


Burroughs 

B 1000 Systems
Data Management
System II
Inquiry

REFERENCE MANUAL

PRICED ITEM

Burroughs 

B 1000 Systems
Data Management
System II
Inquiry

REFERENCE MANUAL

Copyright © 1979 Burroughs Corporation, Detroit, Michigan 48232

PRICED ITEM

Burroughs believes that the information described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, is accepted for any consequences arising out of the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this document should be addressed directly to Burroughs Corporation,
P. O. Box 4040, El Monte, California 91734, Attn: Publications Department, TIO - West.

TABLE OF CONTENTS

Section		Page
	PREFACE	v
1	INTRODUCTION	1-1
2	SYNTAX CONVENTIONS	2-1
	Railroad Diagrams	2-1
	Railroad Components	2-1
	Required Items	2-1
	Optional Items	2-2
	Loops	2-2
	Bridges	2-3
3	GENERAL DESCRIPTION	3-1
	Introduction	3-1
	Description of INQUIRY Statements	3-1
4	USE OF INQUIRY	4-1
	Terminal Use and Data Base Access	4-1
	Entering INQUIRY Statements	4-1
	Basic INQUIRY Process	4-2
	Record Selection (Select)	4-3
	Structure Designation	4-5
	Item Display (Display)	4-6
	Qualification	4-7
	Combining Record Selection and Item Display	4-8
	Output Control (Select/Display)	4-8
	INQUIRY of Embedded Structures	4-9
	Functions	4-11
	Arithmetic Function	4-12
	Displaying Arithmetic Function Results	4-13
	Virtual Items	4-14
	Undefined Items	4-15
	Unquoted Alpha-Literals	4-16
	Output Formatting	4-17
	HEADING	4-17
	TAB	4-18
	SINGLE	4-18
	DEFAULT	4-19
5	FORMAL INQUIRY DESCRIPTION	5-1
	Example Data Base	5-1
	SELECT/DISPLAY	5-2
	Selection Condition	5-5
	DISPLAY List	5-6
	SORT Syntax	5-7
	Functions	5-8
	Boolean Function	5-8
	Arithmetic Function	5-8

TABLE OF CONTENTS (Cont.)

Section		Page
5	FORMAL INQUIRY DESCRIPTION (Cont)	
	Arithmetic Expressions	5-10
	Items	5-11
	Examples of SELECT/DISPLAY Statements	5-12
	ATTACH	5-14
	CLEAR	5-15
	DEFINE	5-16
	DETACH	5-17
	EDIT	5-18
	GENERATE	5-19
	HELP	5-20
	NEXT	5-21
	OPTIONS	5-22
	PRINTER	5-23
	QUIT	5-24
	RECALL	5-25
	REPEAT	5-26
	RESTORE	5-27
	SAVE	5-28
	SET	5-29
	SHOW	5-30
	SORT	5-31
	TERMINAL	5-32
	VIRTUAL	5-33
6	INQUIRY FILES AND SYSTEM GENERATION	6-1
	General	6-1
	Running Environments	6-1
	INQUIRY Generation	6-1
	Security	6-3
	Timeout	6-3

PREFACE

This manual describes the DMSII INQUIRY system. The manual is divided into 6 sections, of which Sections 1 through 4 are somewhat tutorial in nature. The reader can benefit by reading this material thoroughly before proceeding with the more detailed information in Sections 5 and 6.

Section	Contents
1	INTRODUCTION Provides a brief introduction to the INQUIRY system.
2	SYNTAX CONVENTIONS The syntax conventions used throughout this manual are described in Section 2.
3	GENERAL DESCRIPTION Provides a general description of the INQUIRY system and also a brief description of each INQUIRY statement.
4	USE OF INQUIRY Explains INQUIRY through the extensive use of examples.
5	FORMAL INQUIRY DESCRIPTION Provides the formal description of the INQUIRY system. Included in this section is a detailed description of the INQUIRY statements, along with other features to aid and simplify the use of INQUIRY.
6	INQUIRY FILES AND SYSTEM GENERATION Provides information concerning the initialization of the INQUIRY system and also the information needed to execute the DMS/BUILDINQ program.

SECTION 1

INTRODUCTION

The DMS/INQUIRY system (hereinafter referred to as INQUIRY) consists of three basic components: The DMS/BUILDINQ program, the DMS/INQUIRY program, and the DMS/HELPINQ file.

INQUIRY provides a convenient method by which a user can, via a terminal, examine information in a DMSII data base.

The following concepts are incorporated in INQUIRY:

1. INQUIRY is an on-line interactive system which is simple enough that it can easily be used by people who are not trained in data processing.
2. INQUIRY can examine information in any part of a data base regardless of the complexity of the data base.
3. INQUIRY takes advantage of sets, if possible, in extracting information from a data base.
4. INQUIRY always produces the requested information, even if it is necessary to perform linear file searches to satisfy the request. Although these searches require time, an answer can be produced more rapidly than a user can design, code, and debug a program that satisfies the same request.
5. INQUIRY allows a user to examine the description of the data base from a terminal.
6. INQUIRY contains relatively few statements, but these can be combined to perform complex operations.
7. INQUIRY can generate simple reports from data contained in DMSII data bases.

SECTION 2 SYNTAX CONVENTIONS

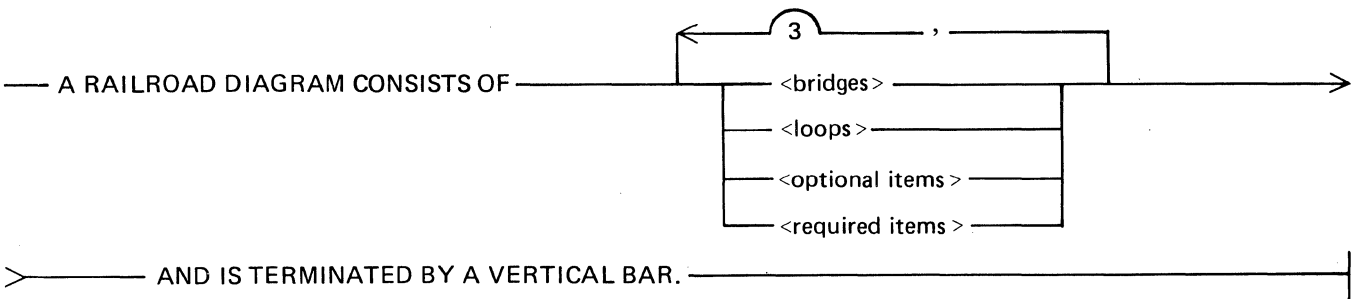
RAILROAD DIAGRAMS

Railroad diagrams show how syntactically valid statements can be constructed.

Traversing a railroad diagram from left to right, or in the direction of the arrow heads, and adhering to the limits indicated by bridges will produce a syntactically valid statement. Continuation from one line of a diagram to another is represented by a right arrow (-->) appearing at the end of the current line and beginning of the next line. The complete syntax diagram is terminated by a vertical bar (!).

Items contained in broken brackets (< >) are syntactic variables which are further defined, or require the user to supply the requested information.

Upper-case items must appear literally. Minimum abbreviations of upper-case items are underlined.



The following syntactically valid statements may be constructed from the above diagram:

A RAILROAD DIAGRAM CONSISTS OF <bridges> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <optional items> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

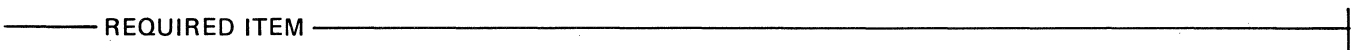
A RAILROAD DIAGRAM CONSISTS OF <optional items>, <required items>, <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

Railroad Components

Required Items

No alternate path through the railroad diagram exists for required items or required punctuation.

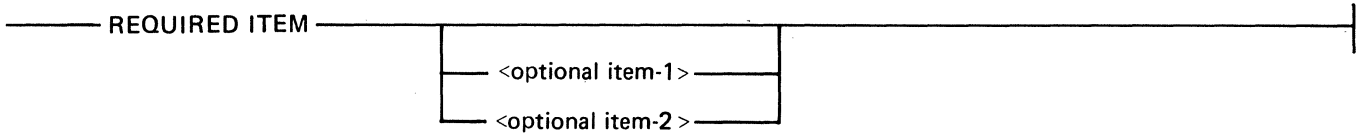
Example:



Optional Items

Items shown as a vertical list indicate that the user must make a choice of the items specified. An empty path through the list allows the optional item to be absent.

Example:



The following valid statements may be constructed from the above diagram:

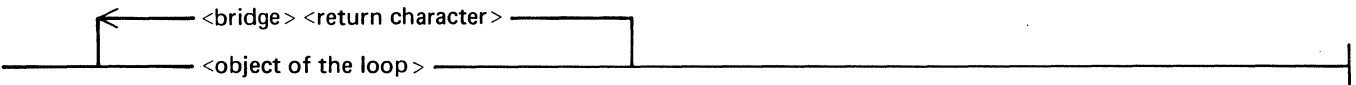
REQUIRED ITEM

REQUIRED ITEM <optional item-1>

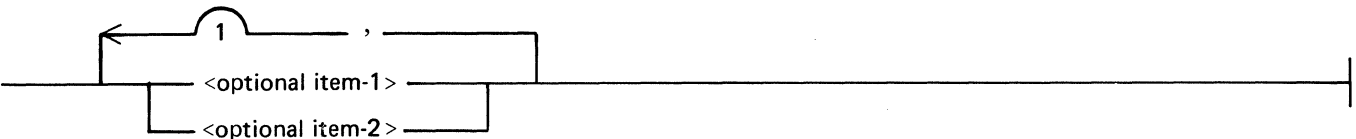
REQUIRED ITEM <optional item-2>

Loops

A loop is a recurrent path through a railroad diagram and has the following general format:



Example:



The following statements can be constructed from the railroad diagram in the example.

<optional item-1>

<optional item-1>,<optional item-1>

<optional item-2>,<optional item-1>

A <loop> must be traversed in the direction of the arrow heads, and the limits specified by bridges cannot be exceeded.

Bridges

A bridge indicates the minimum or maximum number of times a path may be traversed in a railroad diagram.

There are two forms of <bridges>.

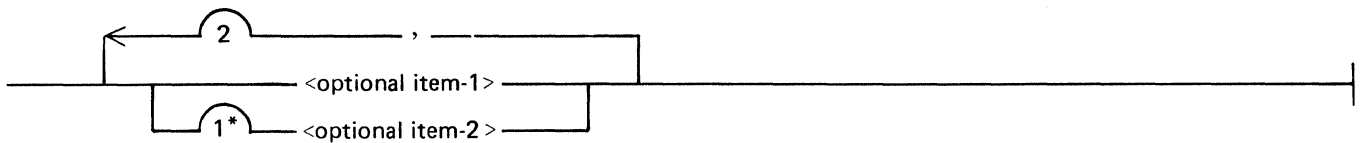


n is an integer which specifies the maximum number of times the path may be traversed.



n is an integer which specifies the minimum number of times the path must be traversed.

Example:



The loop may be traversed a maximum of two lines; however, the path for <optional item-2> must be traversed at least one time.

The following statements can be constructed from the railroad diagram in the example.

<optional item-1>,<optional item-2>

<optional item-2>,<optional item-2>,<optional item-1>

<optional item-2>

SECTION 3

GENERAL DESCRIPTION

INTRODUCTION

The basic function of the INQUIRY system is to select a record from the data base and to display items in that record.

The user must have a basic familiarity with the structure of the data base. For example, the user must know the names of the items to be displayed. If embedded structures are employed, the user must know the "nesting hierarchy" of the structures which are to be referenced. In addition, the user may need to know structure names for item qualification where INQUIRY is unable to determine from context which structures are required.

Data Base Administrators, through the use of DASDL remaps and logical data bases, can protect against unauthorized access to sensitive data by defining which parts of the data base have INQUIRY capability and which users have access.

DESCRIPTION OF INQUIRY STATEMENTS

A brief description of each INQUIRY statement follows:

ATTACH

Allows the user to combine an embedded structure with its owner to establish automatic looping between the two structures.

CLEAR

Discards DEFINE items, VIRTUAL items, and/or generated subsets.

DEFINE

Allows INQUIRY text to be assigned a name. When INQUIRY sees the define name, it replaces the define name with the associated text.

DETACH

Separates an embedded structure from its owner to prevent automatic looping between the two structures.

DISPLAY

Allows items of a selected record to be displayed.

EDIT

Allows a previous INQUIRY statement to be modified without requiring the entire statement to be re-entered.

GENERATE

Creates a temporary subset of a data set. The temporary subset can be referenced by other INQUIRY statements.

HELP

Displays the syntax and semantics for each INQUIRY statement.

NEXT

Causes INQUIRY to resume record selection and item display.

OPTIONS

Allows INQUIRY options to be displayed or altered.

PRINTER

Allows the attributes of the line printer file to be displayed or altered.

QUIT

Terminates the INQUIRY session.

RECALL

Retrieves the text of a prior INQUIRY statement.

REPEAT

Causes re-execution of a previous INQUIRY statement.

RESTORE

Allows previously saved text to be retrieved.

SAVE

Stores the text of DEFINE items, VIRTUAL items, and generated subsets in a file on disk. The saved text can be reloaded and used during subsequent INQUIRY sessions.

SELECT

Locates records which satisfy the selection criteria provided by the user.

SET

Modifies or deletes the text of the most recently entered DISPLAY, REPEAT, or SELECT statement for a given data set.

SHOW

Displays all or selected portions of the data base description and can also be used to display the most recently entered INQUIRY statement.

SORT

Allows a user to control the amount of memory used by the SORT option.

TERMINAL

Allows the attributes of the terminal file to be displayed or altered.

VIRTUAL

Allows new items to be defined which are functions of other items.

SECTION 4 USE OF INQUIRY

TERMINAL USE AND DATA BASE ACCESS

The DMS/INQUIRY program is executed in the normal manner. The DMS/INQUIRY program attempts to open a remote terminal labeled "REMOTE"; this can be changed by a B 1800/B 1700 MCP file equation. After the DMS/INQUIRY program has started executing, it displays on the terminal the message:

WHAT DATA BASE?

Valid responses are:

```
_____ <database-name> _____  
      |  
      | ON <pack-id> |  
      |
```

The <pack-id> is the pack-id of the DMS/INQUIRY program INQCTL file. Refer to Section 6 for details on building the INQCTL file.

The DMS/INQUIRY program then displays the message:

INITIALIZING...PLEASE WAIT...

The user is notified when initialization is complete by the messages:

```
...READY  
#
```

The user can then initiate processing.

