IDS/II

Reference Manual

DPS7000/XTA
NOVASCALE 7000

Database Products: IDS-II

REFERENCE
47 A2 11UD 00

DPS7000/XTA NOVASCALE 7000

IDS/II

Reference Manual

Database Products: IDS-II

February 1984

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 11UD 00

Trademarks and Acknowledgements

Preface

This manual is a companion volume to the IDS/II User Guide and the IDS/II Data Base Administrator Guide; it brings together in a single book all the reference information required for effective use of these other manuals and for a full understanding of the subject.

IDS/II ("Integrated Data Store II") is a set of techniques, utility programs, and special-purpose languages which allow the user to define a data base in accordance with his own requirements, implement it to whatever level of sophistication he chooses, and efficiently access and update it.

The major tools used to this end are the DDL language, for describing the data-base "schema"; the DMCL language, for controlling the storage media; the DML language, for interfacing with standard programming languages (e.g., COBOL); and the software/firmware functions employed by these languages. Each of these tools (and related facilities) is described in detail in the following pages.

The manuals listed below are also recommended:

- IDS/II Data Base Administrator Guide (order number 47 A2 13UD) for further information on designing and managing data bases and their associated schemas,
- IDS/II User Guide (order number 47 A2 12UD) for practical guidance in applying IDS/II to specific requirements,
- IOF Terminal User's Guide (3 volumes: 47 A2 01 UJ/02 UJ/03 UJ) for information on the GCOS Command Language (GCL).
- JCL User Guide (order number 47 A2 12UJ) for further information on the Job Control Language,
- COBOL User Guide (order number 47 A2 02UL) for information on COBOL as used with IDS/II,
- LINKER (order number 47 A2 10UP) for information on constructing and interconnecting user programs.
- General Access Control (Order number 47 A2 11UF) for information on this particular subject (also called GAC).
- TDS Programmers Reference Manual (order number 47 A2 03UT) for information on Transaction Driven Subsystems.

1. Schema Data Description Language

1.1 SCHEMA VIEW

The Schema Data Description Language (DDL) describes the logical structure of the information that the programmer views when writing programs that access the data base. There are two main concepts in the schema view:

- Records and their associated fields which represent external items (for example, orders, invoices, customers etc.).
- Sets which represent relationships between these items (for example, a specific order may belong to a specific customer).

An additional concept, the area, can be used to partition records on a logical basis (one area per geographical branch of the company, for example) but may also be used in a different way when:

- It is not practical to have the whole data base on-line.
- Some areas are security sensitive (pay and personnel records, for example).

1.1.1 Program/Data Independence

If the term data refers to the view of the data base data provided by the schema, then program logic depends on the structure described. If the structure of the data changes, then program logic is affected.

If data refers to the apparently unstructured bit strings on disk, program/data independence exists in two areas:

- There are several options which will map the schema structure on to the bit string. If, for example, in a future release, for performance reasons, a set is implemented using an index instead of a chain of pointers, the logic of NEXT/PRIOR/OWNER is kept and thus the logic of the individual program is kept.
- Use of the same schema on different occurrences of the data base allows use of the same programs. These occurrences may differ in size (test and live versions of the data base).

1.2 SCHEMA DDL OVERVIEW

A schema written in DDL consists of four types of entry that:

- Identify the schema (schema entry).
- Define areas (area entry).
- Define records (record entry).
- Define sets (set entry).

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

- The schema entry must be the first entry.
- An area entry must precede the record entry for all records in that area.
- A record entry must precede a set entry that includes that record within it.
- A record entry comprises one record subentry followed by zero, one or more field subentries.
- A set entry comprises one set subentry followed by at least one member subentry.

An entry/subentry consists of one or more clauses that describe its attributes. Clauses may be written in any sequence provided that the first clause names the entry/subentry. All entries and subentries are terminated by periods. Each entry/subentry is described, in this manual, as follows:

- A description of the function of the entry/subentry.
- An entry/subentry skeleton representing the organization of the entry/subentry into clauses.
- The general formats of the entry/subentry; that is, the general format of each of the clauses that can be specified in the entry/subentry.
- A separate description of each clause.

Each clause is described, in this manual, as follows:

- A description of its function.
- General format.
- The syntax rules that apply.
- The general rules that apply.

A general format is the arrangement of elements that make up a clause.
The subdivisions of a clause are termed "phrases" and must be written in the order in which they are shown. A syntax rule amplifies or restricts the usage of the element within a general format. A general rule amplifies or restricts functions attributed to a general format or to its constituent elements.

1.2.1 DDL Source Format

Schema DDL is a free-format language. There are no special fields for sequence number, comment flag or program identification. DDL is not line oriented but the end of line is a separator when it does not occur within a delimited string (see later this section).

Input enclosures and library members containing DDL must be of type DATA or DATASSF otherwise they are not accepted by the DDL translator.

1.3 NOTATION

The notation used in all formats and the rules that apply to all formats are as follows:

- The elements that make up a clause consist of upper case words, lower case words, special symbols, and special characters.
- All underlined upper case words are required when the format is used (keywords). Example: SCHEMA NAME IS schema-name.
- Upper-case not underlined are optional (optional words) and need not be used. In the preceding example, NAME and IS are optional.
- Lower-case words are generic terms that must be replaced by appropriate names or values. In the preceding example schema-name must be replaced by the user-defined name for the schema.
- The meaning of enclosing a portion of a general format in special symbols is as follows:

  Optional. No selection needed.

```
[a]    at least no occurrences
[b]
[c]    at most one occurrence

Required. Must select one.

{a}    at least one occurrence
{b}
{c}    at most one occurrence

Required. Must select at least one.

||a||  at least one occurrence
||b||
||c||  at most one occurrence of each
```

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

Examples:

Format notation	Coding possibilities
[POINTERS]	blank or POINTERS
$\left\{ \begin{array}{l} \underline{\text{UPDATE}} \\ \underline{\text{RETRIEVAL}} \end{array} \right\}$	UPDATE or RETRIEVAL
$\left\Vert \begin{array}{l} \underline{\text{ACCEPT}} \\ \underline{\text{CONNECT}} \\ \underline{\text{DISCONNECT}} \end{array} \right\Vert$	ACCEPT or CONNECT or DISCONNECT or ACCEPT CONNECT or ACCEPT DISCONNECT or CONNECT DISCONNECT or ACCEPT CONNECT DISCONNECT
[area-name] ...	blank or AR-4 or AR-2 AR-6 AR-3
{record-name} ...	REC-2 or REC-4 REC-8

1.4 CHARACTER SET

The character set for the DDL consists of 52 characters that include letters of the alphabet, digits, and symbols. The specification of this character set defines those characters that may be used in writing a schema:

Character	Name
0, 1, ..., 9	digits
A, B, ..., Z	letters
	space or blank
+	plus sign
-	minus sign
,	comma
;	semicolon
.	period or decimal point
"	quotation mark
(	left parenthesis
)	right parenthesis
'	apostrophe
$	dollar sign
/	slash
*	asterisk
=	equal sign
>	greater than
<	less than

Additional characters, chosen from the complete DPS 7 character set, may be used in delimited strings (non-numeric literals, comments) or in the escape form of names.

1.4.1 Punctuation

There are two types of punctuation characters: delimiters and separators.
The end of a line is treated as a separator.

1.4.2 Delimiters

A delimiter is used to indicate the beginning and the end of a delimited string. A delimited string is a string of any allowable characters in the DPS 7 character set, preceded and followed by the delimiter. If a delimiter is to be included in a string then it must be written twice consecutively. The delimiters are:

- Quotation mark, which is used to delimit alphanumeric and hexadecimal literals and comments,

- Apostrophe, which is used to delimit the alternate form of names.

1.4.3 Separators

A separator is used to separate words and literals and to indicate certain language constructs. A separator is not significant if it occurs within a delimited string. It becomes part of the string if it is a character, it is ignored if it is an end of a line. The separators are:

- Space. One or more spaces acts as a separator. The space or spaces may appear in addition to (following or preceding) any other separator

- Comma. Comma is an obligatory separator between subscripts in the format of data base identifiers. It is optional elsewhere.

- Semi-colon. A semi-colon may be used as, and is treated as, a space.

- Full-stop. A full-stop followed by a space or the end of a line is required to end an entry or subentry.

- Pairs of left and right hand brackets. These are required to delimit a subscript of a data base identifier. Balanced left and right hand brackets can also be used in formulating conditions.

- End of line. End of line has the same rules and effect as a space.

1.5 WORDS

A word is a sequence of not more than 30 characters. Each character is selected from the set A to Z, 0 to 9, or -, except that the -(hyphen or minus) may not appear as the first or last character of a word. A word must begin with a letter.

Examples:

```
Valid           Invalid

A, A-B          -A
Z9999,          A$B, A+B
```

1.5.1 Reserved Words

Reserved words are a list of words that may be used, but must not appear as user-defined names. They include keywords and the optional words.

A keyword is a word that is required whenever the format in which the word appears is used. Within each format, such words are shown as uppercase and underlined.

An optional word is a word that may be included or not as the user prefers. Within each format, such words are shown as uppercase not underlined. Misspelling of an optional word or replacing it by another word is not allowed.

Table 1-1 gives a list of reserved words, with their abbreviations shown in parentheses.

Table 1-1. DDL Reserved Words

ACCESS-CONTROL	EXCLUSIVE (EXCL)	ORGANIZATION
ACTUAL	FIND	OWNER
AFTER	FIRST	PACKED
ALL	FIXED	PACKED-2
ALLOWED	FLOAT	PERMANENT
ALTER	FOR	PICTURE (PIC)
ALWAYS	GE	POSTPONED
AND	GET	PRIOR
ANY	GT	PROCEDURE (PROC)
APPLICATION	IDENTIFIED	PROCESSABLE
ARE	IN	PROTECTED (PROT)
AREA	INDEX	RANGE
AREA-ID	INDEXED	REAL
ASCENDING (ASC)	INSERT	RECORD
AUTOMATIC	INSERTION	RECORD-TYPE
BEFORE	IS	REMOVE
BINARY (BIN)	KEY	RESULT
BIT	KEYS	RETENTION
BY	LAST	RETRIEVAL (RETR)
CALC	LE	SCHEMA
CALC-KEY	LINKED	SEARCH
CALL	LOCATION (LOC)	SELECTION
CHARACTER (CHAR)	LOCK	SEQUENCE
CHECK	LOCKS	SEQUENTIAL
CLOSE	LT	SET
COMMENT	MANDATORY (MAND)	SIGNED
COMPLEX	MANUAL	SORTED
CONSTRAINT	MEMBER	SOURCE
COPY	MEMBERS	STORE
CURRENT	MEMBERSHIP	STRUCTURAL
DATA-BASE-KEY (DB-KEY)	MODE	SYSTEM
DECIMAL (DEC)	MODIFY	SYSTEM-DEFAULT
DECODING	NAME	TEMPORARY (TEMP)
DEFINED	NE	THEN
DELETE	NEXT	THIS
DESCENDING (DESC)	NON EXCLUSIVE	THRU (THROUGH)
DIRECT	NONNULL	TIMES
DISPLAY	NOT	TO
DUPLICATES (DUP)	NULL	TYPE
DURING	OCCURS	UNPACKED
DYNAMIC	OF	UNSIGNED
ENCODING	ON	UPDATE
END	ONLY	USING
END-SCHEMA	OPEN	VALUE
EQ	OPTIONAL (OPT)	VIA
EQUAL	OR	VIRTUAL
ERROR	ORDER	WHERE
		WITHIN

1.6 NAMES

A name is a word beginning with a letter. Types of names are:

```
data-base-data-name
record-name
area-name
set-name
schema-name
data-base-parameter
```

Names may be further grouped into:

```
entry-names = schema-names + area-names + record-names + set-names

data-base-names = entry-names + data-base-data-names
```

The following example illustrates a complete schema for a two-record data base. All the user-defined names are underlined (this is not the "format" notation as defined above and used throughout the body of this manual).

```
SCHEMA NAME IS SCHEMA-NAME.
AREA NAME IS AREA-NAME-A.
AREA NAME IS AREA-NAME-B.
RECORD NAME IS RECORD-NAME-A

    LOCATION MODE DIRECT DATA-BASE-PARAMETER-A

    WITHIN AREA-NAME-A.

    02 DATA-BASE-DATA-NAME-A TYPE CHARACTER 1.

RECORD NAME IS RECORD-NAME-B
    LOCATION MODE VIA SET-NAME-A

    WITHIN AREA-NAME-A AREA-NAME-B

     AREA-ID IS DATA-BASE-PARAMETER-B.

    02 DATA-BASE-DATA-NAME-B TYPE IS SIGNED BINARY 15.


SET NAME IS SET-NAME-A
    OWNER IS RECORD-NAME-A ORDER PERMANENT INSERTION LAST.


    MEMBER IS RECORD-NAME-B

    INSERTION AUTOMATIC RETENTION MANDATORY

    SET SELECTION IS THRU SET-NAME-A

    OWNER IDENTIFIED BY DATA-BASE-KEY

    EQUAL TO DATA-BASE-PARAMETER-C.
END-SCHEMA.
```

1.6.1 Alternate Form of Names

A name can have an alternate form, composed of a string of any allowable characters in the DPS 7 character set, delimited by apostrophes. The string length must not exceed 30 characters. The string may include the apostrophe symbol itself if the symbol is written twice consecutively for each of its occurrences.

Examples:

```
'WORK.SCHEMA', 'O''CLOCK'
```

When a user-defined name is identical to a reserved word, it must be written in the alternate form.

Example:

```
SCHEMA NAME IS 'SCHEMA'.
```

1.6.2 Uniqueness of Names

The following rules apply:

1. There must be no duplicates among entry-names. For example, two areas may not have the same name, nor may an area have the same name as the schema or a record or a set.

2. An entry-name must not be the same as a data-base-data-name.

3. There must be no duplicate data-base-data-names within a record entry.

4. A data item or aggregate within a record entry may have the same name as a data item or aggregate within a different record entry.

5. A data-base-parameter may not be the same as a data-base-name.

1.6.3 Maximum Number of Names

This is as follows:

Item	Maximum Number
Area-names	2048
Record-names	2048
Set-names	2048
Data-base-data-names within a record-type	2048
Data-base-parameters	2048

1.7 LITERALS

A literal is a string of characters representing a value. In IDS/II there are three types of literals:

- Alphanumeric literals

- Numeric literals

- Hexadecimal literals

1.7.1 Alphanumeric Literals

An alphanumeric literal represents a character string. It has a maximum of 256 characters chosen from the DPS 7 character set. It is delimited by quotation marks. The value is the string of character marks. The length of the string is the number of characters within the string. If a quotation mark is to appear in the string it must be written twice consecutively for each occurrence. It is counted once for the purposes of determining the length of the string.

Examples:

```
"JANUARY IST" value JANUARY IST length 11

"CODE ""4"""  value CODE "4"    length 8
```

1.7.2 Numeric Literals

A numeric literal represents a number. Only one form of numeric literal is allowed, the fixed point decimal literal. It must be:

- The digits 0 through 9
- Plus or minus sign
- The decimal point

It has a maximum of 30 digits. When the decimal point is present it must be followed by at least one digit. The value represented is the conventional algebraic value.

Examples:

```
Valid   1   1.0   +322  +0.004  -5789.333  -.004
Invalid 122.      +32.1A4
```

1.7.3 Hexadecimal Literals

A hexadecimal literal is a string of up to 512 characters chosen from:

- 0 through 9
- A through F

It is delimited by quotation marks and the final quotation mark is followed by the character X. The number of characters in the string must be even. The value represented is the EBCDIC equivalent of each pair of characters in the string. The length of the string is the number of characters divided by two. A hexadecimal string can be used to represent characters for which an external symbol does not exist or is not available on an input device.

Example:

```
"9481998388"X value:  march   length 5
```

1.8 COMMENTS

Comments can be included in DDL to document the schema coding. They can be coded wherever the space character can appear as a separator. A comment consists of the following:

either

- the reserved word COMMENT
- 0 to n spaces
- the text of the comment delimited by quotation marks

or

- the text of the comment delimited by and

A quotation mark must be written twice consecutively if it is to be included in a comment

Examples:

```
COMMENT "***********************
         *   INVENTORY SCHEMA  *
         ***********************"

SCHEMA NAME IS INVENTORY.

02  QTY TYPE DEC 6. QUANTITY ORDERED

02  SEX TYPE CHAR 1. COMMENT "VALUE ""M"" IF MALE

                              VALUE ""F"" IF FEMALE"
```

1.9 DATA-BASE-DATA-NAMES (OR FIELD-NAMES)

A data-base-data-name is a user-defined name for a data item or a data aggregate.

1.10 DATA-BASE-IDENTIFIERS (OR FIELD-IDENTIFIERS)

A data-base-identifier is a reference to a data item (stand-alone or part of an aggregate) declared in the schema. It consists of a data-base-data-name followed, as required, by the syntactically correct combination of subscripts and qualifiers necessary to achieve uniqueness of reference. In certain formats, the qualification is not necessary if the ambiguity is removed by the context (specific syntax rules).

The format of a data-base-identifier is:

```
data-base-data-name [(integer-1 [,integer-2] ...)] [{OF} record-name]
                    [          [           ]     ] [{IN}             ]
```

1.10.1 Subscripting

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

The lowest possible subscript value is 1. This value refers to the first occurrence of the data-base-data-name referenced. The highest possible subscript value is 32767.

When more than one subscript is required they are written left to right in the order of increasing data aggregate level numbers.

For example:

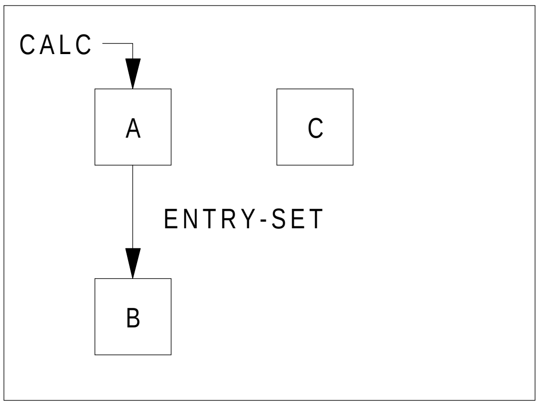
```
RECORD NAME IS R04

     ...
   02  R04-T1 OCCURS 3.
     03 R04-P1 TYPE IS CHAR 2.
     03 R04-P2 OCCURS 2 TYPE IS SIGNED PACKED DEC 3.
     ...
SET NAME IS S17
     ...
       INSERTION IS SORTED BY DEFINED KEYS ...
  MEMBER IS R04 ...
      KEY IS ASCENDING R04-P2 (1,2)
```

The number of subscripts is limited to 3, since this is the COBOL limitation.

1.10.2 Qualification

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

Example:



```
RECORD NAME IS A

     ...
   02 MY-DB-ID TYPE IS CHAR 5.
RECORD NAME IS C
     ...
   02 MY-DB-ID TYPE IS CHAR 5.
SET ENTRY-SET OWNER A... MEMBER B...
      SET SELECTION FOR ENTRY-SET IS THRU ENTRY-SET
         OWNER IDENTIFIED BY CALC-KEY MY-DB-ID EQUAL TO
              MY-DB-ID IN C
```

NOTE: The CALC data-base-data-name item is usually located in the User Work Area (UWA) assigned to record A, but in this example a UWA assigned to a data item in another record is used as an alternate (homonym) source.

1.11 DATA-BASE-PARAMETERS

A data-base-parameter is a user-defined name available at execution time to augment data-base-data-names. The data-base-parameters are not held in records in the data base but are used for record location (DIRECT), area selection (AREA-ID), or set selection (EQUAL TO). They take on the characteristics of the usage specified. That is, the parameter is a data-base-key when used with the DIRECT location clause, it is an area name when used in an area selection clause (WITHIN), and it will be the same as the control field when specified in the EQUAL TO phrase.

1.12 CONDITIONS

In general formats the word condition is used to signify an expression evaluated at run time by IDS/II as either true or false.

- In its general form a condition consists of one or more alternatives separated by the keyword OR. The form is:

```
 ____________________________________________
|                                            |
|   alternative-1 [OR alternative-2]...      |
|____________________________________________|
```

The condition is true if any of the alternatives (and possibly more than one, the OR is not exclusive) is true, otherwise it is false.

- An alternative consists of one or more simple conditions separated by the keyword AND. The form is:

```
 ____________________________________________________
|                                                    |
| simple-condition-1 [AND simple-condition-2]...     |
|____________________________________________________|
```

The alternative is true if all the simple conditions are true, otherwise it is false.

- A simple condition has the form:

```
{relation-condition-1       }
{NOT (relation-condition-2) }
{NOT (condition-1)          }
```

The simple condition is true if:

- Relation-condition-1 is true
- Relation-condition-2 is false
- NOT is omitted and condition-1 is true
- NOT is specified and condition-1 is false

otherwise the simple condition is false.

A relation condition consists of a pair of operands which can be data-base-identifiers or literals separated by a relational operator. A relation condition has the format:

```
                          {LT}
                          {< }
                          {  }
                          {LE}
                          {<=}
                          {  }
{data-base-identifier  }  {EQ}  {data-base-identifier-2}
{                      }  {= }  {                      }
{literal-1             }  {  }  {literal-2             }
                          {GE}
                          {>=}
                          {  }
                          {GT}
                          {> }
                          {  }
                          {NE}
```

The relation-condition is true if the relation holds between the operands, otherwise it is false. The comparison of two data-base-identifiers is permitted in the cases illustrated by Table 1.2 (YES at the intersection).

The comparison of a data-base-identifier and a literal is permitted in the cases illustrated by Table 1.3. If the operands are numeric, there must be no loss of non-null digits (to the left of the integral part and to the right of the fractional part) when the literal is converted to the format of the data-base-identifier.

If the operands are string data items the comparison is performed according to the EBCDIC collating sequence. If the operands are numeric data items, the comparison is performed using the algebraic value of the items. The meaning of the operators is as follows:

```
LT or   <    less than
LE or   <=   less than or equal to
EQ or   =    equal to
GE or   >=   greater than or equal to
GT or   >    greater than
NE      ><   not equal to
```

The operators < <= = >= > must always be preceded by and followed by a space. The two characters that make up the operators >= and <= must be written on the same line without a space.

1.12.1 Use of Brackets

For a condition without brackets, each alternative is evaluated and the truth value of the alternative is used to determine the truth value of the complete condition.

For conditions with brackets, the expressions within brackets are evaluated first and the truth value of the whole bracketed expression is used to determine the truth value of the complete condition. Where brackets are nested, the expression in the innermost set of brackets is evaluated first followed by the next innermost, etc.

If brackets are used, they must always balance.

1.12.2 Examples of Conditions

```
DISCOUNT-RATE = 0.10

AGE >= 25

NOT(AGE < 25)

BIRTH-DATE LE DEATH-DATE

MONTH = "JANUARY" AND DAY <= 31 OR MONTH = "FEBRUARY" AND DAY <=
29

SEX = "M" AND MAIDEN-NAME = " "
OR SEX = "F" AND NOT (MAIDEN-NAME = " ")

RTYPE = "1" AND (SUBTYPE = "1" OR SUBTYPE = "2") OR
RTYPE = "2" AND ((SUBTYPE = "1" AND QTY < 1000) OR
                (SUBTYPE = "2" AND QTY >= 1000))

OR RTYPE = "3"
```

Table 1-2. Comparison of Two Data-Base-Identifiers

Identifier-1 TYPE / Identifier-2 TYPE	DECIMAL m	DECIMAL m,+-p (p<>0)	BINARY	CHARACTER
DECIMAL m	YES		YES	
DECIMAL m,+-p (p<>0)		YES (if same scale)		
BINARY	YES		YES	
CHARACTER				YES

Table 1-3. Comparison of a Data-Base-Identifier and a Literal

Identifier TYPE / Literal TYPE	DECIMAL m,+-p	BINARY	CHARACTER
signed/unsigned integral/fraction decimal	YES	YES	
character string			YES
hexadecimal string			YES

1.13 SCHEMA DDL SYNTAX

A schema is divided into the sections or types of entries:

Schema	names the schema.
Area	names each area in the data base.
Record	names and defines each record type in the data base, specifies data items and storage location method.
Set	names and defines each relationship between records.

Each entry/subentry is terminated with a period followed by a space or an end of line. The entire DDL source text is terminated by an END-SCHEMA statement. The complete DDL for a schema may be summarized as:

```
   Schema Entry

  {Area Entry}...

  {Record Entry}...

  [Set Entry]...
```

Each of the entries will be describe in detail later.

```
COMMENT "****************************************************************"
COMMENT "*            SCHEMA DDL FOR TEST-GROUP *TG19*                 *"
COMMENT "****************************************************************"

COMMENT "********************************"
COMMENT "*            SCHEMA            *"
COMMENT "********************************"

SCHEMA TG19-SH.

COMMENT "********************************"
COMMENT **            AREAS             *"
COMMENT "********************************"

AREA TG19-A00.
AREA TG19-A01.

COMMENT "********************************"
COMMENT "*            RECORDS           *"
COMMENT "********************************"
```

Figure 1-1. Sample DDL Source Schema (1/2)

```
COMMENT "***************************"
COMMENT "*      R01 (PART)         *"
COMMENT "***************************"
RECORD R01
     LOCATION CALC USING R01-NUM DUP NOT
     WITHIN ANY AREA AREA-ID ARID1.
  02      R01-IDENT     TYPE CHAR 7.
  02      R01-NUMH      TYPE CHAR 5.
  02      R01-NUM       TYPE UNPACKED DEC 3.
  02      R01-TRAILER   TYPE CHAR 4.

COMMENT "***************************"
COMMENT "*      R02 (COMPONENT)    *"
COMMENT "***************************"
RECORD R02
     LOCATION VIA S01
     WITHIN AREA OF OWNER.
  02      R02-IDENT     TYPE CHAR 7.
  02      R02-R01-UPH   TYPE CHAR 4.
  02      R02-R01-UP    TYPE UNPACKED DEC 3.
  02      R02-R01-DOWNH TYPE CHAR 6.
  02      R02-R01-DOWN  TYPE UNPACKED DEC 3.
  02      R02-QTYH      TYPE CHAR 5.
  02      R02-QTY       TYPE UNPACKED DEC 3.
  02      R02-TRAILER   TYPE CHAR 4.

COMMENT "***********************************"
COMMENT "*             SETS                *"
COMMENT "***********************************"

COMMENT "***************************"
COMMENT "*     S01  (CALL-OUT)     *"
COMMENT "***************************"
SET S01
     OWNER R01
     ORDER PERMANENT INSERTION SORTED DEFINED KEYS DUP NOT.
  MEMBER R02
     INSERTION AUTO RETENTION MAND
     KEY ASCENDING R02-R01-DOWN
     SELECTION THRU S01
           OWNER IDENTIFIED BY CALC-KEY R01-NUM EQUAL TO R02-R01-UP.

COMMENT "***************************"
COMMENT "*     S02 (WHERE-USED)    *"
COMMENT "***************************"
SET S02
     OWNER RO1
     ORDER PERMANENT INSERTION SORTED DEFINED KEYS DUP NOT.
  MEMBER R02
     INSERTION AUTO RETENTION MAND
     KEY ASCENDING R02-R01-UP
     SELECTION THRU S02
           OWNER IDENTIFIED BY CALC-KEY R01-NUM EQUAL TO R02-R01-DOWN
                              AREA-ID EQUAL TO OTHER-ARID1.

END-SCHEMA.
```

Figure 1-1. Sample DDL Source Schema (2/2)

1.13.1 SCHEMA Entry

Function

To name the schema. This name is entered into the dictionary of the schema file and is used in all further references to the schema. This clause is required.

General Format

```
 ________________________________
|                                |
| SCHEMA NAME IS schema-name.    |
|________________________________|
```

Syntax Rules

1. Schema-name is a name of up to 30 characters in length.

2. This clause must appear first in the schema text.

3. Schema-name must be unique among the data-base-names of the schema.

General Rule

The schema named by schema-name consists of all the DDL entries that appear after the SCHEMA clause and before the END-SCHEMA entry.

1.13.2 AREA Entry

Function

To name an area within the data base.

General Format

```
 ____________________________
|                            |
| AREA NAME IS area-name.    |
|____________________________|
```

Syntax Rules

1. Area-name must be unique among data-base-names of the schema.

1.13.3 RECORD Entry

Function

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

General Format of Record Entry

```
 ______________________________
|                              |
|   record subentry            |
|                              |
|  [data subentry] ...         |
|______________________________|
```

General Format of Record Subentry

```
RECORD NAME IS record-name

                  {DIRECT data-base-parameter-1           }
                  {                                       }
LOCATION MODE IS  {CALC USING {data-base-identifier-1} ... }
                  {     DUPLICATES ARE [NOT] ALLOWED      }
                  {                                       }
                  {VIA set-name-1 SET                     }
                  {                                       }
                  {                                       }
                  {{ ANY AREA         }   AREA-ID IS data-base-parameter-2   }
WITHIN            {{ {area-name-1} ...}                                      }
                  {                                                          }
                  {AREA OF OWNER                                             }
[CHECK IS  condition-1] ...
```

General Format of Data Subentry

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

[          { {              {UNPACKED} }                              }]
[          { { [UNSIGNED]  {[PACKED]} }                              }]
[          { {              {PACKED-2} }                              }]
[          { {                         }  DECIMAL integer-1 [integer-2]     }]
[TYPE IS   { {SIGNED {UNPACKED}        }                              }]
[          { {       {[PACKED]}        }                              }]
[          {                  {15}                                    }]
[          { {SIGNED] BINARY {31}                                     }]
[          {                                                          }]
[          {     CHARACTER integer-3                                  }]

[OCCURS integer-4 TIMES]
[CHECK IS VALUE [NOT] {literal-1 [THRU literal-2 ] } ...] .
```

1.13.3.1 RECORD Clause

Function

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

General Format

```
 _______________________________
|                               |
| RECORD NAME IS record-name    |
|_______________________________|
```

Syntax Rules

1. Record-name must be unique among the data-base-names given by a schema.

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

General Rules

None.

1.13.3.2 LOCATION Clause

Function

To control the assignment of data-base-keys to the record.

General Format

```
                  {DIRECT data-base-parameter-1            }
                  {                                        }
                  {CALC USING {data-base-identifier-1} ... }
LOCATION MODE IS  {     DUPLICATES ARE [NOT] ALLOWED       }
                  {                                        }
                  {VIA set-name-1 SET                      }
```

Syntax Rules

1. Data-base-parameter-1 refers to a data item generated by the COBOL compiler as a data-base-key item. The item is implicitly defined by its appearance in the clause.

2. Data-base-identifier-1..., must refer to data items included in the record-type being described (CALC).

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

General Rules

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

2. The DBCS assigns a data-base-key to a record when a STORE is executed. This may also happen when a MODIFY CALC-key entails a record migration.

3. The assignment of data-base-keys is subject to the overall constraint of the WITHIN clause for the same record subentry.

4. If DIRECT is specified, the contents at execution time of the data item associated with the DIRECT phrase are used in assigning a data-base-key. The contents may be a complete data-base-key that is relative to the entire data base, or an area-key that is relative to only one area. In any case, only the area-key part is used by the DBCS.

5. If CALC is specified, the contents at execution time of the data items associated with the CALC phrase are used in assigning a data-base-key. The data-base-key is developped using a system-supplied algorithm.

6. The DUPLICATES ARE NOT ALLOWED phrase instructs the DBCS to permit no more than one record of this type with identical values for the data items associated with the CALC phrase to be stored in an area of the data base.

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

1.13.3.3 WITHIN Clause

Function

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

General Format

```
        { { ANY AREA          } [AREA-ID IS data-base-parameter-1]}
        { { {area-name-1} ...}                                    }
WITHIN  {                                                         }
        {AREA OF OWNER                                            }
```

Syntax Rules

1. The area names must be the names of areas for which an area entry is included in the schema.

2. If OWNER is specified, the LOCATION mode of the record must be VIA set-name.

General Rules

1. By its appearance in a WITHIN clause, data-base-parameter-1 is implicitly defined to be a data item that contains a character string that conforms to the rules for the formation of area-names (alphanumeric 30 characters long).

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

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

4. If ANY AREA is specified then all declared areas are valid for the placement of this record-type.

5. If ANY AREA is specified and implies several areas or if a list of more than one area-name is specified then AREA-ID must be present.

6. The contents of data-base-parameter-1, which must be one of the area-names specified in the WITHIN clause or any valid area-name if ANY AREA is declared, must be set before the object record can be stored.

Example:

```
Schema:  RECORD A ...
         WITHIN AREA-1, AREA-2, AREA-3, AREA-ID IS
         PARM-1.

