	Error
RECORD CODE IS NOT UNIQUE.	Fatal Error
RECORD ITEM NAME IS TOO LONG. TRUNCATED TO FIVE WORDS.	Error
RECORD NAME IS NOT UNIQUE.	Fatal Error
RECORD NAME IS TOO LONG. TRUNCATED TO FIVE WORDS.	Error
RECORD TYPE record-name APPEARS AS BOTH AN OWNER AND MEMBER OF SET TYPE set-name.	Fatal Error
RECORD TYPE record-name HAS EXCEEDED THE LIMIT FOR AREAS IN WHICH IT MAY BE STORED.	Fatal Error
SCHEMA DESCRIPTION TABLE EXCEEDED 64 SEGMENTS.	Fatal Error
SCHEMA NAME IS TOO LONG. TRUNCATED TO 12 CHARACTERS.	Error
SEQUENCE NUMBER ERROR.	Error
SET CODE IS NOT UNIQUE.	Fatal Error
SET NAME IN NOT UNIQUE.	Fatal Error
SET NAME IS TOO LONG. TRUNCATED TO FIVE WORDS.	Error
SYNTAX ERROR. – A required construction cannot be found. Syntax analysis resumes at the following A-margin entry.	Fatal Error

Error Messages	Classification
THE ASCENDING/DESCENDING SORT KEYS FOR MEMBER RECORD TYPES IN set-name SET DO NOT AGREE IN NUMBER AND/OR DO NOT HAVE IDENTICAL AND/OR TYPE PICTURE AND/OR USAGE CLAUSES.	Fatal Error
THE IDENTIFIER identifier-name IS DUPLICATED IN THE LIST OF USING DATA-BASE-IDENTIFIERS FOR THE CALC LOCATION MODE OF RECORD TYPE record-name.	Fatal Error
THE INTERVAL VALUE IN THE LOCATION MODE CLAUSE OF record-name RECORD IS INCONSISTENT WITH THE VALUE(S) SPECIFIED IN THE ALLOCATE CLAUSE OF AREA area-name.	Fatal Error
THE LAST SET SPECIFIED IN THE SOS CLAUSE FOR MEMBER ENTRY record-name RECORD IN set-name SET MUST BE set-name SET.	Fatal Error
THE LENGTH OF A GROUP ITEM WITH AN OCCURS CLAUSE AND MIXED DATA MUST BE A HALFWORD MULTIPLE.	Error
THE MEMBER ENTRY record-name RECORD IN set-name SET SPECIFIES ALIASES OF DATA-NAMES IN THE LOCATION MODE CLAUSE OF THE OWNER record-name OF THE SET.	Fatal Error
THE MEMBER ENTRY FOR record-name RECORD IN set-name SET SPECIFIED ALIASES OF DATA-NAMES AND/OR FOR IDENTIFIERS FOR THE OWNER (record-name) OF set-name SET. THE ALIASES ARE INCORRECT IN NUMBER AND/OR TYPE FOR THE LOCATION MODE OF THE OWNER.	Fatal Error
THE MEMBER ENTRY FOR record-name RECORD IN set-name SET SPECIFIES A USING CLAUSE FOR THE OWNER record-name OF set-name SET, BUT THE LOCATION MODE OF THIS OWNER IS NOT VIA SET.	Fatal Error
THE SETS SPECIFIED IN THE SOS CLAUSE FOR MEMBER ENTRY RECORD-NAME RECORD IN SET-NAME SET DO NOT FORM A CONTINUOUS PATH RECORD-NAME RECORD IS NOT A MEMBER OF SET-NAME SET.	Fatal Error
THE SOS CLAUSE FOR MEMBER ENTRY RECORD-NAME RECORD IN SET-NAME SET SPECIFIES LOCATION MODE OF OWNER. THE LOCATION MODE OF THE OWNER RECORD-NAME RECORD IS CALC. THE RECORD ENTRY FOR RECORD-NAME MUST HAVE A DUPLICATES NOT ALLOWED CLAUSE IN ITS LOCATION MODE IS CALC CLAUSE.	Fatal Error

Error Messages	Classification
THE SOS CLAUSE FOR MEMBER ENTRY RECORD-NAME RECORD IN SET-NAME SET SPECIFIES LOCATION MODE OF OWNER. THE LOCATION MODE OF THE OWNER RECORD-NAME RECORD IS VIA SET-NAME SET, A SORTED SET. RECORD-NAME MUST HAVE A DUPLICATES NOT ALLOWED CLAUSE IN ITS MEMBER ENTRY IN SET-NAME SET.	Fatal Error
THE SYSTEM GENERATION PARAMETER, HIARCH, WAS EXCEEDED IN BUILDING A LOCATION PATH FOR RECORD-NAME RECORD VIA SET SET-NAME.	Fatal Error
THE SYSTEM GENERATION PARAMETER, HIARCH, WAS EXCEEDED WHILE SEARCHING FOR AREAS IN WHICH RECORD TYPE RECORD-NAME COULD BE STORED.	Fatal Error
THE SYSTEM GENERATION PARAMETER, LRINUM, LIMITS THE NUMBER OF RECORD TYPES WHICH MAY CONTAIN AN INTERVAL CLAUSE. THE LIMIT OF N HAS BEEN EXCEEDED.	Fatal Error
THE USING DATA-BASE-IDENTIFIERS FOR THE CALC LOCATION MODE OF RECORD RECORD-NAME EXCEED TEN ELEMENTARY ITEMS.	Fatal Error
THIS SHOULD BE IN THE A-MARGIN. – A required construction is illegally in the B-margin.	Error
THIS SHOULD NOT BE IN THE A-MARGIN. – A required construction is illegally in the A-margin.	Error
UNDEFINED AREA NAME.	Fatal Error
UNDEFINED CONSTRUCTION – Syntax analysis is resumed at the following A-margin entry.	Fatal Error
UNDEFINED OWNER RECORD.	Fatal Error
UNDEFINED RECORD ITEM. – The diagnostic will be generated in the obvious case in which a reference was made to a record item which was not defined in a record entry in the record section. The diagnostic will also be generated if the record item in the 'ASCENDING/DESCENDING' clause was not defined in the entry of the member record and/or if a record item in the 'SET OCCURRENCE SELECTION' clause was not defined in the entry of the owner record of the corresponding set referenced in the 'SET OCCURRENCE SELECTION' clause.	Fatal Error
UNDEFINED RECORD NAME.	Fatal Error

Error Messages	Classification
UNDEFINED SET NAME.	Fatal Error
USAGE CLAUSE CONFLICTS WITH GROUP USAGE.	Error
USAGE CLAUSE CONFLICTS WITH RECORD MODE.	Error
USAGE CLAUSE CONFLICTS WITH PICTURE. PICTURE CLAUSE IGNORED.	Error
USAGE CLAUSE CONFLICTS WITH PICTURE. USAGE CLAUSE IGNORED.	Error
WITHIN CLAUSES IN RECORD ENTRY record-name DUPLICATE AREA area-name.	Fatal Error
WITHIN CLAUSE IN RECORD ENTRY record-name SPECIFIES UPPER INTEGER LARGER THAN ALLOCATED PAGES FOR AREA area-name.	Fatal Error

APPENDIX E. DATABASE-KEY FORMAT

E.1. GENERAL

A database-key is the unique identifier of a record within a database. One DATABASE-KEY is assigned to each record in a database. The database-key has two parts:

- The area-name, and
- Area-key.

