SPERRY UNIVAC SERIES 90

Data Base Management System/90 (DMS/90)

Data Manipulation Language

Programmer Reference

SPERRY UNIVAC

UP-8036 Rev. 1

SPERRY UNIVAC SERIES 90

Data Base Management System/90 (DMS/90)

Data Manipulation Language

Programmer Reference

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

Sperry Univac is a division of Sperry Rand Corporation.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: Update A – UP-8036 Rev. 1

Part/Section	Page Number	Update Level
Cover		A
Title Page		A*
PSS	1	A
Acknowledgment	1	Orig.
Preface	1	Orig.
Contents	1, 2 3 4 5, 6	Orig. A Orig. A
1	1, 2	Orig.
2	1 2, 3 4 5 thru 9	Orig. A Orig. A
3	1, 2 3, 4 5	Orig. A Orig.
4	1 thru 10	Orig.
5	1	Orig.
6	1	Orig.
7	1 2 3 4, 5	Orig. A Orig. A
8	1, 2 3, 4 5	Orig. A Orig.
9	1 thru 6	Orig.
10	1 thru 3 4 5, 6 7 thru 9	Orig. A Orig. A
11	1, 2 3 thru 6 7 8 thru 10 11	Orig. A Orig. A Orig.
12	1 2, 3 4, 5 6	Orig. A Orig. A
13	1 2 thru 9	A Orig.
14	1 2	Orig. A
Appendix A	1 thru 4	Orig.
Appendix B	1, 2	Orig.
Appendix C	1 thru 8 9, 10	Orig. A
Appendix D	1, 2 3 4 thru 9	Orig. A Orig.
Appendix E	1 thru 4	Orig.
Index	1 thru 5	Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

*New page

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

TABLES

2. Concepts

2.1. GENERAL

The SPERRY UNIVAC Series 90 Data Base Management System (DMS/90) provides separate language facilities for description of data and the manipulation of data. This separation removes the data description function from the scope of the COBOL program. This separation of data description facilities allows integration of all data and data relationships into a base which is common to all applications programs which use it.

DMS/90 supports a network type of data structure. This facility permits the user a definition of structures which are most suitable to the applications which operate upon the data.

2.2. DATA DESCRIPTION LANGUAGE (DDL)

The DMS/90 DDL is a 2-part language used to describe the schema and subschema within the data base. A schema is a complete description of a data base, which includes the names and descriptions of all the data items, records, sets, and areas for all applications which access the data base. The language for describing a schema is called the schema DDL.

The second part of the data description language is called the subschema DDL. A subschema is a separate description of only those data items, records, sets, and areas which are of interest to one or more specific applications programs. In all cases, the schema and subschema descriptions are logically consistent. While there is only one schema for a data base, there may be any number of subschema descriptions, each describing a specific combination of records, sets, and areas which apply to a given application or program.

Both schema and subschema DDLs are within the purview of the data base administration staff which is responsible for the design, maintenance, and protection of the data base. The language is described in a separate reference manual and is a part of the DMS/90 system.

2.3. DATA MANIPULATION LANGUAGE (DML)

The COBOL programmer can access a data base only through a predefined subschema. A subschema is made available to a COBOL program by an INVOKE declarative statement in the Data Division. The DML processor establishes the description of each record included in the subschema as record-level entries in working storage of the COBOL Data Division. The record, sets, and areas included in the subschema place restraints on the use of DML imperative statements within the Procedure Division. By implication, the subschema removes access to all other records, sets, and areas not included in the subschema and thereby provides a measure of data privacy.

The imperative DML statements can be grouped into three categories:

1. Control

2. Retrieval

3. Modification

Control statements are used to establish access to a portion of a data base. The OPEN ALL AREAS statement announces the user's intention to begin processing within all areas specified in the subschema. The corresponding CLOSE ALL AREAS statement announces the end of processing in the subschema areas included in the subschema invoked. An area is a named subdivision of contiguous addressable storage within the data base. The IF statement is used to evaluate the member status in a set or determine if a set is empty. The IF statement provides a DML program with branching capability.

Retrieval statements are primarily concerned with locating data in the data base and making it available to the program in working storage. The DMS/90 FIND, GET, OBTAIN, and MOVE CURRENCY STATUS statements are included in this category. The FIND statement locates a record occurrence in the data base which satisfies the record-selection-expression part of the FIND statement. The GET statement causes a movement of all data items in the record which was most recently located into the like-named location in working storage of the COBOL program. The MOVE CURRENCY STATUS statement causes movement of the data base key associated with the most recently located record of a set, area, or record type into a named location in working storage. A data base key is used as a unique identifier of each record occurrence in the data base. A complete description of the data base key will be presented in a subsequent section. The OBTAIN statement is semantically equal to the combination of the FIND and GET statements.

Modification statements result in a change to the contents of the data base. Changes include the addition of new data, modification of existing data by replacement of existing data item values, update of set relationship by insertion or removal of member occurrences, or deletion of data which currently exists in the data base. The STORE statement uses data established by the user in working storage to create a new record occurrence in the data base. The MODIFY statement is used to change the data content of an existing record in the data base. The INSERT and REMOVE statements cause a change in the set relationship of an existing record occurrence in the data base. The DELETE statement causes an existing record occurrence to be removed from the data base. The space previously used to store a deleted record becomes available for the storage of a new record occurrence. Each DML statement will be discussed in detail in subsequent sections.

2.4. COBOL/DML COMPILATION

The user must identify a predefined subschema in the schema section in the Data Division. The subschema allows the user to manipulate only the sets, data items, records and areas included in the subschema. The DML statements may appear anywhere in the Procedure Division. From a programming viewpoint, DML statements may be considered as an extension to the COBOL language. This means that they may be placed at the appropriate point in any procedure and may be surrounded by other COBOL statements.

The compilation process is illustrated in Figure 2—1. The DMS/90 DML processor reads the COBOL/DML source statements and searches within the Data Division for the Schema Section to obtain the name of the subschema specified in the INVOKE statement. All COBOL (or non-DML) statements are copied directly to output. Once the subschema name has been verified, the DML processor obtains the names of all valid records, sets, and areas. The relationships between records, sets, and areas from subschema information stored in the data base data dictionary are used to validate DML statements in the Procedure Division. The DML processor establishes a 01 level record entry followed by data item entries in working storage for each record included in the subschema. In addition, system control and communication data items are included following the last record description.

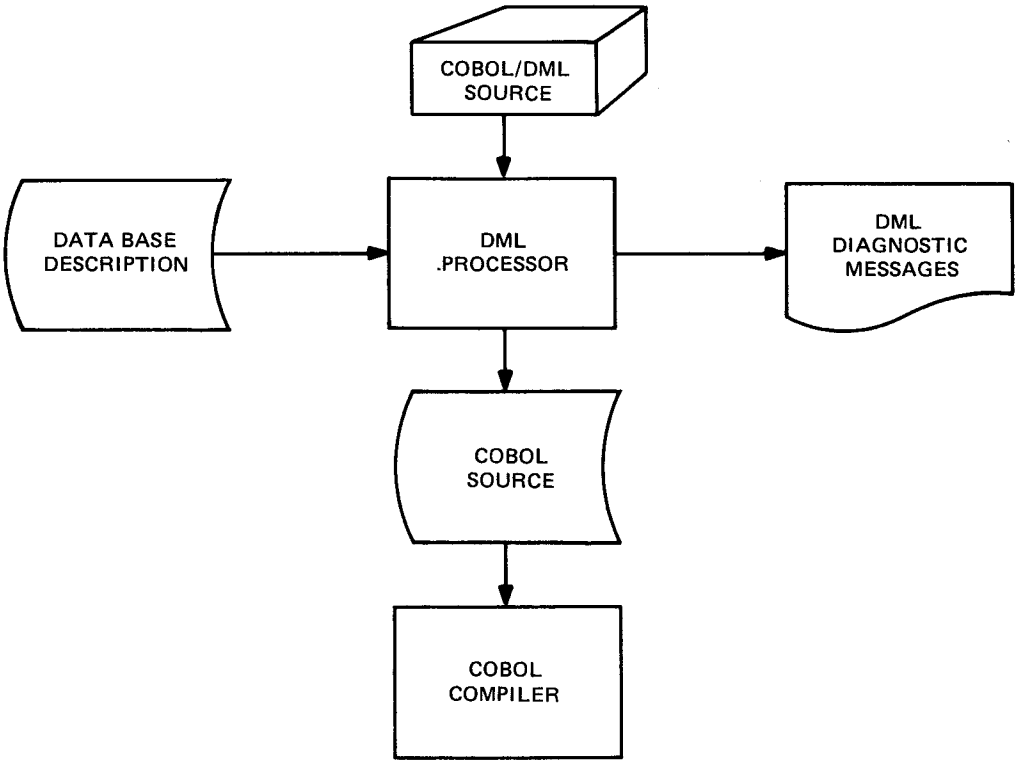


Figure 2—1. DML Compilation Process

The DML processor inserts a group of CALL statements immediately following the Procedure Division statement to bind the necessary working storage location of records and data items to the COBOL object subschema interface routine. Each DML command is validated for correct syntax and usage of record-set-area relationships. If the DML statement is correct, the processor will convert the DML statement into a COBOL comment statement and generate an appropriate COBOL CALL statement to the COBOL object subschema interface routine.

All DML errors detected by the DML processor will be displayed with the source statement which was in error. When input of COBOL/DML source is complete, statements output by the DML processor, along with all other COBOL statements, will be directed to the COBOL compiler.

2.5. COBOL/DML OBJECT PROGRAM EXECUTION

The linkage editor combines the object version of the subschema and the run-time DMS/90 with the user's COBOL relocatable object program prior to execution. Figure 2—2 illustrates the relationships which exist between the data base, operating system, DMS/90, and the COBOL program.

The numbered arrows in Figure 2—2 illustrate the operations which take place when a DML statement is executed. These operations, as numbered, are:

1. A DML statement appears in the object program as a call to the COBOL object subschema interface routine. The call identifies the type of data base service desired and any additional information such as record-name, set-name, or area-name required to properly interpret the call.

2. The COBOL object subschema interface routine analyzes the call using information stored in the object subschema. If the call requires the use of DMS/90, the user-supplied information is augmented by information from the object subschema before control is passed to the DMS/90.

3. The DMS/90 performs the requested data base service using information supplied by the interface routine. The operation which is performed depends upon the type of DML statement executed. In the event of a call to locate a record (FIND statement), DMS/90 searches the system buffers to see whether the requested record occurrence is present. If the record is in the data base, a request will be made to the operating system to input a data base page from the direct access storage to the DMS/90 system buffers. No input occurs if the record is already present in the system buffers.

4. DMS/90 performs the requested data base service and uses additional information contained in the object subschema. The object subschema contains a representation of the data structure, record placement control, record characteristics, set characteristics, currency status, data base operation statistics, and constraints on DML operations. In general, the object subschema controls the operations and access of the data base for each COBOL program which invokes it.

5. Whenever a specified record occurrence is located by the control routines, the data base key and other system information related to the record is moved from the system buffers to locations within the object subschema. This information represents the currency status of the area, sets, and record type of the record occurrence which has been located.

6. If the call to DMS/90 specified movement of the contents of a record to the user working area (GET or OBTAIN statement), data will be moved from a system buffer to a specified record area in working storage. Data movement from working storage to the system buffers occurs in response to a STORE or MODIFY statement.

7. DMS/90 returns to the COBOL object subschema interface routine with an indication of the success or failure of the data base service which was requested in step 3.

8. The COBOL object subschema interface routine moves status information regarding the outcome of the DML statement executed in step 1 to system status information locations within working storage of the COBOL program.

9. Control is returned to the user's COBOL program at the statement following the DML statement just executed.

10. The user must determine the status of the previous DML statement by examining the contents of the system status information data items in working storage. For example, if a FIND statement was just executed, the contents of a data item named ERROR-STATUS would indicate whether the specified record occurrence was located or not.

2.6. COBOL/DML SOURCE PROGRAM

The formats of all data manipulation language (DML) statements used in the SPERRY UNIVAC Series 90 Data Base Management System (DMS/90) were designed to conform as closely as possible to COBOL. From the programmer's viewpoint, DML is treated as an extension to COBOL.

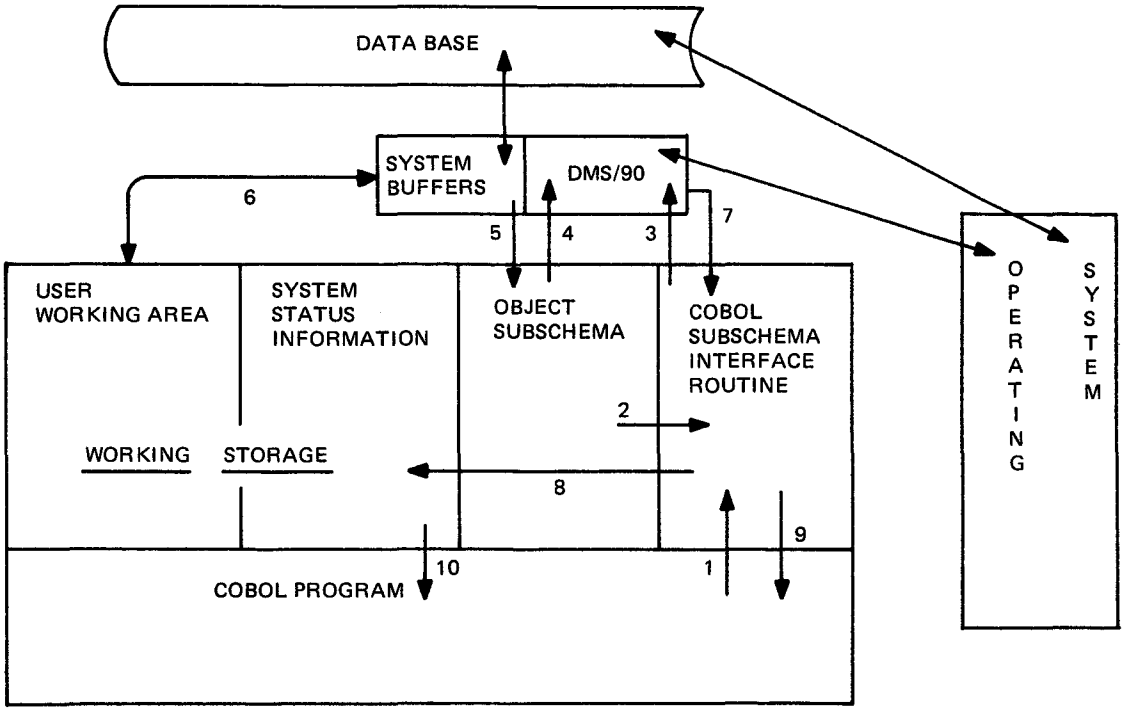


Figure 2—2. DMS/90 Application Program Execution

2.6.1 Data Division Entries

Additional entries in the COBOL Data Division are required to identify the subschema to be invoked in the program and to provide a user working area for storage or output of a record to or from the data base and system status information locations within working storage. Figure 2—3 illustrates the placement of DMS/90 DML statements in the Data Division.