Program: MOVE "AREA-3" TO PARM-1.
         STORE A.
```

At run time, the program makes the selection of which area will contain this record. Before it is stored, the DBCS checks the contents of data item PARM-1 for a valid area-name. If the area-name is invalid then an error of "INVALID REALM" will be returned. Once an area is specified that is contained in this WITHIN clause, then standard key range and location mode considerations apply.

To retrieve a CALC record by a "FIND ANY" statement, the proper area-name must be initialized in the AREA-ID field of PARM-1.

7. If AREA OF OWNER is declared then the record will be stored in the same area as its owner record, as determined by the set selection criteria of the set specified in the VIA LOCATION mode.

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

NOTE: It is recommended that each AREA-ID data-base-parameter be uniquely named. This avoids conflicts in the use of AREA-ID's when set selection is performed.

1.13.3.4 CHECK Clause (Record Subentry)

Function

To specify a validity check on data item values of a record.

General Format

```
 ___________________________
|                           |
| CHECK IS condition-1      |
  -----
```

```
|________________________|
```

Syntax Rules

1. All the data-base-identifiers used as arguments of condition-1 must be data items specified in the record entry.

General Rules

1. The condition is tested whenever a record of this type is stored or any of the arguments of the condition is to be changed by a MODIFY statement.

2. If the condition is found to be false, the DBCS will report an error and the data base will not be changed.

1.13.3.5 Data-base-data-name Clause

Function

To name a data item, a vector or a repeating group or a non-repeating group and indicate its structural level within the record.

General Format

```
 ______________________________________
|                                      |
|  [level-number] data-base-data-name  |
|______________________________________|
```

Syntax Rules

1. Data-base-data-name must be unique among the data-base-data-names declared for this record, and may not be an entry-name.

2. Level-number is an unsigned decimal integer greater than zero and less than 100. Default is 01.

General Rules

1. The content of a record is defined by a series of zero or more data subentries. When a record contains no data subentry it is used in a structural manner to assist in accessing other records ("root" or "relation " records).

2. A data item is described by a data subentry containing a TYPE clause but no OCCURS clause.

3. A vector is described by a data subentry containing a TYPE clause and an OCCURS clause.

4. A repeating group is described by a data subentry containing an OCCURS clause but no TYPE clause.

5. A non-repeating group is described by a data subentry which does not contain a TYPE clause or an OCCURS clause.

6. If a data subentry defines a repeating or non-repeating group, it must be followed by one or more data subentries with higher value level numbers.

1.13.3.6 TYPE Clause

Function

To describe the characteristics of a data item (stand-alone or element ofa vector or repeating group or non-repeating group).

General Format

```
          { {                {UNPACKED}    }                                  }
          { { [UNSIGNED]     {[PACKED]}    }                                  }
TYPE IS   { {                {PACKED-2}    } DECIMAL integer-1 [ integer-2 ]  }
          { {                              }                                  }
          { {SIGNED          {UNPACKED}    }                                  }
          { {                {[PACKED]}    }                                  }
          {                                                                   }
          {    [SIGNED]       BINARY  {15}                                    }
          {                           {31}                                    }
          {                                                                   }
          {    CHARACTER  integer-3                                           }

[OCCURS  integer-4  TIMES]
[CHECK IS VALUE [NOT] {literal-1 [THRU  literal-2] } ...] .
```

Syntax Rules

1. Integer-1 and integer-3 must be unsigned decimal constants with values greater than zero. The maximum value varies by data type and is specified by the General Rules.

2. Integer-2 is a signed decimal constant.

General Rules

1. The TYPE clause defines a numeric data item (decimal or binary) or a string data item.

2. DECIMAL specifies a numeric data item with a decimal base. DECIMAL data items correspond to COBOL PIC 9 items.

3. Integer-1 specifies the number of significant digits to be maintained for all values of a data item. The maximum value of integer-1 is 30.

4. Integer-2, when specified, indicates the scale of a DECIMAL data item. Integer-2 can be any value in the range -29 to 30. When integer-2 is negative, the decimal point is positioned integer-2 places to the right of the rightmost actual digit.

(DECIMAL 4, -1 means PICTURE 9999P in COBOL-74 terms.) When integer-2 is positive, the decimal point is positioned integer-2 places to the left of the rightmost actual digit. (DECIMAL 4, 1 means PICTURE 999V9 in COBOL-74 terms.) When integer-2 is zero, or when not specified, the decimal point is positioned immediately to the right of the righmost digit.

5. Integer-1 and integer-2 must be such that total number of digits (if the data item is displayed) does not exceed 30.

6. The keywords SIGNED/UNSIGNED PACKED/PACKED-2/UNPACKED specify the internal format of the data item. Their COBOL equivalents are shown in Table 1-4.

7. BINARY specifies a numeric item with a binary base. BINARY items correspond to COBOL COMP-1 and COMP-2. They contain signed integer values in 2-byte (15 bits plus sign bit) or 4-byte (31 bits plus sign bit) form.

8. CHARACTER specifies a string data item. CHARACTER items correspond to the COBOL PIC X family. A character data item is assumed to contain characters from the EBCDIC character set, and any comparisons are made using the EBCDIC collating sequence.

9. Integer-3 gives the length of the string data item. The maximum value of integer-3 is theoretically 65535 but the record size cannot exceed the access area page size.

10. Table 1-4 summarizes the correspondence between the TYPE clause definition and the equivalent in COBOL-74.

11. All data items are aligned on a byte boundary when they appear in the UWA record description of a DML program.

Example:

```
RECORD NAME IS data-base-record-name-1 .....

 02 Display-item-1 TYPE IS CHARACTER 3.
 02 Decimal-item-1 TYPE IS SIGNED PACKED DECIMAL 5, +2.
 02 Binary-item-1 TYPE IS SIGNED BINARY 31.
```

NOTES:

1. Display-item-1 will occupy three 8-bit EBCDIC bytes in the record.

2. Decimal-item-1 will occupy five packed decimal digit positions plus an extra sign for a total of six positions, or three 8-bit bytes. The implied decimal point is two places to the left of the last digit, or in COBOL-74 terms:

   ```
   PIC S9(3)V99 USAGE COMP.
   ```

3. Binary-item-1 will occupy four 8-bit bytes starting on the next byte boundary.

Table 1-4. DDL and COBOL Data Types

DDL TYPE			COBOL TYPE	
			PICTURE	USAGE
SIGNED UNPACKED DECIMAL	m		s9 (m)	
	m,p	m>p	s9 (m-p) V9 (p)	
		m=p	sV9 (m)	
		m<p	SP (p-m) 9 (m)	
	m,-p		s9 (m) P (p)	
SIGNED PACKED DECIMAL	{ m m,p m,-p }		same as SIGNED UNPACKED	COMP
UNSIGNED UNPACKED DECIMAL	m		9 (m)	
	m,p	m>p	9 (m-p) V9 (p)	
		m=p	V9 (m)	
		m<p	p (p-m) 9 (m)	
	m,-p		9 (m) P (p)	
UNSIGNED PACKED DECIMAL	{ m m,p m,-p }		same as UNSIGNED UNPACKED	COMP
UNSIGNED PACKED-2DECIMAL	{ m m,p m,-p }		same as UNSIGNED UNPACKED	COMP-8
SIGNED BINARY 15				COMP-1
SIGNED BINARY 31				COMP-2
CHARACTER n			X (n)	

1.13.3.7 OCCURS Clause

Function

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

General Format

```
 ______________________________
|                              |
| OCCURS integer TIMES         |
|                              |
 ______________________________
```

Rules

1. The number of times the group component is repeated is specified by the value of integer, which must be greater than 1.

2. This clause may be supplied to define a vector, in conjunction with a TYPE clause:

   ```
   02 EDX TYPE SIGNED BINARY 31 OCCURS 10.
   ```

   or, it may be used to define a repeating group, if the TYPE clause is omitted:

   ```
   02 EDAB OCCURS 18.

        03 EDA TYPE CHARACTER 20.
        03 EDB OCCURS 8.

           04 EDBX ...
           04 EDBY ...
           04 EDBZ ...
   ```

   This shows both EDAB and EDB as group identifiers.

1.13.3.8 CHECK Clause (Data Subentry)

Function

To specify validity checking that is to be executed when a value is changed or when new data is being stored.

General Format

```
 ____________________________________________________________
|                                                            |
| CHECK IS VALUE [NOT] {literal-1 [THRU literal-2]} ...      |
|____________________________________________________________|
```

Syntax Rules

1. The data subentry containing this CHECK clause must also contain a TYPE clause.

2. The type of literal-1 and literal-2 must match the type of the data subentry as specified in Table 1-3.

3. For string data the literals must be in ascending order as defined by the EBCDIC collating sequence. For numeric data the literals must be in ascending algebraic order.

General Rules

1. The value of the data item is compared with the individual values or ranges specified in this clause. An individual value is specified by a literal. A range is specified by two literals separated by THRU.

2. The value of an item satisfies a comparison if it is equal to the literal, in the case of a single literal. In the case of a range, the value of the item must be greater than or equal to the literal specifying the low end of the range and less than or equal to the literal specifying the high end of the range.

3. The value is valid if it satisfies a comparison and the NOT is omitted or if it fails and the NOT is specified.

4. The specified validity checking occurs whenever a new value of the item is to be stored on the data base (STORE statement) or if the value is to be modified (MODIFY statement).

5. If an invalid value is detected, an error will be reported and the data base will not be changed.

CHECK Clause Examples

```
CHECK IS VALUE "1"

CHECK IS VALUE NOT 1000

CHECK IS VALUE "A" "B" "C"

CHECK IS VALUE 20 THRU 30 50 THRU 60

CHECK IS VALUE NOT 0 THRU 200
```

1.13.4 SET Entry

Function

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

General Format of Set Entry

```
  ________________________
 |                        |
 | Set subentry           |
 |                        |
 | {member subentry} ...  |
 |________________________|
```

General Format of Set Subentry

```
SET NAME IS set-name

OWNER IS record-name-1

ORDER IS PERMANENT INSERTION IS

{FIRST                                                   }
{LAST                                                    }
{NEXT                                                    }
{PRIOR                                                   }
{                                                        }
{                                                        } .
{         {WITHIN RECORD-TYPE                          } }
{         {BY DEFINED KEYS                             } }
{SORTED   {   [RECORD-TYPE SEQUENCE IS {record-name-2} ...] } }
{                                                        }
{         {                  {FIRST      }             } }
{         {  DUPLICATES ARE {LAST       }             } }
{         {                  {NOT ALLOWED}             } }
```

General Format of Member Subentry

```
MEMBER IS record-name-3

             {AUTOMATIC RETENTION IS MANDATORY}
INSERTION IS {                                }
             {MANUAL RETENTION IS OPTIONAL    }

[DUPLICATES ARE NOT ALLOWED FOR {data-base-identifier-1} ...] ...

[         {ASCENDING }    {data-base-identifier-2}                ]
[KEY IS  {DESCENDING}    {RECORD-TYPE          }                ]
[                         {DATA-BASE-KEY          }               ]
[                                                                  ]
[       [ [ASCENDING ]    {data-base-identifier-3}]               ]
[       [ [DESCENDING]    {RECORD-TYPE            }] ...          ]
[                         {DATA-BASE-KEY          }]              ]
[                                                                  ]
[         [                 {FIRST      }]                         ]
[         [  DUPLICATES ARE {LAST       }]                         ]
[         [                 {NOT ALLOWED}]                         ]

SET SELECTION [FOR set-name-1] IS

THRU set-name-2 OWNER IDENTIFIED BY

{APPLICATION                                       }
{                                                  }
{DATA-BASE-KEY [EQUAL TO data-base-parameter-1]   }
{                                                  }
{CALC-KEY [data-base-identifier-4 EQUAL TO ]      }
{         [                                ]      }
{         [ {data-base-identifier-5}       ]      }
{         [ {                       }      ] ...  }
{         [ {data-base-parameter-2 }       ]      }
{                                                  }
{        [AREA-ID EQUAL TO data base-parameter-3] }

THEN THRU set-name-3 WHERE OWNER IDENTIFIED BY

[{                       [         {data-base-identifier-7}]}    ]
[{data-base-identifier-6 [EQUAL TO {data-base-parameter-4 }]} ...] ... .
```

1.13.4.1 SET Clause

Function

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

General Format

```
 ________________________________
|                                |
| SET NAME IS set-name-1         |
|________________________________|
```

Syntax rule

Set-name-1 must be unique among the data-base-names known to the schema.

General Rules

None.

1.13.4.2 OWNER Clause

Function

To specify the name of a record-type, each occurrence of whichestablishes the existence of an occurrence of the set-type named in this set entry.

General Format

```
 ___________________________
|                           |
| OWNER IS record-name-1    |
|___________________________|
```

Syntax Rules

1. Record-name-1 must have been previously declared in a record entry.

2. Record-name-1 must be different from the member record types of this set entry.

3. A record-type may be specified as an owner in more than one set entry. It may also be defined as a member in one or more set entries.

General Rules

1. The OWNER clause specifies the record-type that is the owner in the set-type being described. Each occurrence of the owner record-type entails an occurrence of the set-type being described.

2. A record can only own one occurrence of a given set-type.

1.13.4.3 ORDER Clause

Function

To specify the insertion point of a member record within a set and thereby define the order of sequential progression.

General Format

```
ORDER IS PERMANENT INSERTION IS

{FIRST                                                     }
{LAST                                                      }
{NEXT                                                      }
{PRIOR                                                     }
{                                                          }
{        {WITHIN RECORD-TYPE                            }  }
{        {                                              }  }
{        {BY DEFINED KEYS                               }  }
{SORTED  {                                              }  }
{        { [RECORD-TYPE SEQUENCE IS {record-name-2} ... ]  }  }
{        {                                              }  }
{        {                    {FIRST      }             }  }
{        {     DUPLICATES ARE {LAST       }             }  }
{        {                    {NOT ALLOWED}             }  }
```

Syntax Rules

1. If the SORTED WITHIN RECORD-TYPE phrase is specified, then a KEY clause must be stated in at least one member subentry for this set-type.

2. If the SORTED BY DEFINED KEYS phrase is specified, then a KEY clause must be stated for each member subentry for this set-type.

General Rules

1. ORDER FIRST refers to the position within the set that immediately follows the owner record. This is a reversed chronological sequencing. The most recent member record inserted into the set becomes the first member of the set.

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

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

4. The SORTED phrase allows specification of a set order based on the record-type values or the data-base-keys of the member records of the set, or on the values of data items specified in the KEY clauses for the member records of the set.

5. The WITHIN RECORD-TYPE phrase allows records to be sorted without regard to the order of other record-types in the set. This does not mean that there is an implied major sort by record-type. It means only that for a given type, it is in sequence by its own sort-control-key. The sort-control-keys are specified by the KEY clause for each of the member record-types that are to be sorted.

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

7. The optional RECORD-TYPE SEQUENCE IS phrase allows the declaration of the order-by-record-type sequence so that the record-type can be declared as a key item in the KEY clause for member records of the set.

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

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

1.13.4.4 MEMBER Clause

Function

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

General format

```
 ____________________________
|                            |
| MEMBER IS record-name-3    |
|____________________________|
```

Syntax Rules

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

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

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

4. Record-name-3 must be different from the set owner record-type, as defined in the set OWNER clause.

General Rules

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

1.13.4.5 INSERTION RETENTION Clause

Function

To specify the type of membership within the set-type.

General Format

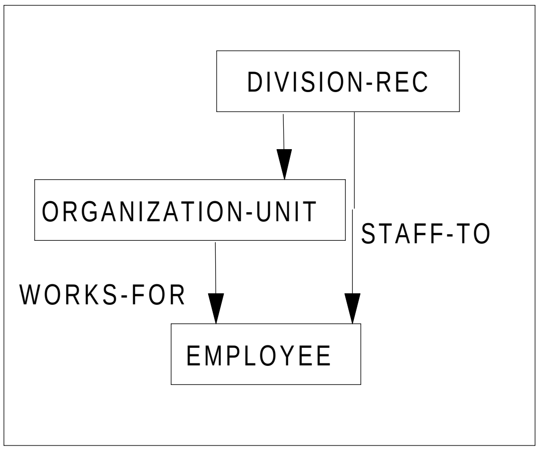
```
 ___________________________________________________________
|                                                           |
| INSERTION IS  { AUTOMATIC RETENTION  IS MANDATORY  }      |
|               { MANUAL    RETENTION  IS OPTIONAL   }      |
|___________________________________________________________|
```

General Rules

1. If AUTOMATIC MANDATORY is used, then an occurrence of the member record is inserted into (made a current member of) the selected occurrence of the set type named in this set entry when the record is added to the data base. It will always be a member of one or another set occurrence of that set-type. The record may be switched between set occurrences of the same set type only by a MODIFY function.

2. If MANUAL OPTIONAL is used, adding an occurrence of the member record-1 to the data base does not cause that record to become a current member of any occurrence of the set-type named in this set entry. Membership in the set is established by a run-unit by means of the DML CONNECT function. Membership can also be cancelled by means of the DML DISCONNECT function. If the location mode is VIA for this record, the record is physically placed near the owner occurrence of the set named in the VIA set-name phrase. After the record has been stored, it may be inserted into the appropriate MANUAL OPTIONAL sets.

Example:



```
RECORD NAME IS EMPLOYEE
 LOCATION MODE IS VIA WORKS-FOR
 SET
...
SET NAME IS STAFF-TO
 OWNER IS DIVISION-REC ...

MEMBER IS EMPLOYEE
 INSERTION MANUAL
      RETENTION OPTIONAL ..
      ...
SET NAME IS WORKS-FOR
 OWNER IS ORGANIZATION-UNIT ...
 MEMBER IS EMPLOYEE
 INSERTION AUTOMATIC RETENTION
          MANDATORY ...
```

In this example, when an employee record is stored, it is automatically inserted into the WORKS-FOR set, and also placed near to the owning ORGANIZATION-UNIT record. If the employee has a staff function with the corporate division, then a CONNECT function can be used to link this employee into the optional STAFF-TO set.

1.13.4.6 DUPLICATES Clause

Function

To check and reject the insertion within the same set occurrence of those member records that have duplicate values for specified data items.

General Format

```
 ______________________________________________________________
|                                                              |
| DUPLICATES ARE NOT ALLOWED FOR { data-base identifier-1}...  |
|______________________________________________________________|
```

Syntax Rules

1. More than one DUPLICATES clause, corresponding to various groups of fields, may be specified in the member subentry.

2. The data-base-identifiers must refer to data items included in the record-type named in this member subentry.

General Rules

1. If the DUPLICATES ARE NOT ALLOWED clause is used, the DBCS rejects the insertion into any given set of a member record whose values, for all data items specified in this clause, duplicate the values contained in an existing member of the set. This may occur during an attempt to store a new record, or to insert an existing record into a set or to modify the value of such data to cause a duplication. The entire set is searched to satisfy this criterion before the ORDER INSERTION clause requirements are processed. In particular, if a field is specified and the INSERTION mode is NEXT, the whole length of the set is checked before the new record is inserted next to the current record in the set.

1.13.4.7 KEY Clause

Function

To specify the sort-control-key for a member record of a sorted set.
A key item may be a data item within the record, or the record-type, or the data-base-key of the record.

General Format

```
        {ASCENDING }  {data-base-identifier-2}
KEY IS  {          }  {RECORD-TYPE           }
        {DESCENDING}  {DATA-BASE-KEY         }

  [               {data-base-identifier-3    } ]
  [[ASCENDING  ]  {RECORD-TYPE               } ] ...
  [[DESCENDING ]  {DATA-BASE-KEY             } ]

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

Syntax Rules

1. The KEY clause must be specified in all member subentries of any set entry that includes the SORTED BY DEFINED KEYS phrase in the ORDER clause.

   It must be specified in at least one member subentry of any set entry that includes the SORTED WITHIN RECORD-TYPE phrase in the ORDER clause.

   The KEY clause must not be specified if the set entry does not include the SORTED phrase in the ORDER clause.

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

3. If RECORD-TYPE is a key item, it must be specified once only and the phrase RECORD-TYPE SEQUENCE IS ... must be present in the ORDER clause of the set entry. All the record-names specified must be members of the set. All the member records of the set must be declared in the RECORD-TYPE SEQUENCE phrase.

4. If DATA-BASE-KEY is a key item, it must be specified once only and as the last key item. The relevant DUPLICATES phrase, whether in the ORDER clause or KEY clause, must specify "NOT ALLOWED".

5. If the set entry includes the SORTED BY DEFINED KEYS phrase in the ORDER clause, corresponding data items must be specified in the KEY clauses of all member record-types. The corresponding data items must have identical data characteristics and must also match in terms of whether ASCENDING or DESCENDING is specified for them. Two data items to which identical TYPE clauses apply have identical data characteristics. If RECORD-TYPE is a key item, the rules of one-to-one correspondence described previously apply only to the data items given to the left of RECORD-TYPE.

6. The DUPLICATES phrase must be specified only if the ORDER clause in the set entry contains the SORTED WITHIN RECORD-TYPE phrase.

General Rules

1. The data-base-identifiers are the key items that together constitute the sort-control-key for the member record-type named in the MEMBER clause of this entry. The key items are stated in the KEY clause in order of decreasing significance; that is, data-base-identifier-2 is the major key item and data-base-identifier-3,... are minor key items.

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

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

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

1.13.4.8 SELECTION Clause

Function

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

General Format

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

THRU set-name-2 OWNER IDENTIFIED BY

{APPLICATION                                          }
{                                                     }
{DATA-BASE-KEY [EQUAL TO data-base-parameter-1]       }
{                                                     }
{CALC-KEY [data-base-identifier-4            ]        }
{         [EQUAL TO {data-base-identifier-5} ] ...    }
{         [         {data-base-parameter-2 } ]        }
{                                                     }
{             [AREA-ID EQUAL TO data base-parameter-3] }

[THEN THRU set-name-3 WHERE OWNER IDENTIFIED BY                   ]
[                                                                 ]
[ {data-base-identifier-6 [EQUAL TO {data-base-identifier-7}]}    ]
[ {                       [         {data-base-parameter-4 }]} ... ] ...
```

Syntax Rules

1. Set-name-1 is the name of the set-type of this set entry.

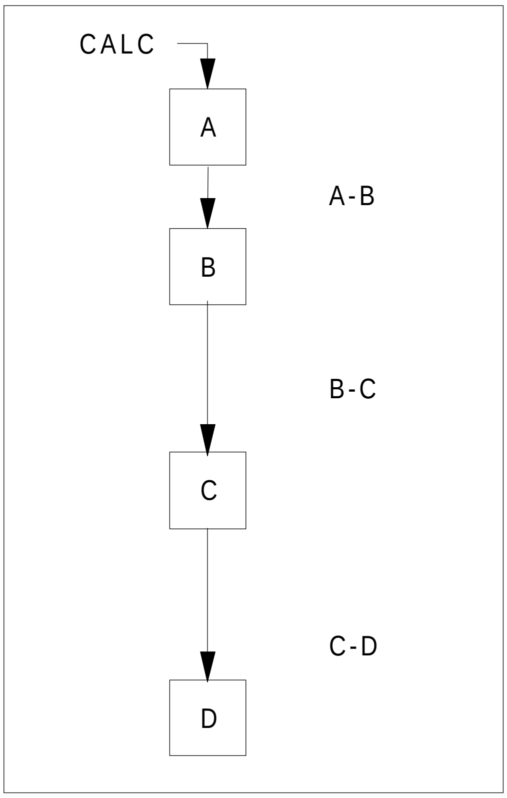
2. If the CALC-KEY option is specified, the owner record type of set-name-2 must have a location mode of CALC with the "DUPLICATES NOT" phrase. Data-base-identifier-5 must have identical data characteristics to those of the CALC-key data-base-identifier-4 as specified in the LOCATION clause. If an EQUAL TO phrase is specified for an item of a multi-item CALC-key, it must be specified for all of them.

3. Data-base-identifier-6 is a declared elementary item in the owner record of set-name-3.

4. Data-base-identifier-7 must have identical data characteristics to those of data-base-identifier-6.

5. Set-name-2, set-name-3,... must form a continuous path in the sense that the owner of set-name-3 is a member of set-name-2..., with set-name-2 as a start point, or root. In that path the same set-name must not appear more than once. The last set-type named must be the subject of the set entry of which this clause is a part.

Example:



```
SET SELECTION FOR C-D IS THRU A-B OWNER IDENTIFIED BY CALC-KEY,
THEN THRU B-C WHERE OWNER IS IDENTIFIED BY FIELD IN-B, THEN THRU
C-D WHERE OWNER IS IDENTIFIED BY FIELD-IN-C.
```

6. The elementary items referenced by data-base-identifier-6 must together uniquely identify the owner of set-name-3 among the members of set-name-2. A DUPLICATES ARE NOT ALLOWED phrase must be declared for those elementary items considered as a group in the ORDER clause of set-name-2 or in the KEY clause, or in a DUPLICATES clause, for the owner of set-name-3 as a member of set-name-2.

7. If DATA-BASE-KEY or CALC-KEY phrase is stated, a location mode of DIRECT or CALC respectively must be specified for the owner record-type of set-name-2.

General Rules

1. The data characteristics of data-base-parameter-1 are those required to hold a data-base-key value.

2. The data characteristics of data-base-parameter-2 are identical to those of the CALC-key as specified in the LOCATION clause in the record entry for the owner record-type of set-name-2.

3. The data characteristics of data-base-parameter-3 are those required to hold an area-name.

4. The data characteristics of the parameter named by data-base-parameter-4 are identical to those of data-base-identifier-6.

5. The SELECTION clause for the appropriate member record-type and set-type combinations governs the selection of a specific set for the purpose of inserting or accessing a member record. The set occurrence is determined by identifying the specific owner record occurrence.

6. Prior to the execution of any DML function involving selection, values must be supplied for the parameters and/or UWA elementary items specified in the EQUAL TO phrase(s) or in the LOCATION clause if one is implied.

7. The APPLICATION option causes the selection from set-name-2 of the occurrence given by the "current-of-set-name-2". The current-of-set can be the owner or a member.

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

9. The CALC-KEY option causes the selection from set-name-2 of that owner record:

   - Having a CALC-key equal to the value contained in the UWA elementary item or parameter named in the EQUAL TO phrase, or in the absence of the EQUAL TO specification, the UWA data items as specified in the LOCATION MODE IS CALC clause for the owner record-type.

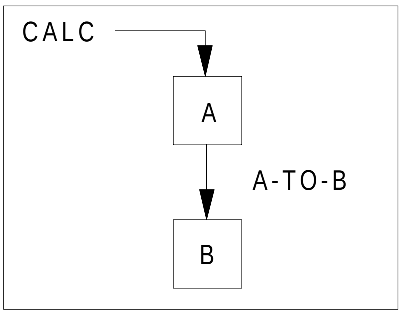
   - Located in the case of a multi-area record-type in the area specified in the parameter named in AREA-ID EQUAL TO, or, if this is absent, in the parameter named in AREA-ID of the WITHIN clause.

10. When only the THRU set-name-2 phrase is specified, the occurrence of set-name-1 is selected as described in General Rules 7 through 9. For example, for set A-B, the clause is:

    ```
    SET SELECTION THRU A-B OWNER IDENTIFIED BY CALC-KEY
    ```

11. Where both the THRU set-name-2 and one or more THEN THRU phrases are specified, the occurrence of set-name-2 is selected as described in General Rules 7 through 9. For each subsequent set in the path, the THEN THRU phrase causes the selection of the owner record of set-name-3... in its capacity as a member of the selected occurrence of the previously named set-type such that the owner record has values for data-base-identifier-6 equal to values contained in the UWA elementary items or parameters named in the EQUAL TO phrase, or if EQUAL TO is absent, the UWA data items data-base-identifier-6.

Examples:



If record A has a CALC location mode, then if the set selection clause is:

```
SELECTION THRU A-TO-B OWNER IDENTIFIED BY APPLICATION.
```

the program selects an occurrence of set A-TO-B before a STORE (or FIND) of record B is executed. The established current position in the set (A-TO-B) is used to identify the set occurrence. It may be the owner or a member. Insertion orders of NEXT and PRIOR are relative to the current set record and place the new record either after or prior to the current one. Orders of FIRST and LAST cause a return to the owner record (A) and the new record is then inserted either first or last in the set A-TO-B.

Processing related to a SORTED insertion depends on the current position in the set:

- If it is the owner, the key comparisons are made going forward through the set.
- If it is a member, the key of the member to be inserted is compared with the key of the current-of-set. If the member is to be inserted after the current-of-set, the key comparison proceeds forward through the set.
- If the new member must be inserted between the owner and the current-of-set, the key comparisons are made starting from the owner, forward.
- If the new member and the current-set are not of the same record type for a SORTED WITHIN RECORD-TYPE set, the key comparisons are made forward from the owner.

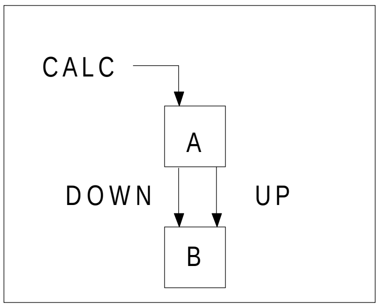
2. If the clause was:

   ```
   SELECTION THRU A-TO-B OWNER IDENTIFIED BY CALC-KEY.
   ```

   the owner record A is always located and then the specified insertion order is used. In each case, the operation begins with the owner record A.

3. Use of EQUAL TO.

   In the example of a network structure where different occurrences of record A are related to each other by means of linking through record B, there is the problem of having one field in record A that is used to identify this record (DIRECT or CALC fields), when actually two different A records must be identified and both accessed to connect record B into both sets. The schema can be stated as follows:



```
SET NAME IS DOWN...

  MEMBER IS B...

  SELECTION THRU DOWN OWNER IDENTIFIED BY CALC-KEY.

SET NAME IS UP...

  MEMBER IS B...

  SELECTION THRU UP

    OWNER IDENTIFIED BY CALC-KEY MYKF EQUAL TO PARM-1.
```

In order to locate the proper A record, the CALC fields named by the LOCATION MODE clause of record A must be initialized to identify the A record. This is sufficient to satisfy the selection criteria for the DOWN set. If the UP set had exactly the same selection clause, there would be a conflict since the same CALC field in record A would be needed to identify the second occurrence of record A. The EQUAL TO clause allows a data-base-parameter to be used as a temporary holder for the CALC field, so that after the new B record is inserted into the DOWN set, the UP set can identify its proper owner by using the control-field value saved in PARM-1.

2. Device Media Control Language

2.1 STORAGE VIEW

Between the schema view composed of records, sets and areas, and the disk extents which represent the final form of the data base, there exists an intermediate view, the storage view, composed of storage records and storage areas (see Figure 2-1).

A storage record has the usual attributes of a record.

A storage area is a group of n pages which acts as a container for the storage records. It corresponds to a UFAS file.

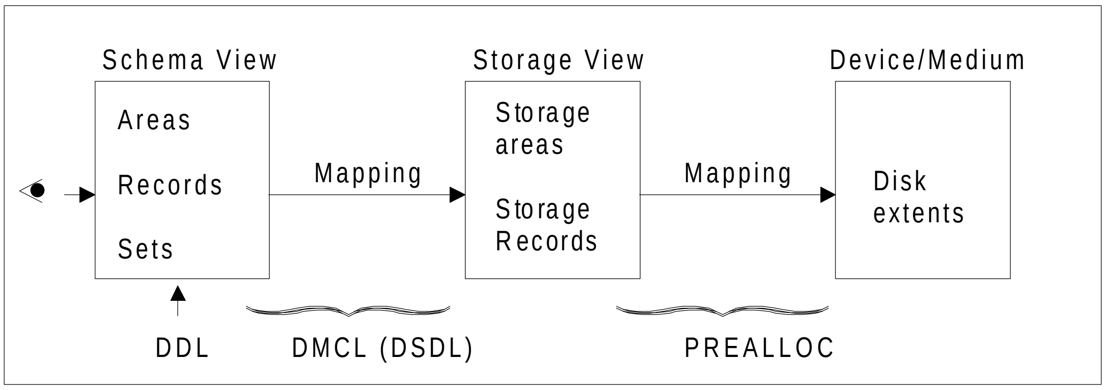


Figure 2-1. Storage View Between Schema View and Device/Medium

2.1.1 DMCL

DMCL stands for Device/Media Control Language; it has been renamed DSDL (Data Storage Description Language) in the CODASYL proposals since it no longer deals with the device/media. In this manual the term DMCL is used for the sake of compatibility and also because media characteristics can influence the choice of certain parameters.

The purpose of DMCL is twofold:

1. To select options for the mapping of the schema onto storage. In the current release, the mapping rules are imposed by IDS/II:

   - Each schema area corresponds to one storage area.

   - Each schema record occurrence corresponds to one storage record occurrence.

   - Each schema set occurrence corresponds to pointers placed as data fields in all the storage records in this set (chain implementation).

2. To provide quantitative information upon the population of the data base, since these are absent from the DDL. Rather than specifying the maximum number of occurrences of each record-type, the DBA specifies the overall size parameters of the containers, that is, of the storage areas.

   Other parameters may be supplied to restrict the placement of storage records to given ranges within storage areas or to allow for the migration of records or to specify the default buffer requirements for run-time execution, etc.

2.1.2 PREALLOC Function

PREALLOC is in charge of mapping one storage area, that is, a UFAS file, onto one or several extents of one or several disks.

This topic is discussed in detail in Section IV of this manual.

2.1.3 Program/Data Independence

Figure 2-2 illustrates what is meant by program/data independence.

- There is no independence between the program and the schema view, since the program logic is adapted to the objects described in this view.

- There is an independence between the program and the storage view, since it is possible to specify several storage views against the same schema view, without the program being aware of it. For example, the same programs can be run against the production data base or a smaller test data base.

  In certain circumstances, such as the computation or interpretation of data-base-keys, the program needs to know DMCL parameters. An extension to the ACCEPT function is provided to allow the program to read this information and thus to adapt itself automatically to the version in use.

  For implementation, the schema view and the storage view are in the same object schema. It is therefore necessary to duplicate the schema view in a second object schema (MOVE command of DDLPROC) if an alternate storage view is desired.

- There is an independence between the program and the device/media, since it is possible to preallocate several occurrences of the data base against the same storage view, without the program being aware of it. This is the case when each programmer uses his own occurrence of the same test data base.

- The link between the program and the occurrence of the data base is made at run-time by selecting, through JCL ASSIGN statements, the needed object schema and storage areas.

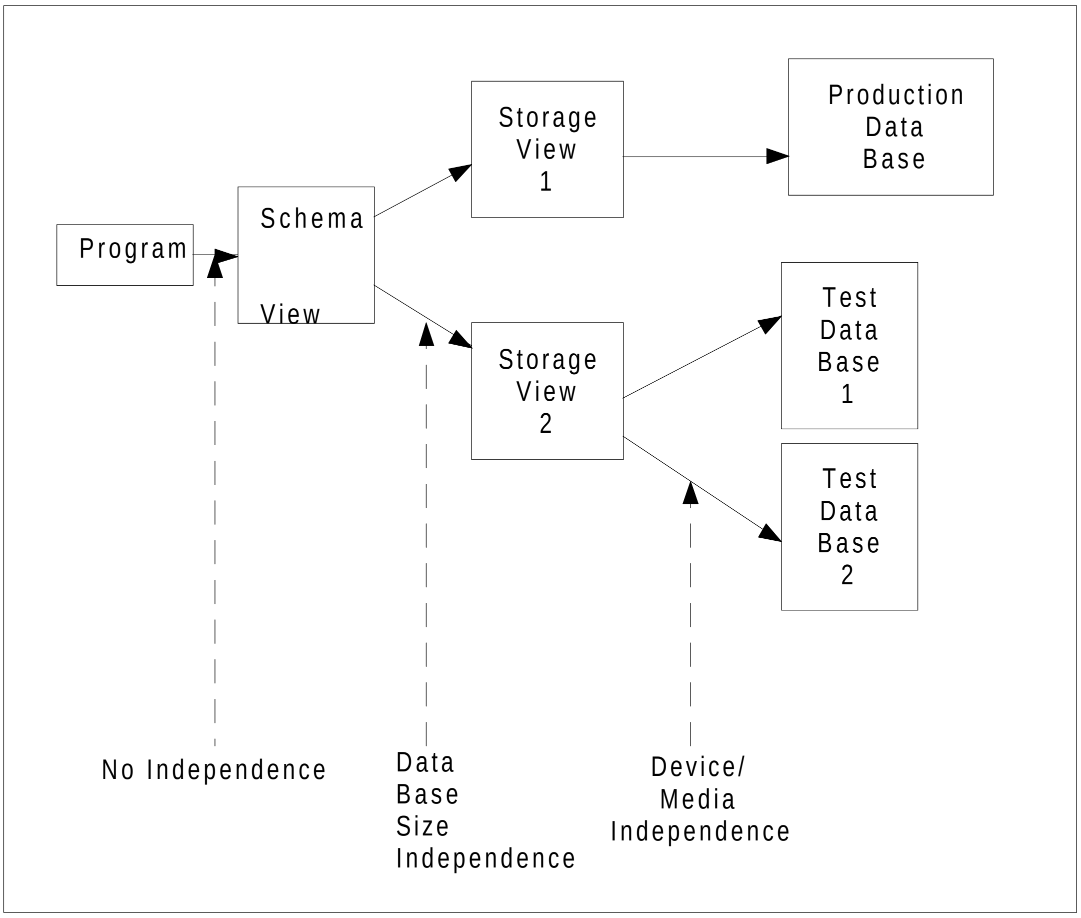


Figure 2-2. Program/Data Independence

2.2 STORAGE AREA STRUCTURE

2.2.1 Area Code Number

The data base is composed of a number NUMA of storage areas. Each storage area is identified by an area code number a, starting from 0:

```
 _______________________
|                       |
| O <= a <= NUMA - 1    |
|_______________________|
```

The area numbers are assigned in sequence, in the order of the schema DDL area entries.

2.2.2 Area Characteristics

- Each storage area is divided into a number NUMP of "pages" of equal length (see Figure 2-). The page is the unit of transfer between the disk and the DBCS buffers.

  `The upper limit of NUMP is (2**24) -1 = 16,777,215.`

  Each page is identified within the area by a page number p, starting from 0:

  ```
   _______________________
  |                       |
  | 0 <= P <= NUMP - 1    |
  |_______________________|
  ```

  Pages are placed sequentially on disk in the order of their page number. Programs may take advantage of this order to optimize disk head movements by a sequential processing of the area.

  NUMP is specified in the NUMBER-OF-PAGES clause of the DMCL area entry. The page size is specified in the PAGE-SIZE clause of the same entry.

- Each page is divided into a number LPP of "lines" (see Figure 2-3). The line is the container for one record.

  `The upper limit of LPP is 255.`

  Each line is identified within the page by a line number l, starting from 0:

  ```
   _____________________
  |                     |
  | O <= l <= LPP - 1   |
  |_____________________|
  ```

  The size of a line is variable according to the type of record it currently contains. A record cannot span pages.

LPP is specified in the LINES-PER-PAGE clause of the DMCL area entry.

- A storage area containing CALC records is also characterized by the size of the buckets which are involved in the hashing algorithm.
  This size is expressed as a number of pages (upper limit 255) in the CALC-INTERVAL clause of the DMCL area entry.
- The number of pages, page size, number of lines per pages and CALC interval may vary between areas.

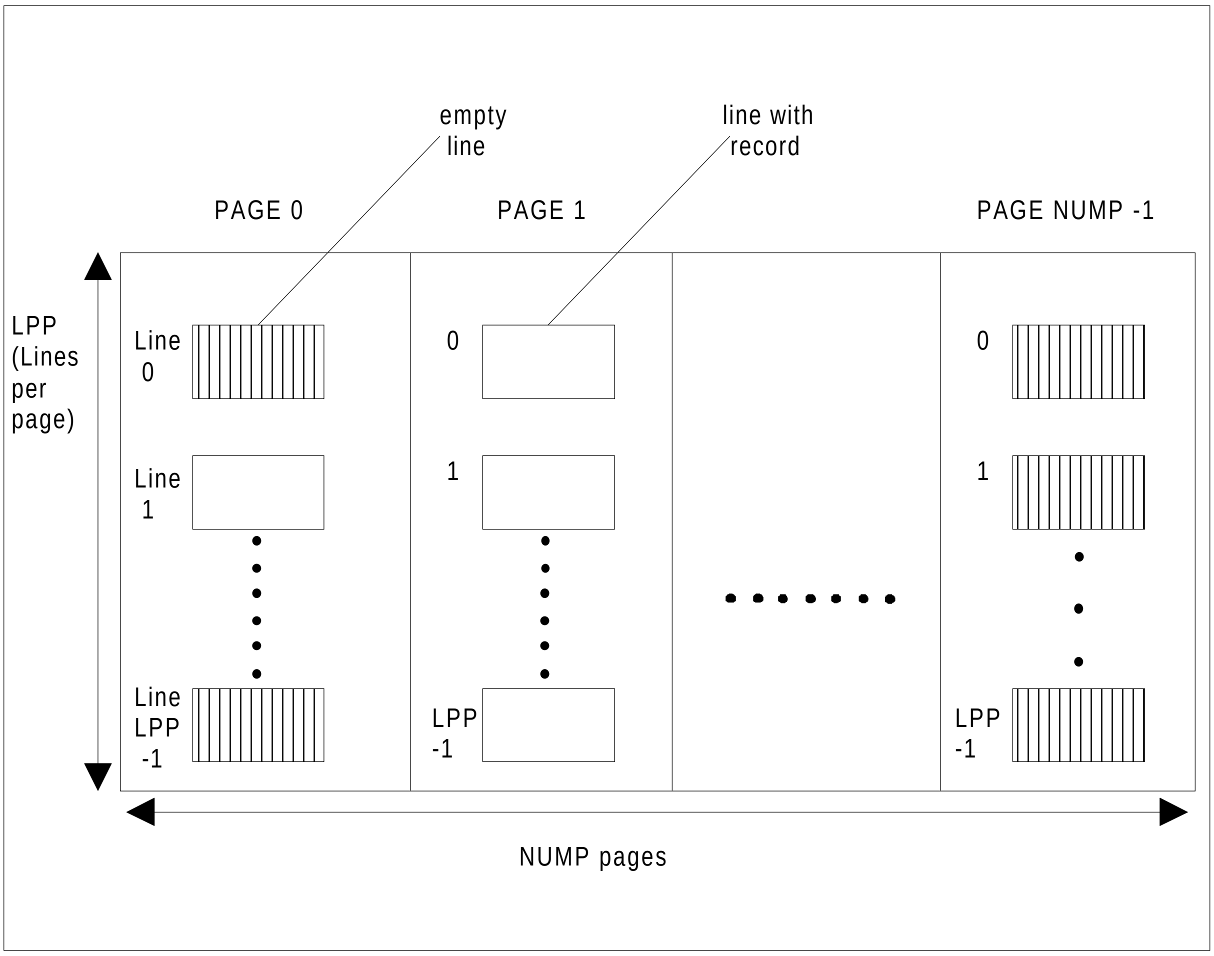


Figure 2-3. Storage Area Subdivisions

2.2.3 Page Structure

The page structure is outlined in Figure 2-4 The page contains:

- A page header of 12 bytes in length, which contains information on the space currently used, the space occupied by logically deleted records, the highest line number in use, etc.
- A variable length array of storage record locators, indexed by the line number from the bottom of the page. Each locator contains the 2 byte offset (relative to the end of the page header) of the storage record corresponding to this line number. The maximum size of the locator array is 2*LPP.
- Storage records composed of a header, a pointer zone and a data zone. These records are referenced by line number. They may be active or logically deleted. Logical deletion allows the DBCS to defer the reorganizaton of the page until compacting becomes necessary to store a new record in the page.
- CALC chain overflow-link records (OWL), of 10 bytes in length, used in the implementation of the CALC chain. These records are not referenced by line number. There is always such a record immediately after the page header (even if the storage area does not contain CALC records).

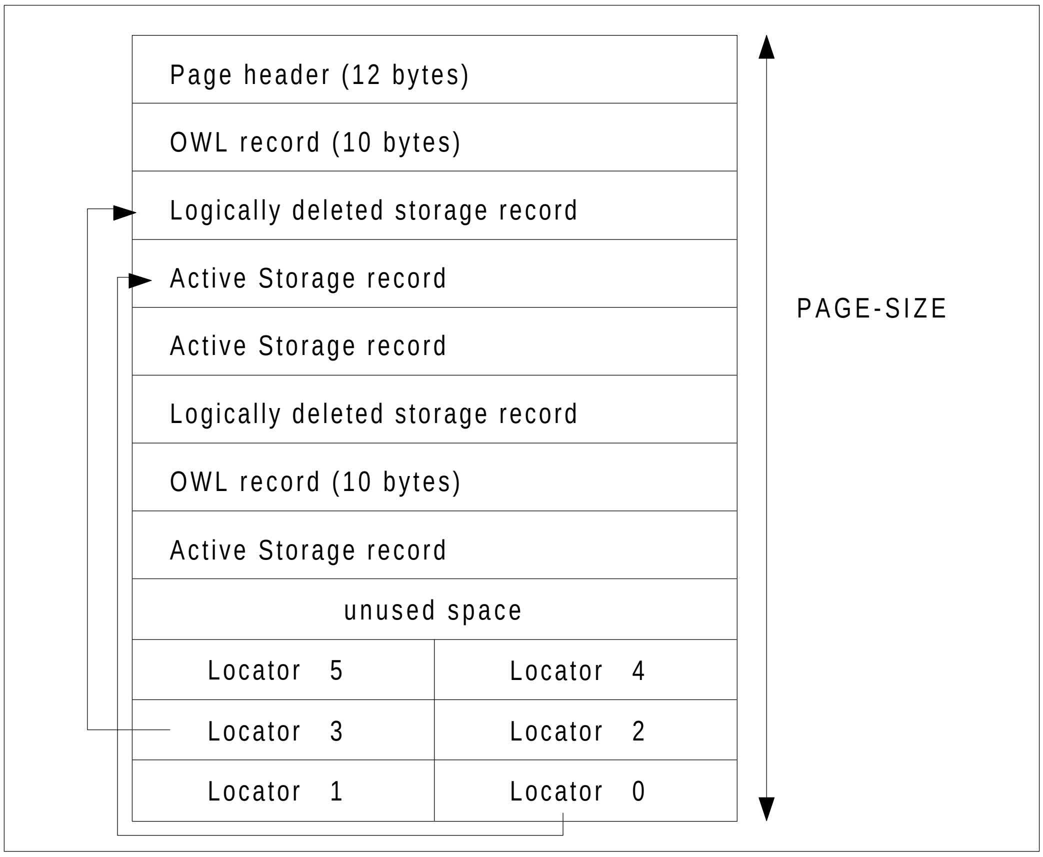


Figure 2-4. Page Structure

2.3 STORAGE RECORD ADDRESSING

2.3.1 Definitions

- When a storage record is stored in an area, or migrates within this area as a result of a MODIFY CALC-key, it is assigned a logical address, called its "area-key", which uniquely identifies the record within this area. The components of this area-key are:

```
 ________________________________________________
|                              |                 |
|  Components of area-key ak:  | page number p   |
|                              | line number l   |
|______________________________|_________________|
```

- This logical address, local to the storage area, gives rise, by addition of the area number, to another logical address, called the "data-base-key", which uniquely identifies the record within the whole data base.

```
 ___________________________________________________
|                                 |                 |
|Components of data-base-key dbk:|  area number a   |
|                                 | page number p   |
|                                 | line number l   |
|_________________________________|_________________|
```

- When the area-key is used in the "chain" implementation of a set, it is represented as a local pointer whose numeric value is calculated as:

```
 ____________________
|                    |
| ak = p * LPP + l   |
|____________________|
```

  where LPP is the number of lines-per-page of the area.

  To optimize the disk space, a local pointer is implemented on the minimum of bytes: 2, 3 or 4 (see Figure 2-5). The local pointer size may be different between areas.

- When the data-base-key is used in the "chain" implementation of a set, it is represented as a global pointer, obtained by concatenation of the area number and the area-key value ak.

  To optimize the disk space, a global pointer is implemented on the minimum of bytes: 2, 3, or 4 (see Figure 2-5). The area code zone is left-justified, except for the 4-byte pointer, where the leftmost bit is reserved for the positive sign of a COBOL DB-KEY item, as defined below. The global pointer size is constant for all areas.

- When a data-base-key appears in COBOL items declared with USAGE DB-KEY (equivalent to USAGE COMP-2), it is equal to the global pointer value right-justified, with zero fill in the case of 3 or 2 byte pointers (see Figure 2-5).

  A DB-KEY item may be involved in any COBOL verb (MOVE, COMPUTE...) which handles COMP-2 fields.

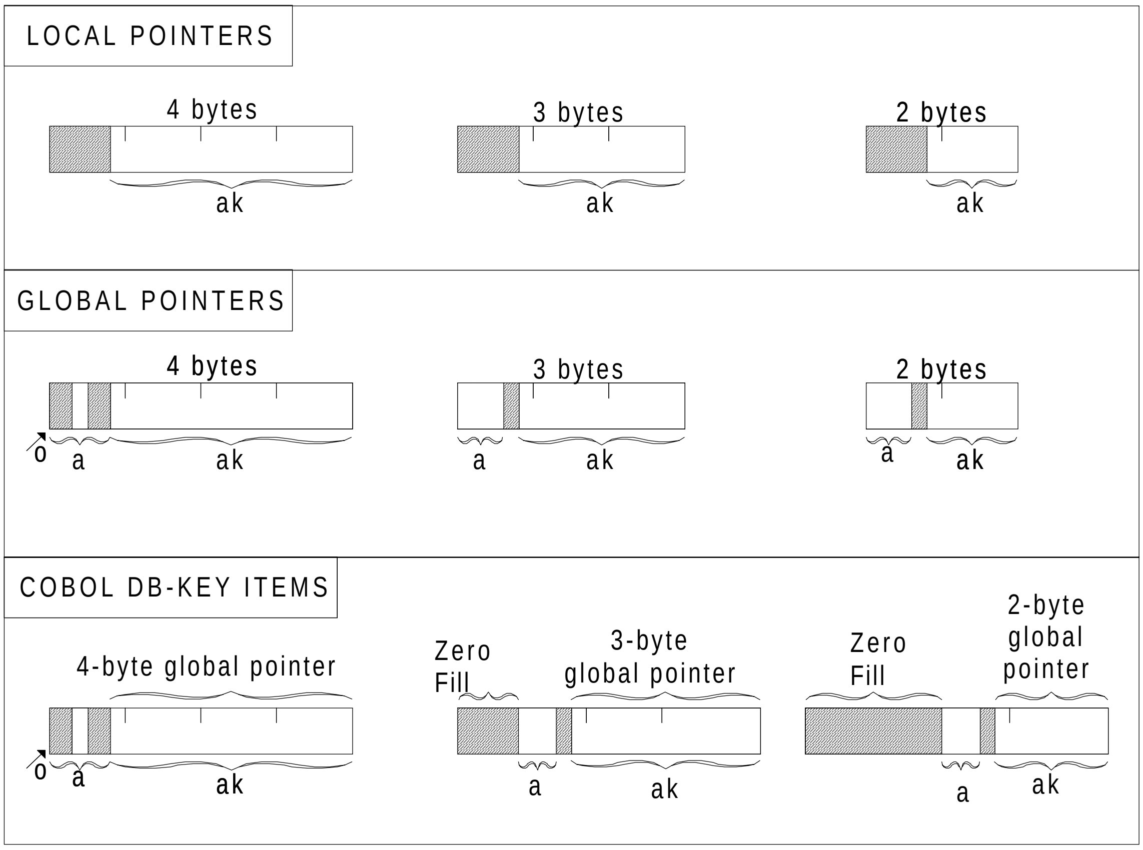


Figure 2-5. Structure of Local Pointers, Global Pointers, DB-KEY Items

2.3.2 Local Pointer Size

- Let LPP and NUMP be respectively the number of lines per page and the number of pages of an area. The number of area-keys available in this area is NUMAK such that:

  ```
  NUMAK = LPP * NUMP
  ```

  The number of bits BL required to represent the highest area-key value (NUMAK-1) as an unsigned binary number is calculated using the formula:

  $2^{BL-1} < \mathtt{NUMAK} <= 2^{BL}$

  Then the local pointer size in bytes is.

  ```
  2,if BL <= 16
  3,if 16 <BL <=24
  4,if 24 <BL <=30
  ```

  Detailed values are given below:

Table 2-1. Local Pointer Size

NUMAK value range	Bits Required (BL)	Bytes Used for Local Pointer	
9-16	4	1	
17-32	5	1	
33-64	6	1	2 bytes are actually used
63-128	7	1	
129-256	8	1	
257-512	9	2	
513-1024	10	2	
1025-2048	11	2	
2049-4096	12	2	
4097-8192	13	2	
8193-16384	14	2	
16385-32768	15	2	
32769-65536	16	2	
65537-131072	17	3	
131073-262144	18	3	
262145-524288	19	3	
524289-1048576	20	3	
1048577-2097152	21	3	
2097153-4194304	22	3	
4194305-8388608	23	3	
8388609-16777216	24	3	
16777217-33554432	25	4	

- The local pointer size may vary between areas since LPP and NUMP depend on the area.

- The local pointer size as calculated above is the default value chosen by DDLPROC. However the EXTEND LOCAL POINTERS clause of the DMCL area entry allows the DBA to select a greater size than required. (See the paragraph entitled "Local sets").

2.3.3 Global Pointer Size

- Let NUMA be the number of storage areas of the data base. The number of bits BLA required to represent the highest area number (NUMA-1) as an unsigned binary number is calculated using the formula:

  $2^{BLA-1} < \text{NUMAK} <= 2^{BLA}$

  Detailed values are given below:

Table 2-2. Global Pointer Size

Number of Areas	Number of Bits (BLA)
1-2	1
3-4	2
5-8	3
9-16	4
17-32...	5...

- Let BLM be the number of bits in the larger local pointers. The total number of bits BLG required for a global pointer is given by:

  ```
  BLG = BLM + BLA
  ```

  Then the global pointer size in bytes is:

  ```
  2, if BLG <= 16
  3, if 16 < BLG <= 24
  4, if 24 < BLG <= 31
  ```

  When BLG is not equal to exactly 16, 24 or 31, the area code zone is left-justified within the required number of bytes, leaving a longer bit field for the area-key zone. In the 4-byte case, the leftmost high-order bit is always reserved and set to zero.

- The maximum addressing capability of global pointers is:

Table 2-3. Global Pointer Addressing

Pointer Size	Number of Records
4 bytes (-1 bit)	2 147 483 648
3 bytes	26 777 216
2 bytes	65 536

- The global pointer size is the same for all areas of the data base.

- The global pointer size, as calculated above, is the default value chosen by DDLPROC. In view of future growth of the data base, the DBA may select the following DMCL options:
  - The EXTEND GLOBAL POINTERS clause of the schema entry increases the global pointer size. The area code zone is left justified within the specified number of bytes.
  - The EXTEND NUMBER OF AREAS clause of the schema entry increases the number of bits reserved for the area code zone of global pointers.
- Figure 2-6 illustrates the format and size of local and global pointers for a data base composed of three areas of different sizes.

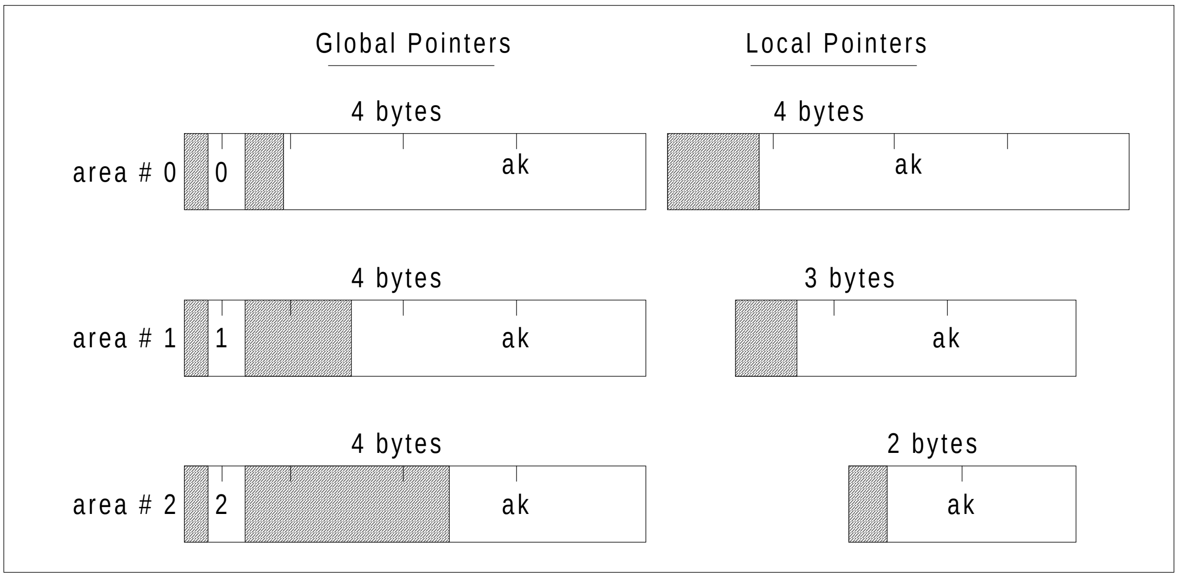


Figure 2-6. Local and Global Pointers in Areas of Different Sizes

2.4 SET IMPLEMENTATION BY A CHAIN OF POINTERS

2.4.1 Set Code Number

Each DDL set-type is identified internally by a set code number s, starting from 1 (s=o is reserved by the system).

Set code numbers are assigned in sequence, in the order of the schema DDL set entries.

The set code number itself is not stored in storage records, but it influences the order of the set pointers embedded in the storage records (see Storage Record Structure).

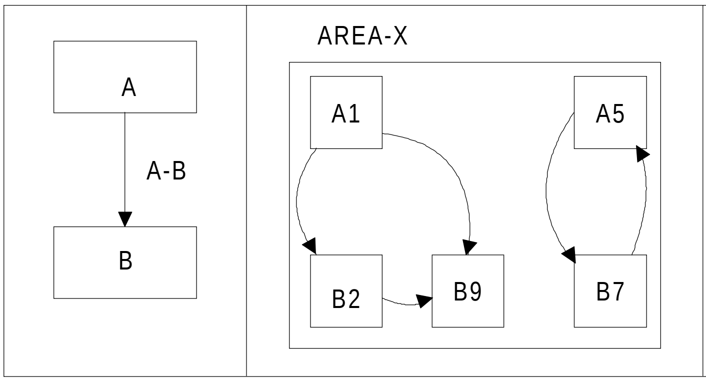
2.4.2 Number of Pointers

When a storage record is the owner of a set, it contains two pointers, NEXT and PRIOR.

When a storage record is a member of a set, it contains three pointers, NEXT, PRIOR and OWNER.

2.4.3 Local Sets

A set is termed local if DDL declarations imply that the owner and the members of a set occurrence are located in the same area. There are two cases of this as illustrated by Figures 2-7 and 2-8.



```
RECORD A LOCATION ...
         WITHIN AREA-X
RECORD B LOCATION
         WITHIN AREA-X
 (or WITHIN AREA OF OWNER)
```

Figure 2-7. Local Set - Mono-area Case

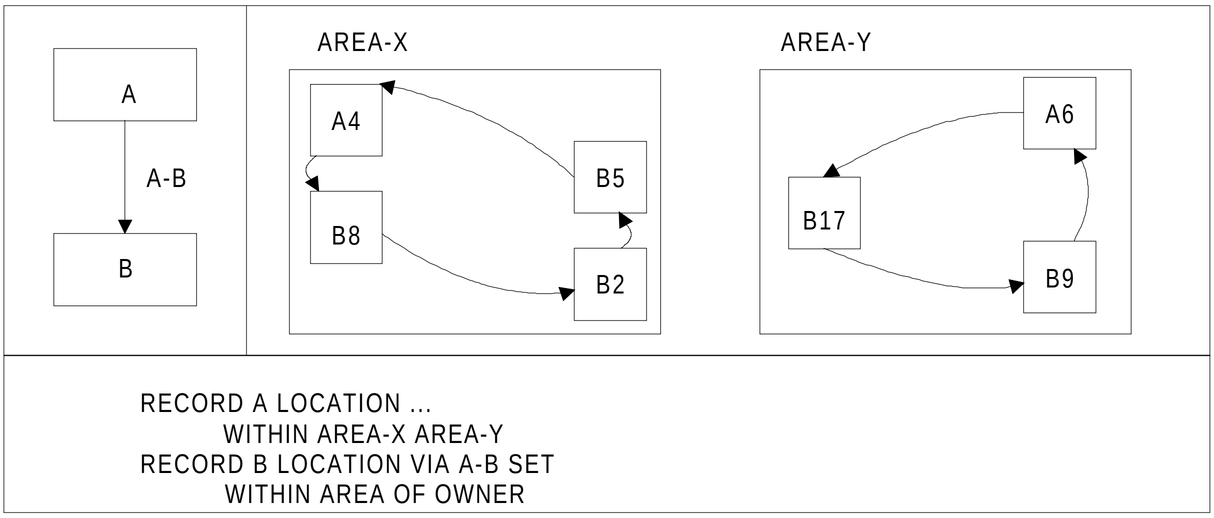


Figure 2-8. Local Set - Multi-area Case

If no contradictory DMCL options are selected, DDLPROC will implement a local set:

- with local pointers in the mono-area case.
- with local pointers in the multi-area case, if the local pointer size in bytes is the same in all the areas involved. (The current implementation imposes that the structure of a storage record-type be the same in all the areas where it can be located).
- with global pointers in the multi-area case, if the local pointer size in bytes is not the same in all the areas involved.

When the DBA knows that some currently local sets will become global sets in the future, he may use:

- either the NO LOCAL POINTERS clause of the schema entry to force all the local sets to be implemented with global pointers.
- or the NO LOCAL POINTERS clause of the set entry to force specific local sets to be implemented with global pointers.

In the multi-area case with different local pointer sizes, the DBA may use the EXTEND LOCAL POINTER clause of the area entry to force a larger local pointer size to be implemented than that currently required, and thus to make local pointer sizes equal in all the areas involved.

Example:

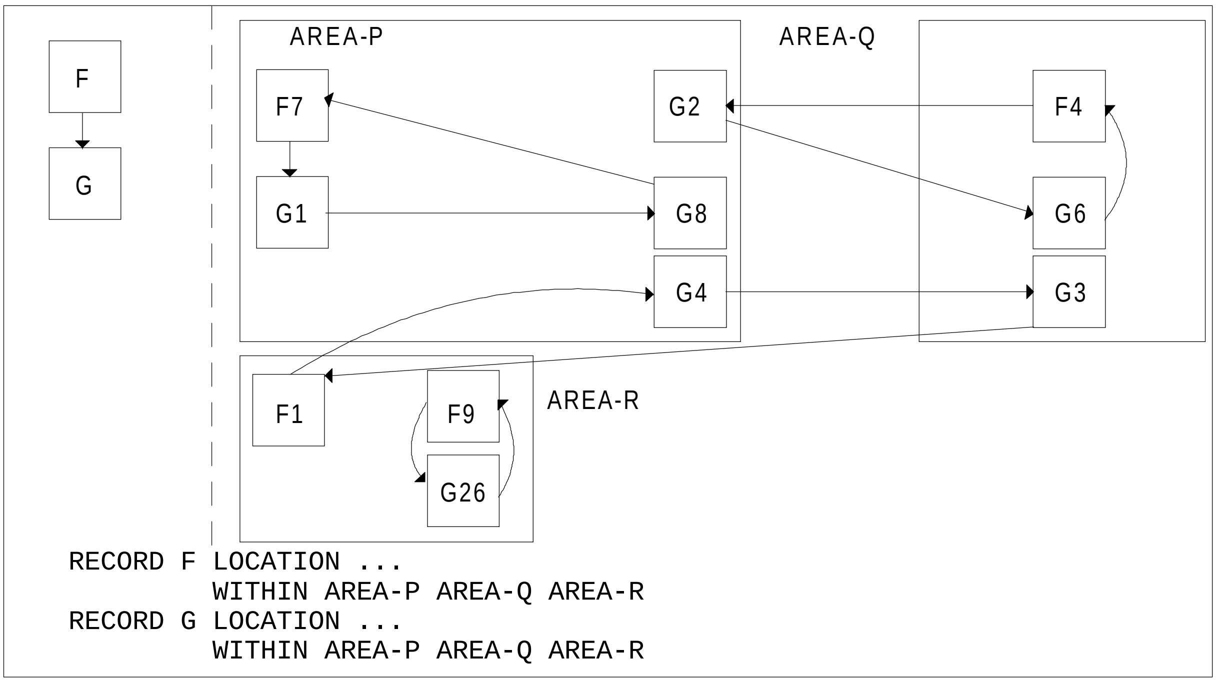
Suppose that the DMCL entries corresponding to figure 2-8 are as.

```
AREA NAME IS AREA-X
  NUMBER-OF-PAGES IS 1500
  NUMBER OF LINES-PER-PAGE IS 50
  ...
AREA NAME IS AREA-Y
  NUMBER-OF-PAGES IS 1300
  NUMBER OF LINES-PER-PAGE IS 50
  EXTEND LOCAL POINTERS TO 3 BYTES
  ...
```

The number of records that should be addressed by a local pointer in AREA-X is 1500 * 50 = 75000 key values; from the list of pointer sizes, 3 bytes are required. The same calculation for AREA-Y gives 2 bytes (1300 * 50 = 65000). Since, for the example, there is a set-type which has local occurrences in both areas, the local pointer length must be forced to the same value for both areas. Therefore there is an EXTEND LOCAL POINTERS entry required in the DMCL entry for AREA-Y.

2.4.4 Global Sets

A set is termed global if DDL declarations imply that the participants in a set occurrence can be located in different areas. Figure 2-9 illustrates this case.