DMS 1100 implements database-keys at two levels as follows:

- A high level implementation for use by the run unit, and
- A low level form for the DMR and the run unit.

The DMR converts from one level to another. Thus a database-key which is initiated by the DMR, for example after storing a record whose location mode is VIA SET, is available in both forms to the run unit. Conversely, the database-key supplied by the run unit before storing a record whose location mode is DIRECT, is passed to the DMR and converted before the record is stored.

E.2. RUN UNIT DATABASE-KEYS

A database-key consists of an area-name and an area-key. It is specified in a run unit in a pair of data-names defined in the data division of the user program.

The user may specify a database-key in the special pair of reserved data-names AREA-NAMES and AREA-KEY. They are produced by the DMLP in the WORKING or COMMON STORAGE SECTION of the user program as part of the DMCA.

The user has the option to specify his own data-names to define a database-key. In this case, the area-name must be in a 77-level single item data-name, defined with USAGE IS AREA-NAME and no picture clause. The DMLP replaces the AREA-NAME usage clause with "PIC IS X(12) USAGE IS DISPLAY" clauses. The user specified area-key must be in a 01-level COMMON OR WORKING STORAGE group item with a USAGE IS AREA-KEY and no picture clause. An area-key of a database-key consists of a page number and a record number. These are always in two DMLP produced subordinate items with "PIC IS 9(5) USAGE IS COMP" clauses for these two items. They have the reserved names PAGE-NUM and RECORD-NUM. The user specified database-key will be an 01- or 77-level data-name defined with USAGE IS DATABASE-KEY and no picture clause. The DMLP replaces the DATABASE-KEY usage clause with a "PIC IS 9(10) USAGE IS COMP" clause.

The data definition language and the data manipulation language pair data-names to specify a database-key. For example, "LOCATION MODE IS DIRECT data-name–1, data-name–2" pairs data-name–1 and data-name–2 to specify a database-key whose area-name is in data-name–2 and whose area-key is in data-name–1.

Example:

Assume the user program contains these DML statements in the WORKING-STORAGE SECTION:

```
77   ANAME USAGE IS AREA-NAME.
01   A-KEY USAGE IS AREA-KEY.
01   LINK USAGE IS AREA-KEY.
```

The COBOL source language which is produced by the DMLP would look like:

```
77   ANAME USAGE IS DISPLAY PIC X(12).
01   A-KEY USAGE IS COMP.
02   PAGE-NUM PIC 9(5).
02   RECORD-NUM PIC 9(5).
01   LINK USAGE IS COMP.
02   PAGE-NUM PIC 9(5).
02   RECORD-NUM PIC 9(5).
```

Thus, the procedure division statements:

```
MOVE 'HISTORY-FILE' TO ANAME.
MOVE 1 TO PAGE-NUM OF A-KEY.
MOVE 17 TO RECORD-NUM OF A-KEY.
MOVE 'HISTORY-FILE' TO AREA-NAME.
```

Define a database-key to the DMR for commands such as:

FIND ADMINISTRATIVE RECORD.
where location mode of ADMINISTRATIVE record type is DIRECT USING A-KEY, ANAME
FIND A-KEY,ANAME.

The last two statements have the same effect. They establish the 17th record of Page 1 of area HISTORY-FILE as the current record.

E.3. DMR DATABASE-KEYS

A database-key internal to the DMR consists of:

- An area-code,
- Page number, and
- Record number,

packed into 3 subfields of a 36-bit word. The format of the three subfields depends on schema and area descriptions. (See following Figure E–1.)

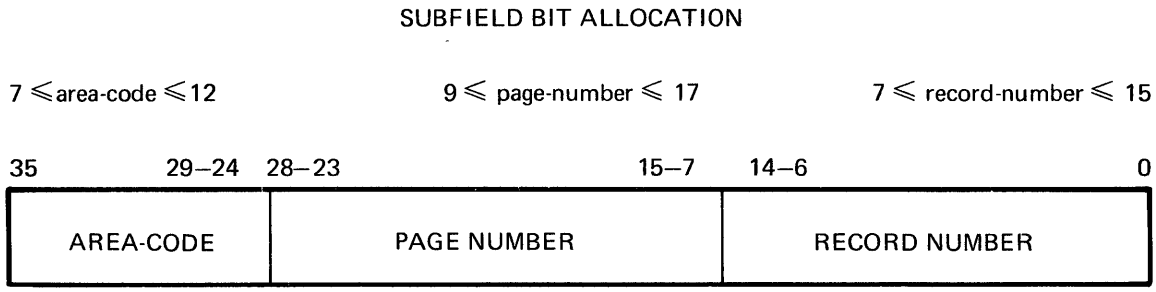


Figure E–1. Subfield Bit Allocation

Figure E–1 shows possible formats of DMR DATABASE-KEYS in an area.

The high order subfield contains the area-code. The size of the area-code field is constant throughout a schema once fixed between a minimum of 7 bits and a maximum of 12. The size of this field is determined by the number of areas specified in the AREA CONTROL clause of the DDL. However, if the number of area is ⩽127, 7 bits will be allocated. If the ALLOCATE clause is not specified, the subfield is assigned 12 bits.

The middle field contains the page number. The size of this field may vary throughout a schema, but is constant throughout an area. The size of the page number field is determined by the larger of two numbers: the number of pages specified on the DDL ALLOCATE clause, and 21 – (Number of bits for area-code). It thus has an absolute minimum in any area of 9 bits and a maximum of 17 bits.

The low order subfield contains the record number. The field must be at least 7 bits and at most 15 bits. Its size in area is the number of bits not assigned to the area code or page number fields.

E.4. USE OF DATABASE-KEYS

The DML Preprocessor provides the basic structure for database-key conversion. When a record whose location mode is DIRECT is stored or found, the DMR must be supplied with a database-key. The database-key (area-name and area-key) must be specified within the run unit before execution of a STORE or FIND command.

APPENDIX F. CALC PROCEDURE USER'S GUIDE

F.1. GENERAL

A CALC Procedure is a user supplied subprogram to generate area-names, page numbers and calc-chain numbers to govern the placement and retrieval of occurrences of specified record types. The Assembly language CALC Routine executes independently of the DMR.