The DMS/INQUIRY program uses dynamic memory for internal working storage. For some applications, the default dynamic memory size is too small. If necessary, the dynamic memory size can be increased by executing the DMS/INQUIRY program, as follows:

```
EX DMS/INQUIRY MEMORY 100000
```

ENTERING INQUIRY STATEMENTS

Normally, INQUIRY statements can be entered on a single line and are executed immediately.

However, an INQUIRY statement can exceed the line width of the terminal. A percent sign (%) character, entered at the end of the line, indicates that more input follows. When the line is transmitted, the DMS/INQUIRY program responds with "#%", and additional input may then be entered. This process can be repeated as many times as necessary. The maximum number of characters for the total input cannot exceed 1919 characters.

When the statement is complete, the user can transmit the last line, and the entire INQUIRY statement is executed. A "%" character should not be entered for the last line of the command.

The user can enter an input statement which is not processed immediately. Instead, the statement is stored by the DMS/INQUIRY program and can be executed later. The user enters the statement as previously described; however, the last line of input must be terminated by the characters "%%".

The SHOW statement can then be used to display the statement. The EDIT statement allows modification of the statement. The REPEAT statement causes execution of the input statement.

Some input commands may produce a large amount of output. This output, shown one page at a time, can be discontinued by entering a nonnull response after a page has been displayed. The DMS/INQUIRY program then responds with a number sign (#) and waits for more input.

BASIC INQUIRY PROCESS

The DMS/INQUIRY program allows the user to select data base records, and then display items within those records.

Example:

Assume that a company has a data base which contains the following information:

```
PERSONNEL DATA SET (  
  NAME           ALPHA (20);  
  EMPNUM         NUMBER (5);  
  DEPT           NUMBER (4);  
  ROOM           NUMBER (3);  
  EXTENSION      NUMBER (4);  
  JOBCLASS       ALPHA (15);  
  SALARY         NUMBER (7,2);  
  HOURS-WORKED   NUMBER (3);  
);
```

To examine the personnel records for employees in department 1800, enter:

```
SELECT DEPT = 1800
```

The DMS/INQUIRY program attempts to locate a record which satisfies the request. If the DMS/INQUIRY program does not locate at least one such record, it displays:

```
#NONE ON SELECTION OF PERSONNEL
```

If the DMS/INQUIRY program locates a record, only the # character is displayed. The user can then display items in the selected record. For example, the employee's name and extension can be displayed by entering:

```
DISPLAY NAME,EXTENSION
```

The DMS/INQUIRY program responds with:

```
NAME = JONES JOHN D
```

```
EXTENSION = 2364
```

To locate other employees in the department, enter:

```
NEXT
```

If another record exists, the DMS/INQUIRY program responds by displaying a # character. If no more records exist, the response is:

```
#NO MORE ON SELECTION OF PERSONNEL
```

RECORD SELECTION (SELECT)

The DMS/INQUIRY program locates all records which satisfy the selection condition specified. A simple selection condition is formed as follows:

<item name> <relational operator> <value>

Examples:

SALARY > 500

JOBCLASS = "PROGRAMMER"

HOURS-WORKED < 40

The following <relational operator>s may be used:

Operator	Alternate Spelling	Meaning
=	EQL	EQUAL TO
<	LSS	LESS THAN
>	GTR	GREATER THAN
≠	NEQ	NOT EQUAL TO
≤, ≧	LEQ	LESS THAN OR EQUAL TO
≥, ≦	GEQ	GREATER THAN OR EQUAL TO

The DMS/INQUIRY program also accepts arithmetic expressions in selection conditions.

Examples:

D1 > D2+D3

D4 < ((D2+D3)*D4)/D5

D7 > D8

The following operators are allowed in arithmetic expressions:

Operator	Meaning
+	Add
-	Subtract
*	Multiply
/	Divide
DIV	Integer Divide
MOD	Remainder Divide

The logical operators AND, OR, and NOT can be used to form more complex selection expressions.

Examples:

SELECT DEPT = 1800 AND JOBCLASS = "MANAGER"

SELECT DEPT = 1800 OR DEPT = 1700

SELECT DEPT = 1800 AND JOBCLASS = "PROGRAMMER" OR SALARY > 500

The following order of precedence controls the sequence in which selection conditions are evaluated.

First : <arithmetic expression>s

Second: <relational operator>s

Third : NOT

Fourth: AND

Fifth : OR

When operators have the same precedence, the operations are performed in order from left to right. Parentheses can be used to override the usual order of evaluation.

Example:

```
SELECT DEPT=1800 AND (JOBCLASS = "PROGRAMMER" OR SALARY > 500)
```

This SELECT statement locates only those employees in DEPT=1800 who are programmers or whose salary exceeds 500.

The logical operator NOT can be used, together with parentheses, to specify an "all except" condition.

Example:

```
SELECT NOT(JOBCLASS = "MANAGER")
```

This statement locates all employees who are not managers.

All items appearing in selection conditions must be contained in currently selected records or in the record being selected.

Example:

```
D DATA SET (  
  D1 NUMBER (4);  
  D2 NUMBER (4);  
  D3 NUMBER (3);  
E UNORDERED DATA SET (  
  E1 NUMBER (3);  
  E2 NUMBER (2);  
  );  
);
```

The statements in the following examples are valid.

Examples:

```
SELECT D1 > D2
```

```
SELECT D3 > (D1+D2)
```

```
SELECT VIA D, SELECT E2 > E1+D1
```

The statement in the following example is invalid because no E record has been selected.

Example:

```
SELECT D1 > D1+E2
```

Structure Designation

For each data set, there may be a number of associated sets. The DMS/INQUIRY program always attempts to locate records by making use of these sets. Subsets may not locate all records contained in the data set; therefore, subsets are not used unless specified by structure designation.

The example data set is used in the examples that follow.

```
PERSONNEL DATA SET (  
  NAME           ALPHA (20);  
  EMPNUM         NUMBER (5);  
  DEPT           NUMBER (4);  
  ROOM           NUMBER (3);  
  EXTENSION      NUMBER (4);  
  JOBCLASS       ALPHA (15);  
  SALARY         NUMBER (7,2);  
  HOURS-WORKED   NUMBER (3);  
);  
EMPNUMSET SET OF PERSONNEL KEY EMPNUM;  
DEPT1800 SUBSET OF PERSONNEL WHERE DEPT=1800 KEY NAME;
```

The following SELECT statement causes the DMS/INQUIRY program to use the set EMPNUMSET.

```
SELECT EMPNUM = 1234
```

The following statement does not use a set or subset.

```
SELECT NAME = "DOE, JOHN"
```

The DMS/INQUIRY program searches the data set directly because subset DEPT1800 may not refer to all records in the data set.

The user can force the DMS/INQUIRY program to use a particular structure by designating the structure name following the word SELECT.

Example:

```
SELECT PERSONNEL AT EMPNUM > 500
```

The DMS/INQUIRY program searches the PERSONNEL data set and selects records in physical order.

The following statement locates all records sequentially using the set EMPNUMSET.

```
SELECT EMPNUMSET
```

The following statement causes the DMS/INQUIRY program to locate records through the subset DEPT1800. The DMS/INQUIRY program only locates JOHN DOEs assigned to department 1800. JOHN DOE entries in other departments are not located.

```
SELECT DEPT1800 AT NAME = "DOE, JOHN"
```

ITEM DISPLAY (DISPLAY)

Once a record has been selected, items within the record can be displayed.

Example:

```
SELECT ROOM = 421
#
DISPLAY NAME
    NAME = SMITH JOHN
DISPLAY EXTENSION
    EXTENSION = 2364
DISPLAY DEPT
    DEPT = 6145
```

Several items can be displayed using a single DISPLAY statement. A comma must separate the item names.

Example:

```
SELECT ROOM = 421
#
DISPLAY NAME,EXTENSION,DEPT
    NAME = SMITH JOHN
    EXTENSION = 2364
    DEPT = 6145
```

To display all of the items in the record, enter the following statement.

```
DISPLAY ALL
```

QUALIFICATION

In the preceding examples, it was not necessary to name the data set being used. The DMS/INQUIRY program attempts to determine the data set by examining the items in the DISPLAY or SELECT statement. This may not be possible in all cases.

Example:

```
INVENTORY DATA SET (  
  UNIT-PRICE      NUMBER (S7,2);  
  PART-NO         NUMBER (6);  
);  
ORDERS DATA SET (  
  UNIT-PRICE      NUMBER (S7,2);  
  PART-NO         NUMBER (6);  
);
```

Both data sets have items with the same names. The statement

```
SELECT PART-NO = 1234
```

causes the DMS/INQUIRY program to request qualification as follows:

```
#WHICH DATA SET?  
1. INVENTORY  
2. ORDERS
```

The user must qualify the request by entering the number associated with the desired data set. Subsequent statements involving items in that data set need not be qualified. The DMS/INQUIRY program uses the specified data set until a different data set is selected or a new qualification clause is specified.

Qualification can be included in the original input request.

Example:

```
SELECT PART-NO OF INVENTORY = 1234
```

The qualification clause "OF INVENTORY" designates that the INVENTORY data set is the data set to be used.

COMBINING RECORD SELECTION AND ITEM DISPLAY

Record selection and item display can be combined by appending a selection condition to the DISPLAY statement.

Example:

```
DISPLAY NAME IN ROOM = 421
```

The word IN separates the item to be displayed from the selection condition. Several items can be displayed with a single statement.

Example:

```
DISPLAY NAME,EXTENSION,DEPT IN ROOM = 421
```

AT, WHERE, and the special character “@” are synonyms for the word IN.

Examples:

```
DISPLAY SALARY AT MAN-NUMBER = 2890
```

```
DISPLAY ROOM WHERE EXTENSION = 2364
```

```
DISPLAY EXTENSION @ ROOM = 421
```

If qualification is required, the qualification clauses must appear immediately before the AT, IN, WHERE, or “@”.

Example:

```
DISPLAY UNIT-PRICE OF INVENTORY WHERE PART-NO = 1248
```

If structure designation is used, the word USING or VIA must appear before the structure name.

Example:

```
DISPLAY UNIT-PRICE USING ORDERS WHERE PART-NO = 1248
```

OUTPUT CONTROL (SELECT/DISPLAY)

Unlike the basic SELECT statement, which locates one record at a time and waits for additional input, the DISPLAY statement can be used to automatically display the information for all records which satisfy the selection condition.

By including a limit in the DISPLAY statement, the user can control the amount of output displayed at one time.

Example:

```
DISPLAY 3 NAME IN DEPT = 1800
```

The integer 3 represents the limit. The DMS/INQUIRY program attempts to display up to three records, and then waits for input from the user.

If the user enters NEXT or null input, the DMS/INQUIRY program responds by displaying three more records. This procedure can be repeated until all records which satisfy the selection condition have been displayed.

If a new INQUIRY statement is entered, further output is discarded and the system processes the new statement.

INQUIRY OF EMBEDDED STRUCTURES

Data sets can appear as items within other data sets.

Example:

```
D DATA SET (  
  D1 NUMBER (3);  
  D2 ALPHA (10);  
  D3 NUMBER (3);  
  E UNORDERED DATA SET (  
    E1 NUMBER (5);  
    E2 ALPHA (8);  
    E3 NUMBER (3);  
    F UNORDERED DATA SET (  
      F1 NUMBER (10);  
      F2 ALPHA (15);  
    );  
  );  
S-D SET OF D KEY D1;
```

The following examples illustrate how to access this data base.

Example:

```
SELECT D1 = 3, THEN DISPLAY E1,E2 WHERE E3 > 5
```

The DMS/INQUIRY program selects all D records where D1=3. For each D record selected, E1 and E2 are displayed for each E record where E3>5.

If structure designation is required, it can be specified as follows:

```
SELECT D WHERE D1 = 3, THEN DISPLAY E1,E2 USING E WHERE E3 > 5
```

Items of F can be displayed:

```
SELECT D1 = 3, THEN SELECT E3 > 5, THEN DISPLAY F1 AT F2 = 7
```

Items from each data set can be displayed:

```
DISPLAY D2 WHERE D1 = 3, THEN DISPLAY E1 WHERE E3 > 5,  
THEN DISPLAY F1 AT F2 = 7
```

To display an item in an embedded data set, a record from the parent data set must be selected.

Records must be selected in ascending hierarchical sequence. To display items in the embedded data set E, a record of D must first be selected.

Example:

```
SELECT D = 3, THEN DISPLAY E1 AT E2 > 0
```

To display items from the embedded data set F, a D record and an E record must be selected in that order.

Example:

```
SELECT D1 = 3, SELECT E1 = 5 THEN DISPLAY F1 at F1 > D1
```

A comma, the word THEN, or a comma followed by the word THEN must be used to separate the clauses in a complex INQUIRY statement.

Since structures must be selected in an ascending hierarchical sequence, a DISPLAY statement can reference any structure previously selected.

Example:

```
SELECT D1 = 3, SELECT E3 > 5, DISPLAY D2,E1,F1 WHERE F2 > 0
```

This example differs from the previous example which displayed items from each data set. In this example, a display occurs only if a record satisfying the selection condition for each structure is located.

If the selected items require qualification, the qualification can be included in the display list.

Example:

```
DISPLAY D2 OF D, E1 OF E, F1 OF F
```

Output Control (Embedded Structures)

The DMS/INQUIRY program displays results until all records selected are displayed (or the screen of a CRT device is full).

The amount of uninterrupted output can be controlled on a structure basis.

Example:

```
SELECT D1 = 3 THEN DISPLAY 3 E1 WHERE E2 = 7
```

The limit 3, following the word DISPLAY, causes the DMS/INQUIRY program to display items for three records of E and wait for further input. Output can be continued by entering NEXT.

Output limit numbers can also be used to selectively bypass some of the output. Referring to the previous example, if the user enters NEXT D following the display of the three items of E, the following actions are performed:

1. Cancel output for E records of the current D record.
2. Select the next D record.
3. Output for the first three E records is selected.
4. Stop and wait for additional input.

Output control can be imposed at each level.

Example:

```
DISPLAY 3 D1 AT D2 = 5 THEN DISPLAY 2 E1 AT E2 = 7
```

The DMS/INQUIRY program stops after displaying two E1 items. The DMS/INQUIRY program also stops after displaying three D1 items.

NOTE

In certain instances, stopping on a limit for one structure coincides with stopping on a limit for another structure. Because of this, it may be necessary to enter NEXT twice, to resume output.

FUNCTIONS

INQUIRY functions consist of Boolean functions and arithmetic functions.

Boolean Function

A Boolean function provides the user with a method of selecting or displaying a record or an item based upon a specified condition of an embedded structure.