- The SCHEMA SECTION statement (sequence number 300010) must appear immediately after the DATA DIVISION statement. The statement may begin in any column after column 7.

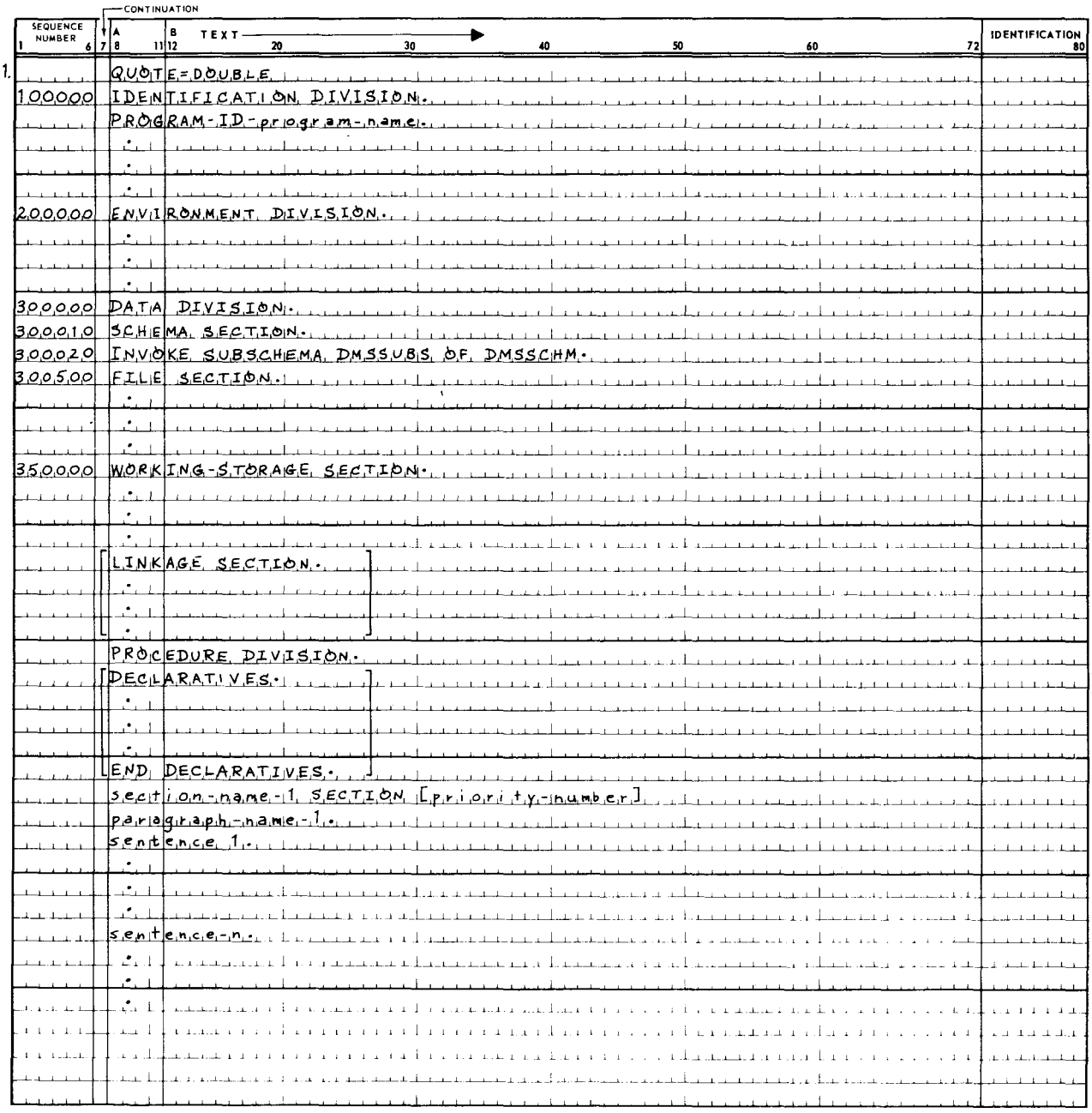
- The INVOKE statement must follow the SCHEMA SECTION statement and may begin in any column after column 7. The INVOKE statement (sequence number 300020) corresponds to the subschema and schema names in Figure 12—1. No other statements are permitted in the Schema Section.

- A WORKING-STORAGE SECTION statement must be present in the position required by COBOL specifications. Any 77-level data item or 01-level record entries are positioned as required by COBOL.

- The DML processor automatically inserts 01-level record descriptions for all subschema records and system status information locations between the last entry in the WORKING-STORAGE SECTION and either the next section statement or the PROCEDURE DIVISION statement if no other sections are present.

- The DML processor automatically changes both SCHEMA SECTION and INVOKE statements to comment statements by inserting an asterisk in column 7 of each statement.

- The user may specify the parameter QUOTE=SINGLE or QUOTE=DOUBLE beginning in column 8 before the IDENTIFICATION DIVISION statement. If QUOTE=SINGLE is specified, single quotes (apostrophes) are used to bound nonnumeric literals generated by the DML preprocessor. If QUOTE=DOUBLE is specified, double quotes are used for the same purpose.



NOTE:

1. QUOTE=SINGLE may also be coded as the first line.

Figure 2—3. Standard COBOL/DML Program Structure

2.6.2 Procedure Division Entries

DML statements entered in the Procedure Division are written as conventional COBOL statements. DML statements may appear anywhere in the Procedure Division and may be intermixed with COBOL statements subject to the following restrictions:

- A DML statement may be preceded by a paragraph name conforming to the naming conventions of COBOL (Figure 13—2, line 5);
- All DML statements must begin with a DML verb and end with a period followed by a space (Figure 13—1, 13—2, or 13—3); and
- A DML statement must not be combined with other COBOL statements to form a COBOL sentence. For example, the following statement is not allowed.

```
IF CUST-IN IS EQUAL TO CUST-TEMP FIND NEXT
  RECORD OF ITEM SET GO TO G410.
```

The DML processor automatically inserts several statements immediately following the first paragraph-name of the Procedure Division to establish address linkages between the user's COBOL program and the object subschema which is linked to the user's COBOL/DML program. The processor also converts the DMS/90 DML statement to a comment statement by inserting an asterisk in column 7, validating the statement, and generating an appropriate CALL statement to the COBOL object subschema interface routine.

Every DMS/90 COBOL/DML program must perform an OPEN ALL AREAS statement before any other DML statement is executed and must execute a CLOSE ALL AREAS statement when all data base operations have been completed.

The DML processor copies a COBOL section named DMS-STATUS into the Procedure Division to aid the programmer in detection and analysis of ERROR-STATUS values other than those specifically related to the logic of the user's program.

DMS-STATUS is a COBOL section which is used to provide a common method of analysis of the ERROR-STATUS location and minimize the amount of COBOL coding required to check the status following a data manipulation language (DML) statement. Figure 2—4 illustrates the coding of the DMS-STATUS SECTION, and its recommended use is illustrated by the following 4-step process associated with the execution of a DML statement (Figures 13—1, 13—2, and 13—3).

1. Execute DML statement.
2. Test for all nonzero values of the ERROR-STATUS location which are used to control logic of the program. This step may be omitted if no nonzero values are valid.
3. Execute a PERFORM DMS-STATUS operation to determine whether an error condition exists.
4. Statements are performed when the DML statement executed in step 1 is successful.

The DMS-STATUS SECTION is copied into the user's program by the DML processor following the last statement of the user's COBOL/DML program.

When the DMS-STATUS SECTION is performed, an IF statement is executed to determine whether the value of the ERROR-STATUS location is zero. If it is zero, the DMS-SUCCESS SECTION is performed and the statement following PERFORM DMS-STATUS statement is executed next. However, if the ERROR-STATUS location is non-zero, an error condition is assumed to exist and additional statements will be executed to document the error on the operator's console and the main system printer. Also a user section called DMS-ABORT is performed to allow the user to terminate report files and produce any other output desired before the program is aborted. Therefore, the user must place sections named DMS-SUCCESS and DMS-ABORT within the Procedure Division. The basic format may be as follows:

503100 DMS-ABORT SECTION.
(user COBOL statements)
503600 DMS-ABORT-EXIT. EXIT.
503700 DMS-SUCCESS SECTION.
(user COBOL statements)
503800 DMS-SUCCESS-EXIT. EXIT.

In Figure 2—4, THE-PRINTER and THE-CONSOLE are used in DISPLAY statements of DMS-STATUS SECTION; therefore, the user must relate these mnemonic-names to the appropriate implementor names in the SPECIAL-NAMES entry of the ENVIRONMENT DIVISION.

```
*
*       ************************************************************
*       *  DMS STATUS CHECK SECTION                                *
*       ************************************************************
*
 DMS-STATUS          SECTION.
 STATUS-PARA.
*
     IF ERROR-STATUS EQUAL TO ZEROS
         PERFORM DMS-SUCCESS
         GO TO ISABEX.
*
         PERFORM      DMS-ABORT.
         DISPLAY      '*********************************'
                      ' ABORTING - ' PROGRAM-NAME
                      ', '           ERROR-STATUS
                      ', '           ERROR-RECORD
                      ' **** RECOVER DMS **** '
                      UPON THE-CONSOLE.
         DISPLAY      'PROGRAM NAME ------ ' PROGRAM-NAME
          UPON THE-PRINTER.
         DISPLAY      'ERROR STATUS ------ ' ERROR-STATUS
          UPON THE-PRINTER.
         DISPLAY      'ERROR RECORD ------ ' ERROR-RECORD
          UPON THE-PRINTER.
         DISPLAY      'ERROR SET --------- ' ERROR-SET
          UPON THE-PRINTER.
         DISPLAY      'ERROR AREA -------- ' ERROR-AREA
          UPON THE-PRINTER.
         DISPLAY      'LAST GOOD RECORD -- ' RECORD-NAME
          UPON THE-PRINTER.
         DISPLAY      'LAST GOOD AREA ---- ' AREA-NAME
          UPON THE-PRINTER.
         DISPLAY      '*** RECOVER DMS ***'
          UPON THE-PRINTER.
*        CLOSE  ALL  AREAS.
         ENTER LINKAGE.
         CALL 'XR7DMS'  USING IDBMSCOM (02).
         ENTER COBOL.
         STOP  RUN.
*
 ISABEX. EXIT.
*
         NOTE
         THIS IS THE END
            OF THE DMS STATUS CHECK SECTION.
```

Figure 2—4. DMS-STATUS Section

2.7 PROGRAMMING REQUIREMENTS

From a programming viewpoint, the DMS/90 input/output area for the data base resides in working storage. Each record included in the subschema is automatically included in working storage as a 01 record level entry followed by the statements which describe the name, picture attribute, and usage of each data item included in the record. Input of a record from the data base always appears in its like-named area in working storage. Storage (or output) of a record to the data base requires the movement of data from various locations in the user's program to each of the data items described for the specific record in working storage. Once this is completed by user COBOL statement, the record is then physically moved from working storage into the data base using the STORE statement.

The user is responsible for initializing all data items required to successfully execute a DML statement and must ensure that the data is correct. DMS/90 has extensive run-time error diagnostic facilities and will update the contents of the location ERROR-STATUS in the Working Storage section of a COBOL program following every DML statement. The user must examine the ERROR-STATUS following each DML statement to determine the action taken by the system in response to a DML statement. Three operations are required each time an access to the data base is required:

1. Initialization of data items as required by the DML statement to be executed

2. Execution of the DML statement to initiate a data base operation

3. Error checking to determine the outcome of the preceding DML statement

Before a programmer can write any DML statements, the names of all areas, records, sets, and data items must be known, along with their characteristics. The existence of data structures present in the data base, but not included in the subschema, may affect the scope of the DML statements which can be used with the subschema. This subschema information and any DML restrictions must be documented by the data base administration staff and understood by the systems analyst and programmer before program specification and implementation can be accomplished.

The remainder of this programmer's reference guide includes a complete description of all DML commands and subschema representation and an example program and subschema illustrating the programming strategy involved in the use of DMS/90.

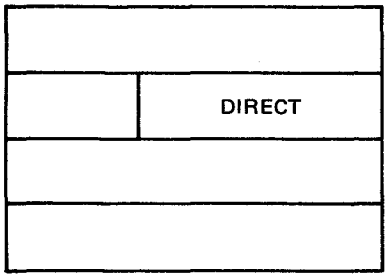
3.5. LOCATION MODE

The location mode of a record defines the way a given record type is stored in the data base and specifies one of the ways of selecting occurrences of the given record in the data base. Any given record type may have only one location mode as specified by the data administration staff. Once specified, the location mode restricts the type of DML statements which apply to the record type to the following three modes:

1. DIRECT
2. CALC
3. VIA

3.5.1 Direct (DIRECT) Location Mode

The format of the location mode for the DIRECT option is:

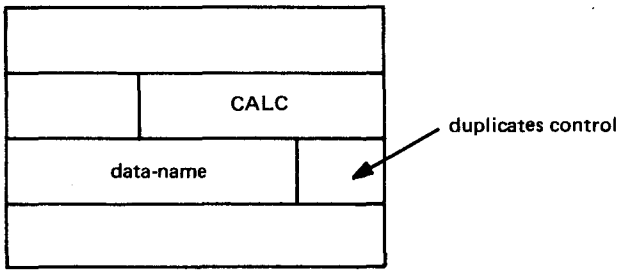


To store the record type with a DIRECT location mode, the user must initialize the system status location named DIRECT-DBK with a —1 or a desired data base key before the execution of a DML STORE statement. For the user's convenience, DIRECT-DBK is initialized with —1 by DMS/90. DMS/90 automatically examines the contents of the location DIRECT-DBK and assigns an appropriate data base key (11.6).

This mode gives the greatest control of placement of record occurrences within the data base. It allows the user to advise DMS/90 to store a record occurrence on (or near) a specific page within an area of the data base. (See STORE statement, 11.6.)

3.5.2. Calculated (CALC) Location Mode

The format of the CALC location mode option is:



where:

data-name
Is the name of a data item within the record.

duplicate control
Is required to specify how additional occurrences of this record are stored if the CALC key values are identical to an occurrence already in the data base.

DN (duplicates not)	Indicates that an additional occurrence with the same CALC key value is not allowed to exist in the data base.
DF (duplicates first)	Indicates that an additional occurrence with the same CALC key value will be stored in a LIFO sequence.
DL (duplicates last)	Indicates that an additional occurrence with the same CALC key value will be stored in a FIFO sequence.

Figure 3—2 illustrates the CUSTOMER record type representation with a location mode of CALC and a data-name of CUST-NO. When an occurrence of a CUSTOMER record is stored in the data base, a system defined procedure within DMS/90 uses the value of data-name CUST-NO to determine a specific page within the data base where the record will be stored.