```
RECORD F LOCATION ...
         WITHIN AREA-P AREA-Q AREA-R
RECORD G LOCATION ...
         WITHIN AREA-P AREA-Q AREA-R
```

Figure 2-9. Global Set

A global set is always implemented with global pointers.

2.5 STORAGE RECORD STRUCTURE

2.5.1 Record Code Number

Each storage record is identified internally by a record code number r, starting from 1 (r=0 is reserved by the system).

Record code numbers are assigned in sequence, in the order of the DDL record entries.

The record code number is stored in the header of each storage record in the data base.

2.5.2 Record Structure

A storage record is composed of three parts, as shown in Figure 2-10.

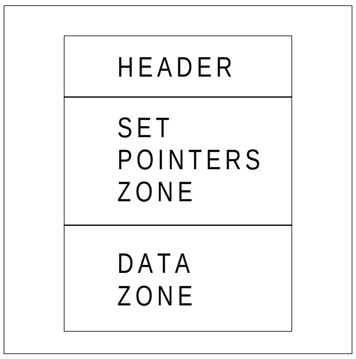


Figure 2-10. Storage Record Structure.

The header is 5 bytes in length for non-CALC records and 9 bytes for CALC records. It contains the code number of the record, its total length, its status indicators and its CALC chain information for CALC records.

The set pointers zone, whose length varies with the record-type, contains the set pointers for the sets in which the record is declared as owner or member.

The number, type, and size of the pointers are determined for each set as explained in the previous paragraphs.

The order of the set pointer groups is:

- first the sets in which the record is declared as the owner. These sets appear in increasing order of code number.

- then the sets in which the record is declared as a member. If the location mode of the record is not VIA set-name, these sets appear in increasing order of code number. If the location mode is VIA set-name-1, set-name-1 appears first, then the others in ascending order of code number.

When a record-type does not participate in any set-type, the set pointers zone does not exist.

The data zone, whose length varies with the record-type, contains the data as defined in the DDL data subentries. In the current implementation, the data format in the storage records is identical to the data format in the UWA records seen by the program.

The data zone may not exist for certain "root" or "relation" records whose role is restricted to the participation in sets.

The PRINT STORAGE DESCRIPTION command of DDLPROC enables the DBA to display the lay-out of the storage records.

2.5.3 Example

Consider the data base of Figure 2-1. Assume that:

- The record RC has a LOCATION mode VIA B-C and contains one data field of TYPE CHARACTER 12.
- The sets are declared in the order A-C B-C C-D C-E in the DDL.
- The local sets B-C C-D are implemented with 2-byte local pointers.
- the global sets A-C C-E are implemented with 3-byte global pointers.

The storage record RC has a header of 5 bytes as it is non-CALC. The total length (header+pointer+data) is 42 bytes.

Set C-D is the first set whose RC is the owner. The pointer group contains 2 local pointers: NEXT, PRIOR.

Set C-E is the second set whose RC is the owner. The pointer group contains 2 global pointers: NEXT, PRIOR.

Set B-C is not the first set whose RC is a member but it is the set involved in the LOCATION mode VIA. The pointer group contains 3 local pointers: NEXT, PRIOR, OWNER.

Set A-C is the first set whose RC is a member. The pointer group contains 3 global pointers: NEXT, PRIOR, OWNER.

The data zone is 12 bytes long.

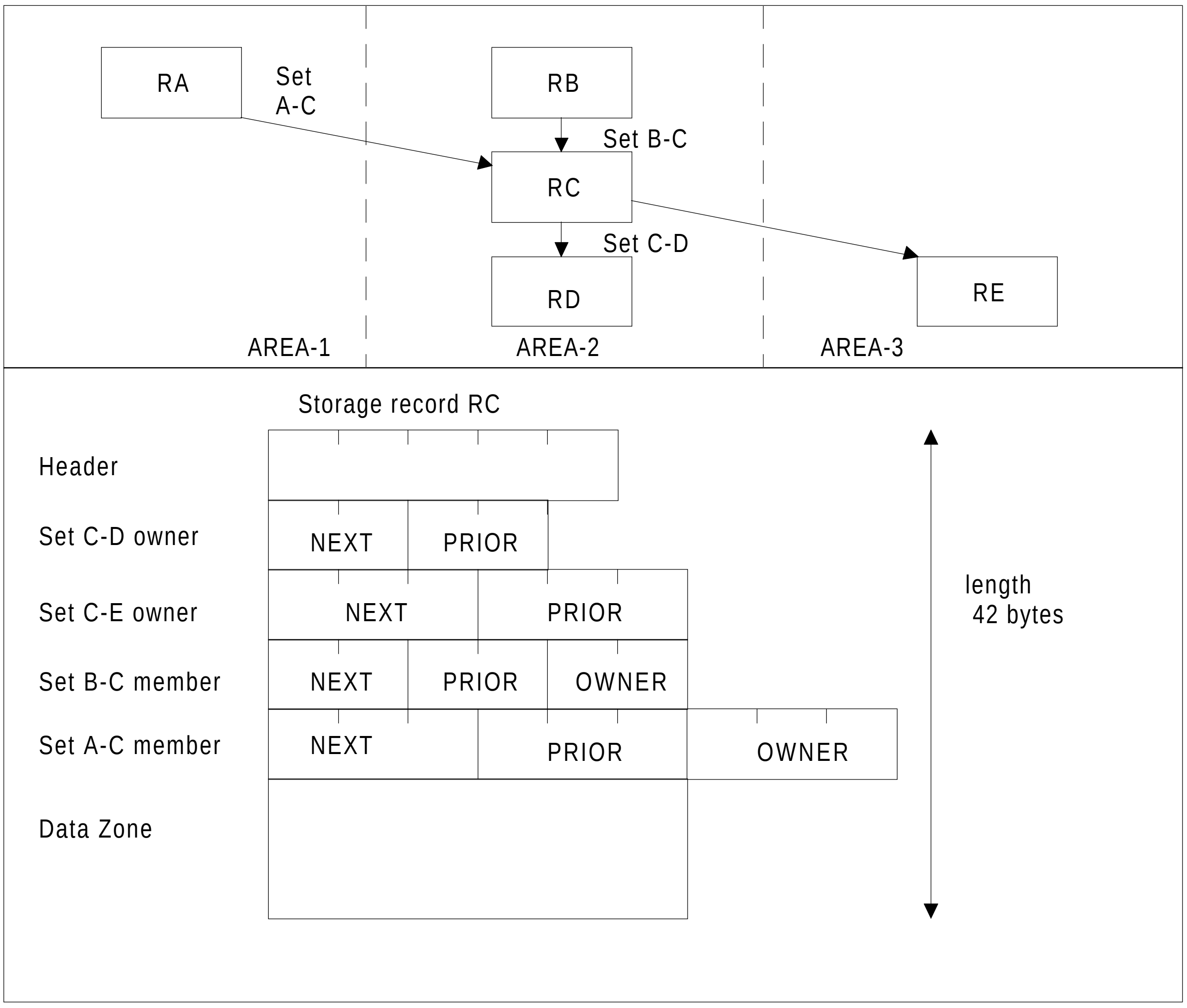


Figure 2-11. Example of Storage Record Lay-out.

2.6 CHOICE OF THE PAGE-SIZE PARAMETER

When the DBA selects the DMCL PAGE-SIZE parameter, he must take several factors into consideration and decide on the best compromise among them.

1. PAGE-SIZE represents the overall size of the page. It includes the page header, the record locators, at least one OWL record, and for each storage record its header, pointer zone, and data zone.

2. The page size must conform to the device characteristics and UFAS rules:

   - PAGE-SIZE must be a multiple of 256 bytes. It is recommended that multiples of 512 bytes be chosen to ease migration to fixed-sector disks.

   - The larger the page size with respect to the track size, the fewer bytes are reserved for inter-page gaps.

   - Pages may not overlap tracks. This restricts the choice to certain values if the disk space is to be used efficiently, in particular as the page size comes near to the track nominal size.

The recommended values are listed in Table 2-4 for the MSU 0300 and MSU 0400 families, the MSU 0500 and the SMD 300.

The percentages show the efficiency of track space use as compared with the nominal track capacity.

3. Large pages are recommended when they contain clusters of records which are processed together. In this case only one I/O is necessary to bring the cluster into memory.

   Large pages are not recommended if they are usually loaded into memory to process only one record at a time; not only is the buffer size required as large as the page size and may far exceed the record size, but also the transfer time is proportional to the page size.

4. In a TDS environment with many simultaneous transactions, large page sizes imply large buffer requirements if one wants to avoid competition for buffers (assuming that there is no concurrent access to the same pages).

5. The page size may vary between areas. If areas of different page sizes are used in the same pool, an extra CPU overhead is associated with the dynamic adaptation of buffer sizes to page sizes. This overhead may be avoided if areas have the same page size.

   For more details on the UFAS buffer pool management, see the UFAS User Guide. The IDS term "page" is a synonym for the "CI" or "Control Interval".

Table 2-4. Recommended Page Size Values

PAGE SIZE	PAGES PER TRACK	PAGES PER CYLINDER
	MSU	0350/400/401/402/452
12800	1 (98%)	19
12288	1 (94%)	19
6144	2 (94%)	38
4096	3 (94%)	57
3072	4 (94%)	76
2048	6 (94%)	114
1536	7 (83%)	133
1024	11 (86%)	209
512	20 (79%)	380
		MSU 0310
6144	1 (84%)	20
3072	2 (84%)	40
2048	3 (84%)	60
1536	4 (84%)	80
1024	6 (84%)	120
512	11 (77%)	220
		MSU 0500
18944	1 (98%)	38
9216	2 (96%)	76
6144	3 (96%)	114
4608	4 (96%)	152
3584	5 (93%)	190
2560	7 (93%)	266
2048	8 (85%)	304
1536	11 (88%)	418
1024	15 (80%)	570
512	27 (72%)	1026
		SMD 300
18944	1 (97%)	19
18432	1 (94%)	19
9216	2 (94%)	38
6144	3 (94%)	57
4608	4 (94%)	76
3584	5 (91%)	95
3072	6 (94%)	114
2560	7 (91%)	134
2048	8 (83%)	152
1536	11 (86%)	209
1024	16 (83%)	304
512	28 (73%)	532

2.7 CHOICE OF THE LINES-PER-PAGE PARAMETER

- LINES-PER-PAGE defines the maximum number of records that can be addressed in a page. It must not exceed 255. It may vary between areas.

- If the PAGE-SIZE parameter has been defined beforehand, LINES-PER-PAGE is derived by considering the size of the records of the area. The value selected must be large enough to address all the records that can fit in the page space.

  If LINES-PER-PAGE is too small, line number saturation occurs before space saturation, giving a poor use of the disk space.

  If LINES-PER-PAGE is too large, space saturation occurs before line number saturation; line numbers are lost, thus reducing the total addressing capability of the data-base-key.

  If the area contains record-types of different sizes, it is recommended that the LINES-PER-PAGE value for the smaller records be chosen. For large records, space will be used up before line numbers are used up.

- The LINES-PER-PAGE parameter may be selected before PAGE-SIZE. This is the case when the area contains groups of member records and one page is supposed to accomodate the average number of records of a group. Then LINES-PER-PAGE is chosen equal to this number and PAGE-SIZE is taken from the record size.

  If the resulting PAGE-SIZE is too large, for reasons discussed in the paragraph Choice of the PAGE-SIZE Parameter, it may be advisable to store the group on a fixed number n of pages (with n = 1). LINES-PER-PAGE is then chosen equal to the number of records of a group divided by n and PAGE-SIZE is derived as above from the record size.

2.8 CHOICE OF THE CALC-INTERVAL PARAMETER

As a rule, assuming that the CALC distribution is uniform, the CALCinterval of a record-type in a range may be chosen as:

- If the number of CALC records is less than the number of pages in the range, for example because their clusters of non-CALC members are located in the same range and spread over 2 or more pages, then the CALC interval value is calculated by dividing the number of pages by the number of CALC records. In the ideal situation, the CALC chain would contain only one record.

- If the number of CALC records is greater than the number of pages in the range, the CALC interval value is 1 page, since the minimum value minimizes the average number of records per CALC chain and consequently the overflow to the following pages.

The CALC interval is constant throughout the area. Its value may therefore be the result of a compromise if several CALC record-types are to be located in the same or different range of the area.

The number of pages of an area or of a range must be a multiple of the CALC interval. It is recommended that the number of CALC buckets, that is the number of pages divided by the CALC interval, should be a prime number or that at least it should not possess small factors: 2, 3, 5, 7, 11, 13, 17, 19. The automatic selection of the nearest prime number may be requested through the OPTIMIZE option of the DMCL area and record entries.

The insertion of new records or retrieval of existing records from the CALC chain may involve a search of the entire list of records from the appropriate CALC bucket. If duplicate records are not allowed then during insertion the entire CALC chain is searched looking for a duplication of the fields used for randomization.

When non-CALC members are clustered near their CALC owners, and when owners are likely to be accessed more often than their members, it may be advisable to load all the owners first and then the members. This will cause the CALC synonyms to be located in the first page of a CALC bucket and therefore require only one I/0 for access. Otherwise, if the distribution is not uniform or if the number of records of a cluster is variable, the members of a CALC record may fill the first page of the CALC bucket and cause the next synonym to be placed in an overflow page.

The CALC overflow rate increases with the number of records stored in the area; areas should therefore be large enough to avoid saturation. A load factor of 70% (space or lines percentage) may be satisfactory.

The CALC interval chosen may be checked for suitability before area allocation, using the simulation function of H_DBUTILITY utility.

2.9 DEVICE MEDIA CONTROL LANGUAGE SYNTAX

DMCL uses the same syntax conventions as DDL. See Section I.

Each DMCL entry references a schema DDL object and describes how it is mapped onto storage.

The general structure of the DMCL is:

```
Schema Entry
(Area Entry) ...
[Record Entry] ...
[Set Entry] ...
```

Entries may appear in any order after the schema entry. Each entry is terminated with a period followed by a space or an end-of-line. The entire DMCL source text is terminated by an END-DMCL statement.

Each of the entries is described in detail below.

Figure 2-12 shows a sample DMCL source.

```
COMMENT "************************************************************"
COMMENT "*             SCHEMA DMCL FOR TEST-GROUP *TG19*              *"
COMMENT "************************************************************"

COMMENT "********************************"
COMMENT "*             SCHEMA            *"
COMMENT "********************************"

SCHEMA TG19-SH.

COMMENT "********************************"
COMMENT **             AREAS             *"
COMMENT "********************************"

AREA TG19-A00.
     NUMBER-OF-PAGES 53
     LINES-PER-PAGE 45
     PAGE-SIZE 2048.

AREA TG19-A01.
     NUMBER-OF-PAGES 52
     LINES-PER-PAGE 45
     PAGE-SIZE 2048.

END-DMCL.
```

Figure 2-12. Sample DMCL Source

2.9.1 SCHEMA Entry

Function

To identify the name of the schema and to specify miscellaneous storage and execution attributes which are not specific to a given area or record or set.

General Format

```
 SCHEMA NAME IS schema-name

    [EXTEND NUMBER OF AREAS TO integer-1]

    [EXTEND GLOBAL POINTERS TO integer-2 BYTES]

    [NO LOCAL POINTERS]

    [BUFFER POOL NAME IS buffer-pool-name]

    [NUMBER OF BUFFERS IS integer-3] .
```

2.9.1.1 SCHEMA Clause

Function

To specify the schema-name.

Format

```
 SCHEMA NAME IS schema-name
```

Rules

1. Schema-name is the name of the schema as specified in the SCHEMA entry of the DDL.

2.9.1.2 EXTEND NUMBER OF AREAS Clause

Function

To force the pointer size of global pointers to allow for a larger number of areas than that declared by the current schema.

Format

```
 EXTEND NUMBER OF AREAS TO integer
```

Rules

1. If present this clause must specify a value greater than the number of areas being described.

2. If this clause is specified, the DBCS will build global pointers with an area code zone large enough to accommodate the number of areas specified.

   For further details see the paragraph Storage Record Addressing in this section.

3. If this clause is not used then the global pointers will have sufficient space to process the number of areas declared. If, for example, 3 areas are declared it is not necessary to specify EXTEND NUMBER OF AREAS TO 4 because there is already space in the global pointers to handle 4 areas.

2.9.1.3 EXTEND GLOBAL POINTERS Clause

Function

To force the size of global pointers for the schema to a size greater than that which is currently required.

Format

```
 ____________________________________________
|                                            |
| EXTEND GLOBAL POINTERS TO integer BYTES    |
|____________________________________________|
```

Rules

1. "integer" can take the values 3 or 4.
2. When specified, the value must be equal to or greater than the size that would be automatically chosen.
3. If this clause is omitted, the calculation of global pointer size is performed using the information declared in DMCL for each area.
4. If this clause is specified, global pointers are implemented with the size specified.
5. For further details, see the paragraph Storage Record Addressing at the beginning of this section.

2.9.1.4 NO LOCAL POINTERS Clause

Function

To force all sets to have global pointers.

Format

```
 _____________________
|                     |
| NO LOCAL POINTERS   |
|_____________________|
```

Rules

1. If this clause is omitted then set-types which are local by schema definition will be held with local pointers, unless otherwise specified in the set entry itself.
2. If this clause is specified, all local sets are implemented with global pointers.
3. For further details see the paragraph Set Implementation by a Chain of Pointers at the beginning of this section.

2.9.1.5 BUFFER POOL Clause

Function

To name a buffer pool to be used by the areas of the schema at execution time.

Format

```
 ________________________________________
|                                        |
| BUFFER POOL NAME IS buffer-pool-name   |
|________________________________________|
```

Syntax Rules

1. The buffer-pool-name must be no more than four characters long.

2. If the blank name is used then it must be in the form '   ' (that is, 4 blanks enclosed by apostrophes).

3. Non-blank names may contain letters, digits, the minus sign, and the underscore. They must be delimited by apostrophes if they do not conform to the rules for the formation of DDL names.

General Rules

1. If this clause is omitted then the default name chosen is DBPL (for Data Base Pool).

2. If the specified value is blank ('   ') then the NUMBER OF BUFFERS clause gives the number of buffers reserved for each area (each area has its own set of buffers).

3. A non-blank entry (the default) means that the NUMBER OF BUFFERS clause gives the total number of buffers that will be shared by all areas.

4. Buffer control information can also be specified in the run-time options. Any DMCL declaration can be overridden.

5. The purpose of the pool of buffers is to reduce the total buffer space requirements when several areas are in the READY state. Suppose that a DML program accesses one area for a certain length of time:

   - If there is no buffer pool, it cannot use more buffer space than that reserved for this area. The space reserved for the other areas is not available.

   - If there is a buffer pool, the total pool space may be used for this area. When the program starts working on a different area, the pool space is assigned to the buffers of the new area.

2.9.1.6 NUMBER OF BUFFERS Clause

Function

To specify the number of buffers to be used at execution time.

Format

```
 ________________________________
|                                |
| NUMBER OF BUFFERS IS integer   |
|________________________________|
```

Rules

1. This statement is optional. If it is omitted then the default chosen is 4.

2. The minimum value that can be specified is 3.

3. If there is a non-blank buffer-pool-name then the value specified is the number of buffers to be shared amongst all areas. In the case where different areas have different page sizes, the largest value will be chosen as the buffer size.

4. If there is a blank pool-name then the value specified (or implied by default) is the number of buffers to be reserved for each area.

5. Buffer control information can also be specified in the run-time options at execution time. Any DMCL declaration can be overridden.

6. The number of buffers to choose at run time is a compromise between the buffer space requirements and the performance requirements.

   The DBCS will function with a minimum of 3 buffers, whether there is a buffer pool or not.

   If a program works for a certain time on a limited set of pages the number of buffers can be chosen to accomodate all these pages.

   If a program accesses pages in a completely random manner, then increasing the number of buffers will not improve the total performance.

2.9.2 AREA Entry

Function

To declare the attributes of the storage area that corresponds to a schema area.

General Format

```
AREA NAME IS area-name

   [AREA INTERNAL FILE NAME IS ifn]

                                [          {HIGH} ]
   NUMBER-OF-PAGES IS integer-1 [OPTIMIZE  {    } ]
                                [          {LOW } ]

   NUMBER OF LINES-PER-PAGE IS integer-2

   PAGE-SIZE IS integer-3 BYTES

   [CALC-INTERVAL IS integer-4 PAGES]

   [EXTEND LOCAL POINTERS TO integer-5 BYTES] .
```

Rules

1. There must be one area entry per area declared in the schema.

2.9.2.1 AREA NAME Clause

Function

To name the schema area which is mapped onto the storage area described in the other clauses of this entry.

Format

```
   ____________________________
  |                            |
  | AREA NAME IS area-name     |
  |____________________________|
```

Rules

1. "area-name" is the DDL name of a schema area.

2. The DMCL name of the storage area is implicitly the same as the DDL name of the schema area.

2.9.2.2 AREA INTERNAL FILE NAME Clause

Function

To define the internal-file-name to be used at run-time in the ASSIGN/DEFINE statements referencing this storage area.

Format

```
 _________________________________
|                                 |
| AREA INTERNAL FILE NAME IS ifn  |
|                                 |
|_________________________________|
```

Rules

1. "ifn" must conform to the rules for internal-file-names defined in the JCL Reference Manual.

   In addition, "ifn" must be different from:

   ```
   IDSOPT
   IDSTRACE
   PRTFILE
   COMFILE
   SLLIBa
   DDLIBb
   SAVEc
   ```

   where a,b,c, are decimal digits or spaces.

2. If this clause is omitted, the default internal-file-name chosen is the storage area-name (right-truncated to 8 characters if necessary).

3. There must be no duplicates among the internal-file-names of a schema.

   If several data bases are accessed in the same step, and if the internal-file-names are not unique among the different schemas, the user must ensure the uniqueness of each internal-file-name by using the INTERNAL FILE NAME command in the run-time options for the relevant schemas.

4. In the TDS generation and operator commands, storage areas are referenced by their internal-file-names as defined, explicitly or implicitly, in the DMCL. These names may be overriden by run-time options.

2.9.2.3 NUMBER OF PAGES Clause

Function

To define the number of pages of the storage area.

Format

```
                              [          {HIGH}]
NUMBER-OF-PAGES IS integer    [OPTIMIZE {    }]
                              [          {LOW }]
```

Rules

1. "integer" represents the number of pages, NUMP. It must be such that:

   ```
   1 <= NUMP <= 16,777,215
   ```

2. If OPTIMIZE is not specified and the storage area does not contain CALC storage records, there is no other constraint on NUMP.

3. If OPTIMIZE is not specified and the storage area contains CALC storage records, NUMP must be a multiple of the CALC-INTERVAL parameter.

   It is recommended that the number of "buckets" (that is, the number of pages divided by the CALC-INTERVAL) should be a prime number or that at least it should not possess small factors: 2, 3, 5, 7, 11, 13, 17, 19.

4. If OPTIMIZE is specified and the storage area contains CALC storage records, DDLPROC increments (HIGH) or decrements (LOW) NUMP in order to obtain the next prime number for the number of buckets.

5. OPTIMIZE must not be specified if the storage area contains no CALC storage records.

2.9.2.4 LINES-PER-PAGE Clause

Function

To define the number of lines of each page of the storage area.

Format

```
 ______________________________________
|                                      |
| NUMBER OF LINES-PER-PAGE IS integer  |
|______________________________________|
```

Rules

1. "integer" is the number of lines per page LPP. It must be such that:

```
1 <= LPP <= 255
```

2. For details on selecting values, see the paragraph Choice of LINES-PER-PAGE in this section.

2.9.2.5 PAGE SIZE Clause

Function

To define the size in bytes of the pages of the storage area.

Format

```
 ________________________________
|                                |
| PAGE-SIZE IS integer BYTES     |
|________________________________|
```

Rules

1. "integer" must be a multiple of 256 and preferably a multiple of 512 for migration to fixed-sector disks.

2. For details on selection values, see the paragraph Choice of PAGE-SIZE in this section.

2.9.2.6 CALC-INTERVAL Clause

Function

To define the number of pages of the "bucket" involved in the randomization algorithm for the placement of CALC records in this storage area.

Format

```
 ______________________________________
|                                      |
| CALC-INTERVAL IS integer PAGES       |
|______________________________________|
```

Rules

1. "integer" is the number of pages B of the bucket. It must be such that:

```
1 <= B <= 255
```

2. This clause must not be specified for a storage area which does not contain CALC storage records.

3. When the clause is not specified and the storage area contains CALC storage records, the default value 1 is assumed.

4. For details on selecting values, see the paragraph Choice of CALC-INTERVAL in this section.

2.9.2.7 EXTEND LOCAL POINTERS Clause

Function

To force the size of the local pointers of the storage area to a size greater than is theoretically required.

Format

```
 ________________________________________
|                                        |
| EXTEND LOCAL POINTERS TO integer BYTES |
|________________________________________|
```

Rules

1. "integer" can take the values 3 or 4.

2. This clause may be specified when records of the same type participate in a local set-type and are stored in several areas with different local pointer sizes. Since pointers within the same storage record-type must be the same size, the user may force the smaller pointer size to that of the larger.

3. For further details, see the paragraph Set implementation by a Chain of Pointers in this section.

2.9.3 RECORD Entry

Function

To declare attributes of the storage record which corresponds to a schema record.

General Format

```
RECORD NAME IS record-name

  [RANGE  [WITHIN area-name] IS {PAGE p-1 THRU p-2     } ]
  [                             {m PAGES FROM PAGE p-1 } ]
  [                                                      ] ...
            [          {HIGH}]                           ]
            [OPTIMIZE {     }]                           ]
            [          {LOW }]                           ]

[MIGRATION IS ALLOWED]  .
```

Rules

1. This entry is optional. If it is not specified, the default options of each clause are assumed.

2.9.3.1 RECORD Clause

Function

To name the schema record-type which is mapped onto the storage record-type whose attributes are described in the other clauses of this entry.

Format

```
  ______________________________
 |                              |
 | RECORD NAME IS record-name   |
 |______________________________|
```

Rules

1. "record-name" is the DDL name of a schema record-type.

2. The DMCL name of the storage record-type is implicitly the same as the DDL name of the schema record-type.

2.9.3.2 RANGE Clause

Function

To specify, for each of the storage areas where the storage record may be located, the range within which the record is to be placed.

Format

```
                                {PAGE p-1 THRU p-2     }
RANGE [WITHIN area-name] IS  {                      }
                                {m PAGES FROM PAGE p-1}

              [          {HIGH} ]
              [OPTIMIZE {     } ]
              [          {LOW } ]
```

Rules

1. If the clause is omitted, the range is assumed to be equal to the area itself.
2. If this clause is specified, the location of the storage record is restricted to the range specified. There can be only one range per record-type in a given area.
3. The WITHIN phrase is required when the record-type is multi-area. It is optional if the record-type is mono-area.
4. The range may be described as either:

   - A range of pages specified by a first page number and a last page number (PAGE THRU). If NUMP is the number of pages of the area, the integers p-1 p-2 must satisfy the condition:

     ```
     0 <= p-1 <= p-2 <= NUMP - 1
     ```

   - A number of pages beginning at a specified page number (PAGES FROM PAGE). The integers m and p-1 must satisfy the conditions:

     ```
     m>O
     O<= p-1 <= NUMP - 1
     p-1 + m <= NUMP
     ```

5. If the record-type is CALC, the first page of the range must coincide with the first page of a CALC bucket.
6. If OPTIMIZE is not specified for a CALC record-type, the number of pages of the range must be a multiple of the CALC-INTERVAL.

   It is recommended that the number of buckets (that is the number of pages divided by the CALC-INTERVAL) should be a prime number or that at least it should not possess small factors: 2, 3, 5, 7, 9, 11, 13, 17, 19.

7. If OPTIMIZE is specified for a CALC record-type, DDLPROC increments (HIGH) or decrements (LOW) the number of pages specified in order to obtain the next prime number for the number of buckets.

   If OPTIMIZE HIGH causes the range to overflow the area, no optimization occurs. The value selected is the value specified, provided that it is a multiple of the CALC-INTERVAL.

8. If two CALC record-types are to be located in two adjacent ranges within the same area, the use of the OPTIMIZE function may cause a slight overlap of the two ranges. As the overlap is not desirable, the range starting addresses may be adjusted in a second run of DDLPROC.

9. The OPTIMIZE phrase must not be specified for non-CALC record-types.

Example

A data base consists of 2 areas (AA, BB), 2 record-types (R01, R02), and 1 set-type (S01) whose owner is R01 and member R02.

R01 is CALC within AA and BB.

R02 is located VIA S01 WITHIN AREA OF OWNER.

Area AA contains 500 pages. Its CALC-INTERVAL is 1. R01 is located in pages 0 thru 102 (103 is prime), R02 in pages 103 thru 499.

Area BB contains 1500 pages. Its CALC-INTERVAL is 1. R01 is located in pages 0 thru 306 (307 is prime), R02 in pages 307 thru 1499.

The DMCL record entries are as:

```
RECORD R01
   RANGE WITHIN AA IS PAGE O THRU 102.
   RANGE WITHIN BB IS PAGE 0 THRU 306.

RECORD R02
   RANGE WITHIN AA IS PAGE 103 THRU 499.
   RANGE WITHIN BB IS PAGE 307 THRU 1499.
```

2.9.3.3 MIGRATION Clause

Function

To specify whether the CALC storage record may be moved to the new CALC bucket as a result of a MODIFY CALC-key function.

Format

```
 ________________________
|                        |
| MIGRATION IS ALLOWED   |
|________________________|
```

Rules

1. This clause must not be specified for non-CALC records.

2. If this clause is specified for a CALC record, the record will be moved to the new CALC bucket during a MODIFY CALC-key function. All the set pointers pointing to this record will be updated. If the data-base-key of the record is a sort-key item in a set, this set will be reordered.

3. If this clause is not specified for a CALC record, the record will not be moved during a MODIFY CALC-key function. The function may fail, however, if an overflow link record (OWL) corresponding to the new CALC chain has to be created in the page of the record and there is no room left.

2.9.4 SET Entry

Function

To specify the implementation of a schema set-type.

General Format

```
 ____________________________
|                            |
| SET NAME IS set-name       |
|                            |
|    [NO LOCAL POINTERS].    |
|____________________________|
```

Rules

1. This entry is optional. If it is not specified, the default options of each clause are assumed.

2.9.4.1 SET Clause

Function

To name the schema set-type whose implementation is described in the other clauses of this entry.

Format

```
 ________________________
|                        |
| SET NAME IS set-name   |
|________________________|
```

Rules

1. "set-name" is the DDL name of a schema set-type.

2.9.4.2 NO LOCAL POINTERS

Function

To force the global pointer implementation of the set-type.

Format

```
 _____________________
|                     |
| NO LOCAL POINTERS   |
|_____________________|
```

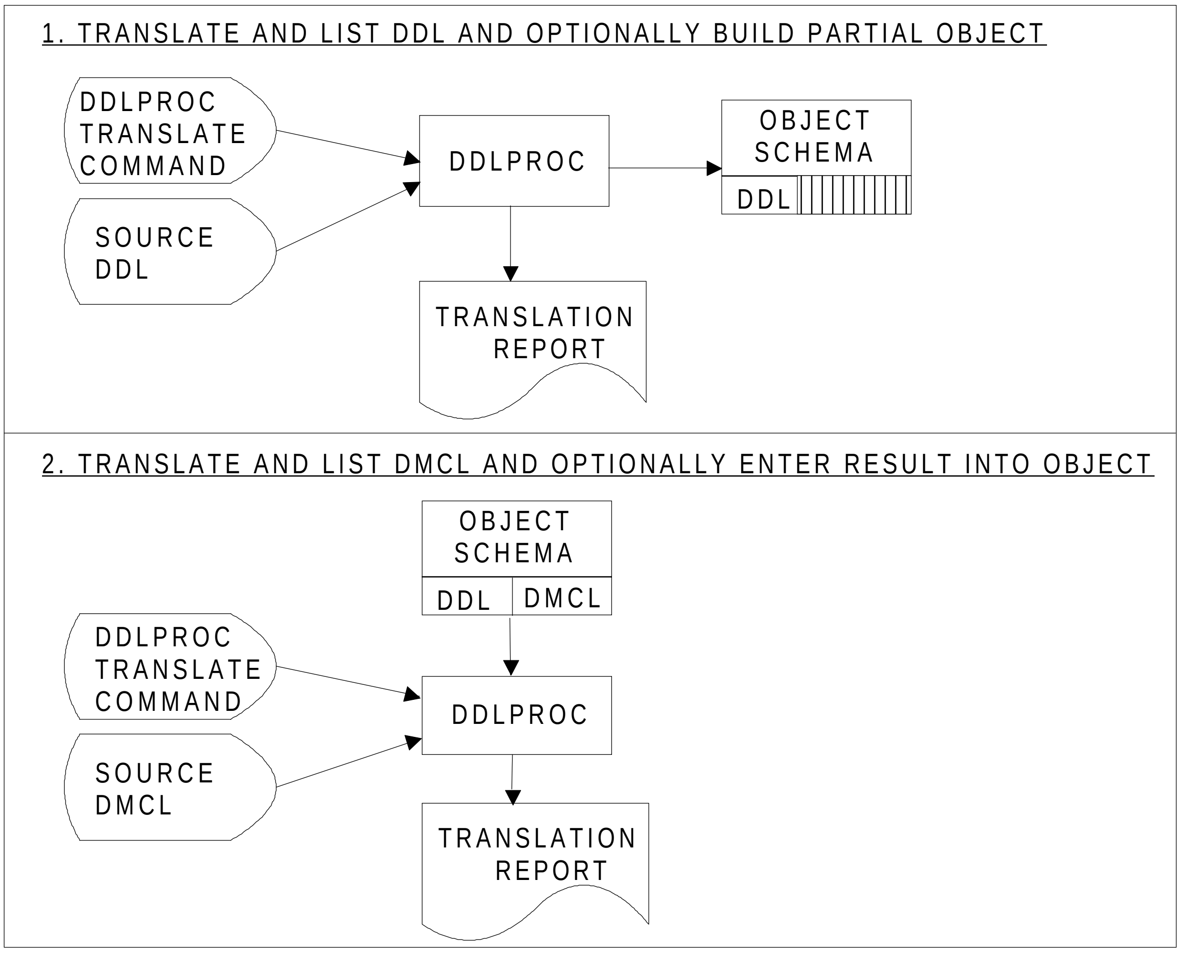
Rules

1. This clause must not be specified for a global set-type.
2. If this clause is omitted and the set-type is local, the set will be implemented with local pointers, unless the set-type is multi-area and the local pointer sizes are unequal or a NO LOCAL POINTERS clause is specified in the schema entry.
3. If this clause is specified and the set-type is local, the set will be implemented with global pointers.
4. For further details, see the paragraph Set Implementation by a Chain of Pointers in this section.

3. Schema Processing

3.1 DDLPROC FUNCTIONS

When the DDL and DMCL are written they must then be processed into an Object Schema by the IDS/II processor DDLPROC. This processor performs five different elementary functions, as shown in Figure 3-1.



1. TRANSLATE AND LIST DDL AND OPTIONALLY BUILD PARTIAL OBJECT
DDLPROC TRANSLATE COMMAND
SOURCE DDL
DDLPROC
OBJECT SCHEMA
DDL
TRANSLATION REPORT
2. TRANSLATE AND LIST DMCL AND OPTIONALLY ENTER RESULT INTO OBJECT
OBJECT SCHEMA
DDL
DMCL
DDLPROC TRANSLATE COMMAND
SOURCE DMCL
DDLPROC
TRANSLATION REPORT

Figure 3-1. DDLPROC Functions (1/2)

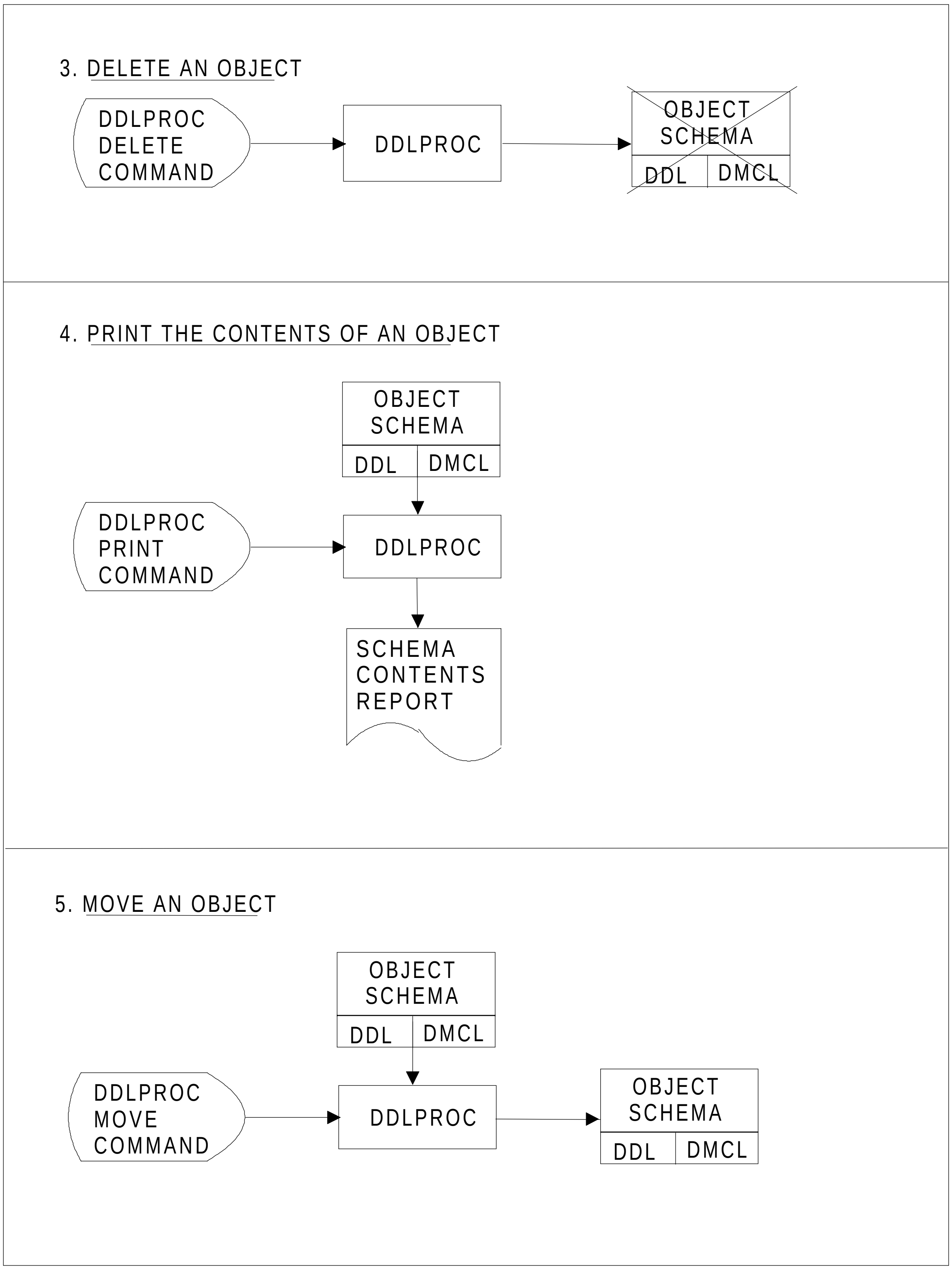


Figure 3-1. DDLPROC Functions (2/2)

3.2 SL AND DD OBJECT USAGE

DDLPROC handles two types of object:

- SL (Source Language) objects corresponding to the schema DDL, DMCL source.
- DD (Data Description) objects corresponding to the schema in object format.

SL objects can reside in:

- Input-enclosures with the TYPE option DATA or DATASSF.
- Members of libraries of type SL. The TYPE option of these members must be DATASSF. Libraries of type SL are preallocated using LIBALLOC SL:

```
LIBALLOC SL  (library-description SIZE=integer)
         [MAXSIZE=integer]
         {DIRSIZE=integer}
         {MEMBERS=integer}
         [COMPACT] ;
```

DD objects reside in members of libraries of type BIN. These are allocated using LIBALLOC BIN:

```
LIBALLOC BIN  (library-description SIZE=integer)
    [MACSIZE=integer]
    {DIRSIZE=integer}   ;
    {MEMBERS=integer}
```

There is no connection between the name X of a schema, as declared in the DDL SCHEMA NAME clause, and the name of the input-enclosure or library member that contains its DDL or DMCL source.

The following convention may be adopted, if the name of the schema is not too long:

- Name of container of DDL source: X-DDL (input-enclosure, member)
- Name of container of DMCL source: X-DMCL (input-enclosure, member)

The number of characters allowed must not exceed:

- 30 for a schema-name.
- 16 for an input-enclosure.
- 31 for a member.

The name of the member containing the object schema must be identical with the name X of the schema, as declared in the DDL SCHEMA NAME clause:

- Name of container of object schema: X (member)

  If different schemas corresponding to the same DDL schema-name are required (initial loading, test, production) they must be located in different libraries since member-names are unique within a library.

Object schemas may be in two states:

- The "DDL" state. This state corresponds to the end of a successful DDL source translation. An object schema in this state is accessible only to COBOL and QUERY for compiling programs/requests.

- The "DDL-DMCL" state. This state corresponds to the end of a successful DMCL source translation. An object schema in this state is accessible to COBOL and QUERY for compiling programs, and to user steps, IDS/II utilities and QUERY for executing DML programs/commands/requests.

Input-enclosures containing SL objects need not be assigned statically since DDLPROC assigns them dynamically (through the ifn INPENC). Such SL objects are referenced in the DDLPROC command language by the name of the input-enclosure, immediately preceded by an *.

Libraries containing SL objects must be assigned statically using the LIB SL INLIB1, INLIB2, INLIB3 in the extended JCL or the reserved ifn's SLLIB1, SLLIB2, SLLIB3 in the basic JCL. In both cases the ifn's used are SLLIB1, SLLIB2, SLLIB3.

SL objects are referenced in the DDLPROC command language by:

- The name of the containing member. In this case the SL object is searched in the libraries assigned, in the order defined by the sequence SLLIB1, SLLIB2, SLLIB3.

- The name of the containing member qualified by the efn of a library or one of the keywords SLLIB1, SLLIB2, SLLIB3. In this case the SL object is searched in the library specified. If libraries have identical efn's but reside on different devices, the qualification by a keyword SLLIBi is required.

Libraries containing DD objects must be assigned statically using the keywords DDLIB1, DDLIB2, DDLIB3 in the extended JCL or the reserved ifn's DDLIB1, DDLIB2, DDLIB3 in the basic JCL. In both cases the ifn's used are DDLIB1, DDLIB2, DDLIB3.

DD objects have the same name as that of their container. They arereferenced in the DDLPROC command language by:

- The name of the schema (explicit or implicitly derived from the DDL, DMCL source). If the DD object is processed in retrieval or update mode, it is searched in the libraries assigned, in the order defined by the sequence DDLIB1, DDLIB2, DDLIB3. If the DD object is processed in creation mode, it is created in the first library assigned, in the order defined by the sequence DDLIB1, DDLIB2, DDLIB3.

- The name of the schema (explicit or implicit) qualified by the efn of a library or any of the keywords DDLIB1, DDLIB2, DDLIB3. In this case, the DD object is searched or created in the library specified. If libraries have identical efn's but reside on different devices, the qualification by a keyword DDLIBi is required.

The previous considerations are summarized in Figure 3-2.

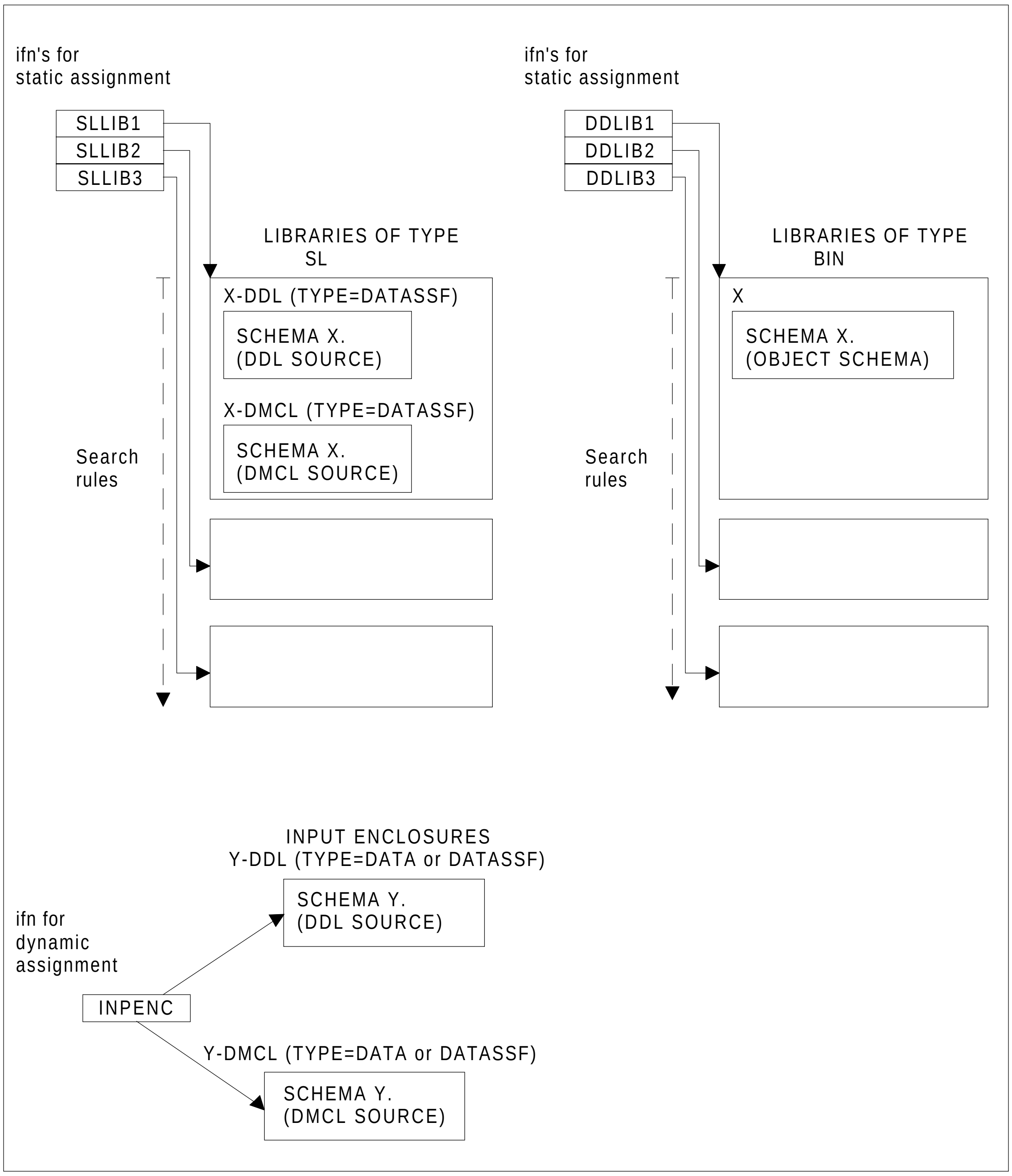


Figure 3-2. DDLPROC Access to SL and DD Objects

3.3 DDLPROC EXTENDED JCL STATEMENT

DDLPROC can be called using the extended JCL statement:

```
[LIB SL INLIB1 = (library-description)    ]
[      [INLIB2 = (library-description)]   ]
[      [INLIB3 = (library-description)] ; ]

DDLPROC

   {          {*input-enclosure-name       }}
   {COMFILE = {                            }}
   {          {(sequential-file-description)}}
   {                                         }
   {                                         }
   {COMMAND = ' {command} ... '              }

   [DDLIB1 = (library-description)]
   [DDLIB2 = (library-description)]
   [DDLIB3 = (library-description)]
   [PRTFILE = (print-file-description)]
   [PRTDEF = (define-parameters)]
   [PRTOUT = (sysout-parameters)]  ;
```

3.3.1 DDLPROC Parameter Description

LIB SL	This statement defines the libraries containing SL objects and their search path.
COMFILE	This keyword is followed by a parameter group defining the input file containing the DDLPROC commands. Input can be from an input-enclosure (TYPE=DATA or TYPE=DATASSF) or from a library member (TYPE=DATASSF) or from a sequential file.
COMMAND	This keyword is used instead of COMFILE to introduce DDLPROC commands directly.
DDLIBi	These keywords define the libraries containing DD objects and their search path.
PRTFILE PRTDEF PRTOUT	These keywords allow a redefinition of the standard options applied to the DDLPROC report.

3.4 DDLPROC BASIC JCL

DDLPROC can also be called using the basic JCL:

```
STEP H_DDLPROC SYS [step-option] ...

{ OPTIONS = ' {command} ...' ;                       }
{ ; ASSIGN COMFILE {*input-enclosure-name      }   }
{                  {sequential_file-description} ; }


[SIZE 100 ; ]
[ASSIGN SLLIB1 library-description;]
[ASSIGN SLLIB2 library-description;]
[ASSIGN SLLIB3 library-description;]
[ASSIGN DDLIB1 library-description;]
[ASSIGN DDLIB2 library-description;]
[ASSIGN DDLIB3 library-description;]
[ASSIGN PRTFILE print-file-description;]
[DEFINE PRTFILE define-parameters;]
[SYSOUT PRTFILE SYSOUT-parameters;]
ENDSTEP;
```

3.4.1 Parameter Description

The correspondence between the extended JCL and the basic JCL is straightforward. The slight differences are listed below.

The COMMAND keyword is replaced by the OPTIONS string.

The LIB SL statement is replaced by the ASSIGN SLLIBi.

3.5 DDLPROC COMMAND LANGUAGE

The syntax rules and descriptive notation are practically the same as those described for the DDL statements in Section I.

A command is an entry composed of several clauses and terminated with a period followed by a space or an end-of-line. Unlike DDL entries, a DDLPROC command can also be terminated with a semicolon (";"). The semicolon can no longer play the role of punctuation character elsewhere in the text. Clauses may appear in any order except for the first which identifies the entry.

A library efn which does not comply with the rules for DDL names:

- Can be written without protection if the only illegal characters it contains are periods (example: PRINT SCHEMA INVENTORY OF MIDS.OD).
- Must be written in the escape form in all other cases.

3.5.1 Reserved Words

DDLPROC reserved words are listed below:

CHECK	QUERY
COBOL	RECORD (RECORDS)
COMPACT	REPLACE
CONVERSION	REPORT
DATA	RESET
DDL	RETRIEVAL
DDLIB1	SCHEMA
DDLIB2	SEVERITY
DDLIB3	SLLIB1
DEBUG	SSLIB2
DELETE	SSLIB3
DESCRIPTION	SOURCE
DMCL	STATEMENTS
DML	STATUS
EXCEPT	STORAGE
FOR	TIME
FORCED	TO
IN (OF)	TRANSLATE
IS	UPDATE
MOVE	USE
OBJECT	UWA
OF (IN)	XREF
PRINT	

3.5.2 Types of Command

There are six types of command. Five reflect the five functions shown in Figure 3-1. The last one is related to the command flow when several commands are submitted in the same run.

The commands:

- `TRANSLATE SCHEMA DDL`
- `TRANSLATE DMCL`
- `PRINT SCHEMA`
- `DELETE SCHEMA`
- `MOVE SCHEMA`
- `STATUS`

are described in detail on the following pages.

3.5.3 TRANSLATE SCHEMA DDL Command

Function

To translate the schema DDL source and optionally create an object schema in the "DDL" state.

General Format

```
TRANSLATE SCHEMA DDL

           {*input-enclosure-name                          }
           {                        [       {library-efn} ] }
SOURCE IS  {                        [{IN}   {SLLIB1     } ] }
           {member-name             [{  }   {SLLIB2     } ] }
           {                        [{OF}   {SLLIB3     } ] }

[       [     {library-efn} ]              ]
[OBJECT [{IN} {DDLIB1     } ]              ]
[       [{OF} {DDLIB2     } ]   [REPLACE] ]
[       [     {DDLIB3     } ]              ]

[REPORT SEVERITY integer]
[       XREF            ]

[FORCED DATE IS {yy mm dd} TIME IS hh mm]
[               {yy ddd  }              ]

              [FOR CONVERSION]

[CHECK FOR COBOL   USE]
[          QUERY      ] .
```

Rules

1. The SOURCE clause defines the input-enclosure or library member containing the DDL statements.

2. If the OBJECT clause is specified, an object schema in the "DDL" state is created, provided that the DDL source is correct.

   The output library may be specified if the default search rules for creation are not appropriate.

   If the operation is to replace an existing member of the same name currently in the output library, then the keyword REPLACE must be coded in the OBJECT clause. Otherwise an error will result.

3. If the OBJECT clause is not specified, no object schema is created. Only a DDL listing is produced.

4. If the REPORT clause is specified, some listing options are provided. The SEVERITY option specifies the minimum level of diagnostics to be produced. There are five levels numbered 0, 1, 2, 3, 4. Level 0 corresponds to messages such as "SYNTAX ANALYSIS RESUMED"... Levels 1 and 2 correspond to warnings. Level 3 corresponds to syntax fatal error messages and level 4 to environment fatal error messages.

   "Integer" may take the values 0, 1, 2, 3. For example, if SEVERITY 3 is coded, only the fatal error messages will appear.
   If the SEVERITY option is not selected, level 0 is assumed.
   The XREF option requests that a cross-reference listing of user-defined names be produced.
   If the XREF option is not selected, no cross-reference listing is produced.

5. If the REPORT clause is not specified, SEVERITY 0 and no XREF are assumed.

6. If the FORCED DATE clause is not specified, the object schema created is given a DDL reference date-time equal to the current date-time.

   Note that the run-time consistency checks between the DDL reference date-time of a schema and the DDL reference date-time of the programs compiled against it are effective only if this clause is omitted.

7. If the FORCED DATE clause is specified, the object schema created is given the date-time indicated.

   The date can be specified as either year, month and day (yy mm dd) or as day number within year (yy ddd). The time is specified as hours and minutes (hh mn). Leading zeroes may be omitted.

   The fact that the date-time has been forced is recorded in the object schema.

   When a schema used under 1D is retranslated under GCOS 7 for compatibility reasons, the phrase FOR CONVERSION may be specified if the reference date of the 1D schema was not forced. In this case the fact that the date-time is forced is not recorded in the schema.

   Apart from the 1D to GCOS 7 conversion, the use of the FORCED DATE clause must be restricted to the following cases:

   - The object schema is destroyed and there are compiled programs which reference the schema. In order for these programs to continue to work without recompilation, the DDL source is retranslated with the old DDL reference date-time.
   - The same DML programs are run against a production data base and a test data base of a smaller size. The test schema has the same DDL as the production schema and must be stamped with the same DDL reference date-time. Its DMCL is different, reflecting the differences in area sizes. Note that the FORCED DATE option may be avoided if the production schema is first moved to another library (see MOVE command) and the test DMCL is applied to the copy.

8. If the CHECK clause is specified, DDLPROC will check that the user-defined names are not reserved words for COBOL or for QUERY.

9. If the CHECK clause is not specified, no check is performed.

10. Figure 3-3 shows a TRANSLATE SCHEMA DDL listing with some errors.

```
    DDLPROC    30.00  X1167.2   DOCLIST  FAYOT         IDS2         IDS2__R5     18:15:33
                MAY 22, 1980      PAGE       4
      TRANSLATE SCHEMA COMMAND   REPORT           SOURCE LISTING AND ERRORS
         ILN      XLN     TEXT


           1       10  COMMENT "***************************************************"
           2       20  COMMENT "*           SCHEMA DDL FOR TEST-GROUP *TG19       *"
           3       30  COMMENT "***************************************************"
           4       40
           5       50  COMMENT "********************************"
           6       60  COMMENT "*            SCHEMA            *"
           7       70  COMMENT "********************************"
           8       80
           9       90  SCHEMA TG19-SH.
          10      100
          11      110  COMMENT "********************************"
          12      120  COMMENT "*            AREAS             *"
          13      130  COMMENT "********************************"
          14      140
          15      150  AREA TG19-A00.
          16      160  AREA TG19-A01.
          17      170
          18      180  COMMENT "********************************"
          19      190  COMMENT "*            RECORDS           *"
          20      200  COMMENT "********************************"
          21      210
          22      220  COMMENT "************************"
          23      230  COMMENT "*     R01  (PART)      *"
          24      240  COMMENT "************************"
          25      250  RECORD R01
          26      260       LOCATION CALC  USING R01-NUM DUP NOT
          27      270       WITHIN ANY AREA .
                            1
***  1  3-03-52  AREA -ID PHRASE MISSING

          28      280    02         R01-IDENT     TYPE CHAR 7.
          29      290    02         R01-NUHN      TYPE CHAR 5.
          30      300    02         R01-NUM       TYPE UNPACKED DEC 3.
          31      310    02         R01-TRAILER   TYPE CHAR 4.
          32      320
          33      330  COMMENT "************************"
          34      340  COMMENT "*     R02 (COMPONENT)  *"
          35      350  COMMENT "************************"
          36      360  RECORD R02
          37      370       LOCATION VIA S01
          38      380       WITHIN AREA OF OWNNER
          39      390    02         R02-IDENT     TYPE CHAR 7.
                         1                                 2
***  1  3-00-02  CLAUSE NOT RECONNIZED WITHIN THE CURRENT (SUB)ENTRY - DATA SKIPPED TILL
                 NEXT RECORD SUBENTRY CLAUSE OR
                 PERIOD
     2  0-00-50  SYNTAX ANALYSIS RESUMED AT THIS POINT

          40      400    02         R02-R01-UPH   TYPE CHAR 4.
          41      410    02         R02-R01-UP    TYPE UNPACKED DEC 3.
          42      420    02         R02-R01-DOWNH TYPE CHAR 6.
          43      430    02         R02-R01-DOWN  TYPE UNPACKED DEC 3.
          44      440    02         R02-QTYH      TYPE CHAR 5.
          45      450    02         R02-QTY       TYPE UNPACKED 3.
                                                                12
***  1  5-07-09  MANDATORY KEYWORD MISSING : DECIMAL  -  TYPE CLAUSE ANALYSIS DISCONTINUED
```

Figure 3-3. TRANSLATE SCHEMA DDL Listing (1/2)

```
    DDLPROC    30.00  X1167.2   DOCLIST  FAYOT         IDS2          IDS2__R5
                  18:15:33MAY 22, 1980      PAGE       5
      TRANSLATE SCHEMA COMMAND   REPORT           SOURCE LISTING AND ERRORS
         ILN      XLN     TEXT

***  2  0-00-50  SYNTAX ANALYSIS RESUMED AT THIS POINT

          46      460    02         R02-TRAILER   TYPE CHAR 4.
          47      470
          48      480  COMMEMT "*************************************"
          49      490  COMMENT "*             SETS                    *"
          50      500  COMMENT "*************************************"
          51      510
          52      520  COMMENT "**************************"
          53      530  COMMENT "*     S01 (CALL-OUT)     *"
          54      540  COMMENT "**************************"
          55      550  SET S01
          56      560       OWNER  R01
          57      570       ORDER  PERMANENT INSERTION SORTED DEFINED KEYS DUP NOT.
          58      580   MEMBER R02
          59      590       INSERTION AUTO RETENTION MAND
          60      600       KEY AS CENDING R02-R01-DOWN
          61      610       SELECTION THRU S01
          62      620       OWNER IDENTIFIED BY CALC-KET R01-NUM EQUAL TO R02-R01-UP.
          63      630
          64      640  COMMENT "**************************"
          65      650  COMMENT "*     S02 (WHERE-USED)  *"
          66      660  COMMENT "**************************"
          67      670  SET S02
          68      680       OWNER  R01
          69      690       ORDER  PERMANENT INSERTION SORTED DEFINED KEYS .
                                                                            1
***  1  4-03-07  MANDATORY KEYWORD MISSING : DUPLICATES DISPOSITION-ORDER CLAUSE ANALYSIS
                 DISCONTINUED
     1  0-00-50  SYNTAX ANALYSIS RESUMED AT THIS POINT

          70      700   MEMBER R02
          71      710       INSERTION AUTO RETENTION MAND
          72      720       KEY R02-R01-UP
                                1
***  1  6-04-02  MANDATORY KEYWORD MISSING : ASCENDING/DESCENDING - KEY CLAUSE ANALYSIS
                 DISCONTINUED

          73      730       SELECTION THRU S02
                            1
     1  0-00-50  SYNTAX ANALYSIS RESUMED AT THIS POINT
                            1
          74      740       OWNER IDENTIFIED BY CALC-KEY R01-NUM EQUAL TO R02-R01-DOWN
          75      750                                  AREA-ID EQUAL TO OTHER-ARID1.
          76      760
          77      770  END-SCHEMA.
```

Figure 3-3. TRANSLATE SCHEMA DDL Listing (2/2)

3.5.4 TRANSLATE DMCL Command

Function

To translate the schema DMCL source and optionally enter the result in the object schema.

General Format

```
TRANSLATE DMCL

          {*input-enclosure-name                }
          {               [        {library-efn}] }
          {member-name    [{IN}]  {SLLIB1     }] }
SOURCE IS {               [{  }]  {           }] }
          {               [{OF}]  {SLLIB2     }] }
          {               [       {SLLIB3     }] }

[       {     {library-efn}]
[SCHEMA {IN}  {DDLIB1     }]
[       {     {DDLIB2     }]
[       {OF}  {DDLIB3     }]

 [OBJECT [REPLACE]]
 [REPORT SEVERITY integer]

[                {yy mm dd}                  ]
[FORCED DATE IS  {yy ddd  }  TIME IS hh mm   ]
[                                            ]
[          [FOR CONVERSION]                  ]
```

Rules

1. The SOURCE clause defines the input-enclosure or library member containing the DMCL statements.

2. If the SCHEMA clause is not specified, the object schema, whose name is derived from the SCHEMA NAME clause of the DMCL, is retrieved according to the search rules.

3. If the SCHEMA clause is specified, the object schema is taken from the library specified.

4. Before processing, the object schema may be in the "DDL" state or in the "DDL-DMCL" state.

5. If the OBJECT clause is specified, the object schema is updated with the DMCL information, provided that the DMCL source is correct.

   If the object schema was already in the "DDL-DMCL" state, (i.e. contained DMCL information), the keyword REPLACE must be specified.

   After a successful processing, the object schema is left in the "DDL-DMCL" state.

6. If the OBJECT clause is not specified, the object schema is left unchanged. Only a DMCL listing is produced.

7. If the REPORT clause is specified, the diagnostics produced are those whose level is equal to or greater than "integer". (See details under TRANSLATE SCHEMA DDL command).

8. If the REPORT clause is not specified, SEVERITY 0 is assumed.

9. If the FORCED DATE clause is not specified, the object schema updated is given a DMCL reference date-time equal to the current date-time.

Note that the run-time consistency checks between the DMCL reference date-time of a schema and the DMCL reference date-time of the areas preallocated against it are effective only if this clause is omitted.

10. If the FORCED DATE clause is specified, the object schema updated is given the DMCL reference date-time indicated.

    The date can be specified as either year, month and day (yy mm dd) or as day number within year (yy ddd). The time is specified as hours and minutes (hh mm). Leading zeroes may be omitted.

    The fact that the date-time has been forced is recorded in the object schema.

    When a schema used under 1D is retranslated under GCOS 7 for compatibility reasons, the phrase FOR CONVERSION may be specified if the reference date of the 1D schema was not forced. In this case the fact that the date-time is forced is not recorded in the schema.

    Apart from the 1D to GCOS 7 conversion, the use of the FORCED DATE clause must be restricted to the following cases:

    - The object schema is destroyed and there are loaded areas which reference the schema. In order that these areas can continue to be used without reloading, the DMCL source is retranslated with the old DMCL reference date-time.

    - In order to speed up the initial loading of the data base, a loading schema is designed whose DDL differs from the DDL of the production schema by some set selection or set ordering criteria. As it references the production areas, the loading schema contains the same DMCL and must be stamped with the DMCL reference date-time of the production schema.

11. Figure 3-4 shows a TRANSLATE DMCL Listing with some errors.

```
     DDLPROC    30.00  X1167.2   DOCLIST  FAYOT       IDS2         IDS2__R5     18:15:33
                MAY 22, 1980      PAGE       8
       TRANSLATE DMCL  COMMAND  REPORT           SOURCE LISTING AND ERRORS
          ILN      XLN     TEXT


            1       10  COMMENT "***************************************************"
            2       20  COMMENT "*       SCHEMA DMCL FOR TEST-GROUP *TG19*          *"
            3       30  COMMENT "***************************************************"
            4       40
            5       50  COMMENT "********************************"
            6       60  COMMENT "*            SCHEMA            *"
            7       70  COMMENT "********************************"
            8       80
            9       90  SCHEMA TG19-SH.
           10      100
           11      110  COMMENT "********************************"
           12      120  COMMENT "*            AREAS             *"
           13      130  COMMENT "********************************"
           14      140
           15      150  AREA TG19-A00
           16      160       NUMBER-OF-PAGES 53.
           17      180       PAGE-SIZE 2040.

  ***     2-00-24  MANDATORY LINES-PER-PAGE CLAUSE MISSING

           18      190
           19      200  AREA TG19-A01
           20      210       NUMBER-OF-PAGES 52.
           21        0       CALC-INTERVAL 3.
                                           1

  *** 1   2-06-60  CALC-INTERVAL VALUE MUST DIVIDE THE AREA NUMBER OF PAGES

           22      220       LINES-PER-PAGE 45
           23      230       PAGE-SIZE 2040.
                                       1   2
  *** 1   2-05-17  PAGE-SIZE VALUE MUST BE AN INTEGER NUMBER OF TIMES 256 BYTES - CLAUSE
                   IGNORED
      2   0-00-50  SYNTAX ANALYSIS RESUMED AT THIS POINT

           24      240
           25      250  END-DMCL.
```

Figure 3-4. TRANSLATE DMCL Listing

3.5.5 PRINT SCHEMA Command

Function

To provide a report on the storage characteristics of the data base and a report on the DML programming rules.

General Format

```
                         [      {library-efn}]
PRINT SCHEMA schema-name [{IN} {DDLIB1     }]
                         [{OF} {DDLIB2     }]
                         [      {DDLIB3     }]

[STORAGE DESCRIPTION [COMPACT]                          ]
[                                                       ]
[    [FOR RECORDS [EXCEPT] {record-name} ...]           ]
[                                                       ]
[            UWA DESCRIPTION                            ]
[     DML   [ [UPDATE   ]            STATEMENTS ]       ]
[           [ [RETRIEVAL]                       ]       ]
[                                                       ] .
[      [FOR RECORDS [EXCEPT]  {record-name} ...]        ]
```

Rules

1. If the STORAGE DESCRIPTION clause is specified, a report giving the storage characteristics of the data base is produced.

   This report contains:

   - general information about the reference dates of the schema, the number of areas, record-types, set-types, fields, the size and segmentation of the schema in main memory, the size and format of the global pointers of the data base, etc.

   - information specific to a given storage record: code number of the record, code numbers of the sets in which it participates, code numbers of its fields, lay out of the storage record (header, pointer zone and data zone), characteristics of the storage areas containing the record.

2. If the FOR RECORDS phrase is not specified, the STORAGE report provides information on all the records of the schema.

3. If the FOR RECORDS phrase is specified, the STORAGE report provides information on:

   - the records in the list, if EXCEPT is not coded.

   - the records in the schema except for those of the list, if EXCEPT is coded.

4. If the COMPACT keyword is not coded, the information on repeating groups and vectors is given for each occurrence of their components.

5. If the COMPACT keyword is coded, the information on repeating groups and vectors is not detailed for each occurrence of their components.

6. The length of a group represents the length of one occurrence of this group, whether COMPACT is coded or not.

   The offset of a group represents the offset of the occurrence of this group if COMPACT is not coded and the offset of the first occurrence of this group (value 1 for all indexes) if COMPACT is coded.

7. If the DML clause is specified, a report giving the COBOL programming rules for this schema is produced.

   The report contains, on a record-type basis, two kinds of information:

   - The COBOL description in the UWA record and the sets in which it participates.

   - The COBOL DML verbs available to process this record and the various fields to be initialized prior to their use.

8. If the UWA phrase is specified, the report contains the COBOL record description and miscellaneous control information.

9. If the STATEMENTS phrase is specified, the report contains the COBOL DML verbs.

   If UPDATE is coded, the update verbs are listed.

   If RETRIEVAL is coded, the retrieval verbs are listed.

   If neither UPDATE nor RETRIEVAL is coded, all verbs are listed.

10. If neither the UWA nor the STATEMENTS phrase is specified, the report contains the COBOL descriptions and all the DML verbs.

11. If the FOR RECORDS phrase is not specified, the DML report provides information on all the records of the schema.

12. If the FOR RECORDS phrase is specified, the DML report provides information on:

    - the records of the list if EXCEPT is not coded.

    - the records of the schema except for those of the list if EXCEPT is coded.

13. If neither the STORAGE clause nor the DML clause is specified, both reports are produced with all options.

14. The STORAGE report is available only if the object schema is in the "DDL-DMCL" state. The DML report is available whether the object schema is in the "DDL" or "DDL-DMCL" state.

15. Figure 3-5 shows a STORAGE report listing without the COMPACT option. Figure 3-6 shows a STORAGE report listing with the COMPACT option. Figure 3-7 shows a DML report listing.

```
___________________________________________________________________________________
|                                                                                 |
|                                                                                 |
|                                                                                 |
|         * SCHEMA TG05-SH *            * PRINT STORAGE REPORT OF RECORD R17 *    |
|                                                                                 |
|                                                                                 |
|                                                                                 |
|                                                                                 |
|RECORD CODE NUMBER : 17                                                          |
|====================                                                             |
|                         S T O R A G E  R E C O R D  L A Y - O U T               |
|**********************************************************************************
|* LEVEL   * FIELD DESCRIPTION *  CODE  *SIZE IN BYTES* OFFSET(HEXA)* OFFSET(HEXA)* |
|*         *  OR FIELD NAME    * NUMBER * (DECIMAL)   * / RECORD    * /USER DATA  * |
|**********************************************************************************
|* HEADER  * UFAS HEADER       *        *           4 *       0000  *             * |
|* HEADER  * STATUS            *        *           1 *       0004  *             * |
|* POINTER * MEMBERSHIP GLOBAL POINTERS FOR SET:      *             *             * |
|*         *    S07                               7   *           6 *       0005  * |
|**********************************************************************************
|* 1       * R17-IDENT         *     0  *    17       *       000B  *      0000   * |
|*  2      * R17-IDENTH        *     1  *           7 *       000B  *      0000   * |
|*  2      * R17-CH            *     2  *           3 *       0012  *      0007   * |
|*  2      * R17-C             *     3  *           2 *       0015  *      000A   * |
|*  2      * R17-FH            *     4  *           3 *       0017  *      000C   * |
|*  2      * R17-F             *     5  *           2 *       001A  *      000F   * |
|* 1       * R17-S2H           *     6  *           9 *       001C  *      0011   * |
|* 1       * R17-S2GROUP1(1)   *     7  *     5       *       0025  *      001A   * |
|*  2      * R17-ITEM1(1)      *     8  *           3 *       0025  *      001A   * |
|*  2      * R17-S2(1,1)       *     9  *           1 *       0028  *      001D   * |
|*  2      * R17-S2(1,2)       *     9  *           1 *       0029  *      001E   * |
|* 1       * R17-S2GROUP1(2)   *     7  *     5       *       002A  *      001F   * |
|*  2      * R17-ITEM1(2)      *     8  *           3 *       002A  *      001F   * |
|*  2      * R17-S2(2,1)       *     9  *           1 *       002D  *      0022   * |
|*  2      * R17-S2(2,2)       *     9  *           1 *       002E  *      0023   * |
|* 1       * R17-S1H           *    10  *           4 *       002F  *      0024   * |
|* 1       * R17-S1H           *    11  *           1 *       0033  *      0028   * |
|* 1       * R17-FILLER        *    12  *         120 *       0034  *      0029   * |
|* 1       * R17-TRAILER       *    13  *           4 *       00AC  *      00A1   * |
|**********************************************************************************
|*                          HEADER SIZE         :                5           *    |
|*                          POINTER ZONE SIZE   :                6            *   |
|*                          DATA ZONE SIZE      :              165            *   |
|*                          RECORD TOTAL LENGTH :              175            *   |
|**********************************************************************************
|                                                                                 |
|                    C O N T A I N I N G   A R E A ( S )                          |
|**********************************************************************************
|* AREA  NAME  * CODE *PAGE SIZE*  RANGE IN  *    RANGE IN   *  LINES  *LOCAL PTR SIZE* |
|*             *NUMBER* IN BYTES*   PAGES    *   AREA-KEYS   *PER PAGE *IN BYTES* BITS* |
|*************************************************************************************
|* TG05-A01      *    1 *  1024   *    0,120   *   0,1200    *   10    *   2    *  11 * |
|**********************************************************************************
|                                                                                 |
|_________________________________________________________________________________|
```

Figure 3-5. PRINT STORAGE DESCRIPTION Listing

```
          * SCHEMA TG05-SH *              * PRINT STORAGE REPORT OF RECORD R17 *



RECORD CODE NUMBER : 17
====================
                          S T O R A G E  R E C O R D  L A Y - O U T
**********************************************************************************
* LEVEL  *  FIELD DESCRIPTION   *  CODE  *SIZE IN BYTES* OFFSET(HEXA)* OFFSET(HEXA)*
*        *    OR FIELD NAME     * NUMBER * (DECIMAL)   * / RECORD    * /USER DATA  *
**********************************************************************************
* HEADER * UFAS HEADER          *        *           4 *       0000  *             *
* HEADER  * STATUS              *        *           1 *       0004  *             *
* POINTER * MEMBERSHIP GLOBAL POINTERS FOR SET:      6 *             *             *
*         *    S07                                   7 *             *             *
*         *                                                    0005  *             *
**********************************************************************************
* 1      * R17-IDENT              *     0  *    17      *       000B  *      0000   *
* 2      * R17-IDENTH             *     1  *         7 *       000B  *      0000   *
* 2      * R17-CH                 *     2  *         3 *       0012  *      0007   *
* 2      * R17-C                  *     3  *         2 *       0015  *      000A   *
* 2      * R17-FH                 *     4  *         3 *       0017  *      000C   *
* 2      * R17-F                  *     5  *         2 *       001A  *      000F   *
* 1      * R17-S2H                *     6  *         9 *       001C  *      0011   *
* 1      * R17-S2GROUP1           *     7  *         5 *       0025  *      001A   *
* 2      * R17-ITEM1              *     8  *         3 *       0025  *      001A   *
* 2      * R17-S2                 *     9  *         1 *       0028  *      001D   *
* 1      * R17-S1H                *    10  *         4 *       002F  *      0024   *
* 1      * R17-S1                 *    11  *         1 *       0033  *      0028   *
* 1      * R17-FILLER             *    12  *       120 *       0034  *      0029   *
* 1      * R17-TRAILER            *    13  *         4 *       00AC  *      00A1   *
**********************************************************************************
*                                  HEADER SIZE          :               5     *
*                                  POINTER ZONE SIZE    :               6     *
*                                  DATA ZONE SIZE       :             165     *
*                                  RECORD TOTAL LENGTH  :             175     *
**********************************************************************************

                      C O N T A I N I N G   A R E A ( S )
**********************************************************************************
* AREA  NAME  * CODE *PAGE SIZE*   RANGE IN   *   RANGE IN  *  LINES   *LOCAL PTR SIZE*
*             *NUMBER* IN BYTES*   PAGES      *   AREA-KEYS *PER PAGE *IN BYTES* BITS*
**********************************************************************************
* TG05-A01      *  1 *  1024  *    0,120   *      0,1200    *  10    *   2    *  11 *
**********************************************************************************
```

Figure 3-6. PRINT STORAGE DESCRIPTION COMPACT Listing

```
        DDLPROC     20.00  X1864.1   DOCLIST   FAYOT     IDS2      IDS2__R5
                 08:02:18AUG 20, 1980     PAGE      19
           * SCHEMA  TG24-SH *                       * PRINT DML REPORT OF RECORD R16 *


*****************************************************************
*               DML RECORD CHARACTERISTICS                      *
*****************************************************************


********************************
*    UWA RECORD DESCRIPTION    *
********************************

 01 R16.
  02 R16-IDENT PIC X(7).
  02 R16-CREATE-IC .
   03 R16-CREATE-CATEH PIC X(10).
   03 R16-CREATE-CATE PIC 9(6).
   03 R16-CREATE-TIMEH PIC X(10).
   03 R16-CREATE-TIME PIC 9(8).
   03 R16-CREATE-TESTH PIC X(10).
   03 R16-CREATE-TEST PIC X(4).
   03 R16-CREATE-SIMUH PIC X(10).
   03 R16-CREATE-SIMU PIC X(12).
  02 R16-MODIFY-IC .
   03 R16-MODIFY-CATEH PIC X(10).
   03 R16-MODIFY-CATE PIC 9(6).
   03 R16-MODIFY-TIMEH PIC X(10).
   03 R16-MODIFY-TIME PIC 9(8).
   03 R16-MODIFY-TESTH PIC X(10).
   03 R16-MODIFY-TEST PIC X(4).
   03 R16-MODIFY-SIMUH PIC X(10).
   03 R16-MODIFY-SIMU PIC X(12).
  02 R16-CALCH PIC X(6).
  02 R16-CALC.
   03 R16-CALC-OCC PIC 9(2).
   03 R16-CALC-NUM PIC 9(3).
  02 R16-MODIFY-DATAH PIC X(10).
  02 R16-MODIFY-DATA PIC 9(8).
  02 R16-TRAILER PIC X(4).