If the user does not wish to write his own CALC procedure, the system supplied CALC procedure, DMSCALC, may be used. See Appendix H for using DMSCALC.

F.2. INTERFACE BETWEEN DMR AND CALC PROCEDURE

- INPUT

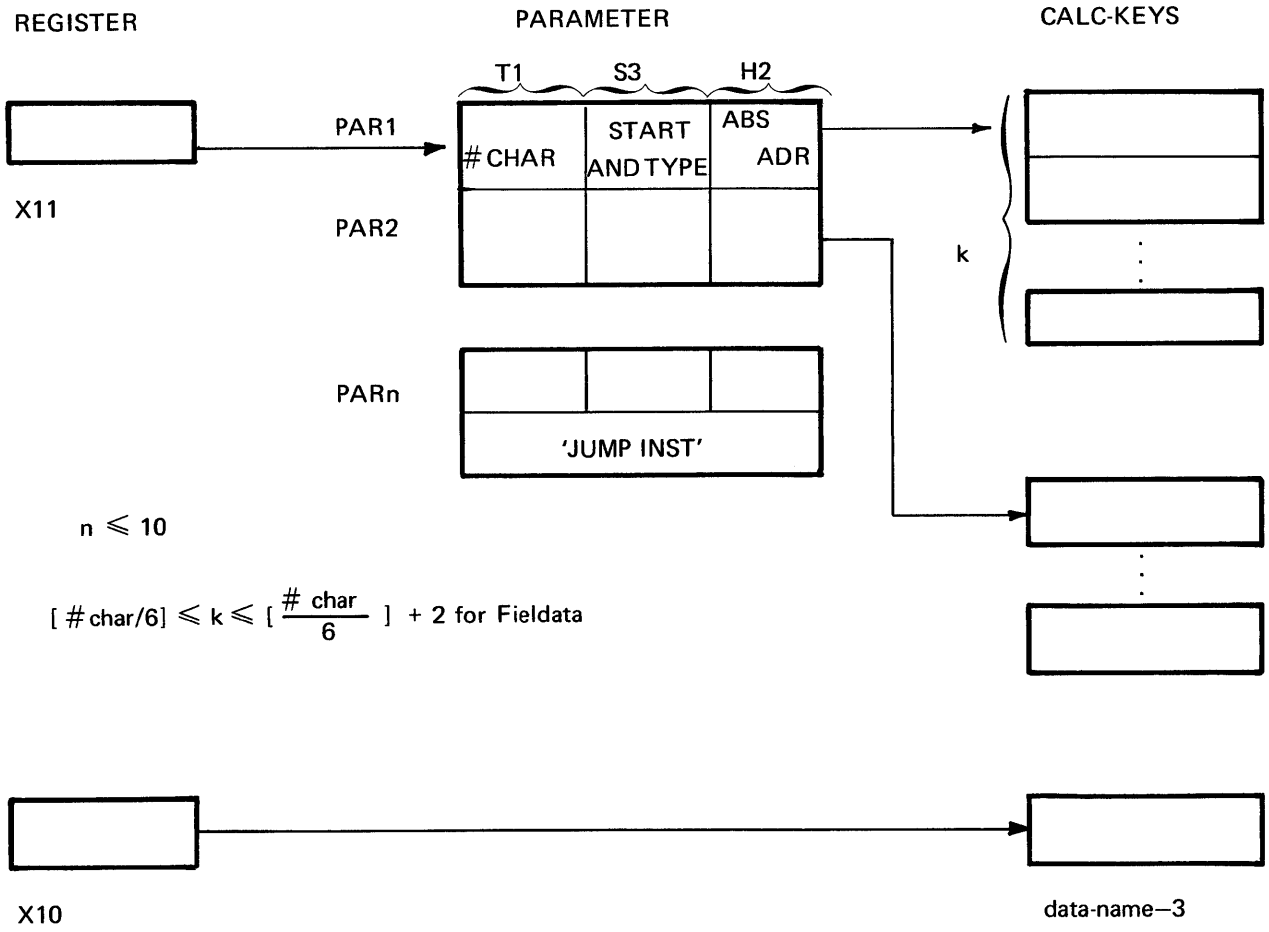
Input to a CALC Procedure is a parameter list containing addresses of the CALC keys (arguments to the CALC Procedure needed by the user to generate a page number and calc-chain number), the address of a data-name to be initialized with an area-name, and a return jump instruction to the calling routine (DMR).

On entrance, Register X11 contains the address of the first entry in the parameter list. The number of parameters, n, in the parameter list is dependent of the number of 'data-names' following the USING clause of the DDL Record Description. (See 3.4.4, "LOCATION MODE IS CALC" clause.) The word following the last parameter in the parameter list is a 'JUMP' instruction back to the DMR (i.e., jump instruction is at (X11) +n). The maximum number of parameter words allowed is 10, not including the 'JUMP' instruction. If the data-base-identifier in the USING clause is a group item, one parameter is generated for each elementary item in the group.

The format for a parameter word is:

T1	=	item length (number of characters).
Bit 23	=	0 PARAMETER IS FIELDATA (6 bit) CHARACTERS
	=	1 PARAMETER IS ASCII (9 bit) CHARACTERS
Bit 22–18	=	starting character position (i.e., number from 1 to 6 corresponding to S1 to S6 for first fieldata character or Q1 to Q4 for first ASCII character).
H2	=	absolute address of the first word containing the starting character of the data item specified as an argument (CALC Key) for the CALC Procedure. (In the DDL Record Section 3.4.4 under "LOCATION MODE", data-base-identifier–1 [,data-base-identifier–2], etc., are the data items referred to.)

Also on entrance, Register X10 points to a data-name which the CALC Procedure may initialize with an Area Name. Each time the CALC Procedure is referenced, the Area Name may be changed, if desired. The data name pointed to is data-base-data-name–3 from the DDL clause LOCATION MODE IS CALC data-base-procedure-name–1 IN data-base-data-name–3 USING. . .