A standard randomizing module, XR7CALC, is provided with DMS/90 and is used by default. Otherwise, if the user wishes to provide his own randomizing algorithm, he may create his own CALC module and link it with all application programs. For details about this process, refer to the DMS/90 system support functions programmer reference for the appropriate operating system.

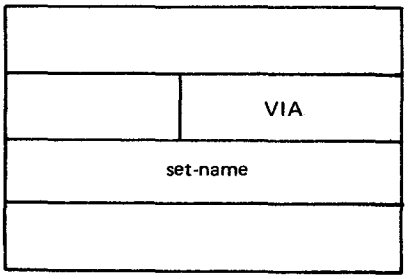
A given CUSTOMER record occurrence may be selected by moving a value into the CUST-NO data item before the execution of the DML statement

```
FIND     CUSTOMER     RECORD.
```

The same DMS/90 procedure (either system standard or user specified) transforms the value of the data-name CUST-NO to a specific data base page as the first step in selecting a record occurrence. DMS/90 then proceeds to locate the specific CUSTOMER record starting with the calculated page. (See FIND statement, 10.2, and STORE statement 11.6.)

3.5.3. VIA Location Mode

The format of the VIA location mode option is:



where:

set-name
Is the name of a set in which the record type participates as a member.

7. System Status Information

7.1. GENERAL

Whenever a run-unit (user's program) executes a call to the SPERRY UNIVAC Series 90 Data Base Management System (DMS/90) for a data base operation, the system furnishes information concerning the outcome of the requested service in the system status locations in working storage of the user's program. This establishes a communication link between the data base management system and the user's program.

The following is a list of the system status locations supplied by the data manipulation language (DML) processor:

PROGRAM-NAME
: Is an alphanumeric data item which is used to hold the name of the program being executed. This value is used for checkpoint and data base recovery operations and is initialized before an OPEN statement is executed.

ERROR-STATUS
: Is a data item that contains a value which indicates the outcome of the last DML statement executed by the program. It is moved to the user program location immediately before returning control to the user's program. The meaning of error status condition will be discussed in a separate section.

DBKEY
: Is the data base key of the last record accessed by the run-unit. For example, when a successful FIND statement has been completed, DBKEY contains the data base key of the record. In the case of a STORE statement, DBKEY will contain the data base key just assigned by DMS/90 when the record was stored in the data base.

RECORD-NAME
: Is the data item that holds the name of the record which has been last accessed by the run-unit. It is left-justified and space-filled.

AREA-NAME
: Is the data item that holds the left-justified, space-filled name of the area in which the last accessed record is stored.

ERROR-SET
: Contains the name of a set that was encountered during an operation which produced an error condition.

ERROR-RECORD
: Contains the name of the record involved in an operation which caused an error condition.

ERROR-AREA
Contains the name of the area involved in an operation which produced an error.

DIRECT-DBK
Is a data-item which should be initialized in the user program with the data base key desired before a record with a DIRECT location mode is stored in the data base. This location is initialized with —1.

Table 7—1 summarizes the update of system status information for each successful and unsuccessful execution of a DML statement. A yes in Table 7—1 indicates that the appropriate value for data base key, area name, and record name will be updated by DMS/90. A no means that the system leaves the previous contents of the data item unchanged. A null data base key is a binary value of minus 1. The ERROR-STATUS location always has a value of zero whenever a DML statement has been completed successfully.

Whenever an error occurs, the ERROR-STATUS location contains a code which identifies the type of DML statement, followed by a numeric code (shown as XX) which identifies the type of error. A complete discussion of each error status condition for each DML statement can be found in Appendix D.

Whenever a statement has been executed successfully, the ERROR-SET, ERROR-RECORD and ERROR-AREA locations will contain spaces. However, whenever a statement is unsuccessful, the DBKEY, RECORD-NAME, and AREA-NAME locations remain unchanged.

7.2. CONTROL STATEMENT SYSTEM STATUS

7.2.1. OPEN ALL Statement

The OPEN ALL statement indicates the user's intention to access information within the data base for all areas included in the subschema invoked by the user. If the open statement is successful, the ERROR-STATUS location is set to zero, the DBKEY location is set to a null value (—1), the RECORD-NAME location is set to spaces, and the AREA-NAME location contains the name of the area just opened. If the subschema includes more than one area, the AREA-NAME location will contain the last area opened.

If the open statement is unsuccessful, the ERROR-STATUS location will contain the error code (09XX), where XX indicates the type of error. The values of the DBKEY, RECORD-NAME, and AREA-NAME locations remain unchanged. The name of the area is placed in the ERROR-AREA location and spaces appear in the ERROR-SET and ERROR-RECORD locations. If the subschema includes more than one area, the location ERROR-AREA contains the first area name.

7.2.2. CLOSE ALL Statement

The CLOSE ALL statement indicates the user's intention to terminate processing of information in the areas included in the subschema. If the close is successful, the ERROR-STATUS location is set to zero and the AREA-NAME location will contain the name of the area just closed. If more than one area is included in the subschema, then the AREA-NAME location will contain only the last area closed. However, if the close statement was unsuccessful, the ERROR-STATUS location will contain the error code (01XX) indicating the type of error. The name of the area in error will be placed in the ERROR-AREA location and spaces will be placed in the ERROR-SET and ERROR-RECORD locations. If the subschema includes more than one area, the location ERROR-AREA contains the first area name.

Table 7–1. Update of System Status Information

	System Status Info	Successful				Unsuccessful			
DML Category	DML Statement	ERROR-STATUS	DBKEY	RECORD-NAME	AREA-NAME	ERROR-STATUS	ERROR-SET	ERROR-RECORD	ERROR-AREA
Control	Open all	0000	Null	Spaces	Yes	09XX	Spaces	Spaces	Yes
	Close All	0000	Null	Spaces	Yes	01XX	Spaces	Spaces	Yes
	If (true)	1601* 0000**	No	No	No	–	–	–	–
	If (false)	1601* 0000**	No	No	No	16XX	Yes	Spaces	Spaces
Retrieval	Find/Obtain	0000	Yes	Yes	Yes	03XX	Yes	Yes	Yes
	Get	0000	No	No	No	05XX	Spaces	Yes	Spaces
	Move Currency Status	No	No	No	No	No	No	No	No
Modification	Delete	0000	Null	No	No	02XX	Yes	Yes	Yes
	Insert	0000	Yes	Yes	Yes	07XX	Yes	Yes	Yes
	Modify	0000	Yes	Yes	Yes	08XX	Yes	Yes	Yes
	Remove	0000	Yes	Yes	Yes	11XX	Yes	Yes	Yes
	Store	0000	Yes	Yes	Yes	12XX	Yes	Yes	Yes

*When the set is not actually empty or the record is not actually a member of the set
**When the set is actually empty or the record is actually a member of a set

7.2.3. IF Statement

The IF statement is used to evaluate the membership status in a set or determine whether a set is empty (that is, does not contain any member record occurrences). If the specified set in format 1 of the IF statement (9.5) is an empty set (either with or without the optional word NOT), or the record of format 2 is a member of the named set (either with or without the optional word NOT), the location ERROR-STATUS is set to zero and the previous state of locations DBKEY, RECORD-NAME, and AREA-NAME is unchanged. If the specified set in format 1 of the IF statement is not an empty set (either with or without the optional word NOT) or the record is not a member of the named set (either with or without the optional word NOT) a value of 1601 is placed in the location ERROR-STATUS, and the previous content of the locations DBKEY, RECORD-NAME, and AREA-NAME is unchanged.

In the execution of an IF statement, an error condition will occur if the current set-name or the current record of run-unit is null. When an error condition exists, the IF statement will be evaluated as false and an appropriate error code will be placed in the ERROR-STATUS location. The name of the set used in the IF statement will be placed in the ERROR-SET location and spaces will be moved to both the ERROR-RECORD and ERROR-AREA locations.

7.3. RETRIEVAL STATEMENT SYSTEM STATUS

7.3.1. FIND/OBTAIN Statement

The successful FIND/OBTAIN statement causes DMS/90 to place a value of zero in the ERROR-STATUS location. The item DBKEY, RECORD-NAME, and AREA-NAME locations are updated with appropriate values to identify the record which has been selected. The selected record is considered to be the current record of the run-unit (or program) until superseded by a subsequent FIND/OBTAIN or STORE statement.

An unsuccessful FIND/OBTAIN statement updates the ERROR-STATUS location to indicate the type of error which has occurred. In any case, the current of run-unit status as indicated by the DBKEY, RECORD-NAME, and AREA-NAME locations remain unchanged. Where appropriate, the ERROR-SET, ERROR-RECORD, and ERROR-AREA locations contain names associated with the error. Only the statements which include a set-name cause an update of the ERROR-SET location.

7.3.2. GET Statement

A successful GET statement causes a zero to be placed in the ERROR-STATUS location. Since this statement is operable upon only the current record of run-unit, the location of the DBKEY, RECORD-NAME, and AREA-NAME locations remain unchanged.

An unsuccessful GET statement causes the system to place an error code (05XX) in the ERROR-STATUS location to define the error which occurred. Only the ERROR-RECORD location contains the name of the record involved in the error condition.

7.3.3. MOVE CURRENCY STATUS Statement

The MOVE CURRENCY STATUS statement involves only movement of appropriate data base keys from the system locations in the object subschema into the user's working storage area. This operation is performed without change to any of the system status locations.

7.4. MODIFICATION STATEMENT SYSTEM STATUS

7.4.1 DELETE Statement

A successful DELETE statement causes a zero to be placed in the ERROR-STATUS location. Since a deleted record is no longer in the data base, the DBKEY location is set to a null value (—1). The locations of the RECORD-NAME and AREA-NAME locations remain unchanged to identify the record which has been deleted.

If the DELETE statement is unsuccessful, the ERROR-STATUS location will be updated with an error code (02XX) to indicate the type of error which has occurred. The locations of the ERROR-SET, ERROR-RECORD, and ERROR-AREA locations are updated to give additional documentation of the error condition.

7.4.2. INSERT Statement

A successful INSERT statement causes a zero to be placed in the location of the ERROR-STATUS location.

An unsuccessful INSERT statement causes an error code (07XX) to be placed in the ERROR-STATUS location to indicate the type of error which has occurred. The name of the set in which the record was to be inserted is placed in the ERROR-SET location. The name of the record specified in the INSERT statement is placed in the ERROR-RECORD location. The name of the area affected is placed in the ERROR-AREA location.

7.4.3. MODIFY Statement

A successful MODIFY statement causes a zero to be placed in the ERROR-STATUS location.

An unsuccessful MODIFY statement causes an error code (08XX) to be placed in the ERROR-STATUS location to indicate the type of error. The contents of the ERROR-SET, ERROR-RECORD, and ERROR-AREA locations are updated appropriately to further document the error.

7.4.4. REMOVE Statement

A successful REMOVE statement causes a zero to be placed in the ERROR-STATUS location.

An unsuccessful REMOVE statement causes an error code (11XX) to be placed in the ERROR-STATUS location to indicate the type of error which has occurred. The contents of the ERROR-SET, ERROR-RECORD, and ERROR-AREA locations are appropriately updated to further document the error.

7.4.5. STORE Statement

A successful STORE statement causes a zero to be placed in the ERROR-STATUS location. The new record in the data base becomes the current record of the run-unit. The data base key assigned to the new record by the system is placed in the DBKEY location. The name of the record is placed in the RECORD-NAME location, and the name of the area in which the record is stored is placed in the AREA-NAME location.

An unsuccessful STORE statement causes an error code (12XX) to be placed in the ERROR-STATUS location to indicate the error which has occurred. The locations of the ERROR-RECORD, ERROR-SET, and ERROR-AREA locations are appropriately updated to further document the error.

8.6.2. Currency GET, INSERT, and REMOVE Statements

These statements always apply to the current record of run-unit. The GET and REMOVE statements do not change any currency information. A successful INSERT statement makes the object record the current of all sets in which it now participates.

8.6.3. Currency DELETE Statement

The DELETE statement always applies to the current record of run-unit. This means that a successful FIND statement must have been executed for this record prior to the delete.

When a record is deleted, the current of run-unit is set to a null value of —1. No other currency information is changed. Retrieving deleted records by using FIND format 2 or FIND NEXT/PRIOR record-name RECORD of set-name SET (format 3) results in a value of 0317 being placed in ERROR-STATUS. Retrieving deleted records by using other formats results in a value of 0326 being placed in ERROR-STATUS.

Whenever a successful DELETE statement has been executed, the records associated with the deleted record are still available. This means that the next, owner, or prior record of the set occurrence of the deleted record is still available, as well as the next or prior of area.

8.6.4. Currency STORE Statement

When the STORE statement is successful, the new record will become the current record of the run-unit and its system-assigned data base key will be stored in location DBKEY. The new record also becomes the current of its record type, current of area in which it has been stored, and current of all sets in which the record participates as either owner or member. All sets in which the record participates as owner or automatic member are empty at the time the record is stored in the data base. The establishment of the stored record as current of all empty sets in which it participates as owner identifies the proper set occurrence for subsequent storage of records that participate as automatic members of the sets. Extension of this principle leads to the requirement that the appropriate currency must be established for all set occurrences in which the stored record participates as an automatic member. In Figure 12—1, this means that the programmer must establish the appropriate ORDOR set occurrence before executing a STORE CUST-ORDER RECORD statement. This is accomplished by finding the CUSTOMER type record which corresponds to the customer who has placed an order.

If a stored record is a manual member of a set, it is not necessary to establish this set occurrence prior to the STORE statement. In this instance, the programmer must find the appropriate owner of a manual set followed by an INSERT statement to properly establish the stored record as a member in the manual set. It is assumed that the record to be inserted has previously been stored in the data base.

8.7. SET CURRENCY FOR RECORDS WITH OPTIONAL SET MEMBERSHIP

When a record is described as having automatic set membership, the record occurrence automatically becomes a member of the set when it is stored, whereas manual set membership requires use of the INSERT statement to establish set membership. In either case, if the record is an optional member of a set, it may or may not participate as a member in that set. When any record is successfully retrieved, it will become the current record of all sets in which it actually participates as either an owner or member. Figure 8—1 is a data structure diagram that includes the LOT-INV record type which is an optional automatic member of the LOT-DET set and an optional manual member of the LOT-OH set. If the PKG-SUM record has just been retrieved, the statement

```
FIND NEXT LOT-INV RECORD OF LOT-DET SET.
```

will select the appropriate record occurrence and make it the current record of the LOT-DET set. In addition, if the record occurrence is a member in an occurrence of a LOT-OH set, then it will become the current record of that set type as well. However, if it has not been previously inserted in a LOT-OH set, no change will be made to the existing currency status of the LOT-OH set. If no change occurs in the currency status of the LOT-OH set, then a different, previously accessed occurrence of a LOT-INV record will remain as the current record of LOT-OH.