Example:

```
EMPLOYEE DATA SET (  
  NAME           ALPHA (20);  
  EMPNUM         NUMBER (5);  
  DEPT           NUMBER (4);  
  JOB-HISTORY UNORDERED DATA SET (  
    JOBCLASS ALPHA (15);  
    TIMEHELD NUMBER (4);  
  );  
);
```

Assume that it is desired to know the names of all employees in DEPT=1800 who have had a JOBCLASS="ENGINEER". The following statement returns this information:

```
DISPLAY NAME WHERE DEPT = 1800 AND ANY(JOBCLASS = "ENGINEER")
```

The following condition is a Boolean function, since it yields a truth value, TRUE or FALSE.

```
ANY(JOBCLASS = "ENGINEER")
```

The DMS/INQUIRY program selects only DEPT=1800 records. For each selected record, the DMS/INQUIRY program searches the employee's job history records, looking for a job class of engineer. If any engineers are found, the employee's name is displayed.

The condition within the parentheses can be complex.

Examples:

```
ANY(JOBCLASS = "ENGINEER" AND TIMEHELD = 18)
```

```
ANY(JOBCLASS = "ENGINEER" OR JOBCLASS = "MANAGER")
```

The first example displays the names of employees who have been engineers for a period of 18 months exactly. The second example displays the names of employees who are, or have been, engineers or managers.

The Boolean function ALL can be used as follows:

Example:

```
DISPLAY NAME WHERE DEPT = 1800 AND ALL(JOBCLASS = "ENGINEER")
```

This statement selects only DEPT=1800 records. For each selected record, the DMS/INQUIRY program searches the embedded job history structure and displays the employee's name if all job history records for that employee contain a job class of engineer.

Arithmetic Function

The second type of INQUIRY function is the arithmetic function. This function provides the user with the ability to select or display a record or a record item based upon the results of a specified arithmetic procedure over an embedded structure. The following examples illustrate the applications of arithmetic functions.

Example:

```
BANK DATA SET (  
  BANK-NAME           ALPHA (30);  
  BANK-ADDRESS        ALPHA (30);  
  STOCK UNORDERED     DATA SET (  
    TENDOR             ALPHA (10);  
    STOCK-VALUE        NUMBER (10);  
  );  
);
```

Each bank record has an embedded STOCK data set which contains information about stock the bank owns. Assume that a list of those banks which own more than one million dollars in stock is required. The following statement returns this list.

```
DISPLAY BANK-NAME WHERE SUM(STOCK-VALUE) > 1000000
```

The following condition is an arithmetic function.

```
SUM(STOCK-VALUE)
```

A complete list and description of the allowable arithmetic functions are contained in Section 5, FORMAL INQUIRY DESCRIPTION.

In the example, the STOCK-VALUE items for all stock of a given parent were summed, then compared with the specified value.

A condition can be specified which allows the selection of only some of the embedded records to be summed.

Example:

```
DISPLAY BANK-NAME  
WHERE SUM(STOCK-VALUE WHERE TENDOR = "ABC") > 1000000
```

This statement displays the names of those banks that own more than one million dollars worth of stock in company ABC.

A special type of arithmetic function, COUNT, is provided which counts the number of occurrences of a specified condition within an embedded data set.

Example:

```
DISPLAY BANK-NAME WHERE COUNT(TENDOR = "ABC") > 9
```

This statement causes the DMS/INQUIRY program to display the names of all banks which hold more than nine stock certificates in company ABC.

Displaying Arithmetic Function Results

It is often desirable to display the results of arithmetic functions. The user is provided with this ability by including an arithmetic function clause as an item in a display list.

Example:

```
DISPLAY COUNT(DEPT = 1800)
```

When using arithmetic functions within a display list or selection condition, the items associated with the arithmetic function must be items of an embedded data set.

Example:

```
D DATA SET (  
  D1 NUMBER (3);  
  D2 NUMBER (4);  
  E UNORDERED DATA SET (  
    E1 NUMBER (4);  
    E2 NUMBER (3);  
  );  
);
```

Example:

```
DISPLAY D1, SUM(E1) AT D2 > 0 AND AVG(E2) > 50
```

The system response to this statement is to display one D1 item and the sum of E1 items. This output is repeated until those records matching the selection condition are exhausted.

The statement in the following example is invalid since D2 is not an item of a data set embedded within the E data set.

Example:

```
DISPLAY E1, SUM(D2)
```

NOTE

Disjoint data sets are considered to be embedded data sets for the purpose of displaying arithmetic functions. (Disjoint data sets are embedded within the data base.)

The following example is a valid statement, assuming that SALARY is an item of a disjoint data set.

Example:

```
DISPLAY AVG(SALARY), MIN(SALARY)
```

To compute the values for the various arithmetic functions it is necessary for the system to access every record of the data set involved; thus, it may take a large amount of time to produce a result.

VIRTUAL ITEMS

It is sometimes desirable to compute and/or display a value which is not physically present in the data base. The concept of virtual items provides this ability.

Example:

It may be desirable to display an employee's salary but only HOURS-WORKED and PAY-RATE are held in the data set. The DMS/INQUIRY program accepts the statement:

```
VIRTUAL SALARY = HOURS-WORKED * PAY RATE
```

which declares SALARY as a virtual item.

The DMS/INQUIRY program remembers SALARY as being equal to HOURS-WORKED * PAY-RATE, and treats SALARY as if it were an actual item in the data base. The user can then use this virtual item in a display list or selection condition.

Examples:

```
DISPLAY NAME,SALARY IN DEPT = 1800
```

```
DISPLAY NAME,DEPT AT SALARY > 400.00
```

Any form of arithmetic expression is allowed to the right of the equal (=) sign in the VIRTUAL statement. The rule is that the items must be capable of yielding a result when the virtual item is referenced.

Example:

```
D DATA SET (  
  D1 NUMBER (3);  
  D2 REAL;  
  E DATA SET (  
    E1 NUMBER (3);  
    E2 NUMBER (3);  
  );  
);
```

The following statements are valid examples of virtual declarations.

```
VIRTUAL DX = D1 + D2
```

```
VIRTUAL DE = E1 + D1
```

Since D exists when E exists, the virtual item DE can be computed using items of D and E. DX is considered a virtual item of D and, DE is a virtual item of E.

The following statements illustrate a valid DISPLAY statement and an invalid DISPLAY statement referencing a virtual item.

Example: (Valid)

```
DISPLAY DX AT D2 = 50
```

Example: (Invalid)

```
DISPLAY DE AT D2 = 50
```

In reference to the invalid example, since E does not exist when D exists, the E item (E1) cannot be part of the virtual DE and a "record-not-selected" error occurs.

The arithmetic expressions associated with a virtual item can also contain arithmetic functions.

Examples:

VIRTUAL DX = D1 + SUM(E1 at E2 > 20)

VIRTUAL DY = AVG(E2)

It is advantageous in terms of efficiency to associate arithmetic functions with virtual identifiers. If a function appears in a display list or selection condition, the function must be evaluated each time the function is referenced. However, if the function is associated with a virtual identifier, the DMS/INQUIRY program "remembers" whether a function has been evaluated and, if so, the value of the function. The DMS/INQUIRY program also recognizes when the value of a function becomes meaningless and re-evaluates it when necessary.

UNDEFINED ITEMS

There are several cases where references can be made which result in undefined situations. These cases occur either in a display list, a selection condition, or an arithmetic expression. The undefined cases are as follows:

1. The value of an item is NULL (as defined in DASDL).
2. The value of a function is undefined.

Example:

AVG(E2 WHERE E1 > 100)

where no records exist in which E1 > 100.

3. Variable format records are used and a record was loaded in which the specified item does not exist.
4. The process of evaluating an arithmetic expression would have resulted in division by zero, or an attempt was made to generate a number exceeding the hardware capacity (integer overflow).

When an undefined situation is encountered in displaying an item, two hyphens are printed in lieu of a value.

The occurrence of an undefined situation becomes more complex when encountered with a selection condition.

Example:

SELECT NOT(A = B) OR C > 50

If either A or B is undefined, the selection expression

NOT(A = B)

is false, regardless of the presence of the NOT and regardless of the relational operator used. The record would be selected dependent only on the truth value of the relational expression C > 50. However, if C is also undefined, for the current record, then the record is not selected.

UNQUOTED ALPHA-LITERALS

When entering <selection condition>s in various INQUIRY statements, relationships in the form

<item> <rel op> <alpha-literal>

are encountered, where <item> is an identifier and <rel op> is a relational operator. An <alpha-literal> is defined as a quotation mark (“), followed by one or more characters, followed by an ending quotation mark.

Example:

“THIS IS AN ALPHA-LITERAL”

One of the most frequent terminal user errors is failure to bracket alpha-literals with beginning and ending quotation marks.

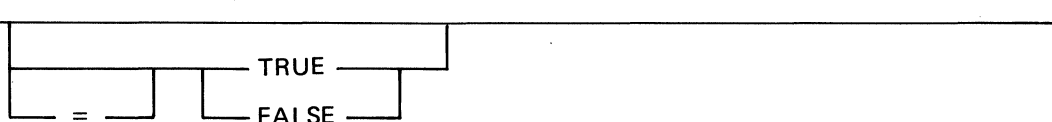
The DMS/INQUIRY program can recognize alphanumeric strings without embedded blanks as alphanumeric literals even though quotation marks are not used.

NOTE

An alphanumeric string is defined as a series of characters containing only lowercase a through z, uppercase A through Z, and the numeric characters 0 through 9. Note that special characters are not allowed.

Since the use of alpha-literals not enclosed within quotation marks can be misinterpreted, the following option can be utilized:

OPTION QUOTES



If QUOTES or QUOTES=TRUE is specified, all alpha-literals must be enclosed within quotation marks. This is the default.

If QUOTES=FALSE is specified, then

1. An alphanumeric string is recognized as an alpha-literal when used in the proper context.
2. Alpha-literals containing special characters (such as “\$”, “%”, or “&”) still must be enclosed within quotation marks.
3. Even if QUOTES=FALSE, alpha-literals within quotation marks are allowed.

As stated previously, the DMS/INQUIRY program may misinterpret the user’s intention when alpha-literals are used without quotation marks. For the following discussion and examples, A represents an alpha identifier in the data set D.

Examples:

SELECT A = 1234

SELECT A = 12AB56

An alpha-literal which starts with numeric characters and is not enclosed in quotation marks will not be misinterpreted.

Example:

```
SELECT A = X13
```

X13 is the name of a data base item. In this case, the DMS/INQUIRY program cannot determine if the user wanted the value of the identifier X13 or the alpha-literal "X13". The DMS/INQUIRY program assumes that the request was for the identifier.

Example:

```
DEFINE X13 = RST  
  
then  
  
SELECT A = X13
```

The DMS/INQUIRY program interprets this as if the user had entered:

```
SELECT A = RST
```

If the alphanumeric string is the name of a DEFINE item, the DEFINE item is expanded and the text of the DEFINE used.

Two safeguards can be used to determine if the DMS/INQUIRY program can recognize the users intent:

1. If a SHOW X13 is entered and this results in an unknown identifier, then X13 can be accepted as an alpha-literal.
2. If a <selection expression> has yielded unexpected results, entering a

```
RECALL D
```

where D is the data set name, indicates what was scanned by the system, and any unquoted alpha-literal is displayed as a quoted alpha-literal.

OUTPUT FORMATTING

The format of the data displayed on the terminal or printer is controlled by the FORMAT attributes of HEADING, TAB, SINGLE, or DEFAULT. The following discussion pertains to output formatting on a terminal. The same principle holds for output to the line printer. Refer to the TERMINAL and PRINTER statements in Section 5.

HEADING

The HEADING attribute causes the output to be displayed in the following form:

N1	N2	N3
V11	V12	V13
V21	V22	V23

where N1, N2, and N3 are the item names; V11, V12, and V13 are the values associated with those items in record 1; V21, V22, and V23 are the values of those items in record 2.

The FORMAT attribute HEADING can be set by entering:

TERMINAL FORMAT HEADING

In some cases the size of the display list is such that all the item names or item values or both do not fit on one line. In this case, the output is in the form:

1: N1	N2	N3
2: N4	N5	N6
<hr/>		
1: V11	V12	V13
2: V14	V15	V16
1: V21	V22	V23
2: V24	V25	V26

where N1 through N6 are the item names, and V11 through V26 are the values associated with those items.

Association between item name and item value is made by line and position within the line. For example, item N1 has a value of V11 in the first record and a value of V21 in the second record. Item N5 has a value of V15 in the first record and a value of V25 in the second record.

TAB

By setting the TAB attribute, the user can force output to be displayed in a format similar to that shown below.

N1 = V11	N2 = V12	N3 = V13
N4 = V14	N5 = V15	

N1 through N5 represent the item names, and V11 through V15 represent the value associated with the items.

The TAB attribute can be set by entering:

TERMINAL FORMAT TAB

SINGLE

The SINGLE attribute causes INQUIRY to display the output in the following format:

N1 = V11
N2 = V12
N3 = V13
<hr/>
N1 = V21
N2 = V22
N3 = V23

As in the previous example, N1 through N3 represent the item names, and V11 through V23 represent the associated item value.

SINGLE causes the system to display one item name and associated value per line.

SINGLE can be set by entering:

TERMINAL FORMAT SINGLE

If more than one record is being displayed, a blank line separates the records.

DEFAULT

The DEFAULT attribute allows the DMS/INQUIRY program to determine the type of output. This is the default attribute.

DEFAULT can be set by entering:

TERMINAL FORMAT DEFAULT

SECTION 5

FORMAL INQUIRY DESCRIPTION

The syntax and semantics of each INQUIRY statement are described in this section.

INQUIRY statements, with the exception of the SELECT and DISPLAY statements, are arranged in alphabetical order. The SELECT and DISPLAY statements, because of their close relationship, are discussed together and appear first.