************************
*    CONTROL FIELDS    *
************************

CALC-KEYS ITEMS (CUP NOT)
     R16-CALC-OCC
     R16-CALC-NUM
SORT KEY ITEMS(DUF NOT) IN SET S10 :
     R16-CALC-OCC                                  ASCENDING
     R16-CALC-NUM                                  ASCENDING
SORT KEY ITEMS(DUP NOT) IN SET S11 :
     R16-CALC-OCC                                  DESCENDING
     R16-CALC-NUM                                  DESCENDING

***************************
*    SET PARTICIPATION    *
***************************

OWNER OF SETS :
```

Figure 3-7. PRINT DML Listing (1/8)

```
            DDLPROC     20.00  X1864.1   DOCLIST   FAYOT     IDS2      IDS2__R5
                 08:02:18AUG 20, 1980     PAGE      20
               * SCHEMA  TG24-SH *                       * PRINT DML REPORT OF RECORD R16 *


           S13
      MEMBER OF SETS :
           S10
                ORDER SORTED BY DEFINED KEYS DUP NOT
                INSERTION AUTOMATIC      RETENTION MANDATORY
           S11
                ORDER SORTED BY DEFINED KEYS DUP NOT
                INSERTION AUTOMATIC      RETENTION MANDATORY

      ******************
      *    LOCATION    *
      ******************

      LOCATION MODE  CALC

      REALMS :
           TG24_A02
           TG24-A05
           TG24-A08
           TG24-A11
           TG24-A14
      AREA-ID DB-PARAMETER : ARID16
```

Figure 3-7. PRINT DML Listing (2/8)

```
          DDLPROC     20.00  X1864.1   DOCLIST   FAYOT     IDS2      IDS2__R5
08:02:18AUG 20, 1980     PAGE      21
               * SCHEMA  TG24-SH *                      * PRINT DML REPORT OF RECORD R16


    ********************************************************************************
    *    DML RETRIEVAL STATEMENTS (FORMAT AND PREREQUISITE ACTION OR CONDITION)    *
    ********************************************************************************


    *********************************
    *    FIND (WITHIN DATA BASE)    *
    *********************************

    FIND |R16|   DB-KEY IS IDENTIFIER   |RETAINING . . .|
       * INITIALIZE IDENTIFIER WITH A DATA-BASE-KEY

    FIND ANY R16 |RETAINING . . . |
       * INITIALIZE AREA-ID DB-PARAMETER ARID16 WITH ANY OF :
              TG24-A02
              TG24-A05
              TG24-A08
              TD24-A11
              TG24-A14
       * INITIALIZE CALC-KEY ITEMS :
              R16-CALC-OCC
              R16-CALC-NUM

    FIND CURRENT |RETAINING . . .|
       * CURRENT-OF-RUN-UNIT MUST REFERENCE A RECORD OF THE SAME TYPE

    ***********************************
    *    FIND (WITHIN RECORD-TYPE)    *
    ***********************************

    FIND CURRENT R16 |RETAINING . . .|
       * CURRENT-OF-RECORD-TYPE MUST REFERENCE A RECORD

    *****************************************************
    *    FIND (WITHIN REALM WHERE RECORD IS LOCATED)    *
    *****************************************************

    FIND FIRST      |R16| WITHIN REALM-NAME |RETAINING . . .|
    FIND LAST       |R16| WITHIN REALM-NAME |RETAINING . . .|
    FIND INTEGER    |R16| WITHIN REALM-NAME |RETAINING . . .|
    FIND IDENTIFIER |R16| WITHIN REALM-NAME |RETAINING . . .|
         WHERE REALM-NAME IS ANY OF :
              TG24-A02
              TG24-A05
              TG24-A08
              TG24-A11
              TG24-A14
       * INITIALIZE IDENTIFIER IF FORMAT 4 IS USED

    FIND NEXT |R16| WITHIN REALM-NANE |RETAINING . . .|
    FIND PRIOR |R16| WITHIN REALM-NAME |RETAINING . . .|
         WHERE REALM-NAME IS ANY OF :
              TG24-A02
              TG24-A05
              TG24-A08
              TG24-A11
```

Figure 3-7. PRINT DML Listing (3/8)

```
       DDLPROC     20.00  X1864.1   DOCLIST   FAYOT     IDS2      IDS2__R5
      08:02:18AUG 20, 1980     PAGE      22
          * SCHEMA  TG24-SH *                       * PRINT DML REPORT OF RECORD R16 *


          TG24-A14
   * CURRENT-OF-REALM MUST REFERENCE A RECORD OR BE VIRTUAL

FIND CURRENT  |R16| WITHIN REALM-NAME |RETAINING . . . |
     WHERE REALM-NAME IS ANY OF :
          TG24-A02
          TG24-A05
          TG24-A08
          TD24-A11
          TG24-A14
   * CURRENT-OF-REALM MUST REFERENCE A RECORD OF THE SAME TYPE

****************************************************
*    FIND (WITHIN SET OF WHICH RECORD IS OWNER)    *
****************************************************

FIND OWNER WITHIN S13 |RETAINING . . .|
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD (OWNER OR MEMBER) OR BE VIRTUAL

FIND CURRENT |R16| WITHIN S13 |RETAINING . . .|
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD(OWNER)

*****************************************************
*    FIND WITHIN SET OF WHICH RECORD IS MEMBER)     *
*****************************************************

FIND FIRST      |R16| WITHIN SET-NAME |RETAINING . . .|
FIND LAST       |R16| WITHIN SET-NAME |RETAINING . . .|
FIND INTEGER    |R16| WITHIN SET-NAME |RETAINING . . .|
FIND IDENTIFIER |R16| WITHIN SET-NAME |RETAINING . . .|
FIND NEXT       |R16| WITHIN SET-NAME |RETAINING . . .|
FIND PRIOR      |R16| WITHIN SET-NAME |RETAINING . . .|
     WHERE SET-NAME IS ANY OF :
          S10
          S11
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD (OWNER OR MEMBER) OR BE VIRTUAL
   * INITIALIZE IDENTIFIER IF FORMAT 4 IS USED

FIND CURRENT |R16| WITHIN SET-NAME |RETAINING . . .|
     WHERE SET-NAME IS ANY OF :
          S10
          S11
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD (MEMBER) OF THE SAME TYPE

FIND R16 WITHIN SET-NAME CURRENT  |USING FIELD-LIST|  |RETAINING . . . |
     WHERE SET-NAME IS ANY OF :
          S10
          S11
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD (OWNER OR MEMBER) OR BE VIRTUAL
   * INITIALIZE ELEMENTARY FIELDS OF THE LIST , IF USING IS SPECIFIED

FIND R16 WITHIN S 10 |USING FIELD-LIST|  |RETAINING . . . |
   * APPLY SET SELECTION FOR S10
     - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
```

Figure 3-7. PRINT DML Listing (4/8)

```
      DDLPROC     30.00  X1159.2   DOCLIST   FAYOT     IDS2      IDS2__R5      08:06:49
                AUG 22, 1980     PAGE      21
         * SCHEMA  TG24-SH *                        * PRINT DML REPORT OF RECORD R16 *


                                          OF SET : S1*
      INITIALIZE AREA-ID DB-PARAMETER  ARID15
      WITH ANY OF :
           TG24-A01
           TG24-A04
           TG24-A07
           TG24-A10
           TG24-A13
      INITIALIZE CALC-KEY ITEMS :
           R15-CALC-OCC
           R15-CALC-NUM
   * INITIALIZE ELEMENTARY FIELDS OF THE LIST IF "USING" IS SPECIFIED

 FIND R16 WITHIN S11  |USING FIELD-LIST | |RETAINING . . . |
   * APPLY SET SELECTION FOR S11
      - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
                                         OF SET : S11
      INITIALIZE AREA-ID DB-PARAMETER  ARID15
      WITH ANY OF :
           TG24-A01
           TG24-A04
           TG24-A07
           TG24-A10
           TG24-A13
      INITIALIZE CALC-KEY ITEMS :
           R15-CALC-OCC
           R15-CALC-NUM
   * INITIALIZE ELEMENTARY FIELDS OF THE LIST IF "USING" IS SPECIFIED

 FIND DUPLICATE WITHIN SET-NAME USING FIELD-LIST |RETAINING . . . |
      WHERE SET-NAME IS ANY OF :
           S10
           S11
   * CURRENT-OF-SET-TYPE MUST REFERENCE A RECORD (MEMBER) OF THE SAME TYPE
   * INITIALIZE ELEMENTARY FIELDS OF THE LIST

 *************
 *    GET    *
 *************

 GET |R16|
 GET FIELD-LIST
   * ESTABLISH OBJECT RECORD AS  :CURRENT-OF-RUN-UNIT
```

Figure 3-7. PRINT DML Listing (5/8)

```
       DDLPROC     20.00  X1864.1   DOCLIST   FAYOT    IDS2      IDS2__R5    08:02:18
                 AUG 20, 1980     PAGE      22
          * SCHEMA  TG24-SH *                        * PRINT DML REPORT OF RECORD R16


 *****************************************************************************
 *    DML UPDATE STATEMENTS (FORMAT AND PREREQUISITE ACTION OR CONDITION)    *
 *****************************************************************************


 *****************
 *     STORE     *
 *****************

 STORE R16 |RETAINING . . .|
    * INITIALIZE ALL FIELDS OF RECORD
    * INITIALIZE AREA-ID DB-PARAMETER ARID16
      WITH ANY OF :
           TG24-A02
           TG24-A05
           TG24-A03
           TG24-A11
           TG24-A14
    * APPLY SET SELECTION FOR S10
      - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
                                     OF SET : S10
      INITIALIZE AREA-ID DB-PARAMETER ARID15
      WITH ANY OF :
           TG24-A01
           TG24-A04
           TG24-A07
           TG24-A10
           TG24-A13
      INITIALIZE CALC-KEY ITEMS :
           R15-CALC-OCC
           R15-CALC-NUM
    * APPLY SET SELECTION FOR S11
      - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
                                     OF SET : S11
      INITIALIZE AREA-ID DB-PARAMETER ARID15
      WITH ANY OF :
           TG24-A01
           TG24-A04
           TG24-A07
           TG24-A10
           TG24-A13
      INITIALIZE CALC-KEY ITEMS :
           R15-CALC-OCC
           R15-CALC-NUM


 ***************************
 *    MODIFY (CONTENTS)    *
 ***************************

 MODIFY |R16|  |RETAINING . . .|
    * ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT (A GET IS NOT REQUIRED)
    * INITIALIZE ALL FIELDS OF THE RECORD WITH THEIR NEW VALUES

 MODIFY FIELD-LIST  |RETAINING . . .|
    * ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT (A GET IS NOT REQUIRED)
    * INITIALIZE ELEMENTARY FIELDS OF THE LIST WITH THEIR NEW VALUES
```

Figure 3-7. PRINT DML Listing (6/8)

```
       DDLPROC     20.00  X1864.1   DOCLIST   FAYOT     IDS2    IDS2__R5    08:02:18
      AUG 20, 1980     PAGE      25
          * SCHEMA  TG24-SH *                    * PRINT DML REPORT OF RECORD R16 *


*****************************
*    MODIFY (MEMBERSHIP)    *
*****************************

MODIFY |R16| ONLY ALL MEMBERSHIP |RETAINING . . . . |
     WHERE "ALL" REPRESENTS THE FOLLOWING SETS :
          S10
          S11
   * APPLY SET SELECTION FOR S10
     - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
                                      OF SET : S10
     INITIALIZE AREA-ID DB-PARAMETER ARID15
     WITH ANY OF :
          TG24-A01
          TG24-A04
          TG24-A07
          TG24-A10
          TG24-A13
     INITIALIZE CALC-KEY ITEMS :
          R15-CALC-OCC
          R15-CALC-NUM
   * APPLY SET SELECTION FOR S11
     - LEVEL 1   -  IN ORDER TO SELECT OWNER : R15
                                      OF SET : S11
     INITIALIZE AREA-ID DB-PARAMETER ARID15
     WITH ANY OF :
          TG24-A01
          TG24-A04
          TG24-A07
          TG24-A10
          TG24-A13
     INITIALIZE CALC-KEY ITEMS :
          R15-CALC-OCC
          R15-CALC-NUM
   * ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT

MODIFY |R16| ONLY SET-NAME-LIST MEMBERSHIP |RETAINING . . .|
     WHERE SET-NAME-LIST IS A SUBSET OF THE LIST OF THE PRECEDING FORMAT
   * APPLY SET SELECTION FOR EACH SET OF SET-NAME-LIST (SEE PRECEDING FORMAT)
   * ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT

******************************************
*    MODIFY (CONTENTS AND MEMBERSHIP)    *
******************************************

MODIFY |R16| INCLUDING ALL MEMBERSHIP |RETAINING . . .|
MODIFY |R16| INCLUDING SET-NAME LIST MEMBERSHIP |RETAINING . . .|
MODIFY FIELD-LIST INCLUDING ALL MEMBERSHIP |RETAINING . . .|
MODIFY FIELD-LIST INCLUDING SET-NAME-LIST MEMBERSHIP |RETAINING . . |
   * PERFORM THE ACTIONS REQUIRED BY THE MODIFY(CONTENTS) STATEMENT
     AND THE MODIFY (MEMBERSHIP) STATEMENT WHICH ARE COMBINED IN THE
     SELECTED MODIFY (CONTENTS AND MEMBERSHIP) STATEMENT

***************
*    ERASE    *
```

Figure 3-7. PRINT DML Listing (7/8)

```
      DDLPROC     20.00  X1864.1   DOCLIST   FAYOT    IDS2      IDS2__R5
   08:02:18AUG 20, 1980     PAGE      26
         * SCHEMA  TG24-SH *                       * PRINT DML REPORT OF RECORD R16 *


***************


ERASE  |R16|
    *  OBJECT RECORD MUST BE OWNER OF EMPTY SETS
    *  ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT
ERASE  |R16| ALL MEMBERS
    *  ESTABLISH OBJECT RECORD AS CURRENT-OF-RUN-UNIT
       - WARNING -  THIS STATEMENT MAY ERASE THE FOLLOWING RECORDS :
       R16
       R17
```

Figure 3-7. PRINT DML Listing (8/8)

3.5.6 DELETE SCHEMA Command

Function

To delete an object schema.

General Format

```
                             [       {library-efn} ]
DELETE SCHEMA schema-name [ {IN} {DDLIB1      } ] .
                             [ {OF} {DDLIB2      } ]
                             [       {DDLIB3      } ]
```

Rules

1. The library containing the object schema to be deleted is determined by the default search rules for update processing or by the qualification.

2. The object schema may be in the "DDL" or "DDL-DMCL" state.

3. The deletion of an object schema may also be obtained using the DELETE command of LIBMAINT BIN.

3.5.7 MOVE SCHEMA Command

Function

To move an object schema from one library to another library.

General Format

```
                              [        {library-efn}]
MOVE SCHEMA schema-name [ {IN}  {DDLIB1     }]
                              [ {OF}  {DDLIB2     }]
                              [        {DDLIB3     }]

                              [   {library-efn} ]
                              [TO {DDLIB1     } ]  [REPLACE] .
                              [   {DDLIB2     } ]
                              [   {DDLIB3     } ]
```

Rules

1. The library containing the object schema to be moved is determined by the default search rules for retrieval processing or by the qualification.

2. The object schema may be in the "DDL" or "DDL-DMCL" state.

3. The library containing the copy is determined by the default search rules for creation processing or by the qualification.

4. If the copy action is to overwrite an existing member of the same name currently in the output library selected, then the keyword REPLACE must be coded, otherwise an error will result.

5. The move of an object schema may also be obtained using the MOVE command of LIBMAINT BIN.

3.5.8 STATUS Command

Function

To control the flow of DDLPROC commands.

General Format

```
 ___________________
|                   |
| STATUS  RESET  .  |
|___________________|
```

Rules

1. After each command other than STATUS, DDLPROC sets a status indicator to one of two positions.

   The NORMAL position corresponds to the normal completion of the command (no fatal errors).

   The ABNORMAL position corresponds to the abnormal completion of the command (fatal errors).

2. When the status indicator is in the NORMAL position, DDLPROC looks for the next command and executes it.

3. When the status indicator is in the ABNORMAL position, DDLPROC looks for the next command, expecting a STATUS RESET command.

   If the next command is not STATUS RESET, DDLPROC terminates abnormally with a severity code SEV3.

   If the next command is STATUS RESET, DDLPROC resets the status indicator to the NORMAL position and continues its processing as defined in rule 2.

4. A STATUS RESET command is ineffective if the status indicator is already in the NORMAL position.

3.6 DDLPROC EXAMPLES

1. The DDL of schema INVENTORY is entered from an input enclosure (INVENTORY-DDL) and is to be translated and listed but not loaded.
   This is only a first pass to detect DDL syntax errors:

```
$JOB TRADDL,USER=PC, PROJECT=MCE;
DDLPROC COMMAND='
TRANSLATE SCHEMA DDL
SOURCE IS *INVENTORY-DDL.
';
$INPUT INVENTORY-DDL;
SCHEMA NAME IS INVENTORY.
...
END-SCHEMA.
$ENDINPUT;
$ENDJOB;
```

2. The DDL of schema PERSONNEL is entered from a source library MIDS.SLLIB (member PERSONNEL-DDL). It is to be translated and loaded into an object schema library MIDS.BINLIB (member PERSONNEL). The DMCL of schema PERSONNEL is also entered from the source library MIDS.SLLIB (member PERSONNEL-DMCL). It is to be translated and loaded into the partial object schema created by the previous command. Information about the storage characteristics of the data base is also requested.

   Figure 3-8 illustrates the sequence of operations.

```
$JOB TDDLDMCL,USER=PC,PROJECT=MCE;
LIB SL INLIB1=MIDS.SLLIB;
DDLPROC COMFILE=*TRACOM
        DDLIB1=MIDS.BINLIB;
$INPUT TRACOM;
TRANSLATE SCHEMA DDL
   SOURCE IS PERSONNEL-DDL
   OBJECT
   CHECK FOR COBOL.
TRANSLATE DMCL
   SOURCE IS PERSONNEL-DMCL
   OBJECT.
PRINT SCHEMA PERSONNEL
   STORAGE DESCRIPTION.
$ENDINPUT;
$ENDJOB;
```

   For the TRANSLATE SCHEMA DDL command, the source PERSONNEL-DDL is retrieved from MIDS.SLLIB (first and only library in SL search path) and the object schema PERSONNEL, in "DDL" state, is stored in MIDS.BINLIB (first and only library in DD search path).

For the TRANSLATE DMCL command, the source PERSONNEL-DMCL is retrieved from MIDS.SLLIB (first and only library in SL search path). The object schema PERSONNEL is retrieved from MIDS.BINLIB (first and only library in DD search path). It is then updated to contain the DMCL and it becomes an object schema in "DDL-DMCL" state, available for run-time execution.

For the PRINT SCHEMA command, the object schema PERSONNEL, in "DDL-DMCL" state, is retrieved from MIDS.BINLIB (first and only library in DD search path).

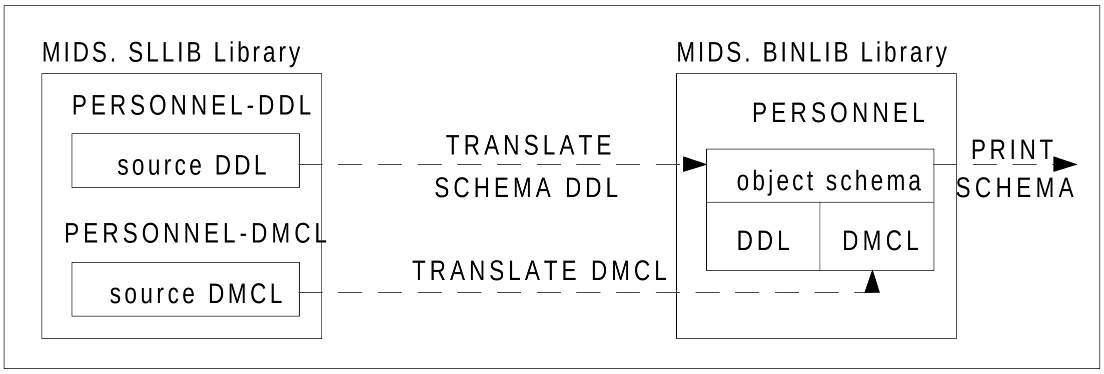


Figure 3-8. Processing Schema PERSONNEL

3. Continuing on from the previous example, the user decides to modify the DMCL parameters, after several runs of the SIMULOAD command of DBUTILITY. He updates the member PERSONNEL-DMCL using LIBMAINT SL then calls DDLPROC to translate the new DMCL and update the object schema.

```
 ________________________________________
|                                        |
|  $JOB TNEWDMCL,USER=PC,PROJECT=MCS;    |
|  LIB SL INLIB1=MIDS.SLLIB;             |
|  DDLPROC COMFILE=*TRACOM               |
|          DDLIB1=MIDS.BINLIB;           |
|  $INPUT TRACOM;                        |
|  TRANSLATE DMCL                        |
|     SOURCE IS PERSONNEL-DMCL           |
|     OBJECT REPLACE.                    |
|  $ENDINPUT;                            |
|  $ENDJOB;                              |
|________________________________________|
```

The keyword REPLACE is necessary to overwrite the old DMCL section.

4. Now the user decides to check out his DML programs against a test data base of a smaller size than that of the production data base.

   The schema DDL is exactly the same for the test and production data bases. The DMCL varies in terms of area sizes.

   The user prepares the source of the test DMCL in the library TEST.MIDS.SLLIB (member PERSONNEL-DMCL). He then moves the production object schema PERSONNEL from the library MIDS.BINLIB to the library TEST.MIDS.BINLIB and translates the test DMCL to replace the production DMCL in the copy, giving the test object schema. Figure 3-9 illustrates the sequence of operations.

```
$JOB TESTDMCL,USER=PC,PROJECT=MCE;
LIB SL INLIB1=MIDS.SLLIB
       INLIB2=TEST.MIDS.SLLIB;
DDLPROC DDLIB1=MIDS.BINLIB
        DDLIB2=TEST.MIDS.BINLIB
        COMMAND='
MOVE SCHEMA PERSONNEL TO TEST.MIDS.BINLIB.
TRANSLATE DMCL
   SOURCE IS PERSONNEL-DMCL OF SLLIB2
   SCHEMA OF DDLIB2
   OBJECT REPLACE.
';
$ENDJOB;
```

In the MOVE command, the object schema PERSONNEL is retrieved from MIDS.BINLIB (first library in DD search path) and moved to TEST.MIDS.BINLIB as specified by the efn of the TO phrase. In the TRANSLATE command, the source PERSONNEL-DMCL is retrieved from TEST.MIDS.SLLIB as specified by the qualifier SLLIB2. The object schema is retrieved from TEST.MIDS.BINLIB as specified by the qualifier DDLIB2.

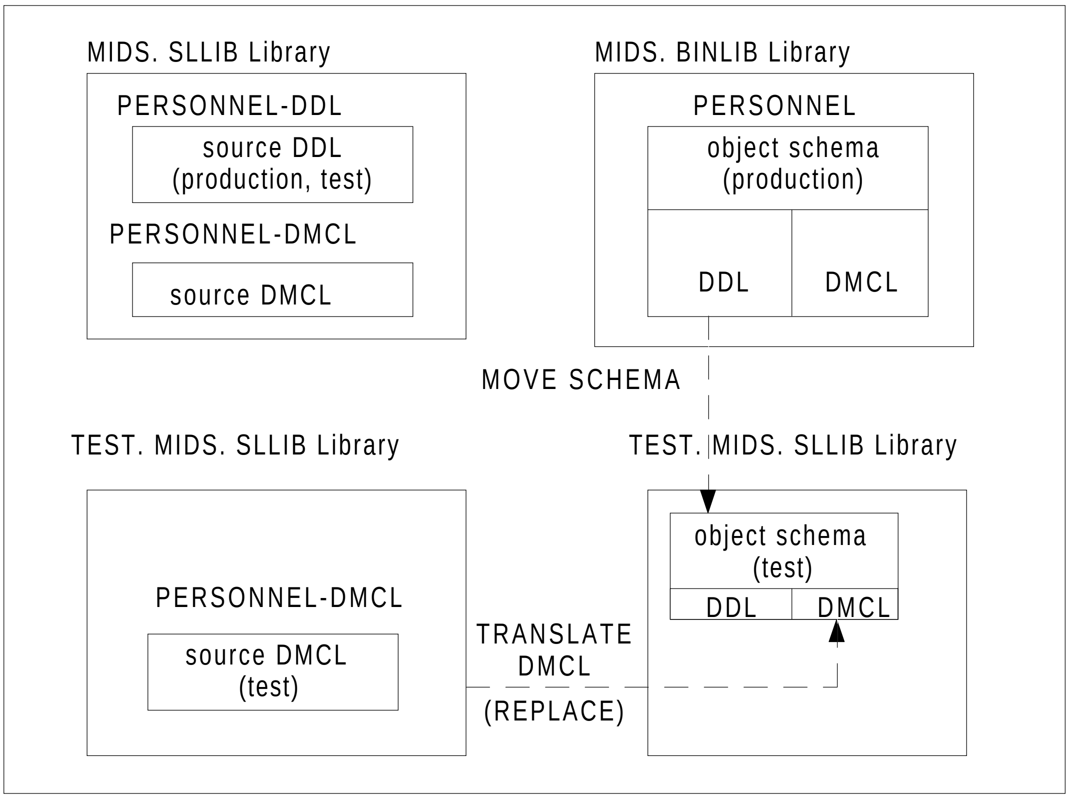


Figure 3-9. Processing Test Schema PERSONNEL

DDLPROC DEBUGGING

If, during the processing of a command, DDLPROC aborts because of an exception or issues an "INTERNAL BUG" message, the field engineer (or the user) is advised to:

1. Rerun DDLPROC with the DUMP=DATA parameter in the extended JCL statement and a command stream composed of the keyword DEBUG, followed by the command that caused the error, for example:

   ```
   $INPUT DDLCOM;
    DEBUG.
    TRANSLATE SCHEMA DDL ...
   $ENDINPUT;
   ```

2. Send the resulting trace output listing, together with the DDL and DMCL source listings, through the STAR system.

4. Storage Area Preallocation

4.1 PREALLOC UTILITY

The purpose of PREALLOC is to map a storage area, equivalent to a UFAS file, onto one or more extents of one or more disk volumes. (See Figure 4-1)

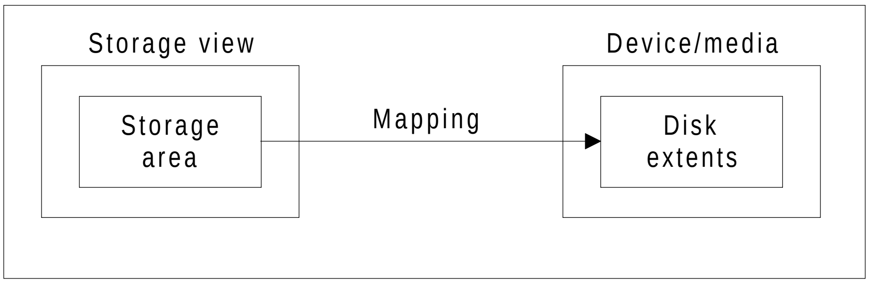


Figure 4-1. PREALOC Function

PREALLOC must be invoked for each storage area of the data base.

The characteristics of the storage area such as the number of pages, the number of lines per page, the page size and the CALC interval are retrieved by PREALLOC from the object schema, which must be in the "DDL-DMCL" state.

The main parameters to be supplied directly to PREALLOC are:

- The external-file-name, which identifies the occurrence of the storage area.
- The disk device type.
- The names of the volumes on which space is to be reserved.
- Optionally, the amount of space to be taken on each volume and the position at which the allocation is to take place.

4.2 SPACE ALLOCATION

1. From the DMCL information (number of pages NUMP, page size) PREALLOC calculates the total size of the file in terms of tracks for the specified device class.

   This calculation is performed automatically. However it is of administrative interest for the user to know how many tracks will be required.

   This calculation is as follows:

   - Find the number of pages per track (PPT) using Table 2-4 in Section II.

   - Calculate the total number of tracks (NUMT) by:

     ```
     NUMT = (NUMP - 1)/PPT +2
     ```

2. Space reservation is done with a disk track as unit. Space is allocated as a series of one or more extents. An extent is a group of one or more contiguous tracks. On any one volume a file may have up to 16 extents. However, for efficiency purposes, the user can restrict the number of extents per volume to less than the maximum (16).

3. In the automatic mode of allocation, the user specifies only the list of volumes on which the area resides.

   PREALLOC allocates on the first volume of the list as many extents as required to accommodate NUMT tracks. If the file fits on this volume, the processing stops and the other volumes are ignored. If the space available on this volume is less than NUMT or if the maximum number of extents allowed is reached too soon, PREALLOC attempts to place the remainder of the file on the next volume of the list, using the same type of allocation, and so on until the total file size is allocated

   For each volume the allocation of extents is done as follows. A list of all the free space extents available is inspected. The smallest extent of all those which are greater or equal to the space required is chosen. Thus if the free extents were 20, 23, 25, 60 cylinders and request was for 24 cylinders then the allocation would be made on the 25 cylinder extent leaving free space extents of 1, 20, 23, 60 cylinders. If the space requested is larger than the largest free space extent, this extent is allocated and the remaining space still required is chosen first by searching for the smallest of the larger free extents or choosing the largest and then searching for space for the remainder. Thus if a request for 86 cylinders was made with the above space list then 60 and 25 extents would be allocated and one cylinder would be used from the 20 cylinders.

4. There is a less automatic mode of allocation, in which the user specifies for each volume the amount of space to be allocated. The allocation fails if this amount of space is not available on the volume or if the number of extents required is larger than permitted.

   The user may also specify the cylinder address at which allocation is to occur. In this case there must be a sufficiently large contiguous free space at the supplied address to allocate in one extent the amount of space specified.

5. The space allocated is formatted by PREALLOC. For each page, the page header and a CALC chain OWL record are written. The rest of the page is filled with binary zeroes.

6. PREALLOC also creates the standard UFAS labels and specific IDS labels. The latter contain, the name and DMCL reference date of the schema used for the preallocation, the storage area name and its DMCL characteristics. This information will have to match at run-time the corresponding information in the schema loaded for execution.

4.3 PREALLOC EXTENDED JCL STATEMENT

PREALLOC can be invoked using the following extended JCL statement:

```
PREALLOC external - file - name

             { CAT    } }
{FILESTAT =  { UNCAT  } }
{            { TEMPRY } }
{TEMPRY

 [CATALOG = integer-1 ] [CATNOW]
            { ddd      } ]
[ EXPDATE = { yy/ddd   } ]
            { yy/mm/dd } ]

[ UNIT = {CYL   } ]
         {TRACK } ]

[ MAXEXT = { 5         } ]
[          { integer-2 } ]

{ RESIDENT                                                                 }
{ DEVCLASS = device-class                                                  }
{    { GLOBAL = MEDIA = ( volume-name [ volume-name ]...))            }    }
{    { SPLIT = ((volume-name SIZE = integer-3                         }    }
{    {                              [ CYL = integer-4 [ TRACK = integer-5 ] ])   }    }
{    {                                                                }    }
{    { [(volume-name SIZE = integer-6                                 }    }
{    {                       [ CYL = integer-7 [ TRACK = integer-8 ] ]) ]...)    }    }

  UFAS = (IDS = (AREA = area-name
         SCHEMA = schema-name
         DDLIB1 = (library-description)))   ;
```

4.3.1 PREALLOC Parameter Description

external-file-name	The name chosen must be unique within the set of file names already present on the volume(s) to be used.
FILESTAT	This keyword indicates whether the area is permanent cataloged(CAT), permanent uncataloged (UNCAT) or temporary (TEMPRY)
TEMPRY	This self-identifying keyword is equivalent to FILESTAT=TEMPRY
CATALOG	integer-1 is a one digit integer which must satisfy the condition: `1 <= integer-1 <= 5` It indicates the rank in the ATTACH statement of the catalog where the file is or is to be catalogued. If CATALOG is not specified when FILESTAT=CAT, the default search rules for catalogs are used.
CATNOW	This keyword indicates that a file description is to be entered in the catalog specified. If a description already exists, CATNOW is ignored.

EXPDATE	This keyword gives the expiry date for the file. If no expiry date is given the default is the current date.
UNIT	This keyword is used in conjunction with the SPLIT parameter. It indicates the unit (cylinder or track) in which SIZE is expressed.
MAXEXT	Integer-2 is a one or two-digit decimal integer which must satisfy the condition: `1 <= integer-2 <= 16` It specifies the maximum number of extents to be selected per volume.
RESIDENT	If this keyword is specified then the allocation will take place on the resident volumes attached to the system.
DEVCLASS	If this keyword is specified then non-resident disk space is allocated. It is followed by a standard disk device class identification.
GLOBAL	This keyword specifies that allocation of extents is to be performed automatically.
MEDIA	This parameter is followed by a list of one or more volume names to be used in the acquisition of space. The order of the list is important. The utility will allocate the maximum number of extents (see MAXEXT above) on the first volume and proceed to the second an so on. In the case of a multi-volume allocation it is possible that the space requirement for the file will be satisfied without all the volumes being needed. If this occurs then the utility will not use the last volumes in any way, and they need not be mentioned in future references to the file. The volume names given must be in the desired order of usage. The pages with the lowest page numbers will be recorded on the first volume and the pages with the highest page numbers on the last.
SPLIT	This keyword specifies that the allocation is to be performed on a volume by volume basis. It is followed by a parameter group which lists each volume and the amount of space required on each. Optionally the user may specify for each volume the disk address at which allocation is to occur. It is possible to give addresses for some of the volumes and not for others. The volume names given must be in the desired order of usage. The pages with the lowest page numbers will be recorded on the first volume and the pages with the highest page numbers on the last.

SIZE	This keyword, supplied with each volume name, gives the amount of space required on the volume. The numeric value supplied, up to five digits long, is expressed in the unit specified by UNIT.
CYL/TRACK	These optional keywords allow the user to specify the disk address within volume, at which allocation is to occur. If this is specified then there is no automatic extent selection. Instead all the amount of SIZE for the volume is allocated starting at the supplied address. The user must ensure that there is a sufficiently large contigous free space at the supplied address; otherwise the allocation will fail and terminate abnormally. The numeric value giving the cylinder address must not be more than three digits long. A partially used cylinder can be given as the starting point for allocation. The numeric value giving the track address must not be more than two digits long. If CYL is not specified for a volume then space is reserved by automatic extent selection.
UFAS	This keyword is followed by a parameter group which defines the file characteristics.
IDS	This keyword signifies that the allocation is for an IDS area.
AREA	This keyword specifies the DMCL name of the storage area (up to 30 characters in length).
SCHEMA	This keyword specifies the name of the schema where the storage area is defined.
DDLIB1	This keyword is followed by a parameter group which identifies the library containing the object schema.

4.4 EXAMPLES

1. Two storage areas are to be allocated. The related DMCL entries are:

```
AREA NAME IS REVISET
    NUMBER-OF-PAGES IS 10000
    PAGE-SIZE IS 1024 BYTES
    LINES-PER-PAGE IS 20.