Even though the most recently accessed LOT-INV record is current of LOT-INV record type and current of LOT-DET set, any operations involving the LOT-OH set will refer to the last LOT-INV or MFG-LOT record which actually participated as member or owner in the LOT-OH set. For this reason, failure to properly test for the existence of set membership before performing operations involving optional sets will give unpredictable results. For example, if the programmer wanted to find a LOT-INV record and then access the MFG-LOT owner record associated with it, the statements

```
FIND NEXT LOT-INV RECORD OF LOT-DET SET.

FIND OWNER RECORD OF LOT-OH SET.
```

would give unpredictable results because the programmer failed to determine if the LOT-INV record was actually a member of any occurrence of a LOT-OH set. The correct sequence of statements in this case would be:

```
FIND NEXT LOT-INV RECORD OF LOT-DET SET.

IF RECORD NOT MEMBER OF LOT-OH SET GO TO G290.
FIND OWNER RECORD OF LOT-OH SET.
```

The following statements summarize the action of the system and the programming required for optional and manual set membership:

- If a record is an optional member of a given set, then the DML IF statement must be performed to determine whether the record is actually a member of any occurrence of the set before any FIND/OBTAIN statements involving the given set are performed. The preceding discussion is an illustration of this rule.

Rules:

1. An OPEN statement for all areas included in the subschema invoked by the program must be executed successfully before any FIND/OBTAIN statements can be executed.

2. The FIND verb causes retrieval of a record occurrence based on the remainder of the FIND statement. A successful FIND statement does not cause movement of the retrieved record into working storage of the program.

3. The OBTAIN verb is identical in operation to the FIND verb except that the successfully retrieved record occurrence is moved to working storage. The OBTAIN statement is a combination of the FIND and GET statements.

   In all cases, the error status conditions following attempted execution of an OBTAIN statement are the same as the FIND statement.

4. The object record of the FIND/OBTAIN statement is specified by the record-selection expression defined by formats 1 through 6.

5. All records names, set names, area names and identifiers specified must have been defined in the subschema invoked by the program. The only exception is identifier in format 1.

6. A successfully executed FIND/OBTAIN statement results in the record occurrence selected becoming the current record of the run-unit. The value of the data base key is placed in the DBKEY location. The name of the area in which the record was stored is moved to the AREA-NAME location. The name of the record is moved to the RECORD-NAME location, The selected record also becomes the current record of its area, of its record-name, and of all sets in which it is defined as an owner or in which it currently participates as a member.

Format 1 Rules:

7. The identifier refers to a location within the program which contains a data base key. The identifier must be described as a computational synchronized data item with a picture of S9(8).

8. The name of the identifier must conform to COBOL data item naming conventions.

9. The user is responsible for initialization of the identifier with a data base key before execution of the format 1 FIND/OBTAIN statement. Any record included in the subschema invoked by the program may be accessed directly in this manner, regardless of the location mode specified in the subschema. If the data base key supplied identifies a record which is not an occurrence of record-name, or if no record is identified by the data base key supplied, an error condition will exist and the statement will not be executed.

Format 2 Rules:

10. This format retrieves the record occurrence that is the current record of the specified record-name, set-name or area-name; or it selects the current record of run-unit. If the OBTAIN option is used, the contents of the record will be moved to working storage of the program after the record has been retrieved.

    If the current record of the type specified is not known or its currency indicator is null, an error condition will exist. Refer to Appendix D.

11. It is possible to establish a currency status for a record occurrence and then at some later time, execute an operation which would delete the record from the data base. Any attempt to select a record which has been deleted since it became current in this run-unit results in the value 0317 being placed in the ERROR-STATUS location before returning to the program.

Format 3 Rules:

12. If record-name and set-name are specified, record-name must have been defined as a member of set-name. If record-name and area-name are specified, the area-name in which the record is stored must be used. Refer to Section 3 and Figure 3—1.

13. If a set-name is specified, the set occurrence from which the object record is to be selected is identified by the current record of the specified set. If the current record of that set is not known, or its currency indicator is null, an error condition will occur. Refer to Appendix D.

    If an area-name is specified, the object record is selected from the named area.

    If the NEXT or PRIOR phrase is used and an area-name is specified, and the current record of the named area is not known or its currency indicator is null, an error condition will occur.

    If the NEXT or PRIOR phrase is used with a set-name specified and a deleted record indicated, an error condition occurs.

    If the optional record-name phrase is used, only occurrences of record-name will be considered in evaluating the statement.

    If the OBTAIN option is used, the contents of the record will be moved to working storage of the program after the record has been retrieved.

    If the sets or areas involved include occurrences of record types not known to the run-unit, then only the records specified in the invoked subschema will be considered in evaluating the statement.

    - NEXT RECORD OF set-name SET means the subsequent record relative to the current record of the named set in the logical order of the set regardless of the data base key sequence. If the set is empty (that is, no member record occurrences participate in the set), or the current record is the last record in the set, the statement will not be executed and the ERROR-STATUS location will contain the value 0307 to indicate the end of set upon return to the program.

    - NEXT RECORD OF area-name AREA means the record with the next higher data base key relative to the current record of the named area. If there is no record in the area named with a higher data base key, the statement will not be executed and ERROR-STATUS location will contain the value 0307 to indicate the end of area upon return to the program.

    - PRIOR RECORD OF set-name SET means the previous record relative to the current record of the named set in the logical order of the set, regardless of data base key sequence. If the set is empty, (that is, no member record occurrence participates in the set), or the current record is the first record in the set, the statement will not be executed and the ERROR-STATUS location will contain 0307 to indicate the end of set upon return to the program.

    - PRIOR RECORD OF area-name AREA means the record with the next lower data base key relative to the current record of the named area. If there is no record in the area named with a lower data base key, the statement will not be executed and the ERROR-STATUS location will contain 0307 to indicate the end of area upon return to the program.

    - FIRST RECORD OF area-name AREA is the record occurrence with the lowest data base key in the named area. If there are no records in the named area, ERROR-STATUS location will contain the value 0307 to indicate the end of area upon return to the program.

    - FIRST RECORD OF set-name SET is the first member occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current record of the set was the owner record and the NEXT RECORD OF set-name SET statement was used. If the set occurrence is empty (that is, no member record occurrences participate in the set), the ERROR-STATUS location will contain the value 0307 upon return to the program.

- **Format 3**

Refer to Figure 12—1. Assume that the PRODUCT-AREA area is to be scanned to find every occurrence of the PRODUCT record. The following statement selects the first PRODUCT record in the PRODUCT-AREA area:

```
604020      FIND FIRST PRODUCT RECORD OF PRODUCT-AREA AREA.
```

The PRODUCT record occurrence selected becomes the current record of the PRODUCT-AREA area. The Following statement retrieves the PRODUCT record occurrence with the next higher data base key.

```
604100      FIND NEXT PRODUCT RECORD OF PRODUCT-AREA AREA.
```

All occurrences of the PRODUCT record are selected by repetition of the above statement until an end-of-area (0307) condition appears in the ERROR-STATUS location.

If the record-name option is not used, the statement

```
600135      FIND FIRST RECORD OF ORDER-AREA AREA.
```

selects one of the records (CUST-ORDER, ORDER-ITEM, or ORD-REMARK), shown in Figure 12—1, with the lowest data base key. The statement

```
600120      FIND NEXT RECORD OF ORDER-AREA AREA.
```

selects the next record from among those of CUST-ORDER, ORDER-ITEM, or ORD-REMARK record occurrences with the next higher data base key. The user must examine the contents of location RECORD-NAME to determine the record which has been retrieved.

Other examples of format 3 statements are included in Figure 13—1 and discussed in 13.5.1.

- **Format 4**

Refer to Figure 12—1. Assume that an occurrence of a CUST-ORDER record has been retrieved. It is now the current record of run-unit, current record of CUST-ORDER, current record of the ORDER-AREA area, and the current record of the ORDOR, SPEC-REMARK, and ITEM sets. The following statement selects the CUSTOMER record occurrence which is the owner of the ORDOR set:

```
600105      FIND OWNER RECORD OF ORDOR SET.
```

- **Format 5**

Refer to Figure 12—1. The CUST-ORDER record is described with a location mode of CALC which uses the data item named FO-NO-620 as the CALC key. The FO-NO-620 data item is initialized in working storage by the statement

```
601000      MOVE FO-NO TO FO-NO-620.
```

The following statement selects an occurrence of the CUST-ORDER record with a factory order number (FO-NO) data-item which matches the value previously stored in location FO-NO-620 in working storage.

```
601900      FIND CUST-ORDER RECORD.
```

Additional examples of format 5 usage are included in Figures 13—1, 13—2, and 13—3 and discussed in 13.5.1, 13.5.2, and 13.5.3.

- **Format 6**

In Figure 12—1, the PROD-ORD set is ordered in ascending sequence, based on the value stored in the LOT-NO-621 data item in each ORDER-ITEM record occurrence. Retrieval of an ORDER-ITEM record with a lot number value equal to 1230427 is accomplished by the following statements:

```
601920      MOVE '1230427' TO LOT-NO-621.
601930      FIND ORDER-ITEM RECORD VIA CURRENT OF PROD-ORD SET USING
601931          LOT-NO-621.
```

The above statements assume that the user had previously established an occurrence of a PROD-ORD set.

10.3 GET STATEMENT

Function:

Transfers the contents of an object record occurrence into working storage of the user's program.

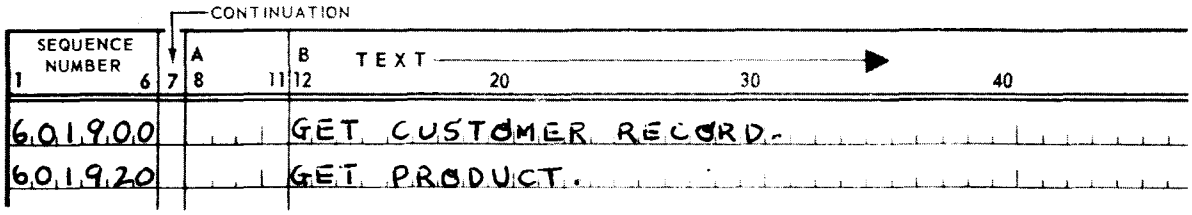
Format:

GET record-name RECORD.

Rules:

1. Record-name must be the name of a record included in the subschema invoked by the program.
2. The GET statement only operates on the current record of run-unit. If record-name is not the same as the name of the record which is the current of run-unit, an error condition will exist.
3. All data items included in the object record are moved to like-named locations in working storage. These locations appear following the 01-level entry in working storage for record-name.
4. The GET statement must be executed before access can be made to data within the record.
5. The GET function can be obtained automatically following a successful FIND statement by replacing the FIND verb with the OBTAIN verb in the FIND statement.
6. If no current record of the run-unit is known, an error condition will result and the GET statement will not be executed.

Example:



10.4 MOVE STATEMENT

Function:

Moves the data base keys which represent the currency status to specified locations within the user's program.

Format:

MOVE **CURRENCY** **STATUS** **FOR** { **RUN-UNIT TO identifier** | **record-name RECORD TO identifier** | **area-name AREA TO identifier** | **set-name SET TO identifier** } .

Rules:

1. Record-name, area-name, and set-name must be names included in the subschema invoked by the program. The identifier refers to a location within the user's program which contains a data base key. The identifier must be described as a computational, synchronized data item with a picture of S9(8) and must conform to COBOL conventions for naming data items.

2. If the RUN-UNIT phrase is specified, the data base key for the current record of run-unit is placed in the identifier. The current record of the run-unit is not altered. The same result can be accomplished by moving the contents of the DBKEY location to the identifier.

   If a record-name, area-name, or set-name is specified, the data base key for the current record of record-name, area-name, or set-name is placed in the identifier. The current record of record-name, area-name, or set-name is not altered.

Examples:

The DMSSUBS subschema (Figure 12—1) is used in the following samples; statement number 610030 is equivalent to statement number 610020.

```
610020     MOVE CURRENCY STATUS FOR RUN-UNIT TO MY-DBKEY.
610030     MOVE DBKEY TO MY-DBKEY.
610040     MOVE STATUS FOR CUST-ORDER RECORD TO SAVE-REC.
610050     MOVE STATUS FOR ORDER-AREA AREA TO SAVE-AREA.
610060     MOVE STATUS FOR ORDOR SET TO SAVE-SET.
```

The following Data Division entries are applicable to the preceding examples.

```
610090 77 MY-DBKEY  COMP SYNC PIC S9(8).
610100 77 SAVE-REC  COMP SYNC PIC S9(8).
610110 77 SAVE-AREA COMP SYNC PIC S9(8).
610120 77 SAVE-SET  COMP SYNC PIC S9(8).
```

7. The DELETE SELECTIVE form of the statement has the same results as the DELETE ONLY statement, with the following exceptions:

   - Optional members are deleted only if they do not currently participate as members in other set occurrences.
   - All deleted records which are, themselves, the owners of any set occurrences are treated as if they were the object of a DELETE SELECTIVE statement.

8. The DELETE ALL form of the statement deletes the object record, together with all of its member records, regardless of whether they are mandatory or optional. As with the DELETE ONLY form of the statement, the delete process continues down the hierarchy, with the difference that all deleted records, which are themselves the owners of any set occurrences, are treated as if they were the object record of a DELETE ALL statement.

9. Following successful execution of a DELETE statement, the current record of the run-unit becomes null. A null value is indicated by a —1 in the DBKEY system status location. All other currency information remains unchanged. Retrieving deleted records by using FIND format 2 or FIND NEXT/PRIOR record-name RECORD of set-name SET (format 3) results in a value of 0317 being placed in ERROR-STATUS. Retrieving deleted records by using other formats results in a value of 0326 being placed in ERROR-STATUS.

10. The subschema documentation indicates restriction on the use of the DELETE statement (Table 12—1). Restrictions are required when the subschema invoked by the program does not include all sets in which a record is an owner or participates as a member.

11. When an error occurs, the ERROR-STATUS location contains a nonzero value which is explained in Appendix D. In addition, the data base and working storage remain in the state existing prior to the attempted execution of the DELETE statement.

Example 1:

This example uses the DMSSUBS subschema in Figure 12—1. Assume that an occurrence of a CUST-ORDER record is current of run-unit. Execution of the statement

```
602020      DELETE CUST-ORDER RECORD ALL.
```

causes the following operations to be performed:

- All occurrences of the ORD-REMARK record in the current SPEC-REMARK set occurrence are deleted.
- Each ORDER-ITEM record occurrence is removed from the related PROD-ORD set occurrence.
- All ORDER-ITEM record occurrences in the current ITEM set occurrence are deleted.
- The CUST-ORDER record is removed from the ORDOR set.
- The CUST-ORDER record is deleted.

Example 2:

Refer to the data structure in Figure 8—1. Assume that an occurrence of a MFG-LOT record is current of run-unit. Execution of the statement

```
609120      DELETE MFG-LOT RECORD ONLY.
```

causes the following operations to be performed:

- Every occurrence of the LOT-INV record which participates in the current LOT-OH set occurrence is removed from the LOT-OH set.
- The MFG-LOT record is removed from the current occurrence of the MFG-LOT-DET set (provided the current MFG-LOT is a member in that set).
- The MFG-LOT record is deleted.