The following INQUIRY statements are provided:

ATTACH	QUIT
CLEAR	RECALL
DEFINE	REPEAT
DETACH	RESTORE
DISPLAY	SAVE
EDIT	SELECT
GENERATE	SET
HELP	SHOW
NEXT	SORT
OPTIONS	TERMINAL
PRINTER	VIRTUAL

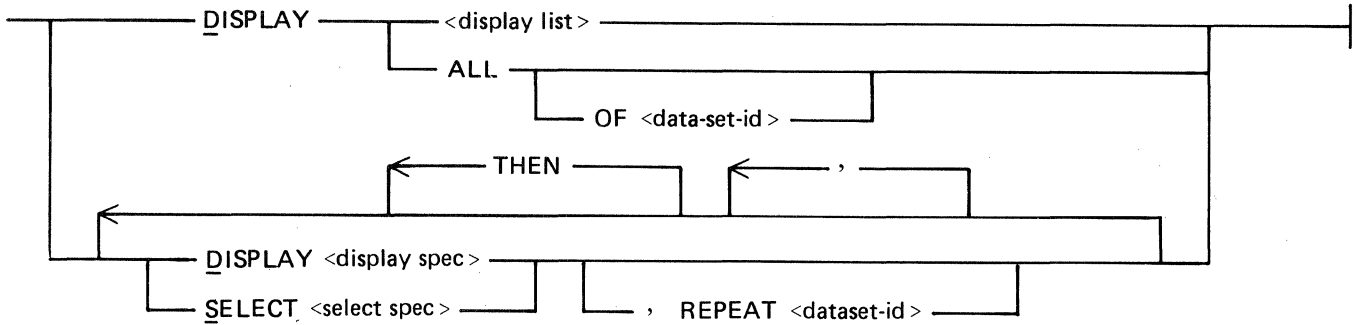
The following data base is used in examples throughout this section.

EXAMPLE DATA BASE

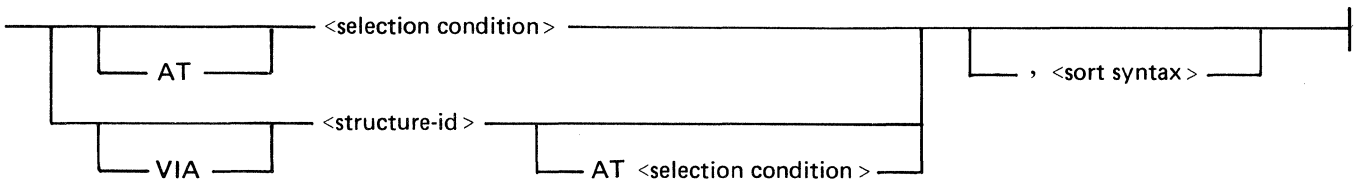
```
A DATA SET(
  A1          NUMBER(4);
  A2          ALPHA(3);
  B DATA SET(
    B1        ALPHA(3);
    B2        NUMBER(4);
  );
  S-B ACCESS TO B KEY (B2);
);
S-A SET OF A KEY A1;
```

SELECT/DISPLAY

Syntax:

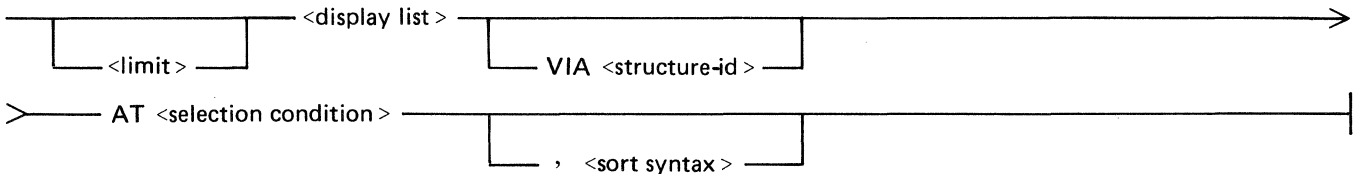


The syntax of the <select spec> is as follows:

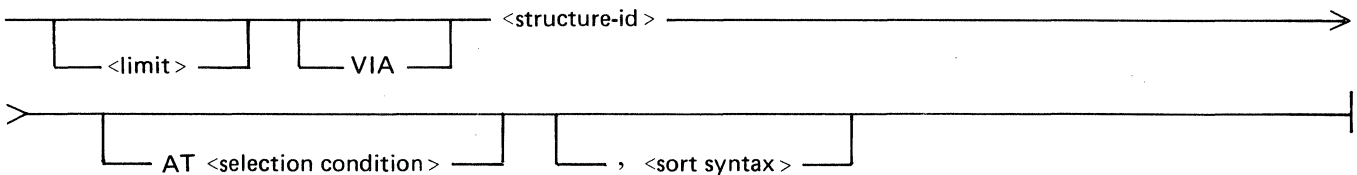


The syntax of the <display spec> is as follows:

Option 1:



Option 2:



WHERE, IN, FOR, and @ are equivalent to AT, and USING is equivalent to VIA.

Semantics:

1. SELECT locates a record which satisfies the <select spec>.
2. SELECT <structure-id> locates a data set record in or via <structure-id>. VIA is optional.
3. SELECT <structure-id> AT <selection condition> locates a record in <structure-id> that satisfies the <selection condition>.
4. The <structure-id> must be a data set, set, subset, or a temporary subset created by the GENERATE statement.
5. The <selection condition> specifies relationships which must be satisfied by items in a record if that record is to be selected.

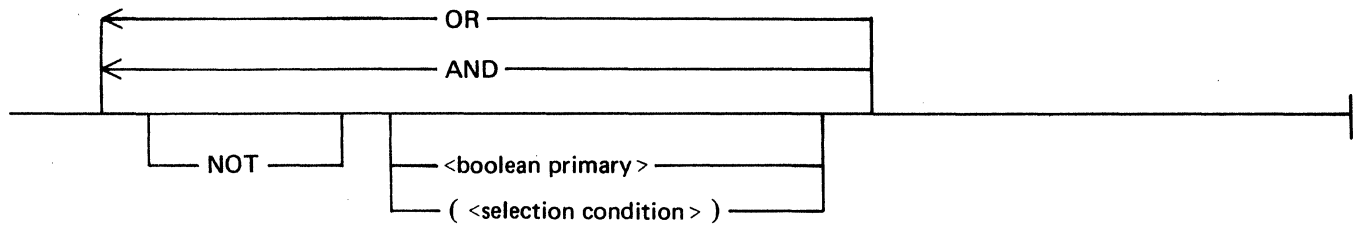
6. The DISPLAY statement displays data item names and their associated values.
7. DISPLAY ALL displays all items of the most recently selected record.
8. DISPLAY ALL OF <data-set-id> displays all items of the most recently selected record of <data-set-id>.
9. DISPLAY <display list> displays the items specified in the <display list> from previously selected records.
10. DISPLAY <display list> AT <selection condition> displays the items specified in the <display list> for all records which satisfy the <selection condition>. If no records satisfy the <selection condition>, no items are displayed.
11. DISPLAY <display list> VIA <structure-id> displays the items specified in the <display list> for all records located through <structure-id>. Records are displayed in the order of physical appearance in <structure-id>.
12. DISPLAY <structure-id> displays all data items for all records located through <structure name>.
13. DISPLAY <structure-id> AT <selection condition> displays all data items for all records located through <structure-id> which satisfy the <selection condition>. If no records satisfy the <selection condition>, no items are displayed.
14. Items specified in the <display list> must be present in currently selected data set records.
15. <Limit> is an integer that indicates how many records to display. If <limit> is larger than the number of records which can be displayed on a CRT, INQUIRY displays as much information as possible and waits for further input.
16. The SELECT and the DISPLAY statement can be combined in a single input command. The DMS/INQUIRY program processes the combined SELECT/DISPLAY statement as a series of nested loops. Structures must be selected in hierarchical sequence throughout the family of embedded structures.
17. The DMS/INQUIRY program processes each command within the statement for the first record which satisfies the selection criteria. When all commands have been processed, one record from each structure has been selected through the innermost nesting level which satisfied a SELECT or DISPLAY.
18. Once a record is selected at the innermost valid nesting level, if a SELECT is specified at that level, the DMS/INQUIRY program responds with a “#” and waits for further input. However, if a DISPLAY is specified, the DMS/INQUIRY program processes all records in that structure which satisfy the <display spec>.
19. When all records at the innermost nesting level are exhausted, the DMS/INQUIRY program attempts to retrieve a new record at the next outer nesting level. If a DISPLAY is specified, the DMS/INQUIRY program automatically retrieves the next record in that structure which satisfies the <display spec> and continues to process the original input statement from that point. If all records at this nesting level are exhausted, the DMS/INQUIRY program proceeds to the next outer nesting level. This process continues until all records are exhausted.
20. If the DMS/INQUIRY program encounters a SELECT command, processing is halted prior to the selection of a new record, and user input is required. To select a new record, the user must enter NEXT. If a new record is located, the DMS/INQUIRY program selects the record and continues processing the original input statement from that point. If no new records are available at that nesting level, the DMS/INQUIRY program responds with a “#” and waits for user input. If NEXT is entered and the entire input statement has been processed, the DMS/INQUIRY program responds with “#NO MORE ON SELECTION OF <dataset-id>”.
21. The <sort syntax> allows records to be selected and displayed in a user specified sequence. The <sort syntax> is described under the heading SORT Syntax in this section.
22. The <repeat option> allows selecting outside the normal family of embedded structures. Specifying REPEAT <data-set-id> is the equivalent of specifying REPEAT <data-set-id>, without interrupting the nesting of the current SELECT or DISPLAY string (refer to REPEAT in this section).

Pragmatics:

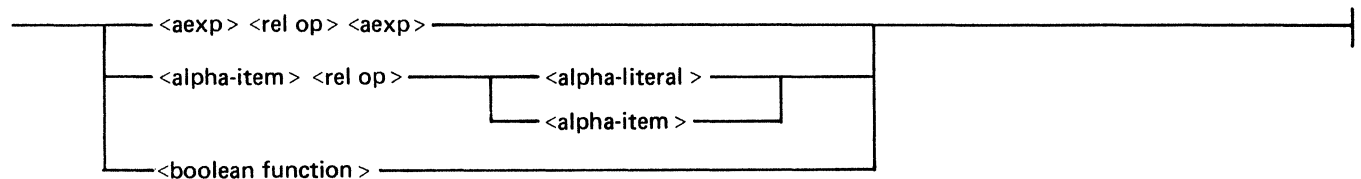
1. The DMS/INQUIRY program determines which data set to use by examining the items contained in the <select spec>. The DMS/INQUIRY program takes advantage of sets, if possible, when selecting records.
2. If an appropriate set does not exist, the DMS/INQUIRY program searches the data set directly.
3. Subsets and temporary subsets may not locate all records of the data set, and are not automatically used by the DMS/INQUIRY program. The <structure-id> must be used to locate records by means of these structures.
4. Data sets must be selected in hierarchical sequence. To select records in an embedded data set, a record from the structure at the outer nesting level must first be selected. All structures at an outer nesting level in the embedded family must have a current record selected.
5. Each data set has only one current record pointer into the data set, regardless of the number of sets or subsets associated with it. Selecting via another set causes the data set current record pointer to be changed to reflect the new set accessing.
6. The DMS/INQUIRY program searches the data base until it finds the specified record(s). To terminate the search while it is in process, enter "?BRK". "?STATUS" can be used to determine the status of the search.

SELECTION CONDITION

The <selection condition> syntax is as follows:



The <boolean primary> syntax is as follows:



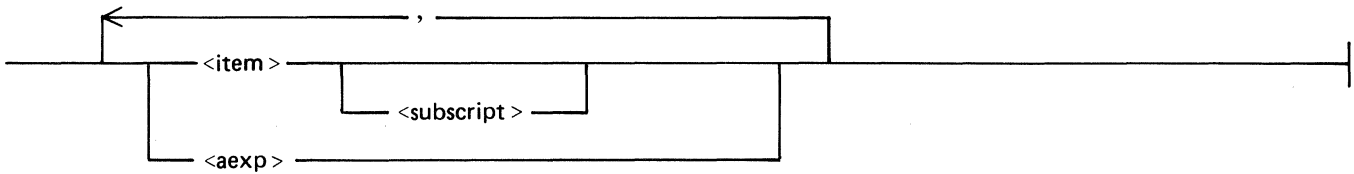
Semantics:

1. A <selection condition> specifies conditions which must be satisfied by a record if that record is to be selected.
2. A <selection condition> is a generalized boolean expression consisting of nested <selection condition>s and <boolean primary>s.
3. Parentheses can be used in conjunction with AND, OR, and NOT to form complex <selection condition>s.
4. <boolean primary>s yield a truth value of either TRUE or FALSE. All items referenced must be contained in the current record or in records previously selected at an outer nesting level.
5. <alpha item> is an ALPHA data item. The <alpha item> must be an item in the selected record. The item can be qualified. Refer to Items in this section.
6. <arithmetic function>s are discussed under the heading Functions in this section.
7. <aexp> (arithmetic expression) is discussed under the heading Arithmetic Expressions in this section.
8. <alpha-literal>s are described in Section 4.
9. <boolean function>s must reference items in an embedded data set declared within the record being selected. Refer to Functions in this section.
10. <rel op> indicates that a relational operator is to be used. The following relational operators are available:

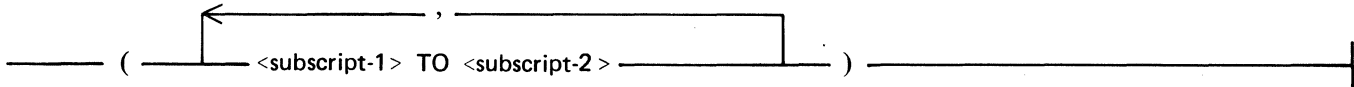
Operators	Symbolic Operators	Definition
EQL	=	Equal to
NEQ	≠	Not equal
LSS	<	Less than
LEQ	<=, ≧	Less than or equal to
GTR	>	Greater than
GEQ	>=, ≦	Greater than or equal to

DISPLAY LIST

The <display list> syntax is used in the DISPLAY statement.



The syntax of the <subscript> is as follows:

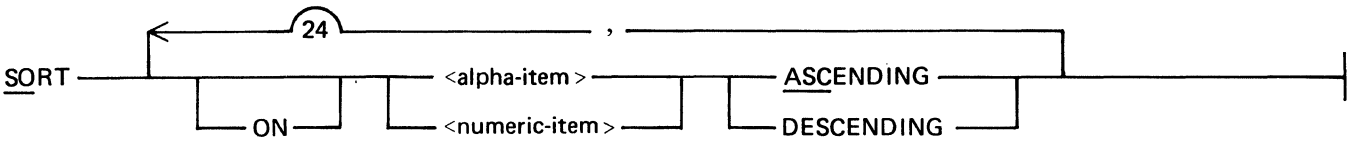


Semantics:

1. <display list> specifies the entities to be displayed.
2. <item> is an item of a data set record. The <item> can be qualified. Refer to Items in this section.
3. <subscript-1> and <subscript-2> are integer constants between 1 and the DASDL defined limit for an item declared with an OCCURS clause in DASDL. If no <subscript>s are given for an occurring item, all occurrences are displayed.
4. <aexp> (arithmetic functions) is discussed under the heading Arithmetic Expressions in this section.