AREA NAME IS ADDSET
    NUMBER-OF-PAGES IS 50000
    PAGE-SIZE IS 2048 BYTES
    LINES-PER-PAGE IS 28.
```

The first area is to be given the external-file-name MIDS.RE and allocated on the MSU 0400 volume BD1023.

The second area is to be given the external-file-name MIDS.LB and allocated on the MSU 0400 volumes BD1024, BD1025.

Both files are protected by a retention period of 300 days.

The object schema is member TPL in library MIDS. BINLIB which is RESIDENT.

The job description to perform allocation is:

```
 _____________________________________________________
|                                                     |
| $JOB DBPREAL USER=MOAN PROJECT=CMSMCETP;            |
| PREALLOC MIDS.RE                                    |
|     FILESTAT=UNCAT                                  |
|     EXPDATE=300                                     |
|     DEVCLASS=MS/M400 GLOBAL=(MEDIA=BD1023)          |
|     UFAS=(IDS=(AREA=REVISET                         |
|                SCHEMA=TPL                           |
|                DDLIB1=MIDS.BINLIB));                |
| PREALLOC MIDS.LB                                    |
|     FILESTAT=UNCAT                                  |
|     EXPDATE=300                                     |
|     DEVCLASS=MS/M400 GLOBAL=(MEDIA=(BD1024 BD1025)) |
|     UFAS=(IDS=(AREA=ADDSET                          |
|                SCHEMA=TPL                           |
|                DDLIB1=MIDS.BINLIB));                |
| $ENDJOB;                                            |
|_____________________________________________________|
```

The space requirements may be calculated using Table 2-4.

```
For file MIDS.RE     NUMP = 10 000    PPT = 11
   Number of tracks = (10 000 - 1)/11 + 2 = 911 (48 cylinders)

For file MIDS.LB     NUMP = 50 000    PPT = 6
   Number of tracks = (50 000 - 1)/6 + 2 = 8335 (439 cylinders)
```

2. A storage area is to be allocated on 3 MSU 0400 volumes VL0015, VL0016, VL0017. The related DMCL entry is:

```
AREA NAME IS PRODUCTS
   NUMBER-OF-PAGES IS 100000
   PAGE-SIZE IS 2048 BYTES
   LINES-PER-PAGE IS 45.
```

As NUMP = 100 000 and PPT=6, the total number of tracks is (100 000 - 1)/6 + 2 = 16 668 which amounts to 878 cylinders.

Assume that:

- 300 cylinders are allocated on VL0015 starting from cylinder 65.
- 300 cylinders are allocated anywhere on VL0016
- the remainder is allocated anywhere on VL0017.
- the file is already cataloged in the site catalog with the external-file-name DEPT1.PRODUCTS
- the schema is member INVENTORY of the catalogued library DEPT1.CONTROL.BINLIB.

The job description to perform allocation is:

```
 ______________________________________________
|                                              |
|  $JOB DBPREAL    USER=HOAN PROJECT=CMS;      |
|  PREALLOC    DEPT1.PRODUCTS                  |
|      FILESTAT=CAT                            |
|      DEVCLASS=MS/M400                        |
|      SPLIT=((VL0015    SIZE=300  CYL=65)     |
|               (VL0016    SIZE=300)           |
|               (VL0017    SIZE=300))          |
|      UFAS=(IDS=(AREA=PRODUCTS                |
|            SCHEMA=INVENTORY                  |
|            DDLIB1=DEPT1.CONTROL.BINLIB));    |
|  $ENDJOB;                                    |
|______________________________________________|
```

The default values for UNIT and MAXEXT are respectively CYL and 5.

Cylinders 65 through 364 must be available on volume VL0015 before the preallocation.

At least 300 cylinders must be available on volume VL0016, in fewer than 6 extents.

The SIZE specified for volume VL0017 is larger than required PREALLOCwill allocate only the space necessary (277 cylinders).

5. COBOL Data Manipulation Language

IDS/II offers a data manipulation facility for accessing a data base described by a schema, composed of:

1. Communication zones between the user and the DBCS. These include:

   - User Working Area (UWA) records, where record occurrences are prepared before storing or placed after retrieval. The format of UWA records corresponds to that defined in the schema.

   - Special registers through which the DBCS informs the user of normal or abnormal termination of a function.

   - Currency indicators where the DBCS maintains pointers to different records in the data base. These indicators are implicit input and output parameters of most manipulation functions.

2. Manipulation functions. These allow the user to store, retrieve and erase records, to modify the contents and membership of records, and to test various conditions in the data base.

5.1 HOST LANGUAGES

The data base facility is not a stand-alone package. It is considered as a service available to a program written in a host language.

The program uses the constructs of this host-language to express the sequence of manipulation functions to be executed, to prepare the contents of UWA records and to interpret the results of the manipulation functions.

The way in which the data base facility is handled in the host-language itself depends on the type of language.

5.1.1 Implementation Languages

In this case the data base facility does not add new constructs to the host-language. The data manipulation functions are invoked by standard procedure calls. The communication zones and function parameters must be defined by the programmer, using the declaratives statements of the host-language.

The link with the schema is not under the compiler's responsibility.
Therefore such languages are particularly adapted to interpretative system processors (QUERY or data base utilities) which must adapt themselves at run-time to any data base schema.

5.1.2 High Level Languages (COBOL)

In this case the data base facility is assimilated by the host language itself. It introduces new constructs recognized by the compiler. The data manipulation functions take the form of new procedural statements in the syntax conventions of the host language. The communication zones are not defined by the programmer; they are derived automatically from the schema by the compiler and made available to the program in the form of declarative statements specific to the host language.

Unlike implementation languages, the program is dedicated to a given schema.

5.2 COBOL DATA MANIPULATION LANGUAGE (DML)

The new constructs introduced by the data base facility in the COBOL language affect the Data Division and the Procedure Division.

1. In the Data Division, a new section, the Sub-Schema Section, specifies the schema which is referenced by the program. Communication zones are automatically derived from this schema.

   Only one schema can be referenced from a single compiled program. If a run-unit is to use more than one schema, then separate compilations are permissible.

2. In the Working-Storage Section and Linkage Section, data items may be defined with a new USAGE: DB-KEY. These data items will contain data-base-key.

3. In the Procedure Division, new statements are introduced:

   - the ACCEPT, CONNECT, DISCONNECT, ERASE, FIND, FINISH, GET, MODIFY, READY, STORE verbs are imperative statements.

   - the IF statement tests data base conditions.

   - The USE statement of the DECLARATIVES part allows a centralized data-base-exception handling.

4. The DDLIST parameter of the COBOL JCL statement provides the programmer with a DATA BASE RECORDS listing containing all selected records within the schema with their corresponding items.

5. The PRINT DML command of DDLPROC provides the programmer with the information required for accessing the data base. This information, given on a record-type basis, includes:

   - the description of the UWA record, in COBOL terms

   - the control-fields of the record (CALC-key, sort-keys, no-duplicate-control-fields)

   - the sets in which the record participates

   - the location mode of the record and its containing areas. Note that the DDL term "area" corresponds to the COBOL term "realm"

   - the list of manipulation verbs authorized for processing the record, together with the actions to be taken prior to their invocation.
     ACCEPT, FINISH, READY, IF, and USE are not mentioned in this list.

5.3 NOTATION

The notation used in all formats and the rules that apply to all formats are as follows:

- The elements that make up a clause consist of upper case words, lower case words, special symbols, and special characters.
- All underlined upper case words are required when the format is used (keywords).
- Upper case words not underlined are optional words and need not be used.
- Lower case words are generic terms that must be replaced by appropriate names or values.
- When a portion of a general format is enclosed in square brackets, [], that portion may be included or omitted at the user's choice. Braces, é è, enclosing a portion of a general format means selection of one of the options contained within the braces must be made. A choice indicator, II and II, enclosing a portion of a general format means that a selection of one or more of the options contained within the choice indicators must not be chosen more than once in that entry or statement. In all cases, a choice is indicated by vertically stacking the possibilities.
  That portion of a general format contained within brackets or braces may be followed by an ellipsis (...) to indicate repetition.

Examples:

1.
```
FIND [record-name] DB-KEY IS identifier-1.
```

   The item record-name may be included or omitted as needed in the program. If included, this indicates a test is to be performed to ensure that the record found is of the named record-type.

2.
```
FIND {ANY      } record-name
     {DUPLICATE}
```

   A choice must be made between ANY and DUPLICATE.

3. FIND record-selection-expression

```
[                          {  MULTIPLE            }]
[                          {  REALM               }]
[ RETAINING CURRENCY FOR   { {SETS          }     }]
[                          { {  {set-name}  } ... }]
[                          {  RECORD              }]
```

This example uses the choice indicators II II to allow at least one or many of the listed forms to be used. The phrase could be:

```
RETAINING REALM RECORD
RETAINING RECORD
RETAINING REALM set-name-1 set-name-2 set-name-3
RETAINING REALM SETS
```

5.4 SYNTAX RULES

5.4.1 COBOL Syntax Conventions

The COBOL-74 rules governing syntax, punctuation, and column alignment apply to all DML statements.

5.4.2 Reserved Words

The list of COBOL/DML reserved words is to be found in the COBOL Reference Manual.

5.4.3 Record Field Qualification

In the schema DDL

- field-names are unique within a record-type
- different record-types may contain fields with identical names.

When duplicate field-names are used in COBOL-74 verbs, they must be qualified by the record-name.

Take as an example the following DDL:

```
RECORD CUSTOMER-1A ...
   02  CUSTOMER-NUMBER  TYPE UNPACKED DEC 6.
   02  CUSTOMER-NAME    TYPE  CHAR 30.
   02  CUSTOMER-ADDRESS TYPE  CHAR 40.
RECORD CUSTOMER-2A...
   02  CUSTOMER-NUMBER  TYPE  UNPACKED DEC 6.
   02  CUSTOMER-NAME    TYPE  CHAR 30.
   02  CUSTOMER-ADDRESS TYPE  CHAR 40 OCCURS 2.
SET CUST-OF-DISTRICT ...
  MEMBER  CUSTOMER-1A ...
  MEMBER  CUSTOMER-2A ...
```

In the DML statement:
FIND CUSTOMER-1A WITHIN CUST-OF-DISTRICT USING CUSTOMER-NUMBER OF CUSTOMER-1A.
CUSTOMER-NUMBER must be qualified even though CUSTOMER-1A is already specified.

In the DML statement:
GET CUSTOMER-ADDRESS OF CUSTOMER-1A CUSTOMER-NAME OF CUSTOMER-1A.
All fields are qualified in order to identify CUSTOMER-1A.

Qualification precedes the subscripts in COBOL while the subscripts precede the qualification in DDL:

```
COBOL: CUSTOMER-ADDRESS OF CUSTOMER-2A (2)
DDL:   CUSTOMER-ADDRESS (2) OF CUSTOMER-2A
```

5.5 SUB-SCHEMA SECTION IN DATA DIVISION

The Sub-Schema Section, which must be the first section of the Data Division, contains only one entry, the sub schema (DB) entry. It specifies the data base schema description to be accessed by the program.
Only one DB entry can appear in a given program.

General Format

```
DATA DIVISION .

SUB - SCHEMA SECTION .

DB schema - name .

[                        {WORKING - STORAGE  }
[ DB - DESCRIPTIONS IN  {                   }
[                        {   LINKAGE         }      SECTION . ]

[ {                   {ALL                      } }    ]
[ {RECORDS ARE        {                         } }    ]
[ {                   {[NOT]  {record-name}... } }    ]
[ {                                                }    ]
[ {                                                }    ]
[ {                   {ALL                      } }  . ]
[ {REALMS ARE         {                         } }    ]
[ {                   {[NOT] {realm-name}...    } }    ]
```

Rules

1. The schema referenced by schema-name must be made available to the program during execution (see Section VII).

2. The DB-DESCRIPTIONS clause specifies where the data base record-descriptions and DBCS control structures are to be placed when the program is compiled.

3. If the program is a main (not called by another) program of a batch step or of a TDS transaction processing routine, the DB-DESCRIPTIONS clause must specify WORKING-STORAGE. If the entire clause is omitted, WORKING-STORAGE is the default value.

4. If the program is a secondary (called by another) program, the DB-DESCRIPTIONS clause must specify LINKAGE SECTION. A secondary program may call another secondary program.

5. In the programs that make up a single batch step or a single TDS TPR, there must be, for a given schema, only one program with DB-DESCRIPTIONS IN WORKING-STORAGE.

6. Any batch step program (main or secondary) may contain READY and FINISH statements. A TPR program does not usually contain READY or FINISH statements; if it does, they are ignored at execution time.

7. Each call to a secondary program must contain, in the USING phrase of the CALL statement, arguments concerning the data-base identifiers that will be used in the execution of the secondary program.

   In the secondary program the USING phrase of the PROCEDURE statement must contain similar arguments in the same order.

   It is the user's responsibility to provide the argument list.
   The argument list must be defined as follows:

   ```
   USING{ua} ... DB-REGISTERS DB-CXT [DB-PARAMETERS] {rn} ...
   ```

   where:

   - ua ... are the user's data arguments that are being passed between programs.

   - DB-REGISTERS, DB-CXT and DB-PARAMETERS pass the necessary control structures to permit the execution of all DML functions.

     DB-PARAMETERS must not be specified if the schema does not contain any data-base-parameters.

   - rn ... is a list of the record-names referenced in the called program. The record-names must be the same as those recorded in the schema. If a record-type has no fields, it must not be specified in the list.

8. The RECORDS clause allows the user to declare that the program references only records of specified types. Only information concerning these record-types is kept in the program.

9. If the word NOT prefixes the list of record-names, all the records, except those declared, are available.

10. If RECORDS ARE ALL is coded, the entire data base defined by the schema may be referenced.

11. As an alternative to RECORDS ARE the user may code REALMS ARE which limits the scope of program reference to the record types defined in the selected areas. This clause is a more convenient way of writing a lengthy list of records.

12. If the word NOT prefixes the list of realm-names, the record-types defined in all areas, except those declared, are available.

13. The effect of REALMS ARE ALL is the same as that of RECORDS ARE ALL.

14. If neither RECORDS nor REALMS is specified then all the record-types of the data base are made available.

15. The use of the RECORDS/REALMS clause reduces the size of the UWA, or LINKAGE area description, that the program will require.

5.5.1 Inter-Program Linkage Example

A master program, PM, makes ready a single area, AREA-X, of a schema SDB-L and calls two programs, P1 and P2, which reference the same schema as PM.

The program P1 references two record-types, RT-A and RT-B, while the program P2 references three record-types, RT-A and RT-D and RT-G.

The structure of the programs is shown on next page.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PM.
 ...
 DATA DIVISION.
 SUB-SCHEMA SECTION.
 DB SDB-L.
     RECORDS ARE RT-A RT-B RT-D RT-G.
 ...
     READY   AREA-X ...
     ...
     CALL P1 USING MYARG DB-REGISTERS DB-CXT DB-PARAMETERS
                   RT-A RT-B.
     ...
     CALL P2 USING MAX MAY DB-REGISTERS DB-CXT DB-PARAMETERS
                   RT-A RT-D RT-G.
 ...
     FINISH AREA-X.
 ...
     STOP RUN.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. P1.
 ...
 DATA DIVISION.
 SUB-SCHEMA SECTION.
 DB SDB-L.
     DB-DESCRIPTIONS IN LINKAGE SECTION.
     RECORDS ARE RT-A RT-B.
 ...
 PROCEDURE DIVISION USING FUNCX DB-REGISTERS DB-CXT
    DB-PARAMETERS RT-A RT-B.
    ...
    FIND RT-A ...
    ...
    EXIT PROGRAM.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. P2.
...
DATA DIVISION.
SUB-SCHEMA SECTION.
DB SDB-L.
    DB-DESCRIPTIONS IN LINKAGE SECTION.
    RECORDS ARE RT-A RT-D RT-G.
...
PROCEDURE DIVISION USING ARA ARB DB-REGISTERS DB-CXT
    DB-PARAMETERS RT-A RT-D RT-G.
    ...
    FIND RT-D ...
    ...
    ACCEPT MID FROM RT-A CURRENCY.
    ...
    ERASE RT-G.
    ...
    EXIT PROGRAM.
```

5.6 DB-KEY DATA ITEMS

In the WORKING-STORAGE or LINKAGE SECTION, data items containing data-base-keys are defined with the type USAGE IS DB-KEY:

```
level-number identifier USAGE IS DB-KEY.
```

This type of data description is required for the "identifier" of the following DML statements:

```
ACCEPT identifier FROM ...CURRENCY
ACCEPT...FROM identifier REALM-NAME

                         {NEXT }
ACCEPT identifier FROM...{PRIOR}
                         {OWNER}

ACCEPT identifier FROM...MINIMUM-DB-KEY
FIND...DB-KEY IS identifier .
```

USAGE DB-KEY is equivalent to USAGE COMP-2 when these data items are used in COBOL-74 statements.

5.7 DATA BASE SPECIAL REGISTERS

Data base special registers are used to communicate data base information between the program and the Data Base Control System (DBCS). The program can read the contents of the registers, but it cannot alter the contents.
The registers are:

DB-STATUS

The reserved word, DB-STATUS, returns the status of every DML statement at the conclusion of the statement. The implicit description is that of an alphanumeric data item of seven characters. If no exceptions are encountered during the execution of the statement, DB-STATUS is set to zero. If an exception is encountered, DB-STATUS is set to the appropriate value. See Data Base Status Indicators.

DB-REALM-NAME

The reserved word, DB-REALM-NAME, returns the name of a realm at the conclusion of certain DML statements. The implicit description is that of an alphanumeric data item of 30 characters. A successful statement other than FIND or STORE leaves this register unchanged.
Each successful FIND and STORE statement updates DB-REALM-NAME with the appropriate realm-name. An unsuccessful function may supply a realm-name in the register in addition to the data-base-exception code supplied in DB-STATUS. If a realm-name is not relevant to the unsuccessful function, the register is set to blanks.

DB-RECORD-NAME

The reserved word, DB-RECORD-NAME, returns the name of a record at the conclusion of certain DML statements. The implicit description is that of an alphanumeric data item of 30 characters. A successful statement other than FIND or STORE leaves this register unchanged.
Each successful FIND and STORE statement updates DB-RECORD-NAME with the appropriate record-name. An unsuccessful function may supply a record-name in the register, in addition to the data-base-exception code supplied in DB-STATUS. If a record-name is not relevant to the unsuccessful function, the register is set to blanks.

DB-SET-NAME

The reserved word, DB-SET-NAME, returns the name of a set at the conclusion of certain DML statements. The implicit description is that of an alphanumeric data item of 30 characters. A successful statement leaves this register unchanged. An unsuccessful statement may supply a set-name in the register, in addition to the data-base-exception code supplied in DB-STATUS. If a set-name is not relevant to the unsuccessful function, the register is set to blanks.

DB-DETAILED-STATUS

The reserved word, DB-DETAILED-STATUS, returns an explanatory message upon the abnormal conclusion of certain DML statements.

The implicit description is the following:

```
01  DB-DETAILED-STATUS.
 02 DB-MESSAGE-LENGTH USAGE COMP-1.
 02 DB-MESSAGE-TEXT PIC LX(256)
                    DEPENDING ON DB-MESSAGE-LENGTH.
```

If no message is present DB-MESSAGE-LENGTH is set to 0.

5.8 DATA BASE STATUS INDICATORS

The execution of any data base manipulation statement gives a value in the special register, DB-STATUS. This value, known as a "data base status indicator", consists of a statement code in the leftmost two character positions and a status code in the rightmost five character positions.
If the execution of any data base manipulation statement results in a data-base-exception, the statement code indicates which type of manipulation statement caused the exception. If the execution of a data base manipulation statement does not result in a data-base-exception condition, both the statement code and the status code are set to zero.

5.8.1 Statement Code and Status Code

The leftmost two characters of the data base status indicator contain the statement code. See Table 5-1 for statement codes.

The rightmost five characters of the data base status indicator contain the status code that indicates a specific data-base-exception condition.
See Table 5-2 for data base status codes.

5.8.2 Statement Code and Status Code Combinations

The valid permissible combinations of statement code and status code are shown in Table 5.3, Statement and Status Code Combinations. An X at an intersection indicates a permissible conbination. This table relates to generic DML functions. For more detailed information about the variations of these functions reference should be made to the description of each DML statement.

5.8.3 Use of Special Registers for Data-base-exception Conditions

When the DBCS recognizes certain data-base-exception conditions, it places values into the special registers DB-REALM-NAME, DB-RECORD-NAME, and DB-DETAILED-STATUS, DB-SET-NAME, when relevant.

DB-REALM-NAME

During the execution of a data base manipulation statement that is not successfully completed, the realm-name of a realm associated with the data-base-exception condition is placed in DB-REALM-NAME.
For example: FIND NEXT WITHIN realm-1 can return and "End of Realm" exception. In this case the realm-name will be placed in DB-REALM-NAME.

DB-RECORD-NAME

During the execution of a data base manipulation statement that is not successfully completed, the record-name of a record-type associated with the data-base-exception condition is placed in DB-RECORD-NAME.

DB-SET-NAME

During the execution of a data base manipulation statement that is not successfully completed, the set name of a set-type associated with the data-base-exception condition is placed in the special register, in DB-SET-NAME. For example, FIND NEXT WITHIN set-name-1 can return an "End of Set" exception. In this case, the set-name is placed in DB-SET-NAME.

DB-DETAILED-STATUS

During the execution of a data base manipulation statement that is not successfully completed, there may be in DB-DETAILED-STATUS an explanatory message complementing the DB-STATUS value. For example, a STORE may return a message indicating which field did not satisfy the validity checks.

Table 5-1. Statement Codes

Statement	Statement Code
ACCEPT	01
CONNECT	02
DISCONNECT	03
ERASE	04
FIND	05
FINISH	06
GET	08
DB Condition (IF)	09
MODIFY	11
READY	13
STORE	15

Table 5-2. Data Base Status Codes

Status Code	**Data-base-exception Condition**
DATA BASE	RETRIEVAL EXCEPTION CONDITIONS
02100	End of a set or realm has been reached.
02300	No set can be located to satisfy the set selection criteria.
02400	No record can be located to satisfy the record-selection-expression.
CURRENCY	INDICATOR EXCEPTION CONDITIONS
03100	Current record of realm, set-type, or record-type is null (or virtual).
03200	Current record of the run-unit is null.
03300	Current record of the run-unit, realm, or set-type is not of correct record-type.
NAME	SPECIFICATION EXCEPTION CONDITIONS
04100	Data-base-key value is inconsistent with realm name.
04300	Realm-name is invalid (not known to the schema or inconsistent with record-type).
DATA ITEM VALUE	EXCEPTION CONDITIONS
05100	Contents of data items are duplicated in the data base.
05200	Contents of data items do not satisfy validity checks.
DELETED RECORD	EXCEPTION CONDITIONS
07200	Deletion of a non-empty set (ERASE).
MEMBERSHIP	EXCEPTION CONDITIONS
08100	Object record is already connected to the set. (CONNECT)
08300	Object recodr is not currently connected to the set. (DISCONNECT OR MODIFY)
READY MODE	EXCEPTION CONDITIONS
09100	Realm or data base not in ready state.
09200	Realm not in update mode.
09300	Realm already in ready state.
GCOS 7 SPECIFIC	
73630	Ordinal null in FIND identifier WITHIN set or realm.
73640	CALC-key modification cannot be performed without migration.
73650	Set membership violates AREA OF OWNER clause (MODIFY or CONNECT).
73660	UWA record-description missing
RESOURCE	ALLOCATION EXCEPTION CONDITIONS
80200	Space in realm is exhausted.

Table 5-3. Statement and Status Code Combinations

STATUS CODE	STAT	EMENT	CODE									CONDITION
	01 ACC	02 CON	03 DISC	04 ER A	05 FIND	06 FINIS	08 GET	09 IF (DB Cond)	11 MODI	13 REA	15 STO	
02100					X							End of realm or set has been reached.
02300		X			X				X		X	No set found to satisfy set selection criteria.
02400					X							No record found to satisfy record selection.
03100	X				X			X				Current record of realm, set-type or record-type is null.
03200	X	X	X	X	X		X	X	X			Current record of run-unit is null.
03300		X	X	X	X		X	X	X			Current record of run-unit, realm or set-type is not correct type.
04100	X				X						X	Data-base-key is inconsistent with realm-name.
04300					X						X	Realm-name is invalid.
05100		X							X		X	Contents of data items duplicated on data base.
05200					X				X		X	Contents of data items do not satisfy validity checks.
07200				X								Deletion of non-empty set specified.
08100		X										Object record is already connected to the set.
08300			X						X			Object record not currently connected to the set.
09100	X	X	X	X	X	X	X	X	X		X	Realm or data base not ready.
09200		X	X	X					X		X	Realm not in update mode.
09300									X			Realm already in ready state.
73630					X							Ordinal null in FIND identifier WITHIN set or realm.
73640									X			CALC-key modification cannot be performed without migration.
73650		X							X			Set membership violates "AREA OF OWNER" clause.
73660							X		X			UWA record-description missing.
80200											X	Space in realm exhausted.

5.9 DATA BASE CURRENCY INDICATORS

In the environment of a data base, the user must be able to access records that are a part of more than one logical relationship. Unlike sequential files, many records can logically follow a given record, depending upon the structure of the data base and the search criteria in use. Thus, there can be several simultaneous "current records" during the execution of a run-unit, and several simultaneous "current record pointers". These simultaneous current record pointers are known as "currency indicators". The following currency indicators are maintained during the execution of a program:

- The current record of the run-unit.
- The current record of each set-type.
- The current record of each realm.
- The current record of each record-type.

The DBCS maintains a pointer to each of these record occurrences. Any of these pointers may be <u>NULL</u>; that is, they may specify no record if there is no current record of that type. The current record of a set-type or a realm may be <u>"virtual"</u> if it identifies a position in the set or realm that is between two records. This can happen because of a DISCONNECT or ERASE statement. A currency indicator of this type is valid for all DML statements except "FIND CURRENT WITHIN set-name or realm-name", "FIND DUPLICATE WITHIN set-name ...", "ACCEPT ... FROM set-name or realm-name CURRENCY or REALM-NAME", and "Data Base Condition (IF)".

5.9.1 Current Record of the Run-unit

The pointer for each run-unit is maintained to identify the object record for certain data manipulation statements that do not specify a particular record. The execution of some types of statements causes the DBCS to update its pointer to a different record, while the execution of other types of statements does not affect the pointer to the current record of the run-unit.

5.9.2 Current Record of a Set-type

For each Set Description entry the DBCS maintains a separate pointer.
The pointer for each set-type is maintained by the DBCS to identify the record in that set-type that was last referenced by a successfully executed statement. A single record, when accessed, sets the currency indicators for all sets of which it is an owner or member, unless a RETAINING phrase was specified in the DML statement. A DISCONNECT or ERASE statement may set the currency indicators to the position between two records that the current record occupied.

5.9.3 Current Record of a Realm

For each Realm Description entry the DBCS maintains a separate pointer.
The pointer for each realm is maintained by the DBCS to identify the record in that realm that was last referenced by a successfully executed statement, unless a RETAINING phrase was specified in the DML statement.
An ERASE statement may set the currency indicator to the position between two records that the current record occupied.

5.9.4 Current Record of a Record-type

For each Record Description entry the DBCS maintains a separate pointer. The pointer for each record-type is maintained by the DBCS to identify the record of that type that was last referenced by a successfully executed statement, unless a RETAINING phrase was specified in the DML statement.

5.9.5 Currency Values

In addition to the currency indicators, the user can keep the data-base-key value for any record or set accessed. A statement such as

```
ACCEPT SAVE-RECORD-03-DBK FROM RECORD-03 CURRENCY.
```

would allow the manual retention of the location for the current record of that type. In some complex structures it may be advisable to keep needed currency values within the control of the program, since many DML functions can change the currency indicators. When the record in the example is needed later, a statement such as

```
FIND RECORD-03 DB-KEY IS SAVE-RECORD-03-DBK.
```

would re-establish the saved record as current of run-unit and reset all appropriate currency indicators for the record.

5.9.6 Currency Indicators and DML Statements

Table 5-4 treats each DML statement with respect to each of the four types of currency indicator. It indicates the currencies used as implicit input parameters and the currencies updated on successful completion.

A statement of the form "FIND FIRST WITHIN realm-1" establishes currency for the run-unit, the realm, the record-type, and any set-types of which the found record is an owner or member. The record located first within the realm establishes currency for the appropriate sets. Thus, if the statement "FIND NEXT WITHIN set-name-1" is issued, then the currency indicators for run-unit, realm, record-type, and set-types are all updated to reflect the first record found in that set. If the set is empty then an "End of Set" exception condition is returned and the currency indicators for run-unit, realm, record-type, and set-types are not updated.

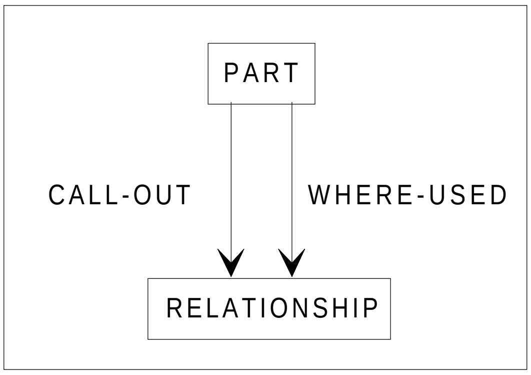
When a new record is added to the data base (STORE), currency indicators for run-unit, realm and record-type are affected. Any set-types of which the new record-type is an owner or member are also affected.

When a record is removed from the data base (ERASE), currency indicators are affected. Current of run-unit is set to null. Current of record-type is set to null if the record erased is the current record of the record-type. Current of set-type and realm are set to virtual if the record erased is the current of set-type and/or realm respectively. Even though the current record of set-type or realm is not available, the currency indicator identifies the position between two records that the current record occupied.

When an unsuccessful DML statement is executed, the currency indicators are not updated. For example, if the statement "FIND NEXT WITHIN set-name-1" returns an "end of set" status, the currency indicator for set-name-1 will still identify the last record in the set occurrence. If the set was empty the currency indicator will identify the owner record since that is what it identified before the FIND statement was executed.
Note that the special registers DB-REALM-NAME, DB-RECORD-NAME, and DB-SET-NAME are updated by an unsuccessful DML statement.

There are some instances when the automatic updating of currency indicators is not desired. The RETAINING CURRENCY FOR phrase can be used with a FIND, STORE, MODIFY or CONNECT statement to inhibit updating of the specified indicators. For example, in a complex network structure (as diagrammed below) involving a PART record and a RELATIONSHIP record used to generate a parts explosion, it may not be desired to affect the currency for the CALL-OUT type when listing out all the parts involved in an assembly.