The effect of these operations is to delete the owner of the LOT-OH set (MFG-LOT record) without also deleting the LOT-INV member records.

Example 3:

Refer to Figure 8—1 and the preceeding example 2. If the statement

```
610090      DELETE MFG-LOT RECORD SELECTIVE.
```

is executed in place of the DELETE ONLY statement in example 2, the same operations will occur, except that each LOT-INV record occurrence will be examined to determine whether it is a member of a LOT-DET set occurrence. If the LOT-INV record is not a member in a LOT-DET set, then the LOT-INV record will be deleted.

11.3. INSERT STATEMENT

Function:

Makes the object record a member of an occurrence of a specified set-name.

Format:

INSERT record-name RECORD INTO set-name SET.

Rules:

1. Record-name must be defined as an optional automatic, optional manual, or mandatory manual member of set-name included in the subschema invoked by the program.

2. All areas included in the subschema invoked by the program must be opened with a usage-mode of exclusive update before the INSERT statement can be executed.

3. The object record occurrence of the INSERT statement is the current record of run-unit.

4. The object record is inserted into the current occurrence of set-name in accordance with the ordering criteria defined in the subschema for that set. The record then becomes the current record of set-name. No other currency information is changed.

5. The user must establish the desired occurrence of set-name before the INSERT statement is executed. The set occurrence is defined by the current record of set-name.

6. Successful execution of an INSERT statement is indicated by a value of zero in the ERROR-STATUS location. A nonzero value in the ERROR-STATUS location indicates an error condition, which is discussed in Appendix D. When an error occurs, the data base and working storage remain in the state existing prior to the attempted execution of the INSERT statement.

Example:

This example uses the DMSSUBS subschema shown in Figure 12—1. Assume that the user has established a current occurrence of a CUST-ORDER record and wants to establish an occurrence of an ORD-REMARK record.

The first operation is to initialize the data items defined in the ORD-REMARK record in working storage. The statement

```
601920        STORE ORD-REMARK RECORD.
```

moves the record from working storage to the data base. The ORD-REMARK record is now the current of run-unit. The statement

```
601960        INSERT ORD-REMARK RECORD INTO SPEC-REMARK SET.
```

establishes the ORD-REMARK record just stored as a member of the current occurrence of the SPEC-REMARK set (established by CUST-ORDER record) and makes the ORD-REMARK record the current record of SPEC-REMARK set.

11.4. MODIFY STATEMENT

Function:

The MODIFY statement:

- replaces the values of all data items of the object record occurrence in the data base with values from like-named locations in working storage; and
- alters the intraset position, so as to maintain the data base in accordance with the set ordering criteria specified in the subschema invoked by the program.

Format:

MODIFY record-name RECORD.

Rules:

1. Record-name must be defined in the subschema invoked by the program. In addition, all other record-names and set-names affected by the MODIFY statement must also be included in the subschema.

2. The areas included in the subschema invoked by the program must be open for exclusive update before a MODIFY statement can be executed.

3. The object record occurrence of the MODIFY statement is the current record of the run-unit. If the current record of run-unit is not an occurrence of the named record type, an error condition will result.

4. The object record in the data base is modified with like-named data item values from working storage.

   If the current record of record-name is not the same record occurrence as the object of a previous STORE, OBTAIN, or GET statement for this record-name, an error condition will result. A successful MODIFY statement occurs only when an occurrence of the named record type is the current record of record-name, is the current of run-unit, and has been the object of a previous successful STORE, OBTAIN, or GET statement. The STORE, OBTAIN, or GET statements provide the required initialization of all data items in working storage before any of the data items are modified.

   Other DML statements which affect the current of run-unit status may be executed between the STORE, OBTAIN, or GET statement and the MODIFY statement. The record occurrence to be modified may be re-established as current of run-unit by execution of the following format 2 FIND statement:

   FIND CURRENT record-name RECORD.

   An error condition results if modification of an ASCENDING/DESCENDING control data item is attempted for a set-name which is not included in the subschema invoked by the program. Restrictions in the use of the MODIFY statement are part of the subschema documentation (Table 12—1).

5. If any of the modified data items are defined as ASCENDING/DESCENDING control data items in the object record for any set occurrence in which the object is currently a member, then its modification causes the intraset occurrence position of the object record to be examined and, if necessary, the object record is removed and reinserted in the set occurrence to maintain the set order specified in the subschema. The NEXT and PRIOR pointers of the set occurrences involved are updated to reflect the new intraset occurrence position of the object record.

6. If a modified data item is defined as a CALC key for the object record, then the execution of the MODIFY statement will enable the record to be found on the basis of the new value for the CALC key. However, the data base key of the object record remains unchanged.

7. In order to change the set occurrence in which a record participates, the following operations must be performed:

   - The REMOVE statement must be performed to remove the record from the current set occurrence.
   - Appropriate DML statements must be performed to select the set occurrence in which the record is to be inserted.
   - The INSERT statement must be performed.

8. If the modification of the value of any data item in the object record would result in the violation of a DUPLICATES NOT ALLOWED clause defined for any of the sets or records involved, then the MODIFY statement is not executed and an error condition results.

9. All data items involved must be updated in working storage with the required values prior to execution of the MODIFY statement.

10. Upon successful completion of a MODIFY statement, the ERROR-STATUS location contains a zero value. The DBKEY, RECORD-NAME, and AREA-NAME locations remain unchanged. A nonzero value for the ERROR-STATUS location indicates an error condition. When an error occurs, the data base and working storage remain in the state existing prior to the attempted execution of the MODIFY statement.

Example:

This example uses the DMSSUBS subschema documented in Figures 12—1 and 12—2 and Table 12—1. Assume that the customer name is incorrect within an existing CUSTOMER record in the data base. The problem is to correct the value of the CUST-NAME-S-611 data item.

The first step is to retrieve the desired CUSTOMER record and move the contents into working storage. This is done by the statements

```
601020      MOVE CUST-NO TO CUST-NO-611.
601030      OBTAIN CUSTOMER RECORD.
601040      PERFORM DMS-STATUS.
```

Statement number 601040 performs an error diagnostic section, which checks for error conditions and returns to the following statement if the ERROR-STATUS location contains a zero value.

The next step is to move the correct customer name (CUST-NAME) into the proper location (Figure 12—2) in the CUSTOMER record in working storage. This is accomplished by the statement

```
601050      MOVE CUST-NAME TO CUST-NAME-S-611.
```

The following MODIFY statement returns all data items in the CUSTOMER record in working storage to the data base.

```
601060      MODIFY CUSTOMER RECORD.
```

11.5. REMOVE STATEMENT

Function:

Cancels the membership of the object record in the occurrence of the specified set-name in which it currently participates as a member, provided that the object record is defined as an optional member of the named set.

Format:

REMOVE record-name RECORD FROM set-name SET.

Rules:

1. Record-name must be defined as an optional member of set-name in the subschema invoked by the program.

2. The object record of the REMOVE statement is the current record of run-unit.

3. The object record must currently participate as a member in an occurrence of the set named. Successful execution of the REMOVE statement cancels that participation. The removed record is then not accessible through this set occurrence, but continues to be accessible through any other sets in which it participates as a member. This record may also be accessible, provided it has been defined as a CALC record. It is always accessible either by means of a complete scan of the area in which it participates or directly, through its data base key, if that is known.

4. No change occurs to any currency information.

5. All areas included in the subschema invoked by the program must be opened with a usage-mode of exclusive update before the REMOVE statement can be executed.

6. When an error condition occurs, the data base and working storage remain in the state existing prior to the attempted REMOVE statement execution. The ERROR-STATUS, ERROR-SET, ERROR-RECORD, and ERROR-AREA locations are updated to document the error condition. Nonzero values of the ERROR-STATUS location are discussed in Appendix D.

7. The ERROR-STATUS location is set to zero to indicate successful completion of a REMOVE statement.

Examples:

Refer to Figure 12—1. Assume that an occurrence of an ORDER-ITEM record is current of run-unit and that it is a member in an occurrence of both the ITEM and PROD-ORD sets. Removal of the record from the current occurrence of the PROD-ORD set is accomplished by the statement

```
601920      REMOVE ORDER-ITEM RECORD FROM PROD-ORD SET.
```

The ORDER-ITEM record remains as current of run-unit, current of ORDER-AREA area, current of ITEM set, and current of PROD-ORD set.

11.6. STORE STATEMENT

Function:

The STORE statement:

- acquires space and a data base key for a new record occurrence in the data base;
- causes the values of the appropriate data items in working storage to be included in the occurrence of the object record in the data base;
- inserts the object record into all sets of the particular subschema for which it is defined as an automatic member in the subschema invoked by the program;
- establishes a new set occurrence for each set for which the object record is defined as owner; and
- establishes the object record as:
  - — the current record of the run-unit;
  - — the current record of the area in which it is stored;
  - — the current record of record-name; and
  - — the current record of set for all set-names in which it is specified as an owner or automatic member.

Format:

STORE record-name RECORD.

Rules:

1. Record-name must be included in the subschema invoked by the program.
2. All areas included in the subschema must be opened with a usage-mode of exclusive update before the STORE statement is executed.
3. All sets in which the named record is defined as owner or member must be included in the subschema before a STORE statement for the named record can be executed. If the subschema does not contain all sets involved, the STORE statement will not be executed; the DMSSUBS subschema includes the PRODUCT and CUSTOMER type records which are examples of this constraint (Figures 12—1 and 12—2 and Table 12—1).
4. Before execution of the STORE statement, the user must establish the required set occurrence for all sets in which the stored record will participate as an automatic member. The set occurrence is defined by the current record of set-name. ←

5. Before execution of the STORE statement, the user must initialize all data items included in the object record defined in working storage. This includes all control data items, such as CALC and ASCENDING/DESCENDING keys.

6. The object record occurrence will be moved from working storage to the data base. The data base key and space for the record will be assigned according to the location mode specified for the record type.

7. The object record occurrence is inserted into the current set occurrence for each set in which the record is defined as an automatic member. The ordering rules for the set govern the insertion point of the object record in all of the relevant set occurrences.

8. The object record is established as the owner of a set occurrence for each set in which it has been defined as an owner. These set occurrences are empty at this time, that is, they have no member records.

9. The successfully stored record occurrence becomes the current record of the run-unit; its area name is placed into the AREA-NAME location, its record name is placed into the RECORD-NAME location, and its assigned data base key is placed in DBKEY location.

10. The object record also becomes the current record of the area in which it is placed, the current record of its record-name, and the current record of all set-names in which it is defined as an owner or automatic member.

11. The location mode defined for the named record type in the subschema invoked by the program affects the physical placement of the object record occurrence in the data base. (Further discussion of location mode is included in 3.5.)

    - In the case of DIRECT location mode, the contents of the location DIRECT-DBK (3.5.1) must be initialized with a data base key or a null data base key value of —1. The DIRECT-DBK data item is provided by the DML processor in working storage as a computational synchronized data item with a picture of S9(8) and the initial value of —1. If the DIRECT-DBK data item contains a valid data base key, the system will assign this data base key if it is available to the stored record occurrence. If the specified data base key is not available, the next available data base key will be selected, subject to the limits of the area in which the record is to be stored.

      If the DIRECT-DBK data item contains a value of —1, the first data base key available in the area in which the record is to be stored will be selected.

      In any case, the data base key of the stored record occurrence is placed in the DBKEY location.

    - In the case of CALC location mode, a data base key is developed within the upper and lower page limit of the area in which the record is to be stored.

    - In the case of VIA set-name location mode, the user must establish a current occurrence of set-name by retrieval of the owner record occurrence, regardless of whether the record being stored is an automatic or manual member of that set. If it is an automatic member of the set, it will be logically inserted into the selected set occurrence. If it is a manual member, it will not be inserted into the selected set occurrence. In all cases, however, the record being stored is placed as close as possible to the owner record of the specified set.

12. If a record to be stored would violate a DUPLICATES NOT ALLOWED clause defined for any of the sets involved, an error condition will occur.

12. Subschema Documentation

12.1. GENERAL

The purpose of a subschema is to provide a definition of only those data items, records, areas and sets which are of interest to one or more application programs. A subschema, by implication, removes all other data items, records, sets and areas in the data base from the view of the programmer and provides a measure of data privacy.

In order to make effective use of the data manipulation language (DML) statements, a thorough understanding of the characteristics of all records and sets is essential. Once this is accomplished, both analyst and programmer can develop the strategy for access of desired information in the data base. Subschema documentation consists of the following:

- The names of all records, sets, and areas
- Names and attributes of all data items within each record
- Location mode for each record
- Set characteristics
- Restrictions on use of DML statements

These are contained in three separate parts: the data structure diagram, record data description, and DML constraints.

12.2. DATA STRUCTURE DIAGRAM

The data structure diagram is a graphical means of showing the set relationships which exist among record types included in a subschema. Records are represented by a rectangular box containing the general characteristics of the record. Records are discussed in Section 3 and illustrated in Figure 3—1. Sets are represented by arrows which connect the rectangular boxes. The arrow points from the owner record type of the set to the member record type. Sets are discussed in Section 4 and illustrated in Figure 4—4.

Figure 12—1 is a simple data structure diagram of a subschema named DMSSUBS consisting of five record types and four sets. It is a typical structure for a manufacturing operation involving customers who place orders for items in inventory. In this illustration, the order items are related both to a particular order for a given customer and to the product inventory. Appendix C shows a real application of this subschema except that the location mode of ORD-REMARK is DIRECT instead of VIA. This is due to the fact that the sample program is intended primarily to demonstrate all three types of location modes. However, in the user's environment, the location mode of ORD-REMARK must be VIA mode to achieve high access performance.

The following detailed discussion of the DMSSUBS subschema in Figure 12—1 demonstrates the usefulness of data structure diagrams. Reference to Figures 3—1 and 4—4 may be required by the reader to understand abbreviations and representations in Figure 12—1.

The CUSTOMER record which is shown as having a location mode of CALC, uses a data item named CUST-NO-611 (customer number) as the CALC control data item. This record, as well as others in this subschema, are stored within the area named CUSTOMER-AREA. The CUSTOMER record is shown as the owner of the ORDOR set which contains CUST-ORDER member records. For a given customer, then, there is one occurrence of a CUST-ORDER record for every order the customer has placed. The set is maintained in ascending sequence based on the value of FO-NO-620 (factory order number), and no duplicate FO-NO-620 values are allowed. Each CUST-ORDER record is the owner of a SPEC-REMARK (special remarks) set containing ORD-REMARK (order remarks) records. The order of the SPEC-REMARK set is LAST, which means the first record stored in the SPEC-REMARK set is the first record encountered in the next direction of the set. This first in, first out (FIFO) sequence causes remark record to be retrieved in the same order as they were written and entered.