SORT SYNTAX

The <sort syntax> can be included in the SELECT/DISPLAY statement.



Semantics:

1. SORT allows records to be accessed in a user specified sequence.
2. All sort keys must be data items of the selected data set. <alpha-item> refers to an ALPHA data item. <numeric item> refers to a NUMBER data item. Sort keys cannot be qualified. All occurring data items must be subscripted using integer constants. The number of sort keys specified cannot exceed 25.
3. If neither ASCENDING or DESCENDING is specified, ASCENDING is assumed.
4. The <sort syntax> can be established, modified, or eliminated using the EDIT or SET statements.

Pragmatics:

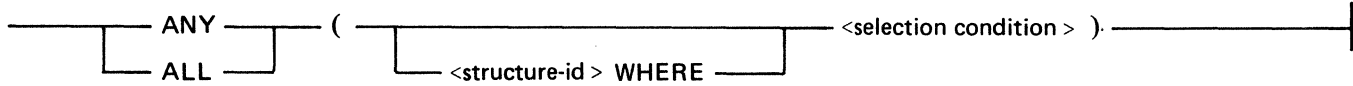
1. SORT utilizes the system SORT intrinsic.
2. Each record of the data set that satisfies the <selection condition> is read from the data base. An entry is made in a tag file which consists of the extracted <key item>s and the data base address of the record. The tag file is passed to the system SORT intrinsic which sorts the tag entries in <key item> sequence.
3. The DMS/INQUIRY program reads the sorted tag file sequentially. For each tag entry, the DMS/INQUIRY program reads the data set record at the address indicated in the tag entry.
4. The DMS/INQUIRY program then performs as if the data set record was obtained directly from the data set; for example, the DMS/INQUIRY program displays items or executes the statement associated with any attached embedded data set.

FUNCTIONS

INQUIRY provides boolean functions and arithmetic functions.

Boolean Function

Syntax:

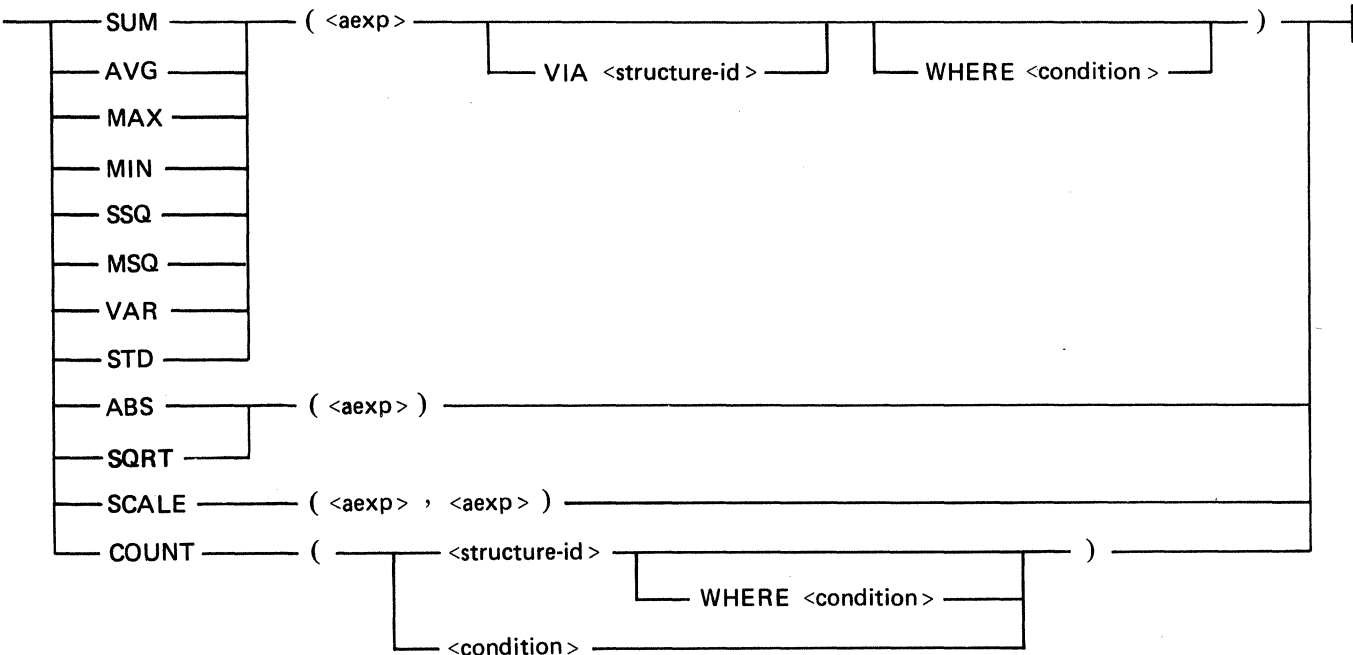


Semantics:

1. ANY yields the value TRUE if any record in the embedded data set satisfies the <selection condition>.
2. ALL yields the value TRUE if all records in the embedded data set satisfy the <selection condition>.
3. If the function is referenced at a given nesting level, the terms in <selection condition> must reference at least one item in a data set at the next inner nesting level.
4. If a <structure-id> is specified, only records retrieved through the specified structure are considered.

Arithmetic Function

Syntax:



Semantics:

1. For the functions COUNT, SUM, AVG, MAX, MIN, SSQ, MSQ, VAR, and STD, if the function is referenced at a given nesting level (excluding the disjoint data set level), then the terms in <aexp> and <condition> must reference at least one item of the data set at the next inner nesting level. For all these arithmetic functions, if the WHERE <condition> is not specified then all records are utilized to produce the function value.

2. <aexp> is an arithmetic expression.
3. For the functions ABS through SCALE, the <aexp> can be any general arithmetic expression.
4. If a <structure-id> is specified, then only records retrieved by means of that structure are considered.
5. An <arithmetic function> is undefined if no records satisfy the <selection condition>. When an undefined situation is encountered, two hyphens are displayed in lieu of a value. Undefined items are described in Section 4, under the heading Undefined Items.

The <arithmetic function>s are defined as follows:

COUNT

Returns the number of records which satisfy the <selection condition>, or the number of records in the structure.

SUM

Sum of the values of all items in the sequence.

AVG

Average is the sum of the specified items in the sequence divided by the number of records summed.

MAX

Maximum is the value of the largest arithmetic item in the sequence.

MIN

Minimum is the value of the smallest arithmetic item in the sequence.

SSQ

Sum of the squares of the specified items in the sequence.

MSQ

Mean square is the sum of the squares of the specified items in the sequence divided by the number of items.

VAR

Variance is $(SSQ - N * AVG^2) / (N - 1)$ where N is the number of records selected.

STD

Standard deviation is the square root of the variance.

SCALE(X,F)

Scale returns the value X, rounded to F fractional digits. The nonfractional digits are unaffected.

ABS(X)

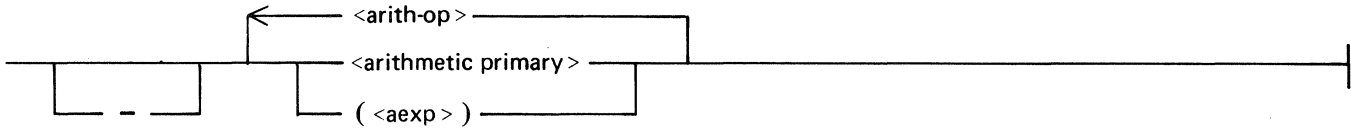
Absolute value of X.

SQRT(X)

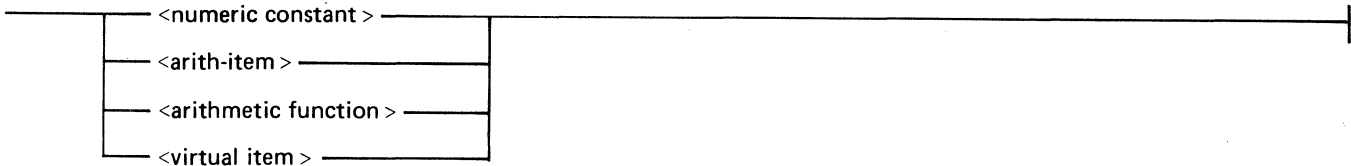
Square root where $X \geq 0$.

ARITHMETIC EXPRESSIONS

The syntax of the <aexp> (arithmetic expression) is as follows:



The syntax of the <arithmetic primary> is as follows:



The <aexp> is a generalized arithmetic expression. If used in a SELECT condition, all arithmetic items must be in data sets having current records (previously selected) or in the data set being selected. Parentheses can be used to combine arithmetic expressions into more complex arithmetic expressions.

The following <arith-op> (arithmetic operators) are allowed.

Operator	Definition
+	Add
-	Subtract
*	Multiply
/	Divide
DIV	Integer Divide
MOD	Remainder Divide

A <numeric constant> is a number.

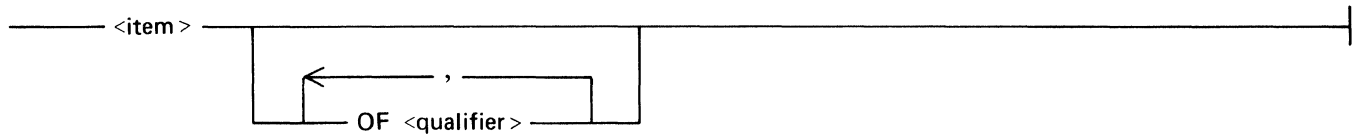
An <arith-item> is an item that yields a numeric value.

An <arithmetic function> is specified under the heading Functions.

A <virtual item> is an item established by the VIRTUAL statement. Refer to Virtual in this section.

ITEMS

The syntax to qualify items is as follows:



A <qualifier> can be used to create a unique name when data items with the same name exist in the data base. A <qualifier> must be a <data-set-id> or <group item name>. Multiple levels of qualification can be used as necessary.

EXAMPLES OF SELECT/DISPLAY STATEMENTS

The following examples illustrate the use of the SELECT/DISPLAY statement.

Example:

```
DISPLAY A1 AT A1 > 0, DISPLAY B2 AT B2 > 1
```

This statement selects a record of A where A1 is greater than zero and displays the value of A1. The DMS/INQUIRY program then examines all B records owned by the current A record, and selects the first record for which B2 is greater than 1.

If a B record is located which satisfies the <selection condition>, the value of B2 is displayed, and the DMS/INQUIRY program attempts to locate another record of B which satisfies the <selection condition>. Selection and display of B records continues until all records which satisfy the <selection condition> have been displayed. When all records of B which belong to the current record of A have been exhausted, a new A record is selected and the process is repeated.

If no B records for the current A record satisfy the <selection condition>, the next record of A is selected and the same process is repeated for it.

This process continues until all A records which satisfy the <selection condition> are exhausted.

Example:

```
SELECT A AT A1 = 1, SELECT B AT B2 > 2
```

The DMS/INQUIRY program first selects an A record for which A1 is equal to 1. The DMS/INQUIRY program next examines all B records owned by the current A record, and selects the first record where B2 is greater than 2.

When a record from A and a record from B have been selected, the DMS/INQUIRY program displays a “#” and waits for further input.

The user can use the DISPLAY statement to examine items in the selected records.

If NEXT is entered, the DMS/INQUIRY program selects the next B record which satisfied the <selection condition>, displays a “#”, and waits for further input

When all B records have been exhausted for the current A record, NEXT causes the DMS/INQUIRY program to select a new A record and the process described above is repeated.

If the A record does not contain any B records which satisfy the <selection condition>, the DMS/INQUIRY program displays a “#” and awaits further input. The user can display items from the selected A record, and can enter NEXT to locate the next A record which satisfies the <selection condition>.

The next record in A can be selected at any time by entering NEXT A. When NEXT A is entered, the DMS/INQUIRY program bypasses any remaining B records and selects the next A record.

When all A records have been exhausted, the DMS/INQUIRY program displays “#NO MORE ON SELECTION OF A”.

Example:

```
SELECT A AT A1 > 0, DISPLAY B2 AT B2 > 1
```

The DMS/INQUIRY program selects a record of A where A1 is greater than zero. All B records belonging to the current A record where B2 is greater than 1 are selected and the value of B2 is displayed.

The DMS/INQUIRY program displays a “#” and waits for further input when all B records for the current A record are exhausted.

The user can enter NEXT to select a new record of A which satisfies the <selection condition>.

Processing of A records and B records continues as described above until no more records in A satisfy the <selection condition>.

When the records of A are exhausted, the DMS/INQUIRY program displays “#NO MORE ON SELECTION OF A”.

ATTACH

Syntax:

ATTACH <data-set-id>

Semantics:

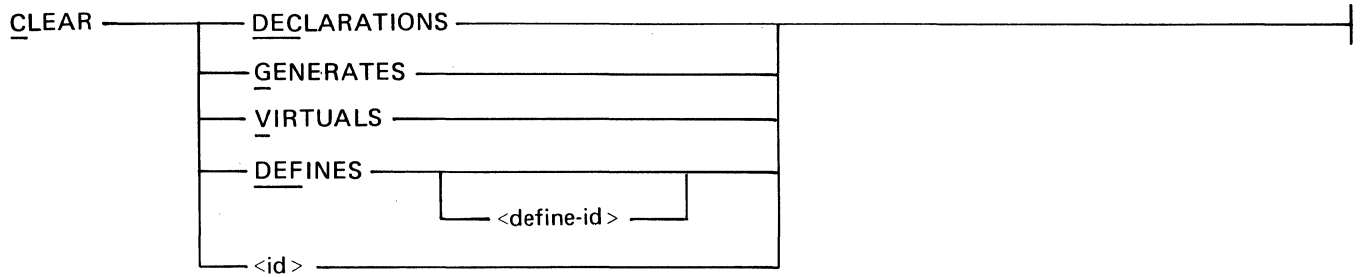
1. The DMS/INQUIRY program remembers the most recently entered statement for each data set. ATTACH can be used to combine the statement entered for an embedded data set with that of its owner. This permits automatic looping between the two structures. Through use of the ATTACH verb, statements entered for individual data sets may be combined as though they were entered in a single SELECT/DISPLAY statement.
2. <data-set-id> must be an embedded data set.
3. All structures selected in a combined SELECT/DISPLAY statement are attached by default.

Example:

```
SELECT A WHERE A1 = 1
SELECT B WHERE B2 > 2
ATTACH B
```

CLEAR

Syntax:



Semantics:

1. CLEAR is used to discard DEFINE items, VIRTUAL items, and generated subsets.
2. CLEAR DECLARATIONS causes the DMS/INQUIRY program to eliminate all DEFINE, VIRTUAL, and GENERATE items.
3. CLEAR DEFINE(S), CLEAR GENERATE(S), and CLEAR VIRTUAL(S) delete all items of the specified type.
4. CLEAR DEFINE <define item name> and CLEAR <define item name> can be used to discard individual DEFINE items.
5. An item that has been cleared can no longer be referenced. However, the item can be re-established by:
 - a. The RESTORE statement (assuming that the item is saved).
 - b. Re-creating the item in the normal manner.