Example:

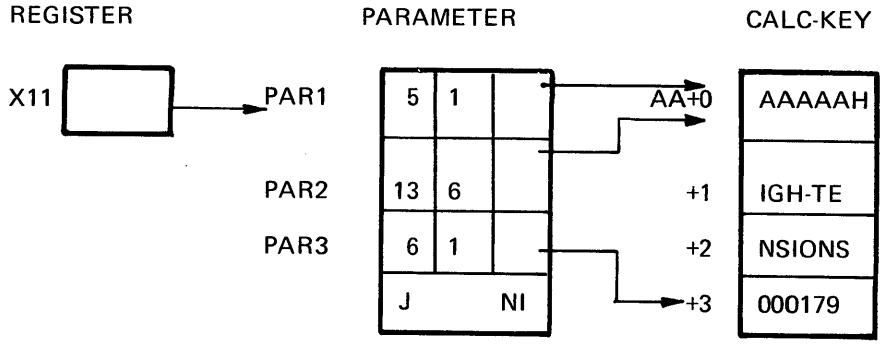
```
LOCATION MODE IS CALC LOCATOR IN ANAME USING PAR1, PAR2, PAR3
    DUPLICATES ARE NOT ALLOWED

WITHIN HISTAREA 1 THRU 20, WITHIN ADMINAREA 5 THRU 17
MODE FIELDATA
```

where, assume that:

PAR1 has PIC X(5) and contains 'AAAAA'
PAR2 has PIC X(13) and contains 'HIGH-TENSIONS'
PAR3 has PIC 9(10) and contains 179

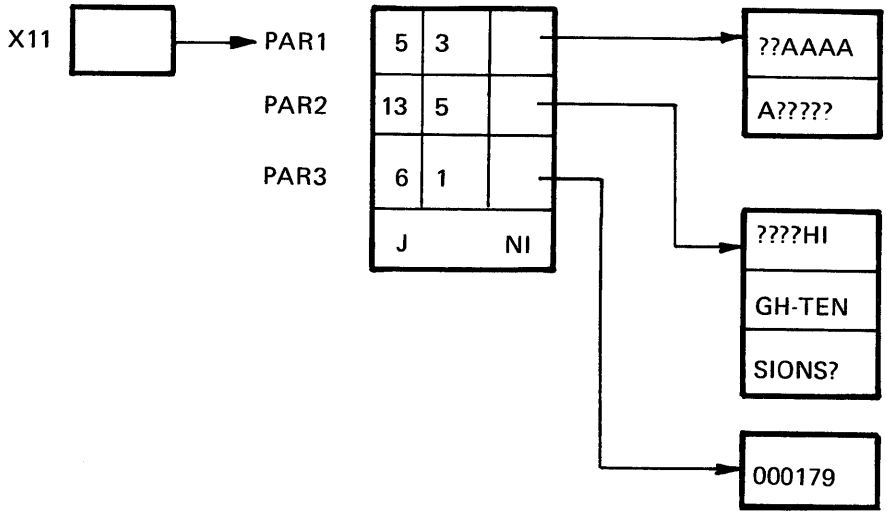
where PAR1, PAR2 have usage is display; PAR3 has usage is comp. The addresses of the CALC-Keys are dependent upon the descriptions of the records of which the parameters are items. The 3 parameters are adjacent items of the same record and PAR1 begins on a word boundary.



where:

AA denotes an absolute address
NI denotes the next instruction in the DMR

More generally, the interface might be:



where:

'?' denotes partial word contents not pertinent here.

In both cases:

The CALC Procedure must define ANAME to contain either 'HISTAREA' or ADMINAREA' and compute Page numberrs which lie within the intervals 1–20, 5–17, respectively.

- OUTPUT

Output of the CALC Procedure is of two types:

(a) A page number and a calc-chain number are calculated; the page number placed in H1 of A15 and calc-chain number in H2 of A15.

(b) Data-base-data-name–3, addressed via X10, is initialized with an AREA-NAME if not initialized by the run unit.

The calc-chain number must be ⩾ 0 and ⩽ (integer–8 minus 1) in the "CALC USES" clause from the Area Section of the DDL (see 3.3.8).

The page number returned must be ⩾ 1 and ⩽ integer–2 in the "ALLOCATE" clause (see 3.3.5) applicable area. In addition, the page number must not be the number of any 'Overflow' page established by the "ALLOCATE" clause.

If the optional portion of the mandatory WITHIN clause (i.e., page limits) of the DDL Record Entry is present, the page number returned by the CALC Procedure must be equal to or lie between these designated page limits.

F.3. DESIGNING A CALC PROCEDURE

Special consideration must be given to the following DDL statements as well as the collection of the user program with the DMR when using CALC Procedures.

- AREA SECTION

ALLOCATE *integer–2* [PRE-INITIALIZED] **PAGES**

[; *Integer–10* OVERFLOW **PAGES AT END**]

[; *Integer–3* OVERFLOW **PAGES** EVERY *integer–4* **DATA PAGES**]

[; EXPANDABLE **TO** *integer–11* **PAGES**]

(See 3.3.5.)

As previously mentioned, the page number returned by the CALC Procedure must be ⩾ 1 and ⩽ 'integer–2'. Overflow pages are used to provide space for records in a specified area whose storage requires more space than is available on the data page.

Example:

A record is stored CALC into AREA1, page 1 and calc-chain 0. Page 1 was previously empty. The record is assigned a record number of 1 by the DMR and linked into the calc-chain 0 on page 1. A second record stored via CALC into AREA1 page 1 will be stored on page 1 if possible, otherwise on an overflow page. In either case it will be linked into a page 1 calc-chain.

The exception to the above is in the case of 'DUPLICATES'. 'DUPLICATE' records are records whose area-name, page, number, calc-chain number, CALC-Keys or input arguments, and Record Code are identical. In the preceding example, if the two records are duplicates, the second record would not be stored if the NOT option was present in the DUPLICATES phrase under the DDL Record Section LOCATION mode clause. Otherwise the record is stored as previously described.

CALC **USES** *integer–8* **CHAINS** [**LINKED** PRIOR]

(See 3.3.8.)

The clause specifies that each page in the area contains a maximum of integer–8 calc-chains. The calc-chains are each circularly linked in one direction unless the optional LINKED PRIOR clause appears. In that case the calc-chain is doubly linked. The calc-chain number generated by the CALC procedure must fall between 0 and integer–8 minus 1 inclusive.

- RECORD SECTION

LOCATION **MODE IS** CALC *data-base-procedure-name–1*
IN *data-base-data-name–3*
USING *data-base-identifier–1* [, *data-base-identifier–2*] ...
DUPLICATES **AREA** [NOT] **ALLOWED**

(See 3.4.4.)

Data-base-procedure-name–1 is an externalized label identifying the entry point to the CALC Procedure. *(See UNIVAC 1100 Series Data Management System (DMS 1100) American National Standard COBOL (Fieldata UP–7908 or ASCII UP–7992) Data Manipulation Language Programmer Reference* (current version), Appendix D, Mapping of the CALC Procedure.)

Data-base-data-name–3 must be defined as a 77-level entry in the Working-Storage or Common-Storage Section of the input to the DML Preprocessor with a USAGE IS AREA-NAME and no PICTURE clause. On entry to the CALC Procedure from the DMR, Register X10 contains the address of data-base-data-name–3. This allows the CALC Procedure to specify AREA-NAME dynamically.

Data-base-identifier–1, [data-base-identifier–2] ... must be defined with an 02–49 level entry in the Record Section as subordinate items of the record under consideration. The contents of these data-item(s) are the CALC-Key inputs to the CALC Procedure.