The ORD-REMARK record has a location mode of VIA SPEC-REMARKS set. This means that ORD-REMARK record occurrences are physically stored as close as possible to the CUST-ORDER records to which they belong. Usually the ORD-REMARK records are on the same page, track, or cylinder in which the CUST-ORDER owner record is stored. The ORDER-ITEM record is a member of the ITEM set, which also is located as close as possible to the CUST-ORDER owner record. This is shown by the VIA ITEM location mode. The CUST-ORDER record, thus, is shown as a member in the ORDOR set and an owner of the SPEC-REMARK and ITEM sets. Note that, when an occurrence of a CUST-ORDER record is selected by the system, all occurrences of ORD-REMARK and ITEM records usually are found on the same page and may be accessed in main storage without additional physical access to the data base. This was done to improve operational performance of the system since greatest access to ORDER-ITEM and ORD-REMARK records is through the SPEC-REMARK and ITEM sets associated with a CUST-ORDER record.

The PRODUCT record has a location mode of CALC based on the value of the data item PROD-NO-631 (product number). It is the owner record of the set named PROD-ORD which contains ORDER-ITEM member records. The PROD-ORD set thus contains an occurrence of an ORDER-ITEM record for every order which contains an item corresponding to the given product identified by the PRODUCT record. Thus, the ORDER-ITEM record, being a member in two different sets, allows selection of all orders placed for a given product by first accessing the PRODUCT record and then selecting every ORDER-ITEM record in the PROD-ORD set. The product items appearing in each order are selected by first accessing the CUST-ORDER record and then selecting each ORDER-ITEM record in the ITEM set. The sample program in Appendix C illustrates the two accessing paths.

The data structure diagram shows the basic characteristics of records and sets included in the subschema. The set relationships and location mode of each record show the paths which may be used to access record occurrences in the data base.

The strategy of using this structure diagram is discussed in Section 13.

12.3. RECORD DATA DESCRIPTION

When a subschema is invoked in a user program, the DML processor automatically inserts the data items contained in each record type of the subschema invoked in working storage of the user's COBOL program.

The order (or position) of each data item within a record is established by the data base administration staff when the record is described in the data base. All of the data items in a record that are not described with VALUE clauses must be initialized by using the COBOL MOVE statement prior to issuing a STORE statement. Whenever access to one or more data items within an existing record occurrence is desired, the DML OBTAIN or GET statement will move all data items from the system buffers into the defined record storage area in working storage.

The name of each data item is established at the time the record is described in the data base and cannot be changed by the COBOL programmer. A given data item containing the same information may appear in more than one record type. For example, the data item PROD-NO-631 (product number) in Figure 12—1 is the CALC control data itern contained in the PRODUCT record. The same product number value is also stored in each ORDER-ITEM record occurrence appearing in the PROD-ORD set for a given PRODUCT record. The reason for this data redundancy is to obtain increased performance of the system. Increased performance is achieved by not requiring an additional access to the PRODUCT record (to obtain product number) whenever a CUST-ORDER record is accessed along with its nearby ORDER-ITEM and ORD-REMARK record occurrences. In this way, the required product number is immediately available along with the other data items which make the ORDER-ITEM records. The trade-off between disc space and time was decied in favor of time.

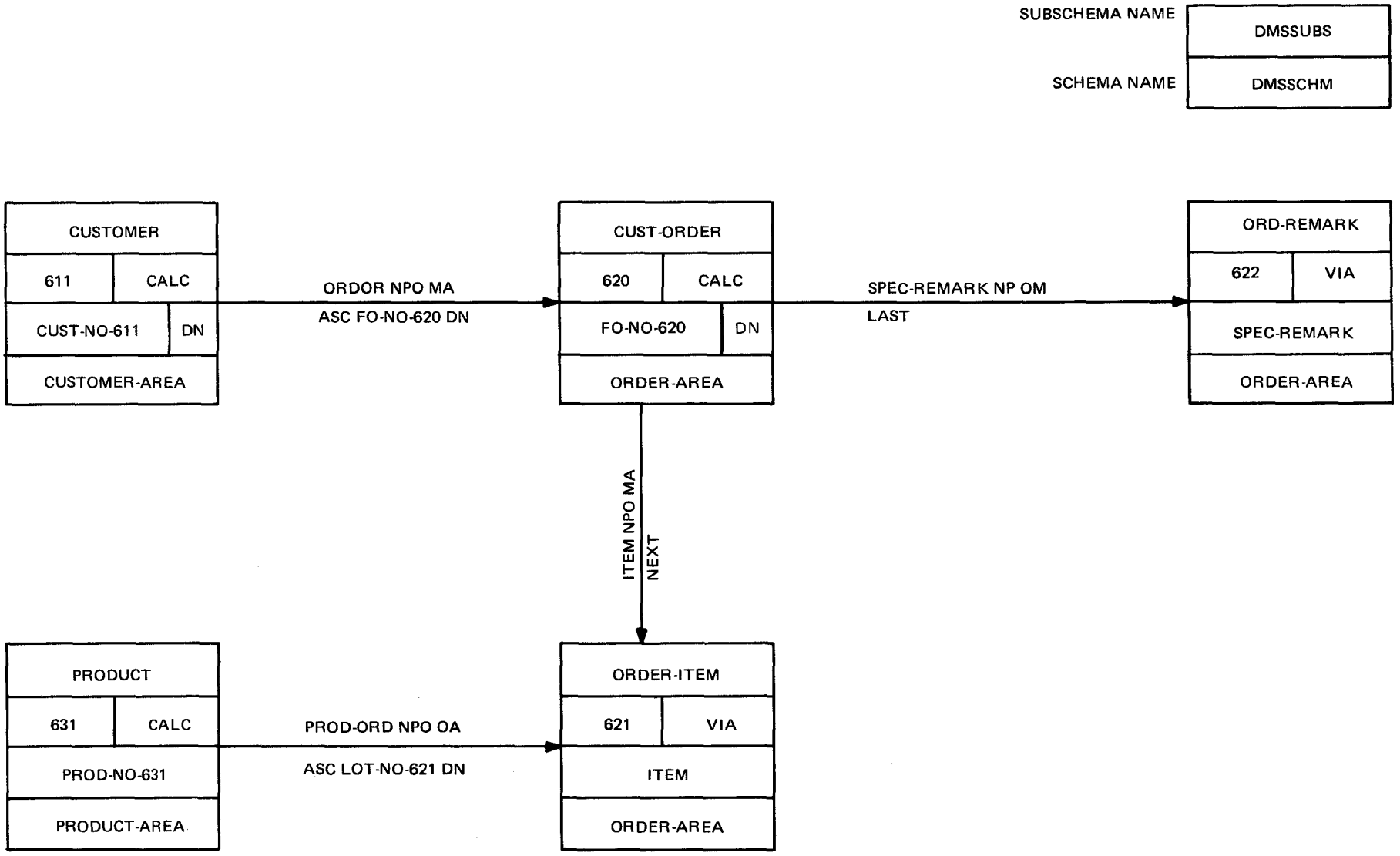


Figure 12—1. Data Structure Diagram

All data items in the example data base are made unique by appending a hyphen followed by the numeric record identification to each data name. The product number which appears in both PRODUCT and ORDER-ITEM records is made unique by naming it PROD-NO-631 in the PRODUCT record and PROD-NO-621 in the ORDER-ITEM record.

Figure 12—2 includes the record data description for the DMSSUBS subschema. The 01 level entry is the name of the record and corresponds to the top line of each rectangular record representation shown in Figure 12—1. This entry is preceded by a note comment which also contains the name of the record and the record identification number. The hyphen followed by the corresponding record identification number is appended to the name of each data item following the 01 level entry.

12.4. DATA MANIPULATION LANGUAGE CONSTRAINTS

The areas, records, and sets which are included in a given subschema represent only the records and sets of interest to a given application. By implication, all other areas, sets, and records are removed from view. Since the subschema is a part of the total schema of the data base, set relationships may exist between records which are not included in the subschema and records which are included. Whenever this condition occurs, certain restrictions must be placed on the use of DML statements which modify the data base.

- The STORE statement is not executed for a record that participates either as a member or as the owner of a set which has not been specified as part of the subschema.

- The DELETE statement is not executed for any record that is an owner or participates as a member in any set which has not been specified as part of the subschema.

  This rule applies to the record named in the DELETE statement, as well as any other records affected by a DELETE ALL or DELETE SELECTIVE statement.

- The MODIFY statement is not executed whenever modification of a data item value requires a logical repositioning of the record occurrence in a set which has not been defined as part of the subschema.

The subschema DML constraints also contain characteristics of records which cannot be conveniently shown on the data structure diagram. The constraint tabulation should show the following information for each record type.

Record-name

- STORE constraints
- DELETE constraints
- MODIFY constraints

STORE constraints should specify whether or not a record can be stored by the user. If a record can be stored and it is an automatic member in one or more sets, the names of the owner record types which must be selected prior to the store of the subject record will be listed.

The duplicate control information for all sets in which the subject record participates as a member is shown in the set representation. However, a CALC record may be defined to not allow or to allow records with duplicate values for the CALC control data item. Duplicate control information for CALC records are included. If a record type is an owner of a singular set, then only one occurrence of the owner and set exists in the data base.

DELETE constraints arise from data structure and other data relationships which are not included in a given subschema but are related to records included in the subschema. In some cases a record cannot be deleted under any circumstances. In other cases a record may be deleted under one or more of the options provided by the DELETE statement. In addition, any other special precautions or procedures which are involved in the delete process are listed.

```
*                **********  CUSTOMER RECORD  ****  ID 611
 01  CUSTOMER.
     05  CUST-NO-611                 PIC X(11).
     05  CUST-NAME-S-611             PIC X(35).
     05  DIV-NAME-S-611              PIC X(35).
     05  STRT-ADDR-S-611             PIC X(35).
     05  CITY-ADDR-S-611             PIC X(35).
     05  AREA-CD-611                 PIC X.
     05  COUNTRY-CD-611              PIC XXX.
     05  AR-STMT-CD-611              PIC X.
     05  INV-DIST-CD-611             PIC X.
     05  LOCK-BOX-CD-611             PIC X.
     05  NORM-PAY-T-611              PIC XX.
     05  SLS-CLAS-CD-611             PIC X.
     05  CR-CLASS-611                PIC X.
     05  SPLC-611                    PIC X(6).
     05  SLS-TAX-CD-611              PIC X.
     05  TAX-CERT-NO-611             PIC X(12).
     05  DATE-TAX-EXP-611            PIC X(6).
     05  PROD-CERT-611               PIC X.
     05  QLTY-RPT-CD-611             PIC X.
     05  FILLER-611                  PIC X(15).

*                **********  ORD-REMARK RECORD  **  ID 622
 01  ORD-REMARK.
     03  ORD-REM-CD-622.
     05  REMARK-CD-622               PIC X.
     05  REMARK-SEQ-622              PIC X.
     05  REMARK-622                  PIC X(75).
     05  FILLER-622                  PIC XXX.

*                **********  PRODUCT RECORD  ******  ID 631
 01  PRODUCT.
     05  PROD-NO-631                 PIC X(12).
     05  GRADE-631                   PIC X.
     05  COMPANY-CD-631              PIC X.
     05  PROD-DES-INT-631            PIC X(15).
     05  PROD-DES-EXT-631            PIC X(30).
     05  SLS-BRKD-CD-631             PIC X(4).
     05  ALT-SBC-631                 PIC X(4).
     05  INTER-DIV-PR-631  COMP-3    PIC 9V9999.
     05  PRICE-PKG-ID-631            PIC X.
     05  PRICE-YR-631                PIC XX.
     05  PRICE-CAT-631               PIC XX.
     05  BASE-PRICE-631    COMP-3    PIC 9V9999.
     05  PROPERTY-1-631              PIC X(5).
     05  PROPERTY-2-631              PIC X(5).
     05  PROPERTY-3-631              PIC X(5).
     05  PROPERTY-4-631              PIC X(5).
     05  IND-MGR-631                 PIC XX.
     05  FRT-CATEGORY-631            PIC XXX.
     05  SCHED-GRP-631               PIC XX.
     05  FILLER-631                  PIC X(7).

*                *****  CUST-ORDER RECORD  **  ID 620
 01  CUST-ORDER.
     05  FO-NO-620                   PIC X(8).
     05  CUST-PO-NO-620              PIC X(18).
     05  LOC-NO-620                  PIC XX.
     05  ORD-STAT-620                PIC X.
     05  DATE-STAT-CHG-620           PIC X(6).
     05  DEPT-NO-620                 PIC X(6).
     05  ACCT-NO-620                 PIC XXX.
     05  BILL-CD-620                 PIC X.
     05  GOVT-REGN-620               PIC XX.
     05  GOVT-PRTY-620               PIC X.
     05  PAY-TERM-620                PIC XX.
     05  FOB-TERM-620                PIC X.
     05  HOW-SHIP-620                PIC X.
     05  TYPE-ORD-620                PIC X.
     05  DATE-SHIP-620               PIC X(6).
     05  DATE-REQ-620                PIC X(6).
     05  DATE-PROM-620               PIC X(6).
     05  ROUTING-620                 PIC X(39).
     05  MILES-OW-620                PIC X(4).
     05  TRANS-TIME-620              PIC XX.
     05  SHIP-MODE-620               PIC X.
     05  SLSM-NO-620                 PIC XXX.
     05  TRADE-CD-620                PIC X(6).
     05  INVOICE-DIST-620            PIC X.
     05  INSTAT-NO-620               PIC X(4).
     05  SHIPSTAT-NO-620             PIC X(4).
     05  DATE-DELIVER-620            PIC X(6).
     05  DATE-PROCESS-620            PIC X(6).
     05  TIME-PROCESS-620            PIC X(4).
     05  DATE-ENTERED-620            PIC X(6).
     05  DATE-T4-REL-620             PIC X(6).
     05  DATE-CR-APP-620             PIC X(6).
     05  TIME-CR-APP-620             PIC X(4).
     05  REASON-CR-HELD-620          PIC X.
     05  UNITS-CD-620                PIC XXX.
     05  SLS-TAX-NO-620              PIC X(12).
     05  SLS-TAX-CD-620              PIC X.
     05  CLERK-CD-620                PIC X.
     05  FILLER                      PIC X(11).

*                ****  ORDER-ITEM RECORD ****  ID 621
 01  ORDER-ITEM.
     05  PROD-NO-621                 PIC X(12).
     05  GRADE-621                   PIC X.
     05  PKG-CD-621                  PIC XX.
     05  LOT-NO-621                  PIC X(7).
     05  SLS-BRKD-CD-621             PIC X(4).
     05  QTY-ORD-621       COMP-3    PIC S9(7).
     05  QTY-SHIP-621      COMP-3    PIC S9(7).
     05  NO-ITEMS-621      COMP-3    PIC S9(4).
     05  PRICE-UNIT-621    COMP-3    PIC 9V9999.
     05  CUST-MTL-CD-621             PIC X(14).
     05  RES-CD-621                  PIC X.
     05  REASON-LATE-621             PIC XX.
     05  FILLER-621                  PIC X(11).
```

Figure 12—2. Record Data Description for DMSSUBS Subschema

MODIFY constraints arise from the fact that one or more data items in a given record may be a control data item for a set which is not included in the given subschema. If a data item is an ascending key control data item, modification would adversely affect the logical order of the set in the data base.