Examples:

```
CLEAR DECLARATIONS
CLEAR GENERATES
CLEAR VIRTUALS
CLEAR DEFINES XYZ
CLEAR XYZ
```

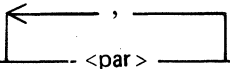
DEFINE

Syntax:

Option 1:

DEFINE <define-id> = <text> _____|

Option 2:

DEFINE <define-id> () = <text> _____|

Semantics:

1. DEFINE establishes <define-id> as a synonym for the <text>.
2. <define-id> is an <identifier> and must be unique. <define-id> cannot be an INQUIRY verb, a data base item identifier, or a generated subset name.
3. A parametric DEFINE (option 2) allows parameters to be supplied at the time the define item is referenced.
4. When a parametric define is referenced, the formal parameters appearing in the text of the define are replaced by the actual parameters. For example, the define

DEFINE GET2(M,N) = DISPLAY A2 AT M GTR N

can be invoked as follows:

GET2(A1,5)

The DMS/INQUIRY program replaces M by A1 and N by 5. The DMS/INQUIRY program interprets this as:

DISPLAY A2 AT A1 GTR 5

DETACH

Syntax:

DETACH <data-set-id>

Semantics:

1. DETACH separates an embedded data set from its owner and prevents automatic looping between the two data sets. Refer to ATTACH in this section for additional information.
2. <data-set-id> must identify an embedded data set.
3. If the parent and embedded data set are specified in the same SELECT or option 2 of the DISPLAY statement, they are attached by default.
4. When the owner data set is selected, the embedded data set is marked as having no current record. The DMS/INQUIRY program displays an appropriate message if an attempt is made to display any items contained in the embedded data set.

Example:

```
SELECT A AT A1 = 1, THEN SELECT B AT B2 > 2  
DETACH B
```

EDIT

Syntax:

EDIT _____ <delim> <text> <delim> _____
REPEAT _____ <text> _____

Semantics:

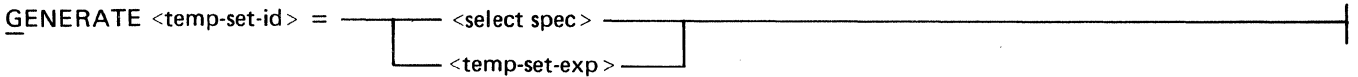
1. EDIT modifies the text in the command buffer.
2. The command buffer normally contains the text of the most recently entered DEFINE, DISPLAY, GENERATE, SELECT, SET, or VIRTUAL statement. RECALL can be used to replace the current contents of the buffer by the text of a previously entered statement.
3. If the second <text> is specified, the first occurrence of the first <text> is replaced by the second <text>; otherwise, the first <text> is eliminated from the command buffer. <delim> may be any special character not contained in the <text>; however, both <delim>s must be the same character.
4. After each EDIT is entered, the contents of the command buffer are displayed. EDIT can be performed as many times as necessary to alter the statement.
5. REPEAT causes the edited command in the command buffer to be executed. The REPEAT statement can be used after the EDIT statement. Refer to REPEAT in this section for additional information.
6. EDIT does not repeat or reprocess the modified text; therefore, a sequence of modifications may be lost if REPEAT is not entered prior to other input. REPEAT can be used to cause the new text to be processed immediately.

Example:

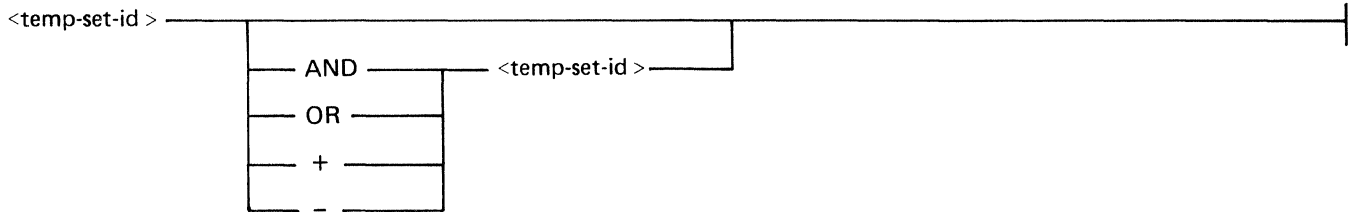
```
SELECT A WHERE A1 GTR 3 EDIT /GTR/ GEQ
```

GENERATE

Syntax:



<temp-set-exp> syntax is as follows:



Semantics:

1. GENERATE creates a temporary subset which can be referenced by SELECT and DISPLAY statements. <temp-set-id> can be used in lieu of <structure-id> in a SELECT or DISPLAY statement.
2. The generated subset contains only the records which satisfy the condition at the time the subset is created.
3. <temp-set-id> must be a unique identifier. That is, <temp-set-id> cannot be identical to any data base item name, define item name, virtual item name, generated subset name, or INQUIRY verb.
4. The functions AND, OR, +, and - can be used to form a generated subset from two existing generated subsets. All generated subsets referenced in a <temp-set-exp> must refer to the same data set.

The operators have the following meanings:

Operation	Meaning
$T = T1 \text{ AND } T2$	Each entry in T is contained in both T1 and T2. (subset intersection)
$T = T1 \text{ OR } T2$	Each entry in T is contained in either T1, or T2, or both. (subset union)
$T = T1 + T2$	Each entry in T is contained in either T1 or T2, but not both. (subset exclusive OR)
$T = T1 - T2$	Each entry in T is present in T1 and is absent from T2. (subset difference)

5. Although the text used to generate the generated subset can be saved, INQUIRY discards the generated subset at the end of each session.

Pragmatics:

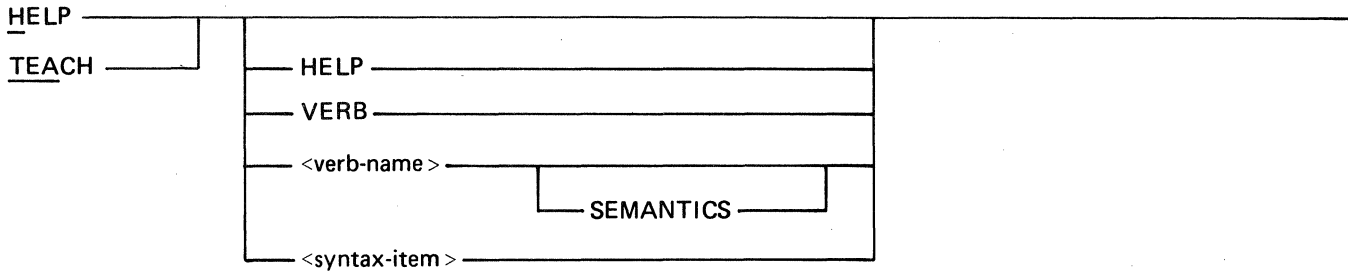
1. The DMS/INQUIRY program searches the data base until the temporary data set is built. To terminate the search while it is in process, enter "?BRK". "?STATUS" can be used to determine the status of the search.

Examples:

```
GENERATE V = A1 GTR 3
G X = A1 = 4
G Y = V + X
G Z = V AND X
```

HELP

Syntax:



Semantics:

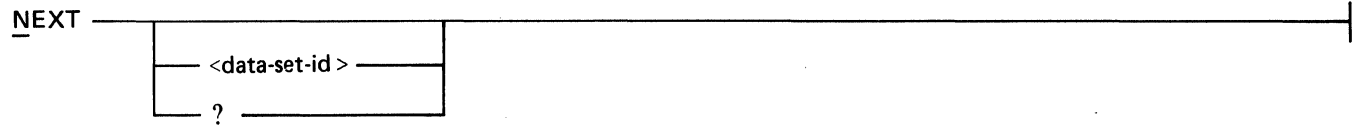
1. HELP displays information regarding the DMS/INQUIRY program. TEACH is equivalent to HELP.
2. When HELP is entered alone, the name of the DMS/INQUIRY program is displayed.
3. HELP HELP displays the syntax of the HELP statement.
4. If HELP VERB or HELP VERBS is entered, the system displays a list of the INQUIRY verbs.
5. HELP <verb-name> (without SEMANTICS) displays the syntax of the verb.
6. HELP <verb-name> SEMANTICS displays the semantics for the verb.
7. When <verb-name> is included, the entire <verb-name> must be entered. Abbreviations are not allowed.
8. HELP <syntax-item> displays the syntax of the syntax-item. A "syntax-item" is any name enclosed in "<" and ">" in any syntax diagram displayed by HELP. The "<" and ">" must be included when specifying <syntax-item>.

Pragmatics:

1. If the file labeled DMS/HELPIHQ is not present, HELP information is not available and the DMS/INQUIRY program responds with a message indicating that this file is not present.

NEXT

Syntax:

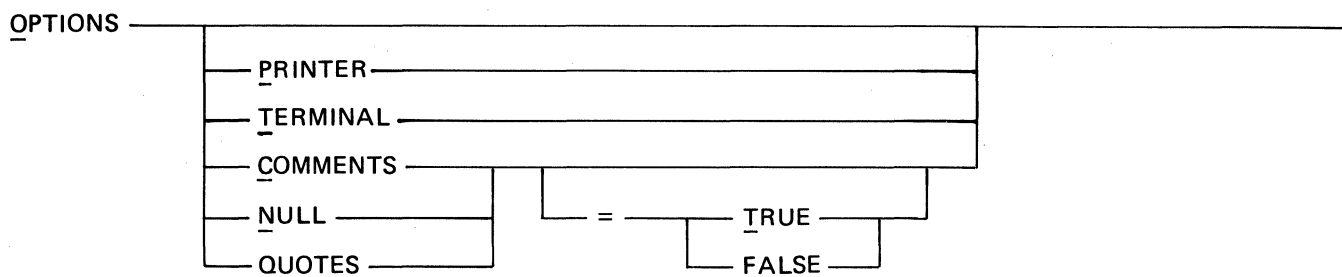


Semantics:

1. NEXT causes the DMS/INQUIRY program to continue record selection and item display of the most recently entered command from the point at which it stopped.
2. NEXT <data-set-id> causes the DMS/INQUIRY program to continue to process the most recently entered command at the point where the next record is to be selected for <data-set-id>.
3. NEXT ? causes the DMS/INQUIRY program to display the name of the next data set to be selected.

OPTIONS

Syntax:



Semantics:

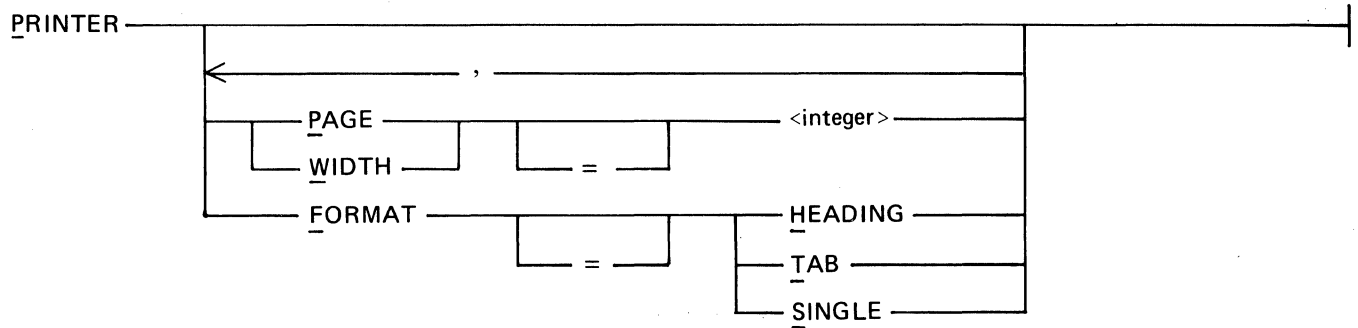
1. OPTIONS without any parameters displays the current setting of the options.
2. OPTIONS TERMINAL causes output generated by DISPLAY statements to be listed on the user's terminal (resets OPTIONS PRINTER). TERMINAL is set by default.
3. OPTIONS PRINTER causes output generated by DISPLAY statements to be printed on the system line printer (resets OPTIONS TERMINAL). The number of lines per page and the number of characters per line can be specified using the PRINTER statement. Refer to the PRINTER statement for additional information.
4. TERMINAL or PRINTER can be set at any time. The option remains set throughout the DMS/INQUIRY session or until it is explicitly altered.
5. COMMENTS or COMMENTS = TRUE causes quoted DASDL comments to be displayed along with the identifier when using the SHOW statement. COMMENTS = FALSE suppresses the display of DASDL comments. COMMENTS = FALSE is set by default.
6. NULL = FALSE suppresses display of items whose values are NULL when TERMINAL FORMAT TAB or TERMINAL FORMAT SINGLE is used. NULL or NULL = TRUE displays all items regardless of their value. Two hyphens are displayed for items whose value are NULL. NULL is TRUE by default.
7. QUOTES or QUOTES = TRUE requires alpha literals to begin and end with quotation marks. QUOTES = TRUE is set by default. QUOTES = FALSE allows quotation marks to be omitted.

NOTE

Before using OPTIONS QUOTES, see the description of Unquoted Alpha-literals in Section 4.

PRINTER

Syntax:



Semantics:

1. PRINTER displays or alters the printer attributes assigned by the DMS/INQUIRY program.
2. When PRINTER alone is specified, the current printer attributes are displayed.
3. PAGE specifies the number of lines displayed per page.
4. WIDTH specifies the number of characters displayed on each line.
5. FORMAT is used to select one of three output formats. The HEADING, TAB, and SINGLE options are described in Section 4. SINGLE is the default format type.
6. The printer attributes are active only if OPTIONS PRINTER is set. Refer to the OPTIONS statement for additional information.