WITHIN *area-name–1* [{ *integer–3* / *data-base-data-name–4* } { THRU / THROUGH } { *integer–4* / *data-base-data-name–5* }]

(See 3.4.5.)

The WITHIN clause specifies page limits in an area within which the record occurrence is to be placed.

For a given area, only one range of page numbers may be specified. The upper limit must not exceed the number of pages specified in the PAGES clause of the respective area entry.

The page number supplied by the CALC Procedure to the DMR must fall within the area(s) and page limits imposed by the WITHIN clause. If not, an error status is returned.

Because the DMR is a REP and when it is in control the user I-bank is no longer available, *the CALC Procedures must be assembled under an even control counter* (D-bank code) for the DMR to be able to find it.

APPENDIX G. SAMPLE DDL RUN STREAM

G.1. GENERAL

The following paragraph contains a run stream of the type necessary to use the DDL translator for creating an absolute SCHEMA. The third paragraph discusses the following run stream.

G.2. SAMPLE RUN STREAM

```
@RUN    SCHOOL,491195,1,200
@FREE    TPF$.
@ASG,T    TPF$.,F///2000
@ASG,T    DDLIN.,T,4906
@COPIN,C    DDLIN.,TPF$.
@FREE    DDLIN.
@ASG,C    SCHOOLBASE.,F
@DDL,IBL    EXAMPLE,SCHOOLBASE.SCHEMA
          IDENTIFICATION DIVISION
          SCHEMA NAME IS SCHOOLBASE
          :
          AREA SECTION
          :
```

```
RECORD SECTION
:
SET SECTION
:
@LIST,AO   SCHOOLBASE.SCHEMA

@FIN
```

G.3. DISCUSSION

The preceding run stream assumes that the DDL translator is an element on the first file of the tape 4906. The COPIN causes the DDL translator to become an element in TPF$. In this example, SCHOOLBASE, the SCHEMA NAME, is also the name assigned to an Executive file on FASTRAND mass storage in which the DDL translator will output the absolute SCHEMA. The file SCHOOLBASE, catalogues at the end of the run, is the file which will be used by the DML since it will contain the absolute SCHEMA. The 'I' option of the DDL processor call statement specifies that the source input is from cards or card images in the control stream. The preceding example assumes that the source cards follow the DDL statement. However, the source input could exist on some storage device. In the latter case, the source input could be read into TPF$ with the element name EXAMPLE. The 'B' option on the DDL processor call statement indicates to the translator that sequence numbers on the source input statements are to be ignored. The 'L' option requests a complete listing of source input and diagnostics generated by the DDL.

APPENDIX H. SYSTEM CALC PROCEDURE

H.1. GENERAL

The purpose of the System CALC Procedure, DMSCALC, is to provide the Data Administrator the facility to define records having a LOCATION MODE of CALC without writing an ASM CALC procedure for this purpose. Before discussing the details of using DMSCALC, several general comments are in order.

The algorithm used by DMSCALC has been mathematically validated on several representative ranges of input values and output area-keys. It may be used in a production type environment. However, it is a commonly recognized fact that given exact knowledge of the range of input values, any possible skewness, location of "breaks", etc., a CALC procedure may be written by the Data Administrator which will outperform a generalized CALC procedure. The same is true of DMSCALC. DMSCALC is best used in an educational type environment where the concepts embodied in the DDL are under study, not methods of writing CALC procedures. DMSCALC is also suited to initial production efforts where the emphasis is upon debugging of application programs and "getting on the air" in short order, not upon optimizing mass storage/throughput requirements.

DMSCALC may be used as an integral part of the full DDL syntax. However, there are certain restrictions upon its use. These are explained below.

H.2. DMSCALC

The DDL record description clauses required for DMSCALC are:

LOCATION **MODE IS** CALC DMSCALC
IN *data-base-data-name–1*
USING *data-base-identifier–1*
DUPLICATES **ARE** [NOT] **ALLOWED**
WITHIN *area-name–1*

NOTE:

Only one input data item (data-base-identifier–1) is allowed, and a page range phrase may not be specified in the WITHIN clause.

Data-base-identifier–1 must be defined as an 02–49 level elementary item in the record being described.

Note that data-base-data-name–1 corresponds to a like named word entered in the run unit by the Application Programmer as a 77-level entry having USAGE IS AREA-NAME and, optionally, a VALUE IS area-name–1 clauses. For example, if CALC-AREA is submitted for data-base-data-name–1, then:

77 CALC-AREA USAGE IS AREA-NAME VALUE IS *area-name–1.*

If additional record types are defined having a LOCATION MODE IS CALC DMSCALC, then, for these definitions, CALC-AREA may be replaced by a unique name for each LOCATION MODE clause. Each unique name must be defined in the run unit by the Application Programmer as a 77-level entry having USAGE IS AREA-NAME and, optionally, a VALUE IS area-name-n clause. Area-name-n must be the area-name specified on the corresponding WITHIN clause in the record description.

The area-name specified on the within clause must not have overflow pages interspersed with data pages; that is, all overflow pages must be at the end of the area. Thus, the allowable syntax is:

ALLOCATE *integer–2* **PAGES** [, *integer–10* OVERFLOW **PAGES AT END**]

Every area which contains records stored through a LOCATION MODE IS CALC DMSCALC clause must have a

CALC **USES** *integer–8* **CHAINS** [**LINKED** PRIOR]

clause, with or without the optional LINKED PRIOR phrase. DMSCALC attempts to store each record into a calc-chain of length one. Thus it will interpret integer–8 as being the maximum number of records which may be stored in each page. Each record is assigned a page number between 1 and integer–2 minus integer–10, and a chain number between 0 and integer–8 minus 1. Thus integer–2 minus integer–10 of the ALLOCATE clause times integer–8 equals the maximum number which are expected in the area. The integer–10 overflow pages may be used in the event that DMSCALC places a record in a page that does not have sufficient space.

DMSCALC is an integral part of the DMR. Therefore, *UNIVAC 1100 Series Data Management System (DMS 1100) American National Standard COBOL (Fieldata UP–7908 or ASCII UP–7992) Data Manipulation Programmer Reference* (current version), *Appendix D, MAPPING OF A CALC PROCEDURE,* may be ignored when using DMSCALC.