Table 12—1 illustrates the DML constraints for the DMSSUBS subschema. Here, the CUSTOMER and PRODUCT records are associated with other sets in the schema DMSSCHM of the data base but not included in the DMSSUBS subschema. For this reason the STORE and DELETE DML statements are not allowed. Whenever either record type is retrieved, only one record occurrence will be found for a given CUST-NO-611 or PROD-NO-631 CALC control data item. All data items within each record type may be modified except for the CALC control data items.

The CUST-ORDER record can be stored by a program which uses the DMSSUBS subschema. This record also has a CALC location mode; duplicate records with the same factory order number (CALC control data item FO-NO-620) are not allowed. Remarks indicate that the appropriate CUSTOMER record (which is the owner of the ORDOR set) must be retrieved prior to a store of a CUST-ORDER member record.

The ORDER-ITEM record can be stored successfully, provided that the user has first retrieved the appropriate CUST-ORDER and PRODUCT record occurrences. ORDER-ITEM owner record occurrences also may be deleted and modified.

The ORD-REMARK record may be stored, provided that the appropriate CUST-ORDER owner record is first retrieved.

Table 12—1. DMSSUBS Subschema DML Constraints

Record Name	Store	CALC Duplicates	Delete	Modify	Remarks
CUSTOMER	Not Allowed	No	Not Allowed	All Except CUST-NO-611	
PRODUCT	Not Allowed	No	Not Allowed	All Except PROD-NO-631	
CUST-ORDER	Allowed	No	Allowed	Allowed	Store requires find of CUSTOMER.
ORDER-ITEM	Allowed	—	Allowed	Allowed	Store requires find of CUST-ORDER and PRODUCT.
ORD-REMARK	Allowed	—	Allowed	Allowed	Store requires find of CUST-ORDER.

13. Programming Strategy

13.1. GENERAL

Programming strategy is a plan whereby the subschema documentation, currency status, and the data manipulation language (DML) statements are used to access data in the data base as required by an application program. In many cases, required data may be accessed from several paths. One path or approach may be more efficient than another in terms of processing efficiency. Good strategy involves the proper use of the structure paths provided by sets and access entries (e.g., CALC) in such a way as to reduce the total number of record occurrence accesses to a minimum.

This section discusses various approaches to the solution of data access requirements through the use of example problems based on Figures 12—1 and 12—2 and Table 12—1.

13.2. ENTRY INTO THE DATA BASE

Knowledge of all possible points of entry into the data base is the first step in developing an access strategy for the data to be retrieved. The choice of entry point depends upon judgments involving the problem to be solved, processing efficiency, and available input information.

13.2.1. CALC Entry

Entry into the data base can be made through any CALC record, provided the CALC control data item is known. In Figure 12—1, the CUSTOMER, CUST-ORDER, and PRODUCT records can be accessed directly if the values of CUST-NO-611, FO-NO-620, or PROD-NO-631 are known beforehand. Once accessed successfully, any set associated with the record provides a path leading to other records in the data base. For example, if a PRODUCT record was successfully accessed, the PROD-ORD set would provide a means of accessing the ORDER-ITEM record for all orders associated with the PRODUCT record.

13.2.2. Direct Entry

Direct access of any record occurrence in the data base is possible provided that its data base key is known beforehand. A data base key for a given record may be saved when the record was current of run-unit by executing the statement:

```
MOVE DBKEY TO MY-DBKEY
```

The data base key in the item named MY-DBKEY can be used at any later time to directly retrieve the record identified by the contents of the MY-DBKEY item. Since the data base key assigned to a record remains unchanged for the entire time that the record is present in the data base, the value of the MY-DBKEY record may be output from the run-unit for reentry at some later time.

A more common use of this type of data base entry is in situations where the programmer has found a record which will be reaccessed after performing several intervening data base operations. For performance reasons, the programmer may decide that use of the direct access path is more efficient than the path used to find the record the first time. In either case, the programmer must know, the type of record which is to be accessed by the data base key. The statement

```
FIND ORDER-ITEM RECORD USING MY-DBKEY
```

is successful provided that the record occurrence identified by MY-DBKEY is an ORDER-ITEM type record.

13.2.3. Area Entry

Access to an occurrence of any record in the subschema, regardless of location mode, can be made by the FIND statement option which specifies either the first or last record of the area. If the statement

```
FIND FIRST CUSTOMER RECORD OF CUSTOMER-AREA AREA.
```

is successfully executed, the CUSTOMER record with the lowest data base key value in the CUSTOMER-AREA area will be made current of run-unit, current of CUSTOMER record type, current of ORDOR set, and current of CUSTOMER-AREA area. The statement

```
FIND NEXT CUSTOMER RECORD OF CUSTOMER-AREA AREA.
```

would access the CUSTOMER record occurrence with the next higher data base key. Repetitive execution of this statement would proceed to access all occurrences of the CUSTOMER record in the CUSTOMER-AREA area in data base key sequence.

13.3. DATA ACCESS THROUGH SET RELATIONSHIPS

Once an entry has been made into the data base, currency status is updated for all sets in which the record participates, and the record becomes the current of its record type, the current of the sets in which the record is an owner or participates as a member, current of area, and current of run-unit. This enables the programmer to step in any pathway provided by the set relationships. Two general rules for set processing are:

- If the record participates as either an owner or member of a set, then the FIND NEXT statement will access the next record in the named set. If the set is defined as linked prior, then the FIND PRIOR statement may also be used.

- If the record is a member of a set, the FIND OWNER statement can be used to access the owner record occurrence of the set.

If a set is ordered through use of an ascending or descending key, the record occurrence in a set with the desired key may be selected by the following statements, which find a specific CUST-ORDER record in the ORDOR set with a factory order number of 70016934, assuming that the appropriate CUSTOMER record has previously been found.

```
MOVE '70016934' TO FO-NO-620.
FIND CUST-ORDER RECORD VIA CURRENT OF ORDOR SET USING FO-NO-620.
```

The system searches the current ORDOR set occurrence to find the CUST-ORDER record occurrence which contains a value of 70016934 for FO-NO-620.

14. Diagnostic and Error Messages

14.1. GENERAL

The first part of this section includes the diagnostic error messages produced by the data manipulation language (DML) processor in its analysis and conversion of DML statements into COBOL CALL statements.

The second part includes a description of each ERROR-STATUS condition which can occur during an attempted execution of a DML statement and a discussion of possible causes for each condition.

14.2. DML PROCESSOR ERROR MESSAGES

The DML processor displays the statement which is in error followed by an error message that indicates the type of error which has occurred. Appendix E contains a listing of these errors and an explanation to aid in correction.

14.3 ERROR-STATUS CONDITIONS

Whenever a DML statement is executed, extensive error checking occurs to ensure that the requested data base operation can be performed without affecting the integrity of the data base. The nonzero values placed in the ERROR-STATUS location indicate errors of the following types:

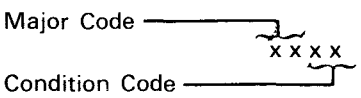
- Conditions which are useful in the logical operation of the program. These are not errors as far as the data base is concerned and are passed to the program for further analysis. Examples of this condition are END OF SET and NO RECORD FOUND.

- Conditions which indicate an incorrect preparation for the statement to be executed. Examples of this condition are NO CURRENT RECORD OF RUN-UNIT and NO CURRENT OF SET-NAME.

- Conditions which would violate the integrity of the data base as defined by the subschema invoked by the program and the usage-mode of the OPEN statement. Examples of this condition are INCORRECT USAGE-MODE and VIOLATION OF DUPLICATES NOT ALLOWED.

- Conditions which indicate system-level errors that require attention of a systems programmer.

The ERROR-STATUS condition is described as a 2-part data item consisting of a major code and a condition code formatted as follows:



The 2-byte major code identifies the type of DML statement which was attempted prior to the error condition. The values of each major code and its corresponding DML verb are shown in Table 14—1.

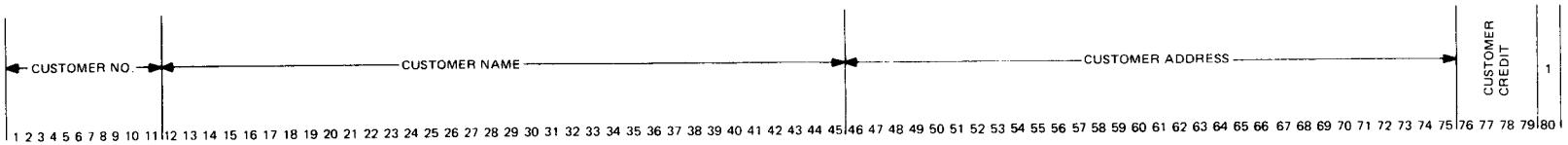
The 2-byte condition code defines the type of error which occurred. Condition code values from 01 through 59 are reserved for all error types other than system type errors. Condition code values from 60 through 99 are reserved for system type errors.

Appendix D contains an explanation of the error and the possible causes to aid correction.

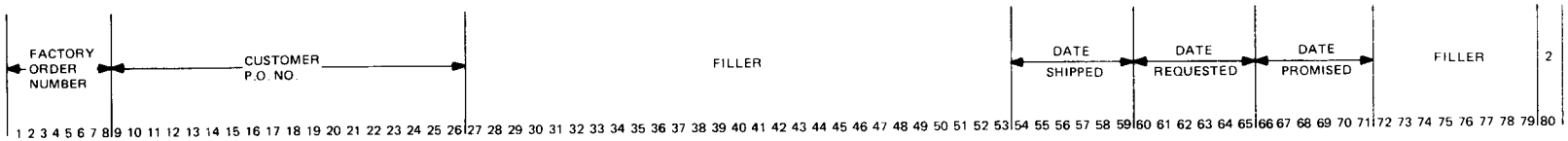
Table 14—1. DML Verb Codes

Major Code	DML Verb
01	CLOSE
02	DELETE
03	FIND/OBTAIN
05	GET
07	INSERT
08	MODIFY
09	OPEN
11	REMOVE
12	STORE
16	IF

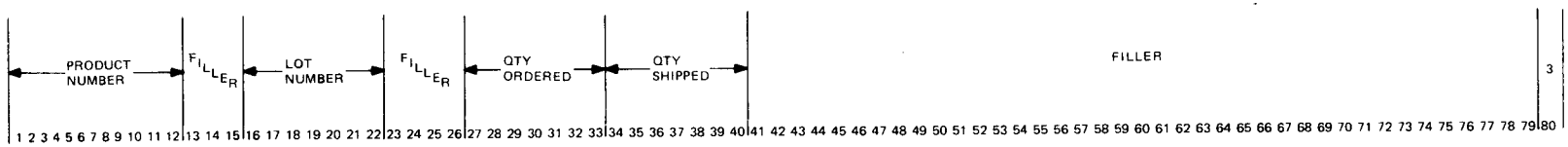
C.5. DATA INPUT RECORDS



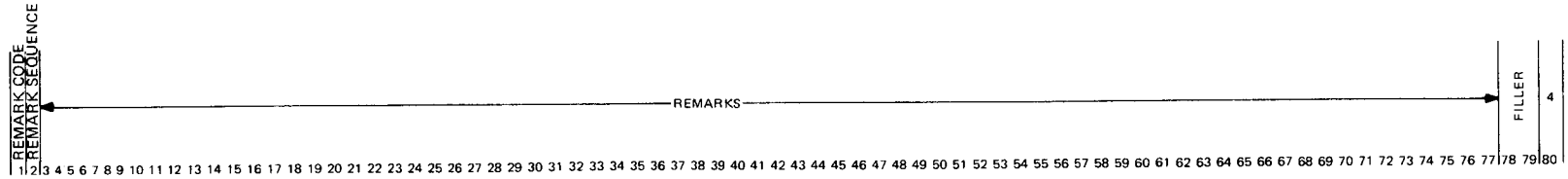
CUSTOMER RECORD TYPE



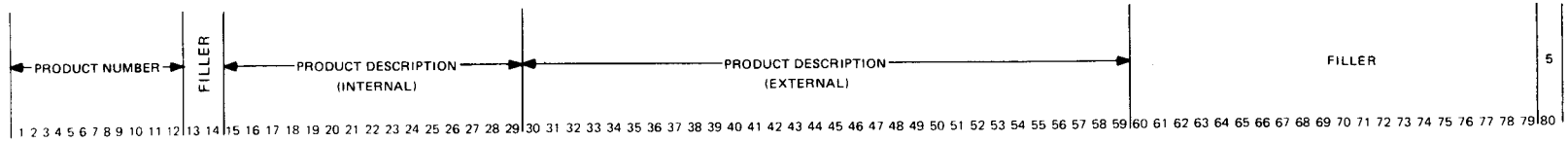
CUSTOMER ORDER RECORD TYPE



ORDER ITEM RECORD TYPE



ORDER REMARK RECORD TYPE



PRODUCT RECORD TYPE

C.6. SAMPLE PROGRAM DATA INPUT

```
31THIS IS GENERAL REMARK 3 - USE ONLY FOR AAA CREDIT RATINGS                    4
PRODUCT   01  CODE 5301      SUPER V RADIAL SPARE  FR70-14                     5
PRODUCT   02  CODE 5302      TUBELESS BELTED       GR70-15                     5
PRODUCT   03  CODE 5303      CUPROUS SULFATE       CU2S04                      5
PRODUCT   04  CODE 5304      BURN IT AWAY CLEANSER H2S04                       5
PRODUCT   05  CODE 5305      SUPER CLEANER    HCL                              5
PRODUCT   06  CODE 5306      TUBE VALVE STEMS      TVS-N15                     5
PRODUCT   07  CODE 5307      DMS/90 PKG.           RFS-001                     5
CUSTOMER 01MELCHER OIL COMPANY              2100 SOUTH BAY STREET             1
ORDER 01MEL PO # 15                           08157208307208307 2             2
PRODUCT   06   LOT  01     00000560000056
PRODUCT   01   LOT  01     00000180000012
PRODUCT   03   LOT  01     00000010000001
PRODUCT   05   LOT  01     00000200000019
11SEND DIRECT TO STEVE SINGER,                                                  4
12     1885 MAIN ST                                                             4
21SPECIAL TERMS ARE - 75% OFF                                                   4
CUSTOMER 02RED STAR SERVICE                 15201 N. MAIN                     1
ORDER 02RED PO # 82                           06177206157206167 2             2
PRODUCT   06   LOT  02     00000160000015
PRODUCT   01   LOT  02     00000220000021
PRODUCT   04   LOT  02     00000180000018
PRODUCT   07   LOT  02     00000010000001
PRODUCT   05   LOT  02     00000360000018
PRODUCT   03   LOT  02     00000520000005
ORDER 03RED PO # 56                           02137203187203187 2             2
PRODUCT   01   LOT  03     00000190000018
PRODUCT   06   LOT  03     00001500000080
PRODUCT   05   LOT  03     00010000001000
PRODUCT   02   LOT  03     00000210000021
CUSTOMER 03ATLANTIC TIRES, INC.             1802 HIGHLAND SQUARE RD           1
CUSTOMER 04WALLHAVEN CHEMICALS              111 CASCADE PLAZA                 1
ORDER 04WAL PO # 93                           05117205107205107 2             2
PRODUCT   07   LOT  04     00000010000001
CUSTOMER 05CHIPPEWA SUPPLY CO.              83 W. HAWKINS AVE.                1
CUSTOMER 06SHOE, BERT ENTERPRIZES INC       ONE AMERICAN WAY BLVD.          AAA1
ORDER 05SHO PO # 56                           12187212157212257 2             2
PRODUCT   03   LOT  05     00000180000017
PRODUCT   06   LOT  05     00000150000010
PRODUCT   07   LOT  05     00000100000010
CUSTOMER 07QUIK STOP SERVICE                1895 MAIN STREET                  1
CUSTOMER 08GOWAY, INC.                      8888 OAK TREE BLVD.             AAA1
ORDER 06DON PO # 13                           09027208167209017 2             2
PRODUCT   01   LOT  06     00000010000001
PRODUCT   02   LOT  06     00000020000002
PRODUCT   03   LOT  06     00000030000003
PRODUCT   04   LOT  06     00000040000004
PRODUCT   05   LOT  06     00000050000005
PRODUCT   06   LOT  06     00000060000006
PRODUCT   07   LOT  06     00000150000015
EOF
```