QUIT

Syntax:

QUIT

BYE

Semantics:

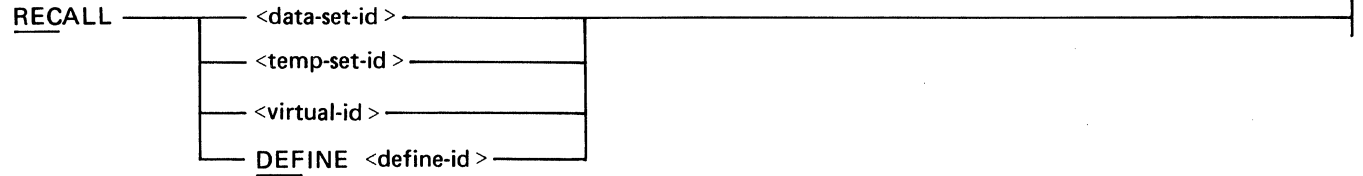
1. QUIT terminates the DMS/INQUIRY program. BYE is equivalent to QUIT.
2. Prior to termination of the session, the text of all DEFINE, GENERATE, or VIRTUAL items must be saved or discarded.
3. If QUIT is entered, and DEFINE, GENERATE, or VIRTUAL items have been added, deleted, or modified but have not been saved, the following message is displayed.

DECLARATIONS NOT SAVED

4. If SAVE is entered, the declarations are saved and the DMS/INQUIRY program is terminated.
5. If QUIT is entered again, the DMS/INQUIRY program is terminated and the declarations discarded.
6. Any other input causes the DMS/INQUIRY program to ignore the attempted "QUIT" and to process the new command.

RECALL

Syntax:



Semantics:

1. RECALL loads the command buffer with the specified item and displays that item on the terminal.
2. Once the text has been recalled, it can be modified using the EDIT statement or executed using the REPEAT statement.
3. If a DEFINE <define-id> is specified, the DMS/INQUIRY program loads and displays the text associated with the <define-id>.
4. If a <data-set-id> is specified, the DMS/INQUIRY program loads and displays the text of SELECT and DISPLAY statements associated with the data set.
5. If a <temp-set-id> is specified, the DMS/INQUIRY program loads and displays the text of the associated GENERATE statement.
6. If a <virtual-id> is specified, the DMS/INQUIRY program loads and displays the text of the VIRTUAL statement.

Example:

RECALL A

REPEAT

Syntax:

REPEAT _____
 ┌──────────────────┐
 <data-set-id >

Semantics:

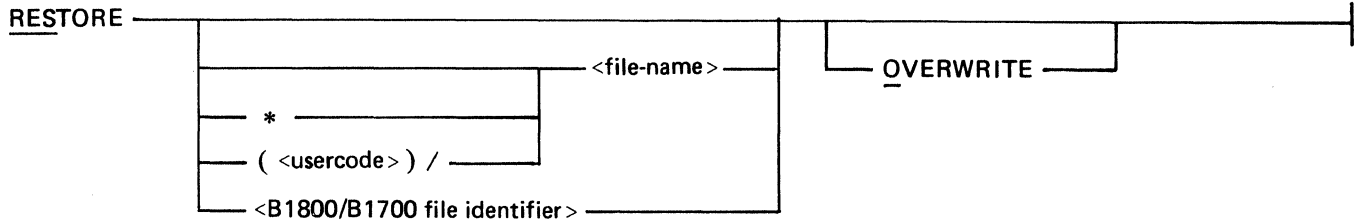
1. REPEAT causes the command in the command buffer to be executed.
2. If a DEFINE, VIRTUAL, or GENERATE command is presently in the command buffer, REPEAT causes the DMS/INQUIRY program to remember the text associated with the command. If RECALL is entered to place the text of a DEFINE item, VIRTUAL item, or generated subset in the command buffer, and the text is altered, REPEAT causes the DMS/INQUIRY program to discard the old text and retain the new text. If the text associated with these commands has been altered and REPEAT is not entered, the old text is retained.
3. Similarly, the text of the most recent SELECT or DISPLAY statement associated with each data set is remembered by the DMS/INQUIRY program. Therefore, if REPEAT <data-set-id> is entered, statements associated with the data set and any statements associated with attached embedded data sets are re-executed. REPEAT starts the selection process from the beginning.
4. REPEAT can be used to modify and execute the text in the current buffer, within one operation. Refer to the EDIT statement for details.

Example:

REPEAT A

RESTORE

Syntax:



Examples:

RESTORE

RESTORE OVERWRITE

RESTORE XYZ

RESTORE *XYZ OVERWRITE

RESTORE (USERA)/XYZ

Semantics:

1. RESTORE is used to reload previously saved declarations.
2. The <file-name> is a 10-character file identifier.
3. The <B1800/B1700 file identifier> is a standard B 1800/B 1700 file identifier.
4. If an asterisk (*) is specified, the file is assumed to reside on the system disk.
5. If a (<usercode>) is specified, the file is assumed to have that usercode as the multi-file-id and is assumed to be on that usercode's default pack.
6. If no file name is specified, the DMS/INQUIRY program restores declarations from the default definitions file. If DMS/INQUIRY is not run under file security, the file name is DEF.DECLAR/<database name>; if DMS/INQUIRY is run under file security, then the file name is (<usercode>)/XXXXXX.DEF, where XXXXXX is the first six characters of the <database name>.
7. The file name must have previously been created using the SAVE statement. An asterisk or a (<usercode>) must be used if the declarations file was saved using an "" or (<usercode>).
8. Normally, when an existing declaration and a saved declaration have the same name, RESTORE retains the existing declaration and ignores the saved declaration. OVERWRITE causes the saved declaration to replace the existing declaration.

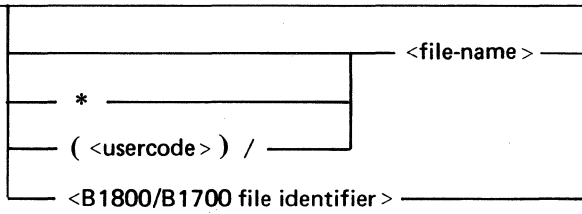
Pragmatics:

1. Following a RESTORE, the SHOW DEFINES, SHOW GENERATES, and SHOW VIRTUALS statements can be used to display the current value of the restored declarations.
2. RESTORE does not execute any of the declarations or regenerate any subsets created by the GENERATE statement. RECALL and REPEAT can be used to re-establish generated subsets.
3. Nonprivileged users cannot access files saved with a (<usercode>) other than their own.

SAVE

Syntax:

SAVE

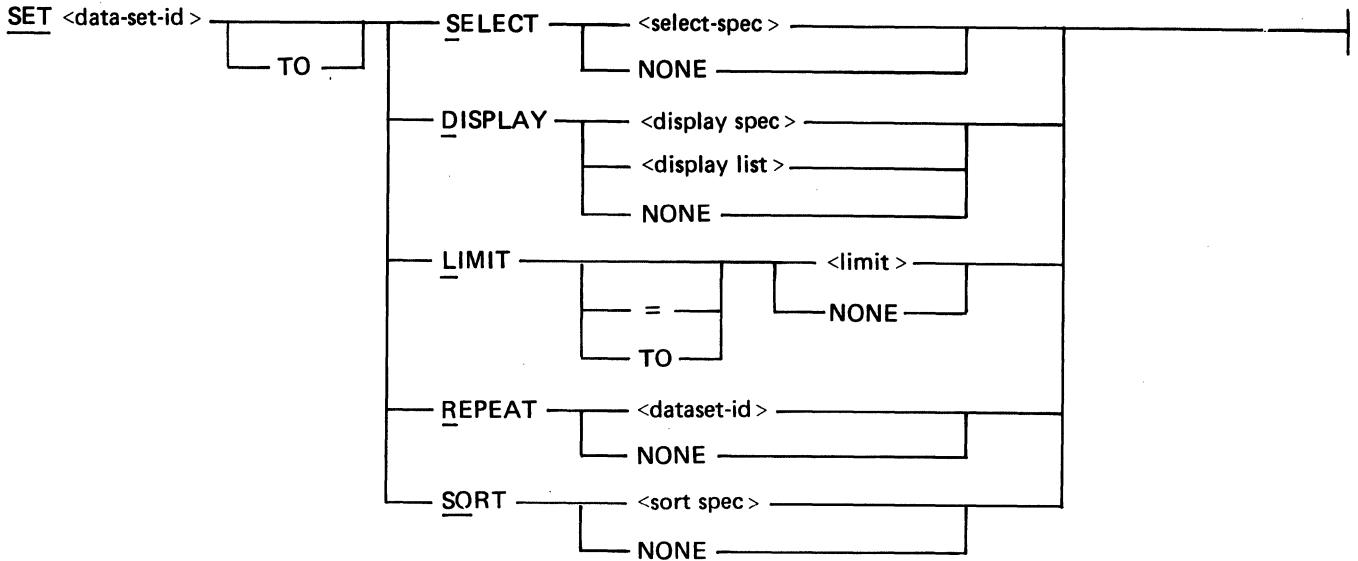


Semantics:

1. SAVE stores the text of DEFINE, VIRTUAL, and GENERATE statements. This allows the user to declare these items during one INQUIRY session and save them for use during subsequent sessions.
2. Refer to the RESTORE statement for details on file names.

SET

Syntax:

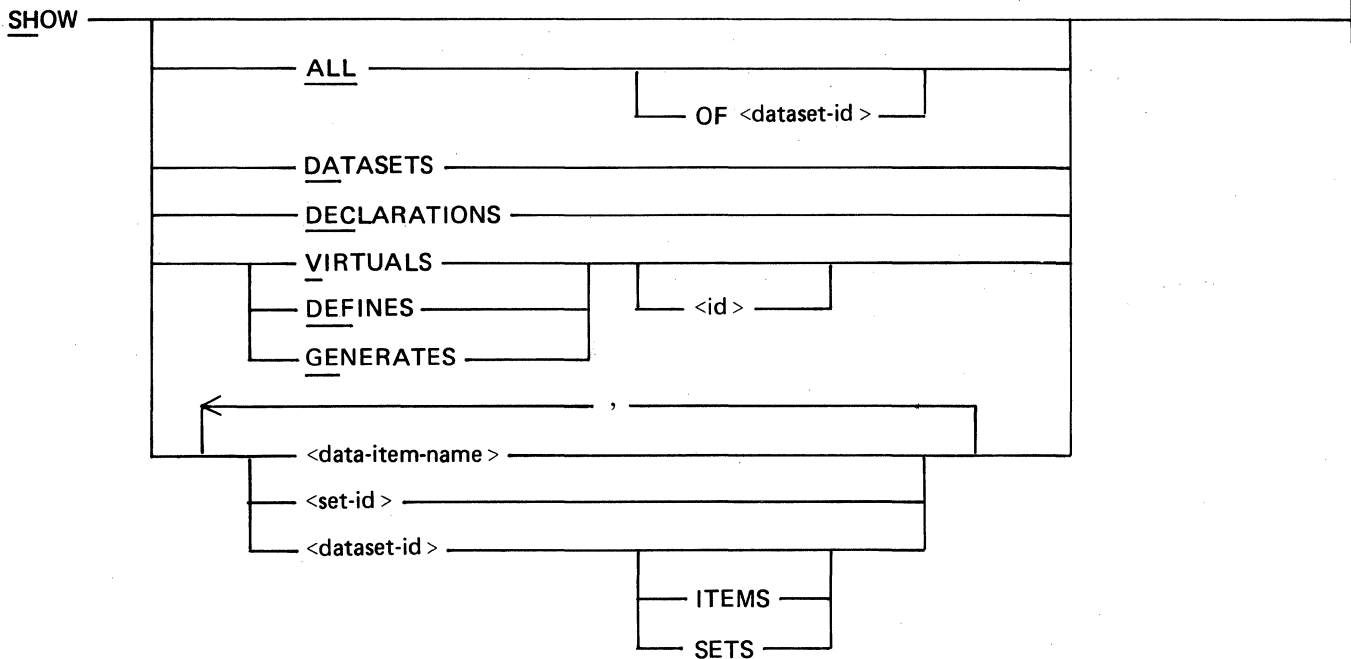


Semantics:

1. The DMS/INQUIRY program remembers the most recently entered DISPLAY, LIMIT, SELECT, and SORT text for each data set. SET can be used to modify or delete this text.
2. SET only changes the text associated with the data set; it does not select a record or display an item.
3. SET DISPLAY changes or eliminates the <display list> for the data set. If NONE is specified, the <display list> is eliminated; otherwise, the <display list> is changed. A new <select spec> or <limit> can be included in the <display spec>.
4. SET LIMIT modifies or eliminates the <limit> for the data set.
5. SET REPEAT modifies or eliminates the repeat text for the data set.
6. SET SELECT changes or eliminates the <select spec> for the data set. If NONE is specified, the <select spec> is eliminated; otherwise, the <select spec> is changed. When the <select spec> for a data set is changed or eliminated, there is no longer a current record selected for that data set or for any data set embedded within it.
7. SET SORT modifies or eliminates the sort text for the data set.
8. The RECALL command can be used to load the text for a data set into the command buffer. This text can be edited and then it may be executed using REPEAT.

SHOW

Syntax:

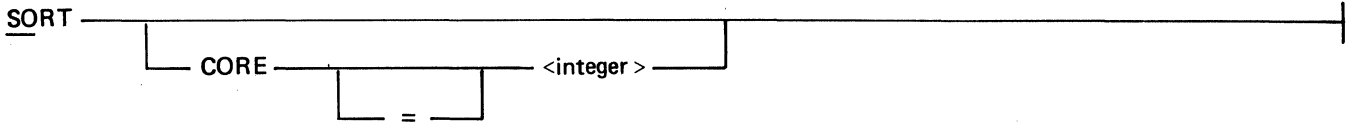


Semantics:

1. If SHOW alone is entered, the contents of the command buffer are displayed.
2. SHOW ALL displays the DASDL description of the entire data base.
3. SHOW DATASETS displays the names of all data sets in the data base.
4. SHOW DECLARATIONS displays the names and text associated with all DEFINE, GENERATE, and VIRTUAL items.
5. SHOW DEFINES displays the names and text of all DEFINE items.
6. SHOW DEFINES <id> displays the text of the specified DEFINE item.
7. SHOW GENERATES displays the names and text of all generated subsets.
8. SHOW VIRTUALS displays the names and text of all VIRTUAL items.
9. SHOW <dataset-id> displays the DASDL description of the data items, sets, and subsets associated with the data set. SHOW <dataset-id> ITEMS displays the DASDL description of the data items in the data set. SHOW <dataset-id> SETS displays the DASDL description of the sets associated with the data set.
10. SHOW <set-id> displays the DASDL description of the set and indicates which data set it references.
11. SHOW <data-item-name> displays the DASDL description of the data item. It also displays the name of the data set which contains the item.
12. The words ALL, DATASETS, DECLARATIONS, DEFINES, GENERATES, GLOBAL, or VIRTUALS must be preceded by a “#” character if the word is also the name of a data base identifier. For example, if ALL is a data set name, SHOW ALL displays the description of the data set. To display a description of the entire data base, SHOW #ALL must be entered.
13. If the COMMENTS option is TRUE (refer to the OPTIONS statement), comments appearing in the DASDL declaration of the data base are displayed.
14. SHOW output cannot be listed on the line printer.