H.3. EXAMPLES:

The following example shows the use of the System CALC Procedure, DMSCALC, to store Employee Records into EMP-AREA, 10 records per page, using the employee life number as the key item.

```
IDENTIFICATION DIVISION
        :
        DATA DIVISION
        AREA SECTION
             AREA NAME IS EMP-AREA
             :
             ALLOCATE 200 PAGES, 50 OVERFLOW PAGES
             PAGES ARE 140 WORDS
             :
             CALC USES 11 CHAINS
```

NOTE:

The PAGES and CALC-USES changes are related in that page size is determined by the number of expected records per page, in this example, limited to 10.

```
        RECORD SECTION
        RECORD NAME IS EMP-RECORD
             :
             LOCATION MODE IS CALC DMSCALC
                  IN CAL-AREA
                  USING EMP-LIFE-NO
                  DUPLICATES ARE NOT ALLOWED
             WITHIN EMP-AREA
             02 EMP-NAME      PIC X(12)
             02 EMP-LIFE-NO   PIC X(9)
                  :
```

Thus, prior to a DML operation which stores or directly or indirectly accesses an Employee Record, the 77-level item CALC-AREA must have been initialized with EMP-AREA; and the EMP-LIFE-NO item initialized with the appropriate Employee Life Number.

APPENDIX I. SCHEMA REPORT

I.1. GENERAL

When the R option is specified on the DDL processor call statement, the following schema reports are generated:

- One preface report for the schema;
- One area report for the schema;
- One set-area cross reference table for the schema; and
- One record report for each record in the schema.

The following sections of this appendix describe the form and content of each report type. Do not assume all samples have been generated from the same schema. Consider each sample report independently.

I.2. PREFACE TO SCHEMA REPORTS

Information contained in this report is relatively self-explanatory (refer to Figure I–1). The number of area control bits and subsequently the maximum number of areas calculated from the AREA CONTROL source input statement, or are default values (12 and 4000, respectively), if no AREA CONTROL statement is specified. The average record and set description table lengths are obtained by straight forward summing of the individual lengths and subsequently dividing (integer) by the number of each, respectively. The maximum table length of both types is determined during the latter process.

If the Schema Description Table is segmented, the number of words in each segment indicated in the report is the maximum segment length, regardless of the actual size of the last segment. If the table is not segmented, the number of words shown is the actual number of words (rounded to the nearest sector) in the table.

SCHEMA TEST1

GENERATED 11:43:42 APRIL 20, 1972

AREA CONTROL BITS: 12 MAXIMUM NUMBER OF AREAS: 4000

AVERAGE RECORD DESCRIPTION TABLE LENGTH IN WORDS: 260

MAXIMUM RECORD DESCRIPTION TABLE LENGTH IN WORDS: 1584

AVERAGE SET DESCRIPTION TABLE LENGTH IN WORDS: 11

MAXIMUM SET DESCRIPTION TABLE LENGTH IN WORDS: 20

SCHEMA DESCRIPTION TABLE LENGTH: 3 SEGMENT(S) OF 1764 WORDS (EACH)

Figure I-1. Preface to Reports

I.3. THE AREA REPORT

The areas in the schema are listed in order of user specified area code (Figure I–2), although the code is not listed on the report. The records which could be stored in each area are determined using the following logic for each record:

(1) If a record description contains a WITHIN clause(s), the area(s) in which the record may be stored is (are) clearly determined; however, if the record description does not contain a WITHIN clause, the location mode of the record must be VIA SET; (the record description for the owner of the latter set is scanned for a WITHIN clause).

(2) The process continues up the hierarchy until a record's description contains a WITHIN clause; the area(s) in which the original record (the first record in the hierarchy whose description contained no WITHIN clause) can be stored is (are) thus determined.

The composition of the database key for records stored in each area is calculated as follows: the AREA CONTROL statement determines the number of area bits as explained in Appendix I.2. The minimum number of area code bits is seven.

The number of page bits is determined from the page value specified in the ALLOCATE clause according to these constraints:

- The number of allocated pages must be less than or equal to 99999 which is contained within 17 bits;
- The number of area bits plus the number of page bits must be greater than or equal to 21 bits.

AREA(S) IN THE SCHEMA

AREA NAME	RECORDS WHICH MAY BE STORED IN THE AREA	AREA BITS	PAGE BITS	MAXIMUM PAGE NUMBER	RECORD BITS	MAXIMUM RECORD NUMBER	@ASG REQUIREMENTS POSITIONS	OR TRACKS
A00001	R00001 R00047 R00052 R00105 R00107 R00110 R00151 R00152 R00155 R00167	12	9	511	15	32767	1	11
A00002	R00161 R00162 R00163 R00164 R00165	12	9	511	15	32767	1	4
A00003	R00017 R00020 R00021 R00022 R00023 R00112 R00153 R00154	12	9	511	15	32767	1	5

Figure I-2. Area Report

If the latter condition is not met, the number of page bits is increased to meet the condition. The number of record bits is equal to 36 minus (area bits plus page bits). The maximum page number and record number for each area, are determined from the number of page bits and record bits, respectively.

The storage assignment requirements for each area are calculated by the following method: the number of word allocated the area (allocated pages X words per page) is divided by 1792 words per track to produce the number of tracks required by the area. The number of tracks is divided by 64 tracks per position to produce the number of positions required by the area. (A remainder in either division adds one to the quotient.)

I.4. THE SET-AREA CROSS REFERENCE TABLE

The sets are listed in order of user specified set code (refer to Figure I–3), although the code is not included in the report. For each set, the owner record is listed first, followed by each member record of the set. For each of the records (owner and members) listed for the set, the area(s) in which each record could be stored is (are) listed opposite the record name. The user specified area protection is listed opposite each area name. If the set is not contained within one area, the AREA LOOKS specified for each area spanned by the set must be identical. If the AREA LOOKS are not identical, the following message will appear in the report directly below the set name in question:

*** ERROR ***

INCONSISTENT PROTECTION FOR THE AREAS SPANNED BY THE ABOVE SET

SET-AREA CROSS REFERENCE TABLE

SET NAME	OWNER AND MEMBER RECORD(S)	AREA THE SET IS CONTAINED WITHIN OR AREA(S) SPANNED BY THE SET	AREA PROTECTION
S00014	R00126	A00013	NO-LOOKS
	R000130	A00013	NO-LOOKS
S00015	R00024	A00007	NO-LOOKS
	R00032	A00007	NO-LOOKS
S00016	R00027	A00013	NO-LOOKS
	R00030	A00014	BEFORE-LOOKS

*** ERROR ***

INCONSISTENT PROTECTION FOR THE AREAS SPANNED BY THE ABOVE SET

S00020	R00030	A00010	NO-LOOKS
	R00036	A00010	NO-LOOKS
S00021	R00015	A00007	NO-LOOKS
	R00016	A00007	NO-LOOKS
	R00046	A00010	NO-LOOKS

Figure I–3. Set-Area Cross Reference

I.5. THE RECORD REPORT

The area(s) in which the record may be stored is (are) determined as explained in Appendix I.3. (Refer to Figures I–4a, I–4b, and I–4c, one sample record report.) If a record does not contain a WITHIN clause, the phrase 'DETERMINED BY RECORD record-name' appears below the area name(s) in the report.