DML Verb	ERROR-STATUS Code	Explanation
FIND/ OBTAIN (cont)	0307	End of set or area. Normally this condition exists when the next or prior record of set or area is referenced and the most recently retrieved record was the last occurrence in the set or area. This condition also occurs if a set occurrence is empty.
	0308	Invalid record-name or set-name was used. The causes are: ■ Misspelled record-name or set-name ■ Incorrect subschema invoked by program ■ Record-name not defined as a member of set-name
	0313	No current record of run-unit. The current of run-unit option in format 2 was used and no current status exists for run-unit.
	0317	Deleted record involved. The record name has been deleted by this or another run-unit.
	0323	Illegal area-name used. This is caused by using an area-name which was not included in the subschema. This also occurs when an area-name and record-name are used in format 3 and record-name is not defined within area-name.
	0326	The object record of this statement was not found. This applies to formats 1, 5, and 6.
	0331	The statement format used conflicts with the location mode defined for the object record. Causes are: ■ Use of format 5 to retrieve a record which does not have a CALC location mode ■ Use of format 6 to retrieve a record which is not stored on the basis of an ascending or descending key
	0332	Format 5 with the NEXT DUPLICATE option was attempted and the value of the CALC data item in working storage does not equal the value of the CALC data item in the current record of run-unit in the data base. The first record of a number of duplicate records must be retrieved with a FIND statement, without the NEXT DUPLICATES option. Subsequent use of FIND with the NEXT DUPLICATE option retrieves the remaining records with CALC keys equal to the CALC key of the first record.
	0335	In format 1, the identifier is not aligned on the word boundary. The user must include SYNC in the description of the identifier.
GET	0508	Record-name specified is not included in this subschema. Causes are: ■ Wrong subschema invoked by this program ■ Incorrect or misspelled record-name
	0513	No current record of run-unit. This statement operates on the current of run-unit. Causes are: ■ A previous DELETE statement was executed and made the current of run-unit null. ■ A previous FIND statement was not executed successfully.

DML Verb	ERROR-STATUS Code	Explanation
GET (cont)	0520	The current record of the run-unit is not of the same type as that specified by record-name in this statement. Causes are: ■ Incorrect record-name in the statement ■ Failure to properly establish the current record of run-unit
INSERT	0701	The areas included in the subschema were not opened prior to the execution of this statement. Causes are: ■ The OPEN statement is missing or was bypassed by program logic. ■ The CLOSE statement was executed prior to this statement.
	0705	Insertion of the named record violates the duplicates not allowed restriction on set membership. Possible causes are: ■ Program logic error ■ Input data error
	0706	No current record of record-name exists. Possible causes are: ■ Failure to establish the record to be inserted as the current of record-name ■ Program logic error
	0708	Record-name specified in this statement is not included in the subschema invoked by the program. Possible causes are: ■ Incorrect subschema invoked ■ Misspelled record-name
	0709	Areas included in this subschema are open with an incorrect USAGE-MODE option. Causes are: ■ The INSERT statement is disallowed for this subschema. ■ The OPEN statement specified USAGE-MODE IS RETRIEVAL instead of EXCLUSIVE UPDATE.
	0714	The object record is not defined as an optional automatic, optional manual, or mandatory manual member of set-name. Cause is an incorrect use of the INSERT statement or an incorrect set-name.

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

CUT

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

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

POSTAGE WILL BE PAID BY

SPERRY UNIVAC

P.O. BOX 500
BLUE BELL, PA.
19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT

Changes to UP-8036 Rev. 1 (with Update B)

Page 2-5

The third paragraph (second bulleted item) of 2.6.1 should be changed as follows:

- The INVOKE statement must follow the SCHEMA SECTION statement and may begin in any column after column 7, but cannot continue onto separate lines. The INVOKE statement (sequence number 300020) corresponds to the subschema and schema names in Figure 12-1. No other statements are permitted in the Schema Section.

Page 2-7

In the sixth paragraph, line 3, of 2.6.2, change ''Figure 2-4 illustrates'' to ''Figure 2-4a and Figure 2-4b illustrate...''

Page 2-8

Delete the paragraph preceding the illustration which reads:

> ''In Figure 2-4, THE-PRINTER and THE-CONSOLE are used in DISPLAY statements of the DMS-STATUS SECTION; therefore, the user must relate these mnemonic-names to the appropriate implementor names in the SPECIAL-NAMES entry of ENVIRONMENT DIVISION.''

Replace Figure 2-4 with Figure 2-4a and Figure 2-4b.

Page 7-3

Change the line in Table 7-1 of the ''Move Currency Status'' DML Statement. The ERROR-STATUS column under ''Unsuccessful'' should read ''15XX''; and, the ERROR-SET, ERROR-RECORD, and ERROR-AREA columns should read ''Yes''.

Page 7-4

In 7.3.3, delete the last sentence in the first paragraph that begins with ''This operation is performed...''

Replace the second paragraph that begins ''A MOVE CURRENCY STATUS...'' with:

> An unsuccessful MOVE CURRENCY STATUS statement causes the system to place an error code (15XX) in the ERROR-STATUS location to define the error which occurred. The ERROR-SET, ERROR-RECORD, and ERROR-AREA locations contain names associated with the error.

Page 8-1

In the last sentence of the last paragraph of 8.2, delete ''INSERT,''.

Page 8-3

In the title of 8.6.2, insert ''MODIFY'' after ''GET''.

In the second sentence of 8.6.2, insert ''MODIFY'' after ''GET''.

```
00010000      *
00020000      *          ************************************************************
00030000      *          *  DMS STATUS CHECK SECTION                                *
00040000      *          ************************************************************
00050000      *
00060000       DMS-STATUS            SECTION.
00070000       STATUS-PARA.
00080000      *
00090000                 IF          ERROR-STATUS EQUAL TO ZEROS
00100000                   PERFORM DMS-SUCCESS GO TO ISABEX.
00110000                 DISPLAY     '** DMS/90 RUN-UNIT TERMINATED BY DML ERROR'
00120000                             UPON CONSOLE.
00130000                 DISPLAY     '   PROGRAM NAME ------ ' PROGRAM-NAME
00140000                             UPON CONSOLE.
00150000                 DISPLAY     '   ERROR STATUS ------ ' ERROR-STATUS
00160000                             UPON CONSOLE.
00170000                 DISPLAY     '   ERROR RECORD ------ ' ERROR-RECORD
00180000                             UPON CONSOLE.
00190000                 DISPLAY     '** EXECUTE DBBAR IF DATA BASE WAS UPDATED'
00200000                             UPON CONSOLE.
00210000                 DISPLAY     '** DMS/90 RUN-UNIT TERMINATED BY DML ERROR'
00220000                             UPON TERMINAL.
00230000                 DISPLAY     '   PROGRAM NAME ------ ' PROGRAM-NAME
00240000                             UPON TERMINAL.
00250000                 DISPLAY     '   ERROR STATUS ------ ' ERROR-STATUS
00260000                             UPON TERMINAL.
00270000                 DISPLAY     '   ERROR RECORD ------ ' ERROR-RECORD
00280000                             UPON TERMINAL.
00290000                 DISPLAY     '   ERROR SET --------- ' ERROR-SET
00300000                             UPON TERMINAL.
00310000                 DISPLAY     '   ERROR AREA -------- ' ERROR-AREA
00320000                             UPON TERMINAL.
00330000                 DISPLAY     '   LAST GOOD RECORD -- ' RECORD-NAME
00340000                             UPON TERMINAL.
00350000                 DISPLAY     '   LAST GOOD AREA ---- ' AREA-NAME
00360000                             UPON TERMINAL.
00370000                 DISPLAY     '** EXECUTE DBBAR IF DATA BASE WAS UPDATED'
00380000                             UPON TERMINAL.
00390000                 PERFORM     DMS-ABORT.
00400000      *          CLOSE ALL AREAS.
00410000                 ENTER LINKAGE.
00420000                 CALL  XR7DMS  USING IDBMSCOM (02).
00430000                 ENTER COBOL.
00440000                 STOP RUN.
00450000      *
00460000       ISABEX. EXIT.
00470000      *
00480000                 NOTE
00490000                 THIS IS THE END
00500000                    OF THE DMS STATUS CHECK SECTION.
```

Figure 2-4A. DMS-STATUS SECTION for VS/9

```
00010000      *
00020000      *          ************************************************************
00030000      *          *  DMS STATUS CHECK SECTION                                *
00040000      *          ************************************************************
00050000      *
00060000       DMS-STATUS            SECTION.
00070000       STATUS-PARA.
00080000      *
00090000                 IF          ERROR-STATUS EQUAL TO ZEROS
00100000                   PERFORM DMS-SUCCESS GO TO ISABEX.
00110000                 DISPLAY     '** DMS/90 RUN-UNIT TERMINATED BY DML ERROR'
00120000                             UPON CONSOLE.
00130000                 DISPLAY     '   PROGRAM NAME ------ ' PROGRAM-NAME
00140000                             UPON CONSOLE.
00150000                 DISPLAY     '   ERROR STATUS ------ ' ERROR-STATUS
00160000                             UPON CONSOLE.
00170000                 DISPLAY     '   ERROR RECORD ------ ' ERROR-RECORD
00180000                             UPON CONSOLE.
00190000                 DISPLAY     '** EXECUTE DBBAR IF DATA BASE WAS UPDATED'
00200000                             UPON CONSOLE.
00210000                 DISPLAY     '** DMS/90 RUN-UNIT TERMINATED BY DML ERROR'
00220000                             UPON TERMINAL.
00230000                 DISPLAY     '   PROGRAM NAME ------ ' PROGRAM-NAME
00240000                             UPON TERMINAL.
00250000                 DISPLAY     '   ERROR STATUS ------ ' ERROR-STATUS
00260000                             UPON TERMINAL.
00270000                 DISPLAY     '   ERROR RECORD ------ ' ERROR-RECORD
00280000                             UPON TERMINAL.
00290000                 DISPLAY     '   ERROR SET --------- ' ERROR-SET
00300000                             UPON TERMINAL.
00310000                 DISPLAY     '   ERROR AREA -------- ' ERROR-AREA
00320000                             UPON TERMINAL.
00330000                 DISPLAY     '   LAST GOOD RECORD -- ' RECORD-NAME
00340000                             UPON TERMINAL.
00350000                 DISPLAY     '   LAST GOOD AREA ---- ' AREA-NAME
00360000                             UPON TERMINAL.
00370000                 DISPLAY     '** EXECUTE DBBAR IF DATA BASE WAS UPDATED'
00380000                             UPON TERMINAL.
00390000                 PERFORM     DMS-ABORT.
00400000      *          CLOSE ALL AREAS.
00410000                 ENTER LINKAGE.
00420000                 CALL  XR7DMS  USING IDBMSCOM (02).
00430000                 ENTER COBOL.
00440000                 STOP RUN.
00450000      *
00460000       ISABEX. EXIT.
00470000      *
00480000                 NOTE
00490000                 THIS IS THE END
00500000                    OF THE DMS STATUS CHECK SECTION.
```

Figure 2-4B. DMS-STATUS SECTION for OS/3

In the second sentence of the first paragraph of 8.6.3, insert ''or OBTAIN'' after ''FIND''.

Page 9-2

Insert an additional sentence at the end of rule 2 that reads ''The INVOKE statement should not span more than one line.''

Page 10-9

Change the last part of the format of the MOVE statement to read:

```
{ RUN-UNIT          }
{ record-name RECORD } TO identifier.
{ area-name AREA    }
{ set-name SET      }
```

Page 11-3

Replace rule 10 with:

The DELETE statement is not executed for any record that is:

- an owner or member of any set that has not been specified as part of the subschema,
- an owner of a set whose members are not specified as part of the subschema, or
- an owner of a set whose members are themselves nondeletable.

Page 11-5

In 11.3, insert an additional sentence at the end of rule 1 that reads:

All member record types of set name must also be included in the subschema.

Pages 11-6 and 11-7

Delete the last sentence of rule 1.

Insert a new rule 2 that reads:

2. The MODIFY statement is not executed for any record that is a member of a sorted set that has not been specified as part of the subschema or is a member of a sorted set whose numbers are all not part of the subschema.

Change the rule numbers for 2 through 10 to 3 through 11.

Page 11-8

Insert an additional sentence at the end of rule 1 that reads:

All member record types of set name must also be included in the subschema.

Page 11-9

Replace rule 3 with:

All sets in which the named record is defined as an automatic member must be included in the subschema before a STORE statement for the named record can be executed. Also, for each set type in this category, all member record types must be included in the subschema. If the subschema does not contain all required records and sets, the STORE statement will not be executed.

Page 12-6

In the second sentence of the second paragraph, replace ''associated with'' with ''owners of''. In the same sentence following ''of the data base but'' insert ''the sets and member records are''. In the third sentence replace ''now'' with ''not''. In the last sentence delete ''except for the CALC control data items''.

In Table 12-1, in the CUSTOMER row under the Modify column, replace ''All Except CUST-NO-611'' with ''Allowed''; under the Remarks column insert ''CUST-NO-611 cannot be duplicated.'' In the PRODUCT row under the Modify column replace ''All except PROD-NO-631'' with ''Allowed'' and under the Remarks column insert ''PROD-NO-631 cannot be duplicated.''

Page D-5

Under the DML Verb MODIFY, delete the 0806 ERROR-STATUS code and its Explanation.

Page D-8

Between the STORE and IF DML Verbs, insert the DML Verb MOVE with one ERROR-STATUS code: 1508.

The Explanation is:

The record-name specified is not defined within the subschema involved by this program. Possible causes are:

- Incorrect subschema involved.
- Misspelled record name.