SORT

Syntax:



Semantics:

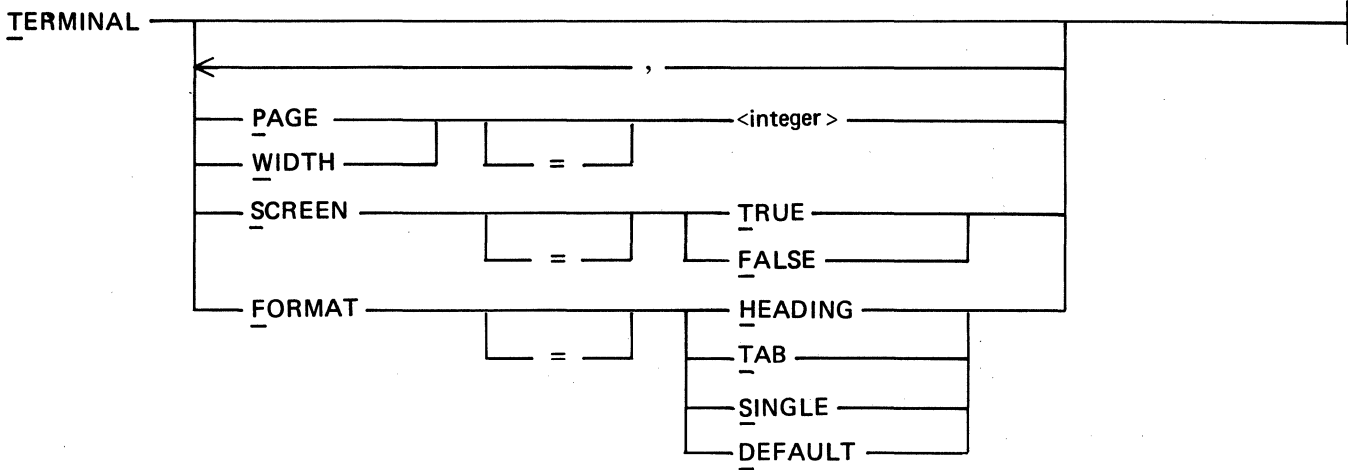
1. SORT is used to display or alter the SORT memory parameter used by the <sort syntax> of the SELECT and DISPLAY statement.
2. When SORT alone is entered, the current value for SORT memory is displayed.
3. CORE = <integer> can be used to change the sort memory parameter.
4. By default, the SORT memory size is 8000 bytes.

Pragmatics:

1. In general, increasing the size of <integer> increases the speed of the sort.

TERMINAL

Syntax:



Semantics:

1. **TERMINAL** displays or alters the terminal attributes assigned by the DMS/INQUIRY program. The initial values for the **PAGE**, **WIDTH**, and **SCREEN** attributes depend upon the terminal being used.
2. When **TERMINAL** alone is specified, the current terminal attributes are displayed.
3. **FORMAT** is used to select one of four output formats. The four options, **HEADING**, **TAB**, **SINGLE**, and **DEFAULT** are described in Section 4. **DEFAULT** is the default format.
4. **PAGE** specifies the number of lines displayed per page on CRT terminals.
5. **WIDTH** controls the number of characters displayed on each line. Some terminals are designed to perform an automatic line advance after printing to the last character position of the line. This can cause the output to be double spaced. This can be avoided by setting the width to one character less than the screen width.
6. **SCREEN** should be **TRUE** for CRT terminals. **SCREEN = TRUE** causes the DMS/INQUIRY program to suspend the displaying of records after displaying a full page until a null (blank) input is received from the user.

VIRTUAL

Syntax:

VIRTUAL _____ < virtual-id > _____ = _____ < aexp > _____ |

Semantics:

1. **VIRTUAL** creates a virtual item and assigns the value of the associated expression to it. Virtual items can be used in <display list>s or <selection condition>s.
2. <virtual-id> is an identifier and must be unique.

Pragmatics:

1. When a virtual item appears in a <display list> or <selection condition>, the items appearing in the expression must be contained in the current record or in records previously selected at a outer nesting level.
2. If the expression contains functions, these functions must apply to items contained in records at the next inner nesting level.
3. The DMS/INQUIRY program recognizes when the value of a virtual item must be recomputed. When this occurs, the DMS/INQUIRY program recomputes the <arithmetic expression> and assigns the new value to the virtual item.

SECTION 6

INQUIRY FILES AND SYSTEM GENERATION

GENERAL

The DMS INQUIRY system consists of three basic components: the DMS/BUILDINQ program, the DMS/INQUIRY program, and the DMS/HELPIQ file.

The DMS/BUILDINQ program is run before any inquiry into the data base in order to set up the necessary information for the DMS/INQUIRY program. The DMS/BUILDINQ program builds the control file labeled <database-name>/INQCTL. This control file is associated with only one data base. The DMS/BUILDINQ generates separate INQCTL files for separate data bases.

The DMS/INQUIRY program is used for the actual selection and display of the data base records.

The DMS/HELPIQ file is a data file which must be present in order for the HELP statement to function.

RUNNING ENVIRONMENTS

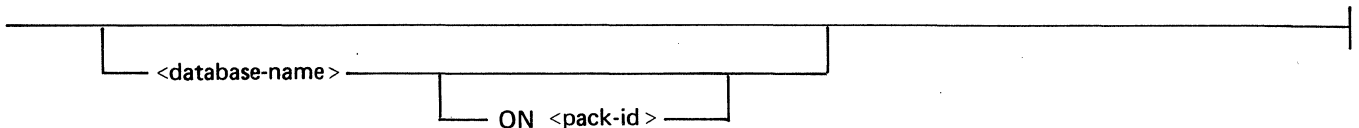
Both the DMS/BUILDINQ and DMS/INQUIRY programs are data communication programs. These programs are designed to use the CANDE request set in the network controller. Both programs must have the internal file labeled REMOTE equated to the user's terminal. Optionally, the DMS/BUILDINQ program can be executed from cards if an appropriate switch is set.

INQUIRY GENERATION

The DMS/BUILDINQ program can be executed from a terminal or as a batch job receiving input from a card file. If SW 0 is set to 1, a card file labeled CARDS is expected.

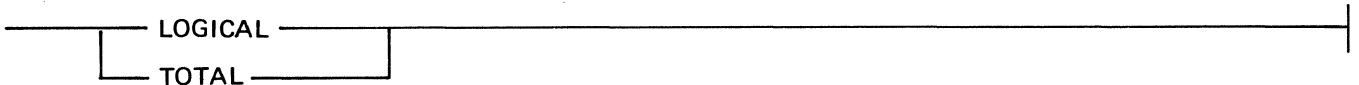
When the input device is a terminal, the DMS/BUILDINQ program asks for the information needed. The program first identifies itself, then asks for the following parameters:

1. The data base name. Valid responses are:



If a null input is entered, the DMS/BUILDINQ program terminates immediately.

2. After finding the data base dictionary, the DMS/INQUIRY program queries whether to use the total or logical data base. The valid responses are:



If the response is LOGICAL, the program asks for the logical data base name. If a logical data base is used, the INQCTL file is labeled <logical database-name>/INQCTL instead of <database-name>/INQCTL. If the input is null or the response is TOTAL, the entire data base is used.

3. The disk pack for the inquiry file labeled <database-name>/INQCTL. If the file is to reside on a user pack, the pack-id must be given. A null input indicates the system disk.
4. The timeout time. If DMS/INQUIRY detects that the user's terminal has not had any input for the length of time specified as the timeout time, then the DMS/INQUIRY program terminates. The response is the time in minutes, within the range 1 to 999. Refer to Timeout in this section.
5. Whether to restrict data base access through DMS/INQUIRY to certain usercodes. If the response is yes, then the DMS/INQUIRY program expects the usercodes to be entered, separated by commas.

Example:

USERA,USERB,USERC

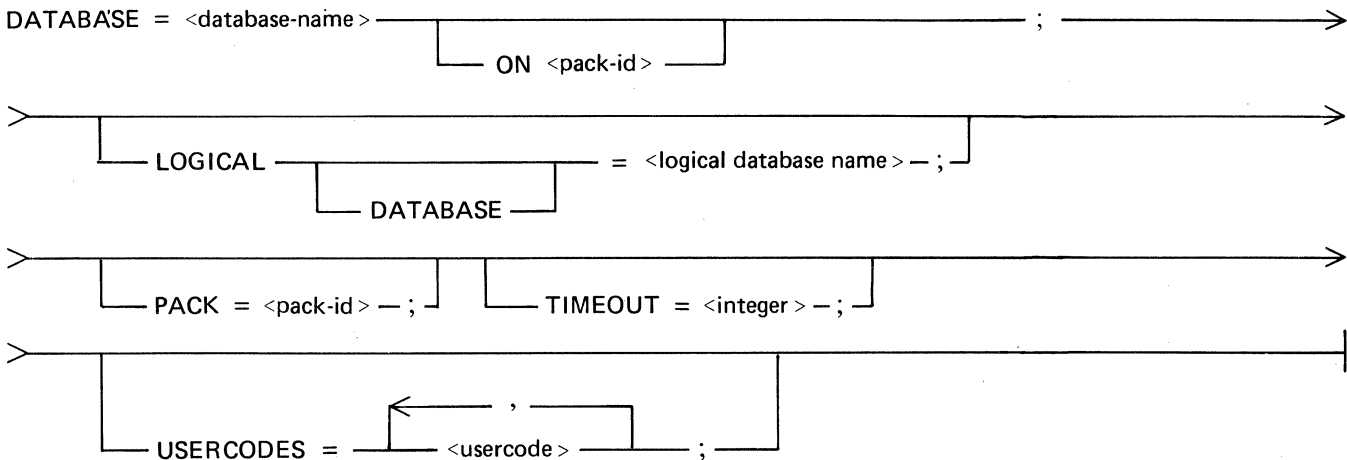
If there is more input than can fit on one input line, then terminate the line with a comma.

After generating the "INQCTL" file, the DMS/BUILDINQ program requests another data base name, for generating another INQCTL file. A null input terminates the program.

To execute the DMS/BUILDINQ program from cards, a card deck with the following statements must be used:

```
?EX DMS/BUILDINQ
?DATA CARDS
  <card specifications>
?END
```

The <card specifications> are in free-form format, one specification per card, and have the following format:



The control file is built from the information supplied by the <card specifications> or the specifications entered on the terminal. Generation of this file requires several system sorts and therefore invokes the system sort intrinsic SORT/VSORT.

SECURITY

Security in the form of usercode checking is available in the INQUIRY system. This checking can be invoked when the DMS/BUILDINQ program is executed. The DMS/BUILDINQ program should be executed with a public, privileged usercode with no default pack-id so that the INQCTL file can be created with the appropriate name.

TIMEOUT

If a user's terminal remains inactive for a specified length of time (specified in the DMS/BUILDINQ program), the DMS/INQUIRY program terminates the session and goes to end of job. If DEFINE, VIRTUAL, or GENERATE specifications exist which have not been saved, then they are saved in a file called TIMEOUTXXX where XXX is the station LSN number of the user's terminal.

INDEX

- Absolute Value Function, 5-9
- ALL Function, 4-12, 5-8
- ANY Function, 4-11, 5-8
- Arithmetic Expressions, 4-3, 5-10
- Arithmetic Functions, 4-12, 5-8
- Arithmetic Operators, 4-3, 5-10
- ATTACH Statement, 5-14
- Average Function, 4-13, 5-9

- Boolean Functions, 4-11, 5-8
- Break (BRK), 5-4
- BYE Statement, 5-24

- CLEAR Statement, 5-15
- COUNT Function, 4-13, 5-9

- Data Base Access, 4-1
- Data Set Qualification, 4-7
- Default Format, 4-19
- DEFINE Statement, 5-16
- DETACH Statement, 5-17
- DISPLAY Limit, 4-8
- DISPLAY List, 5-6
- DISPLAY Spec, 5-2
- DISPLAY Statement, 4-2, 4-6, 5-2, 5-12
- Displaying Arithmetic Functions, 4-13
- DMS/BUILDINQ Program, 6-1
- Dynamic Memory, 4-1

- EDIT Statement, 5-18
- Embedded Structures, 4-9
- Entering Inquiry Statements, 4-1

- Functions, 4-11, 5-8

- GENERATE Statement, 5-19

- Heading Format, 4-17
- HELP Statement, 5-20

- INQUIRY Generation, 6-1
- INQUIRY Statements, 3-1
- Item Qualification, 5-11

- Logical Operators, 4-3

- Maximum Function, 5-9
- Mean Square Function, 5-9
- Minimum Function, 5-9
- Multiple Line Input, 4-1

- NEXT Statement, 4-2, 5-21

- Operator Precedence, 4-4
- OPTIONS Statement, 5-22
- Output Formatting, 4-17

- PRINTER Statement, 5-23

- QUIT Statement, 5-24

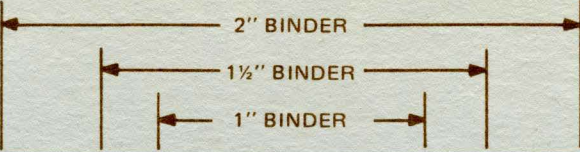
- Railroad Diagrams, 2-1
- RECALL Statement, 5-25
- Relational Operator, 4-3, 5-5
- REPEAT Statement, 5-18, 5-26
- RESTORE Statement, 5-27
- Running Environment, 6-1

- SAVE Statement, 5-28
- SCALE Function, 5-9
- Security, 6-2
- SELECT Spec, 5-2
- SELECT Statement, 4-2, 5-2, 5-12
- Selection Condition, 5-5
- Set Designation, 4-5
- SET Statement, 5-29
- SHOW Statement, 5-30
- Single Format, 4-18
- SORT Statement, 5-31
- SORT Syntax, 5-7
- Square Root Function, 5-9
- Standard Deviation Function, 5-9
- Status (ST), 5-4
- SUM Function, 4-12, 5-9
- Sum of the Squares Function, 5-9

- Tab Format, 4-18
- TEACH Statement, 5-20
- Temporary Subsets, 5-19
- TERMINAL Statement, 5-32
- Terminal Use, 4-1
- Timeout, 6-2

- Undefined Items, 4-15
- Unquoted Alpha Literals, 4-16

- Variance Function, 5-9
- VIRTUAL Items, 4-14
- VIRTUAL Statement, 5-33



B 1000 Systems
Data Management System II Inquiry
REFERENCE MANUAL

1108875

Printed in U.S.A.