The number of control words is calculated as explained in 6.2. The number of data words is determined from the data description in the source input of the record description. If the record is variable length, i.e., if any item(s) contains an OCCURS clause, both the maximum and minimum record length are calculated and would appear on the report as follows:

RECORD LENGTH IN WORDS

(MAXIMUM)

CONTROL	DATA	TOTAL
m	n	m+n

(MINIMUM)

CONTROL	DATA	TOTAL
m	k	m+k

A listing of the number and type of each pointer in the control section of the record always follows the record length information in the report.

If the record is a member of any set(s), the set(s) is listed with either the word (AUTOMATIC) or (MANUAL) directly below the set name, indicating the record's automatic or manual participation in the set. The pointers for the record's participation in the set are listed opposite the set name. Given the proper occurrence of the set, "LOCATION CRITERIA AS A MEMBER" indicates how the record is selected from within the set occurrence. If the order of the set is first or next, the location criteria appears as:

SET ORDERED FIRST
FIRST MEMBER OF TYPE IS ALWAYS SELECTED

If the order of the set is last or prior, the location criteria appears as:

SET ORDERED LAST
LAST MEMBER OF TYPE IS ALWAYS SELECTED

If the order of the set is sorted, the sort keys, and their respective types, which must be initialized to determine the proper occurrence of the record type within the set, are listed. The listing of the sort keys is major to minor, top to bottom.

RECORD R00013

AREA(S) IN WHICH THE RECORD MAY BE STORED:

A00003

RECORD LENGTH IN WORDS		
CONTROL	DATA	TOTAL
7	4	11

2 OWNER POINTER(S)
4 AUTOMATIC POINTER(S)

SET PARTICIPATION	POINTERS	LOCATION CRITERIA AS A MEMBER
S00010 (AUTOMATIC)	NEXT	SET ORDERED FIRST FIRST MEMBER OF TYPE IS ALWAYS SELECTED
S00013 (AUTOMATIC)	NEXT PRIOR	R00013-ID (DESCENDING KEY) R00013-NO (DESCENDING KEY)
S00020 (AUTOMATIC)	NEXT	RECORD-NAME (DESCENDING KEY) R00013-NO (DESCENDING KEY)

Figure I-4a. Record Report

RECORD R00013

SET PARTICIPATION	POINTERS	LOCATION CRITERIA AS AN OWNER
S00011	NEXT	ESTABLISH A CURRENT OF THIS SET THE CURRENT OF SET MAY BE AN OCCURRENCE OF THE OWNER OR OF ANY MEMBER RECORD. CURRENT OF SET SPECIFIED IN SOS CLAUSE OF MEMBER R00010
S00014	NEXT	***WARNING*** SOS USING CLAUSE IDENTIFIERS IGNORED LOCATION MODE OF THIS RECORD IS CALC USING CLAUSE SPECIFIED IN SOS CLAUSE OF MEMBER R00016 IN THIS SET
		WARNING SOS USING CLAUSE IDENTIFIERS IGNORED LOCATION MODE OF THIS RECORD IS CALC USING CLAUSE SPECIFIED IN SOS CLAUSE OF MEMBER R00017 IN THIS SET

Figure I-4b. Record Report

RECORD R00013

PATHS BY WHICH THE RECORD MAY BE LOCATED

PRIME

CALC-AN-A
R00013-AK
FOR CALC LOCATION MODE OF TYPE
R00013

ALTERNATE
VIA SET S00010
ESTABLISH A CURRENT OF SET
S00010
THE CURRENT OF SET MAY BE AN OCCURRENCE
OF OWNER R00003
OR OF ANY MEMBER TYPE

ALTERNATE
VIA SET S00013
ESTABLISH A CURRENT OF SET
S00012
THE CURRENT OF SET MAY BE AN OCCURRENCE
OF OWNER R00014
OR OF ANY MEMBER TYPE

ALTERNATE
VIA SET S00020
ESTABLISH A CURRENT OF SET
S00012
THE CURRENT OF SET MAY BE AN OCCURRENCE
OF OWNER R00014
OR OF ANY MEMBER TYPE

FIRST OCCURRENCE OF MEMBER TYPE
R00013
IS SELECTED FROM THE OCCURRENCE OF
S00010

R00015-SORT-KEY
(ASCENDING RANGE KEY)
FOR R00015
FOR LOCATION MODE VIA SET
S00012

R00015-SORT-KEY
(ASCENDING RANGE KEY)
FOR R00015
FOR LOCATION MODE VIA SET
S00012

R00013-ID
(DESCENDING KEY)
R00013-NO
(DESCENDING KEY)
FOR R00013
FOR LOCATION MODE VIA SET
S00013

RECORD-NAME
(DESCENDING KEY)
R00013-NO
(DESCENDING KEY)
FOR R00013
FOR LOCATION MODE VIA SET
S00020

Figure I-4c. Record Report

If the record is an owner of any set(s), the set(s) are listed with the pointer(s) for the record's ownership listed opposite the set-name. 'LOCATION CRITERIA AS AN OWNER' is determined as follows for each set of which the record is an owner. The SET OCCURRENCE SELECTION clause specified in each member entry in the set determines the ways in which the owner record is selected, and consequently the ways in which the set occurrence is selected. One of the following types of output (1) through (7) is generated for the SET OCCURRENCE SELECTION clause of each member entry:

OUTPUT TYPES

(1)

ESTABLISH A CURRENT OF THIS SET
THE CURRENT OF SET MAY BE AN OCCURRENCE
OF THE OWNER OR OF ANY MEMBER RECORD.
CURRENT OF SET SPECIFIED IS SOS CLAUSE
OF MEMBER record-name

(2)

REFER TO PRIME PATH FOR THIS RECORD
(LOCATION MODE OF OWNER SPECIFIED IN
SOS CLAUSE OF MEMBER
record-name
BUT NO ALIAS/USING KEYS SPECIFIED)

WARNING

(3)

SOS USING CLAUSE IDENTIFIERS IGNORED
LOCATION MODE OF THIS RECORD IS { DIRECT / CALC }
USING CLAUSE SPECIFIED IN SOS CLAUSE OF
MEMBER record-name
{ IN THIS SET / IN set-name SET }
identifier
:

(4)

FOR LOCATION MODE VIA SET
set-name
SPECIFIED IN THE SOS CLAUSE OF
MEMBER record-name
{ IN THIS SET / IN SET set-name }

(5)

data-name
ALIAS OF data-name
data-name
ALIAS OF data-name
FOR DIRECT LOCATION MODE
SPECIFIED IN THE SOS CLAUSE OF
MEMBER record-name
{ IN THIS SET }
{ IN SET set-name }

(6)

data-name
ALIAS OF data-name
data-name
ALIAS FOR identifier
⋮
FOR CALC LOCATION MODE
SPECIFIED IN THE SOS CLAUSE OF
MEMBER record-name
{ IN THIS SET }
{ IN SET set-name }

(7)

data-name
ALIAS FOR identifier
⋮
FOR LOCATION MODE VIA SET
set-name
SPECIFIED IN THE SOS CLAUSE OF
MEMBER record-name
{ IN THIS SET }
{ IN set-name SET }

When each member entry SET OCCURRENCE SELECTION clause has been exhausted, one of types (3) through (7) above is output for each appearance of this set, (of which the report record is an owner) in the SET OCCURRENCE SELECTION clause of any member entry in any other set in the schema.

The number of 'PATH(S) BY WHICH THE RECORD MAY BE LOCATED' is equal to the number of sets in which the record is a member plus one, the prime path. The 'PRIME' path is the location mode of the record. If the location mode of the record is either direct or calc, output of type (1) or (2), respectively, from the following output types, would appear as the PRIME path on the report. If the location mode of the record is VIA SET, the PRIME path is formed in the same manner as an ALTERNATE path, a path which ends in a set in which the record is a member. The paths are listed from the top of the hierarchy to the bottom, i.e., to the

object record occurence. A path listed on the report consists of a combination of output types (1) through (13), one type for each level of the hierarchy. (Section 4, "Hierarchical Storage Structures", should be studied in conjunction with this appendix.)

OUTPUT TYPES

(1)

data-name
data-name
FOR DIRECT LOCATION MODE OF TYPE
record-name

(2)

data-name
identifier
•
•
•
FOR CALC LOCATION MODE OF TYPE
record-name

(3)

FIRST OCCURRENCE OF MEMBER TYPE
record-name
IS SELECTED FROM THE OCCURRENCE OF
set-name

(4)

LAST OCCURRENCE OF MEMBER TYPE
record-name
IS SELECTED FROM THE OCCURRENCE OF
set name

(5)

ESTABLISH A CURRENT OF SET
set-name
THE CURRENT OF SET MAY BE AN OCCURRENCE
OF OWNER record-name
OR OF ANY MEMBER TYPE

(6)

data-name
ALIAS OF data-name
data-name
ALIAS OF data-name
FOR DIRECT LOCATION MODE OF TYPE
record-name

(7)

data-name
ALIAS OF data-name
data-name
ALIAS FOR data-name
⋮
FOR CALC LOCATION MODE OF TYPE
record-name

(8)

data-name
ALIAS FOR data-name
⋮
FOR record-name
FOR LOCATION MODE VIA SET
set-name

(9)

*** WARNING ***
SOS USING CLAUSE IDENTIFIERS IGNORED
data-name
data-name
FOR DIRECT LOCATION MODE OF TYPE
record-name

(10)

*** WARNING ***
SOS USING CLAUSE IDENTIFIERS IGNORED
data-name
data-name
FOR DIRECT LOCATION MODE OF TYPE
record-name

(11)

identifier
⋮
FOR record-name
FOR LOCATION MODE VIA SET
set-name

(12)

*** ERROR ***
IMPOSSIBLE TO CONTINUE PATH
REFER TO LISTING OF ERRORS

(13)

```
 ⎧        identifier         ⎫
 ⎪    (ASCENDING KEY)        ⎪
 ⎨ (ASCENDING RANGE KEY)     ⎬
 ⎪    (DESCENDING KEY)       ⎪
 ⎩ (DESCENDING RANGE KEY)    ⎭
             :
      FOR record-name
 FOR LOCATION MODE VIA SET
         set-name
```

Included on the following pages are the record and set descriptions which may be referenced in the analysis of the prime and alternate paths in the sample report produced for record R00013.

```
SEQUENCE NUMBER 1-6 | CONTINUATION 7 | A 8-11 | B 12 TEXT ... 20 30 40 50
:
RECORD  NAME IS R00013; RECORD CODE IS 11:
        LOCATION MODE IS CALC CALCAK IN CALC-AN-A
        USING R00013-AK
            DUPLICATES ARE NOT ALLOWED;
        WITHIN A00003;
        02 R00013-ID   PIC X(6)
        02 R00013-NO   PIC X(6)
        02 R00013-AK   USAGE IS AREA-KEY
        02 R00013-SEARCH-KEY  PIC 9(6)

RECORD  NAME IS R00015; RECORD CODE IS 13:
        LOCATION MODE IS VIA S00012 SET;
        WITHIN A00003:
        02 R00015-ID   PIC IS X(6)
        02 R00015-NO   PIC IS 9(6)
        02 R00015-SORT-KEY   PIC IS 9(6)
        02 R00015-SEARCH-KEY   PIC IS 9(6)
:
```

```
:
SET NAME IS S00010; SET CODE IS 8;
       MODE IS CHAIN;
       ORDER IS FIRST;
       OWNER IS R00003;
       MEMBER IS R00013 AUTOMATIC;
       SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

SET NAME IS S00012; SET CODE IS 10;
       MODE IS CHAIN LINKED PRIOR
       ORDER IS SORTED;
       OWNER IS R00014;
       MEMBER IS R00015 AUTOMATIC LINKED TO OWNER;
       ASCENDING RANGE KEY IS R00015-SORT-KEY
           DUPLICATES ARE NOT ALLOWED;
       SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

SET NAME IS S00013; SET CODE IS 11;
       MODE IS CHAIN LINKED PRIOR;
       ORDER IS SORTED;
       OWNER IS R00015;
       MEMBER IS R00013 AUTOMATIC;
       DESCENDING KEY IS R00013-ID, R00013-NO
           DUPLICATES ARE NOT ALLOWED;
       SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER;
       MEMBER IS R00003 AUTOMATIC;
       DESCENDING KEY IS R00003-ID, R00003-NO
           DUPLICATES ARE NOT ALLOWED;
       SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
           USING R00015-SEARCH-KEY.
SET NAME IS S00020; SET CODE IS 16;
       MODE IS CHAIN;
       ORDER IS SORTED;
       OWNER IS R00015;
       MEMBER IS R00013 AUTOMATIC;
       DESCENDING KEY IS RECORD-NAME R00013-NO
           DUPLICATES ARE NOT ALLOWED;
       SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
       MEMBER IS R00003 AUTOMATIC;
       DESCENDING KEY IS RECORD-NAME R00003-NO
           DUPLICATES ARE NOT ALLOWED;
       SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
           USING R00015-SEARCH-KEY.
```

APPENDIX J. LIST OF ABBREVIATIONS

AUTORP	–	Automatic Recovery Point
CD	–	Communication Description
DDL	–	Data Definition Language
DMCA	–	Data Management Communications Area
DML	–	Data Manipulation Language
DMR	–	Data Management Routine
DMS	–	Data Management System
DMSCALC	–	Data Management System-Supplied Calc
DMSSGP	–	Data Management Systems Support Generation Parameter Element
MCS	–	Message Control System
RDA	–	Record Delivery Area
TIP	–	Transaction Interface Package