5.9.7 Example



A specific PART is located by a FIND ANY PART statement that establishes that record as being the current of the CALL-OUT and WHERE-USED sets.
When traversing the CALL-OUT set to determine all the components of the PART, it is necessary to preserve the set currency when going from each RELATIONSHIP record up to the owner PART record. The operation would be coded as:

```
   FIND ANY PART.
LOOP-ON-COMPONENTS.
   FIND NEXT RELATIONSHIP WITHIN CALL-OUT.
   IF DB-STATUS EQUAL "0502100" GO TO LAST-COMPONENT.
   FIND OWNER PART WITHIN WHERE-USED
   RETAINING CURRENCY FOR CALL-OUT.
   GET.
   .
   .
   .
   GO TO LOOP-ON-COMPONENTS.
```

If the RETAINING CURRENCY FOR CALL-OUT phrase was omitted, then each subordinate PART record found would reset the currency for that set and probably send the reporting program down the wrong path.

Table 5-4. Currency Usage and Update (1/3)

STATEMENT	**CURRENCIES USED IN INPUT**	**CURRENCIES UPDATED ON SUCCESSFUL COMPLETION**
ACCEPT...FROM CURRENCY	current-of-run-unit	
ACCEPT...FROM realm-name CURRENCY	current-of-realm specified	
ACCEPT...FROM record-name CURRENCY	currend-of-record-type specified	
ACCEPT...FROM set-name CURRENCY	current-of-set-type specified	
ACCEPT...FROM REALM-NAME	current-of-run-unit	
ACCEPT...FROM record-name REALM-NAME	current-of-record-type specified	
ACCEPT...FROM set-name REALM-NAME	current-of-set-type specified	
ACCEPT...FROM set-name NEXT/PRIOR/OWNER	current-of-set-type specified	
CONNECT...TO set-name	.current-of-run-unit, as object record .if First level of set selection path is identified by APPLICATION, the corresponding current-of-set-type is used. In addition, if the set selection path has only one level and if ORDER is NEXT or PRIOR, the current-of-set-type identifies the insertion point	current-of-set-type specified, if not retained
DISCONNECT...FROM set-name	current-of-run-unit, as object record	current-of-set-type specified is made "virtual" if the object record was the current of this set
ERASE... [ALL MEMBERS]	current-of-run-unit, as object record	.current-of-run-unit is nulled .current-of-realm is made "virtual" if an erased record was current of its realm .current-of-record-type is nulled if an erased record was current of its record-type .current-of-set-type of the sets in which an erased record was an owner is nulled if this record was the current of set. .current-of-set-type of the sets in which an erased record was an actual member is made "virtual" if this record was the current of set.
FIND...DB-KEY IS...		.current-of-run-unit .current-of-realm, if not retained

Table 5-4. Currency Usage and Update (2/3)

FIND ANY record-name		.current-of-record-type, if not retained
FIND DUPLICATE record-name	current-of-run-unit, to identify CALC record-type and position in "synonym"chain	.current-of-set-type, if not retained, of all the sets in which the record is an owner or an AUTOMATIC-MANDATORY member or a MANUAL-OPTIONAL member currently connected.
FIND FIRST/LAST/ ordinal... WITHIN realm-name		.current-of-run-unit .current-of-realm, if not retained
FIND NEXT/PRIOR... WITHIN realm-name	current-of-realm specified, to identify position in realm	.current-of-record-type, if not retained
FIND FIRST/LAST/ ordinal...WITHIN set-name	current-of-set-type specified, to identify set occurrence	.current-of-set-type, if not retained, of all the sets in which the record is an owner or an AUTOMATIC-MANDATORY member or a MANUAL-OPTIONAL member currently connected
FIND NEXT/PRIOR ...WITHIN set-name	current-of-set-type specified, to identify set occurrence and position in set occurrence	
FIND OWNER WITHIN set-name	current-of-set-type specified, to identify set occurrence	
FIND CURRENT	current-of-run-unit	
FIND CURRENT record-name	current-of-record-type specified	
FIND CURRENT... WITHIN realm-name	current-of-realm specified	
FIND CURRENT... WITHIN set-name	current-of-set-type specified	
FIND record-name WITHIN set-name CURRENT [USING...]	current-of-set-type specified, to identify set occurrence	
FINDrecord-name WITHIN set-name [USING...]	if first level of set selection path is by APPLICATION, the corresponding current-of-set-type is used	
FIND DUPLICATE WITHIN set-name	currend-of-set-type specified, to identify record-type, set occurrence and position in set occurrence	
GET	current-of-run-unit, as object record	
OWNER/MEMBER/TENANT CONDITION	current-of-run-unit	
set-name EMPTY CONDITION	current-of-set-type specified, to identify set occurrence	
MODIFY... (no membership)	current-of-run-unit, as object record	

Table 5-4. Currency Usage and Update (3/3)

MODIFY... MEMBERSHIP	.current-of-run-unit, as object record .if first level of set selection path of a new set occurrence is identified by APPLICATION, the corresponding current-of-set-type is used. Also, if the set selection path has only one level and if ORDER is NEXT or PRIOR, the current-of-set-type identifies the insertion point.	.current-of-realm, if not retained .current-of-record-type, if not retained .current-of-set-type, if not retained, of all the sets in which the record is an owner or an AUTOMATIC-MANDATORY member or a MANUAL-OPTIONAL member currently connected
READY...		
STORE record-name	if first level of a set selection path is identified by APPLICATION, the corresponding current-of-set-type is used. In addition, if the set selection path has only one level and if ORDER is NEXT or PRIOR, the current-of-sey-type identifies the insertion point	.current-of-run-unit .current-of-realm, if not retained .current-of-record-type, if not retained .current-of-set-type, if not retained, of all the sets in which the record is an owner or an AUTOMATIC-MANDATORY member or a MANUAL-OPTIONAL member currently connected.
IDS session initial conditions		all currencies are nulled

5.10 DATA MANIPULATION LANGUAGE STATEMENTS

The Data Manipulation Language consists of the following statements:

ACCEPT	Obtain the data-base-keys or realm-names held within the DBCS currency indicators, or obtain DMCL information from the object schema.
CONNECT	Insert a new record occurrence into a set in which the record has been defined to be a MANUAL OPTIONAL member.
DISCONNECT	Remove a record from a set in which it resides as a MANUAL OPTIONAL member.
ERASE	Remove a record from the data base, provided it has no members.
FIND	Locate any record in the data base subject to a variety of record-selection-expression options.
FINISH	Make a realm unavailable for further accesses.
GET	Obtain the contents of a current record.
DB Condition (IF)	Test data base conditions.
MODIFY	Alter the contents of data items on the data base and/or the set relationships of a record.
READY	Make the contents of a realm available for processing.
STORE	Add a new record occurrence to the data base.
USE	Define a DB-EXCEPTION procedure to be automatically invoked when needed.

Each DML statement are described in detail below. The tables showing the DB-STATUS values are interpreted as follows:

A "_" in the columns REALM, RECORD or SET indicates that the corresponding register DB-REALM-NAME, DB-RECORD-NAME or DB-SET-NAME is left unchanged by the DBCS.

A blank in the columns REALM, RECORD or SET indicates that the corresponding register is filled with blanks by the DBCS.

An "X" in the columns REALM, RECORD or SET indicates that the corresponding register is filled by the DBCS with the name of a realm, record or set. If "X" is enclosed in parentheses, the register is filled with either a name or blanks, depending on the error.

The order in which the error conditions are presented for a given DML statement reflects, in most cases, the sequence of tests performed by the DBCS during the execution of the DML function.

5.10.1 ACCEPT

Function

Causes the contents of the specified currency indicators to be made available to the program, provides a means for deriving the realm name that corresponds to a data-base-key value, supplies the data-base-keys of the NEXT, PRIOR, OWNER of a record within a set and reads DMCL information from the object schema.

General Format

```
Format 1

                           [record-name]
ACCEPT identifier-1 FROM   [set-name   ]  CURRENCY
                           [realm-name ]


Format 2

                           [record-name ]
ACCEPT identifier-2 FROM   [set-name    ] REALM-NAME
                           [identifier-3]


Format 3

                                   {NEXT  }
ACCEPT identifier-4 FROM set-name  {PRIOR }
                                   {OWNER }


Format 4

ACCEPT identifier-5 FROM realm-name LINES-PER-PAGE


Format 5

ACCEPT identifier-6 FROM realm-name MINIMUM-DB-KEY
                                    [ OF record-name ]


Format 6

ACCEPT  identifier-7 FROM realm-name NUMBER-OF-PAGES
                                     [ OF record-name ]
```

Syntax Rules

1. Identifier-1, identifier-3, identifier-4 and identifier-6 must be DB-KEY items.

2. Identifier-2 must be an alphanumeric elementary item. It must be large enough to handle any REALM-NAME. The minimum size is 30 (PIC X(30)).

3. Identifier-5 must be an elementary integer item with a range at least equal to the range of a COMP-1 item.

4. Identifier-7 must be an elementary integer item with a range at least equal to the range of a COMP-2 item.

General Rules

Format 1

1. If record-name, set-name or realm-name is specified, the data-base-key value for the current record of record-name, set-name or realm-name respectively is placed in the data item referenced by identifier-1.

2. If a record-name, set-name or realm-name is not specified, the data-base-key value for the current record of the run-unit is placed in the data item referenced by identifier-1.

3. The currency indicator involved must be neither NULL nor virtual.

Format 2

1. The realm-name that is derived from the data-base-key value in the data item referenced by identifier-3 or from the specified currency indicator is placed in the data item referenced by identifier-2, according to the rules for an alphanumeric elementary move.

2. If identifier-3 or a record-name or a set-name is not specified, the name of the realm that is associated with the current record of the run-unit is placed in the data item referenced by identifier-2, according to the rules for an alphanumeric elementary move.

3. If a currency indicator is involved, it must be neither NULL nor virtual.

Format 3

1. The currency indicator of set-name must point to a record (owner or member) or indicate a position (virtual).

2. The data-base-key of the NEXT or PRIOR or OWNER record within set-name of this record or position is placed in identifier-4.

   The NEXT or PRIOR record may be the owner itself. (There is no "end-of-set" condition). The set may be empty.

Format 4

1. The number of lines per page of realm-name is moved into the user defined item identifier-5.

Format 5

1. The lowest (first) data-base-key value of realm-name is placed in identifier-6. If the optional record-name is present in the clause then the value supplied is the lowest (first) data-base-key in the realm for the range of this record-type.

Format 6

1. The number of pages of realm-name is placed in identifier-7. If the optional record-name is present then the size, in pages, of the record range within realm-name, is placed in identifier-7.

DB-STATUS Values

FUNCTION CATEGORY	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
	None	0000000	-	-	-
	Data base not open	0109100			
ACCEPT ...CURRENCY	CRU null	0103200			
	Current-of-area null or virtual	0103100	X		
	Current-of-record-type null	0103100		X	
	Current-of-set-type null or virtual	0103100			X
ACCEPT ...REALM-NAME	CRU null	0103200			
	Current-of-record-type null	0103100		X	
	Current-of-set-type null or virtual	0103100			X
	Data-base-key inconsistent	0104100			
ACCEPT ... NEXT/PRIOR/ OWNER	Inconsistent	0104100			
	Current-of-set-type null	0103100			X

5.10.2 CONNECT

Function

Causes a record stored in the data base to become an actual member of a named set in which the record has been declared to be a MANUAL OPTIONAL member. The current record of the run-unit is the object of the statement.

General Format

```
CONNECT      [record-name ] TO set-name
    [RETAINING CURRENCY FOR     {SETS      }  ]
    [                           {set-name  }  ]
```

Syntax Rules

1. Record-name is not required. If specified, record-name must be the name of the current record of the run-unit.

2. The current record of the run-unit must be defined to be a MANUAL OPTIONAL member of the set-type specified in the CONNECT statement.

General Rules

1. Execution of the CONNECT statement causes the current record of the run-unit to become an actual member of the specified set-type provided that it is not currently connected to an occurrence of this set-type.

2. The set occurrence to which the current record of the run-unit is to be connected is determined by the set selection criteria. The record is connected to the set occurrence in accordance with the set ordering criteria of the set-type.

3. The realms in which the current record of the run-unit is stored and in which the records of the affected set are stored must be open in update.

4. If the RETAINING phrase is not specified, the current record of the run-unit becomes the current record of the set-type into which the record has been connected. Other currency indicators are not affected.

5. If the RETAINING phrase is specified, no currency indicator is affected.

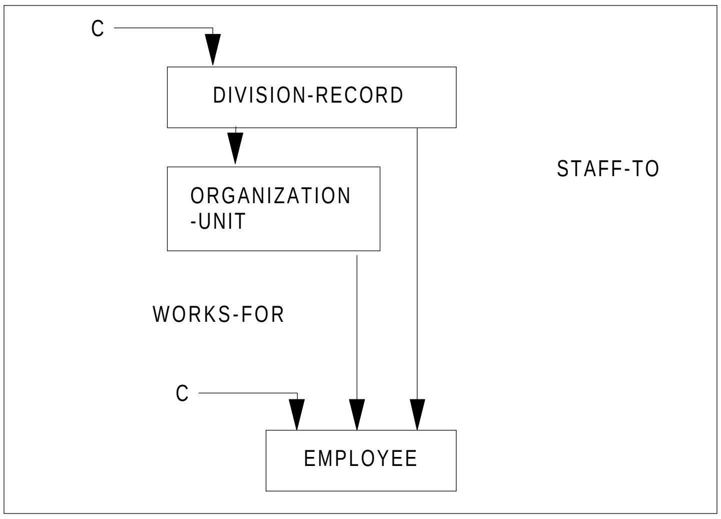
NOTES:

1. The set selection process uses the field or db-parameter values of the UWA. The set insertion process uses the sort-key and no-duplicate-control-field values of the record on the data base.

2. If a data-base-exception occurs during the execution of a CONNECT statement, the data base is restored to the state it was in before the statement was executed (this is performed without using the journals). No currency indicators are affected by an unsuccessful statement.

Example

The EMPLOYEE record is a MANUAL OPTIONAL member of the STAFF-TO set, with a set selection path consisting of only 1 level identified by CALC-KEY.



The DML sequence connecting an EMPLOYEE to a DIVISION-RECORD is the following:

```
MOVE "mmm" TO EMPLOYEE-NUMBER.
FIND ANY EMPLOYEE.              Establish current-of run-unit
MOVE "nnnn" TO DIVISION-NUMBER. Prepare set selection path
CONNECT EMPLOYEE TO STAFF-TO.   Connect
```

FUNCTION **CATEGORY**	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
	None record connected	0000000	-	-	-
	Data base not open	0209100			
	CRU null	0203200			
	CRU not of correct type (when specified) or not declared as OPTIONAL member of the set	0203300		X (cru)	(X)
	CRU is an OPTIONAL member of the set but is currently connected	0208100		X	X
	Area not in ready state	0209100	X	X	(X)
	Area not in update mode	0209200	X	X	(X)
	Owner of new set occurrence does not respect the "AREA OF OWNER" condition	0273650	X (owner)	X	X
	Set selection failed	0202300		X	X
	Duplicate not allowed (sort-key or other control field)	0205100		X	X

5.10.3 DISCONNECT

Function

Logically removes a record from a specified set if the record is a MANUAL OPTIONAL member of the set. The current record of the run-unit is the object of the statement.

General Format

```
 ___________________________________________
|                                           |
| DISCONNECT [record-name] FROM set-name    |
|___________________________________________|
```

Syntax Rules

1. Record-name is not required. If specified, record-name must be the name of the current record of the run-unit.

2. The current record of the run-unit must be defined to be a MANUAL OPTIONAL member of the set-type specified in the DISCONNECT statement.

General Rules

1. Execution of the DISCONNECT statement causes the current record of the run-unit to be removed from membership in the specified set-type provided that it is currently connected to an occurrence of this set-type.

2. The realms in which the current record of the run unit is stored and in which the records of the affected set are stored must be open in update.

3. If the current record of the run-unit is the current of the set-type specified, the currency indicator for the set-type is updated to identify the position between the two records that the disconnected record occupied. If the current record of the run-unit is not the current record of the set-type specified, the currency indicator is not affected. Other currency indicators are not affected.

NOTES:

1. The current record of the specified set-type may become unavailable, as described in General Rule 3. However, the NEXT, PRIOR, and OWNER records are available. This means that the currency indicator is valid for all subsequent DML statements which use a set position rather than a particular record. For example, the following sequence is valid:

```
DISCONNECT FROM set-name-1.
FIND NEXT WITHIN set-name-1.
```

2. If a data-base-exception occurs during the execution of a DISCONNECT statement, the data base is restored to the state it was in before the statement was executed (without using the journals). In addition, no currency indicators are affected by an unsuccessful statement.

DB-STATUS Values

FUNCTION CATEGORY	ERROR CONDITION	DB-REGISTERS			
		DB-STATUS	REALM	RECORD	SET
	None; record connected	0000000	-	-	-
	Data base not open	0309100			
	CRU null	0303200			
	CRU not of correct type (when specified) or not declared as OPTIONAL member of the set	0303300		X (cru)	(X)
	CRU is an OPTIONAL member of the set but currently not connected	0308300		X	X
	Area not in ready state	0309100	X	X	
	Area not in update mode	0309200	X	X	

5.10.4 ERASE

Function

Removes one or more records from the data base. The current record of the run-unit is the object of the statement.

General Format

```
 ________________________________________
|                                        |
| ERASE [record-name] [ALL MEMBERS]      |
|________________________________________|
```

Syntax Rule

Record-name is not required. If specified, record-name must be the name of the current record of the run-unit.

General Rules

1. Execution of the ERASE statement causes one or more records to be removed from the data base.

2. If ALL is not specified and the current record of the run-unit is not the owner of a set that currently has members, the record is disconnected from the sets to which it is currently connected, if any, and then removed from the data base.

3. If ALL is not specified and the current record of the run-unit is the owner of a set that currently has members, an exception condition results.

4. If ALL is specified, and the current record of the run-unit is the owner of a set that currently has members, then the removal of the current record of the run-unit is executed as explained in General Rule 2 and is followed by the removal of all of these member records. Any record so removed is treated as though it were the object record of an ERASE...ALL statement.

   The process is repeated until all hierarchically related records have been removed from the data base.
   Note: Therefore ALL must always be used WITH CARE.

5. If ALL is specified, and the current record of the run-unit is not the owner of a set that currently has members, execution proceeds as if ALL had not been specified.

6. The realms in which the current record of the run-unit is stored and in which the records of the affected sets are stored must be open in update.

7. Currency indicators are affected as follows:

   - The current record of the run-unit is nulled.

   - If any record removed from the data base is the current record of its record-type, the currency indicator for the record-type is nulled.

   - If any record removed from the data base is the current record of a set-type of which it is the owner, the currency indicator for the set-type is nulled.

   - If any record removed from the data base is the current record of a set-type to which it is connected, the currency indicator for the set-type is updated to identify the position between the two records that the removed record occupied.

   - If any record removed from the data base is the current record of its realm, the currency indicator for the realm is updated to identify the position that the removed record occupied.

NOTES:

1. As specified in General Rules 2 and 4, each object record is disconnected from any sets to which it is currently connected. This means no special consideration need be made for MANUAL OPTIONAL sets. If an object record is currently connected to a MANUAL OPTIONAL set, it will be disconnected before the record is removed from the data base. If an object record is currently disconnected from a MANUAL OPTIONAL set, no error will result and the record will be removed from the data base.

2. The current record of a set-type may become unavailable as described in General Rule 7d; however, the NEXT, PRIOR, and OWNER records are available. This means the currency indicator is valid for subsequent DML statements referencing a position within the set rather than a particular record of the set. For example, the following sequence is valid:

   ```
   ERASE B.
   FIND NEXT WITHIN A-B.
   ```

   where the B record is a member of the A-B set.

3. The current record of a realm may become unavailable as described in General Rule 7e; however, the NEXT and PRIOR records are available. This means that the currency indicator is valid for subsequent DML statements referencing a position within the realm rather than a particular record of the realm.

4. a) If a data-base-exception occurs during the execution of an ERASE statement without member deletion, the data-base is restored to the state it was in before the statement was executed (without using the journals). The currency indicators are not affected and the control is given back to the calling program with the appropriate DB-STATUS value.

   b) If a data-base-exception occurs during the execution of an ERASE statement with member deletion, but before the updating of the data-base has actually begun, then the DBCS proceeds as described above.

   c) If a data-base-exception occurs during the execution of an ERASE statement with member deletion, but after the updating of the data base has actually begun (an area is not in ready state or not in update mode), then the processing depends on the protection level of the data base:
   - If the data base is protected by the BEFORE journal, in BATCH, IOF or TDS environment, the data base is restored to the state that it was in before the statement, the currency indicators are not affected and the control returns to the calling program with the appropriate DB-STATUS value.
   - If the data base is not protected by the BEFORE journal in BATCH or IOF environment, the data base is left in an inconsistent state and the step is aborted. The data base must be restored by FILREST or VOLREST.
   - If the data base is protected by the AFTER journal and deferred updates in TDS environment, the data base is restored to the state it was in at the last commitment point, then the TPR is aborted and not restarted.

DB-STATUS Values

FUNCTION CATEGORY	ERROR CONDITION	DB-REGISTERS			
		DB-STATUS	REALM	RECORD	SET
General	None; record(s) erased	0000000	-	-	-
	Data base not open	0403100			
	CRU null	0403200			
	CRU not of correct type (when specified)	0403300		X (cru)	
	Area not in ready state	0409100	X	X	
	Area not in update mode	0409200	X	X	
ERASE [record-name]	Record is owner of a non-empty set occurrence	0407200	X	X	X

5.10.5 FIND

Function

Establishes a specific record occurrence in the data base as the object of subsequent statements.

General Format

```
FIND record-selection-expression

  [                           { MULTIPLE          }  ]
  [                           { REALM             }  ]
  [ RETAINING CURRENCY FOR    { {SETS         }   }  ]
  [                           { { {set-name}...}  }  ]
  [                           { RECORD            }  ]
```

General Rules

1. The FIND statement causes the record referenced by the record-selection-expression to become the current record of the run-unit. The realm-name for the record found is placed in the special register DB-REALM-NAME and the name of the record found is placed in DB-RECORD-NAME.

2. If the RETAINING phrase is not specified, the record referenced by the record-selection-expression becomes the current of its realm, the current record of its record-type, and the current record of all set-types in which it is a tenant.

3. If the RETAINING phrase with the optional word REALM is specified, the realm currency indicator is not changed.

4. If the RETAINING phrase with the optional word RECORD is specified, the record-type currency indicator is not changed.

5. If the RETAINING phrase with the optional word SETS is specified, no set-type currency indicators are changed.

6. If the RETAINING phrase with the optional set-name, etc. is specified, the set currency indicators for the named set-types are not changed.

7. If the RETAINING phrase with the optional word MULTIPLE is specified then the realm, record-type and set-type currency indicators are not changed.

NOTES:

1. The execution of a FIND statement does not make the selected record contents available to the program; it merely identifies the record for use in subsequent statements. To obtain the contents of data items within the record, a GET statement must be issued.

2. If an exception condition is encountered, no currency indicators are changed. However, the special registers DB-REALM-NAME, DB-RECORD-NAME, and DB-SET-NAME are updated.

3. The currency indicators for a MANUAL OPTIONAL set will not be updated if the record found is not currently connected to the set. The currency indicators will be updated if the record found is currently connected to the set.

Format 1

```
 ________________________________________________
|                                                |
| FIND [record-name-1] DB-KEY IS identifier-1    |
|________________________________________________|
```

Rules

1. Identifier-1 must be defined as a DB-KEY item.

2. The record identified is the record whose data-base-key value is equal to the value of the data item referenced by identifier-1.

3. Record-name-1 is not required. If specified, the record found must be of the record-type specified by record-name-1.

4. The value in identifier-1 must be a complete data-base-key and not an area-key. ACCEPT statements can be used to obtain data-base-key values.

Format 2

```
FIND {ANY      } record-name-2
     {DUPLICATE}
```

Rules

1. The record referenced by record-name-2 must have a CALC location mode.

2. For FIND ANY, a realm is selected, if there is a choice. The schema-supplied area identification parameter is used (WITHIN clause in DDL).

3. For FIND ANY, the CALC-key value for the record named by record-name-2 is used to calculate the data-base-key value for the identified record. The value must be moved into the CALC-key fields in the UWA before the FIND statement is issued.

4. For FIND DUPLICATE, the DBCS searches the CALC chain beginning with the current record of the run-unit, looking for the next record of the same type whose CALC-key is equal to that of the current of run-unit. The FIND DUPLICATE statement assumes that the current of run-unit is already one of the duplicates. The LOCATION MODE CALC declaration in the schema must have DUPLICATES ARE ALLOWED.

NOTE: The value of the CALC-key used during the execution of a FIND DUPLICATE statement is the one found in the record on the data base. The value in the UWA is not used during this execution.

Format 3

```
 ___________________________________________________________
|                                                           |
| FIND DUPLICATE WITHIN set-name-1 USING {identifier-2} ... |
|___________________________________________________________|
```

Rules

1. Identifier-2, ... must be defined in the record entry for the current record of the set-type referenced by set-name-1.

2. The record identified:

   - is a member of the set occurrence identified by the currency indicator for set-name-1.

   - has a record-type equal to the current record of the set type referenced.

   - has the contents of the data items referenced by identifier-2, ... equal to those in the current record of the set-type referenced.

3. The DBCS begins its search at the next record in the forward direction.

NOTES:

1. The values of identifier-2, ... used in the search are those in the record on the data base. The values in the UWA are not used during the execution of the FIND DUPLICATE statement.

2. If the owner record is encountered before a duplicate record is found, a "record-not-found" (02400) exception condition results. If the duplicate record exists between the first and the current record of the set, it will not be found.

Format 4

```
      {NEXT          }
      {PRIOR         }
      {FIRST         }                        {set-name-2   }
FIND  {              } [record-name-3]WITHIN {              }
      {LAST          }                        {realm-name-1 }
      {integer-1     }
      {identifier-4  }
```

Rules

1. Integer-1 may be signed. This allows a relative forward or backward reference.

2. The data item referenced by identifier-4 must be a signed elementary integer, and must have a non-null value.

3. If record-name-3 is not specified, all record-types defined in the schema are considered in evaluating the record-selection-expression.

4. If record-name is specified, only the record-type of the record referenced by record-name-3 is considered in evaluation of the record-selection-expression. All other record-types are ignored.

5. If the NEXT phrase and realm-name are specified, the record identified is the one whose data-base-key value is the next higher, relative to the data-base-key value of the current record of the referenced realm.

6. If the PRIOR phrase and realm-name are specified, the record identified is the one whose data-base-key value is the next lower, relative to the data-base-key value of the current record of the referenced realm.

7. If the FIRST phrase and realm-name are specified, the record identified is the one whose data-base-key value is the lowest, relative to all records stored in the referenced realm.

8. In the LAST phrase and realm-name specified, the record identified is the one whose data-base-key value is the highest, relative to all records stored in the referenced realm.

9. If integer-1 or identifier-4 is specified, and realm-name is specified, the record identified is the one whose ordinal position in the realm is equal to the value of integer-1 or the contents of the data item referenced by identifier-4. If the specified value is positive, the ordinal position is relative to the record whose data-base-key value is the lowest in the referenced realm. If negative, the ordinal position is relative to the record whose data-base-key value is the highest in the referenced realm.

10. If the NEXT phrase and set-name are specified, the record identified is the next record in the set relative to the current record of the set, according to the conventional set order.

11. If the PRIOR phrase and set-name are specified, the record identified is the prior record in the set, relative to the current record of the set, according to the conventional set order.

12. If the FIRST phrase and set-name are specified, the record identified is the first member of the set in which the current record of the set is a member, according to the conventional set order.

13. If the LAST phrase and set-name are specified, the record identified is the last member of the set in which the current record of the set is a member, according to the conventional set order.

14. If an integer-1 or identifier-4 is specified, and set-name is specified, the record identified is the one whose ordinal position in the set in which the current record of the set is a member is equal to the value of integer-1 or the contents of the data item referenced by identifier-4. If the value specified is positive, the ordinal position is relative to the first member of the set, according to the conventional set order. If negative, the ordinal position is relative to the last member of the set, according to the conventional set order.

Format 5

```
                              [        {realm-name-2} ]
FIND CURRENT [record-name-4] [WITHIN {             } ]
                              [        {set-name-3   } ]
```

Rules

1. If set-name-3 is specified, the record identified is the current record of the set-type indicated.

2. If realm-name-2 is specified, the record identified is the current record of the realm indicated.

3. If record-name-4 and set-name-3 are specified, then record-name-4 must be the current record of the set-type indicated.

4. If record-name-4 and realm-name-3 are specified, then record-name-4 must be the current record of the realm indicated.

5. If record-name-4 is specified and the WITHIN clause is omitted, the record identified is the current record of the record-type indicated.

6. If neither realm-name-2, set-name-3, nor record-name-4 are specified, the record identified by the statement is the current record of the run-unit.

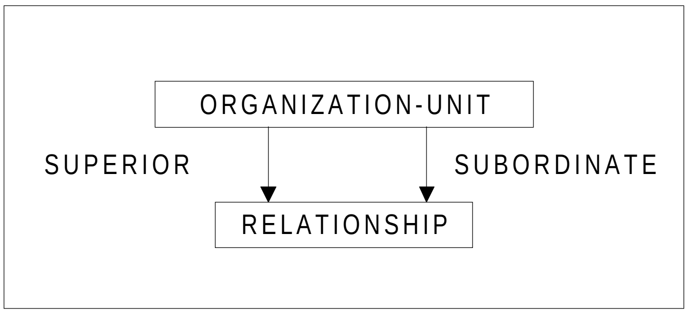
Format 6

```
 ________________________________
|                                |
| FIND OWNER WITHIN set-name-4   |
|________________________________|
```

Rules

1. The record is the owner of the set occurrence identified by the currency indicator for set-name-4.

2. In some complex network structures it is advisable to use the optional RETAINING CURRENCY FOR... clause to insure that other set currencies are not altered by the execution of the FIND. When the owner record is found, it is made the current record of all set-types in which it is a tenant. For example, in a network structure for an organization chart application:



When traversing the SUPERIOR set to locate all other ORGANIZATION-UNITs which are subordinate to the current ORGANIZATION-UNIT record, the FIND OWNER WITHIN SUBORDINATE statement is issued after each RELATIONSHIP record has been found. The new ORGANIZATION-UNIT record will reset the current of set-type indicator for the SUPERIOR set, thereby terminating the initial retrieval. If, instead, the statement were written: FIND OWNER WITHIN SUBORDINATE RETAINING CURRENCY FOR SETS., then when the new owner ORGANIZATION-UNIT record is found, the current record of SUPERIOR set is unchanged, allowing the next RELATIONSHIP record to be found.

Format 7

```
 _________________________________________________
|                                                 |
| FIND record-name-5 WITHIN set-name-5 [CURRENT]  |
|                                                 |
|      [USING {identifier-5}...]                  |
|_________________________________________________|
```

Rules

1. Identifier-5, ... must be defined in the record entry for the record-type record-name-5.

2. Record-name-5 must be defined in the schema DDL as a member of the set-type set-name-5.

3. The record identified has a type equal to that of the record referenced by record-name-5.

4. If USING is specified the record identified has the contents of the data items referenced by identifier-5, ... equal to these data items in the UWA. If USING is not specified, the record identified is the first record in the set occurrence.

5. The DBCS proceeds in its search as follows:

   - If CURRENT is not specified the correct set occurrence is selected according to the set selection rules specified in the schema DDL.

     The entry point record is selected by one of three methods: by APPLICATION (currency indicator for set-type); CALC-KEY (randomize a specified control field); or DATA-BASE-KEY (data-base-key of record supplied by program).

     Each intervening set is searched based on the keys defined in the schema DDL. Each set searched corresponds to a THEN THRU clause in the set entry of the schema DDL.

     The last set searched identifies the set occurrence in which the identified record is a member. If there are no THEN THRU clauses in the schema DDL, the entry point record identifies the set occurrence in which the identified record is a member.

   - If CURRENT is specified, the set occurrence is identified by the current of the set-type.

   - The set occurrence identified as described above, is searched in the forward direction to locate, if USING is specified, the record in which the contents of identifier-5 ... are equal to the contents of those data items in the UWA. If USING is not specified, the first record is selected. Only the records of the type referenced by record-name-5 are considered in this search.
     Other record-types are ignored.

NOTES:

1. If the entry point record as described in Rule 5a.- is identified by CALC-KEY and the record can reside in more than one realm, realm selection will be invoked. The schema-supplied area identification parameter is used. The CALC-key for this record must be initialized in the UWA before executing the FIND statement.

2. If the entry point record as described in Rule 5a.- is identified by APPLICATION, the current record of the set type specified in the schema DDL must be established before executing the FIND statement.
   This set-type is the one referenced by the first THRU phrase of the SELECTION clause of the MEMBER subentry in the schema DDL. The record identified may be either an owner or member of the set-type.

3. If the entry point record as described in Rule 5a.- is identified by DATA-BASE-KEY, the data-base-parameter specified in the schema DDL must be set to the correct value before executing the FIND statement. The record identified must be the owner of the set.

4. The set-selection-keys (if any) as described in Rule 5a.- must be set to the correct values in the UWA before executing the FIND statement.

5. If a set occurrence is not found which satisfies the set selection criteria as described in Rule 5a, the exception condition "set selection not satisfied" (02300) is returned. This can occur during any of the steps described in Rule 5a.

6. If the correct set occurrence is found, but a record cannot be found that satisfies the record-selection-expression as described in Rule 5b, the exception condition "record-selection expression not satisfied" (02400) is returned.

7. The search of the occurrence as described by Rule 5b will search the entire set occurrence, beginning at the first record of that type in the set occurrence.

DB-STATUS Values

FUNCTION CATEGORY	ERROR CONDITION	DB-REGISTERS			
		DB-STATUS	REALM	RECORD	SET
General	None; record found	0000000	X	X	-
	Data base not open	0509100			
	Area not in ready state	0509100	X	X	
FIND DB-KEY IS...	Data-base-key inconsistent or does not correspond to record-type range	0504100			
	Valid data-base-key but no record found or record found or record found has not the correct type (when specified)	0502400	X	(X)	
FIND ANY... (CALC)	Area-name supplied in AREA-ID is unknown or does not correspond to record-type	0504300			
	Illegal decimal data in UWA CALC-key	0505200		X	
	No record found with this CALc-key	0502400	X	X	
	UWA record-description missing	0573660		X	
FIND DUPLICATE... (CALC)	CRU null	0503200			
	CRU not of correct type	0503300		X (cru)	
	No duplicate found	0502400	X	X	
FIND relative within area	Current-of-area null (NEXT,PRIOR)	0503100	X		
	Ordinal= 0	0573630			
	Record not found (end of realm)	0502100	X	(X)	
FIND relative within set	Current-of-set null	0503100			X
	Ordinal= 0	0573630			
	Record not fount (end of set)	0502100		(X)	X
FIND OWNER within set	Current-of-set null	0503100			X
FIND CURRENT	CRU null	0503200			
FIND CURRENT record-name	Current-of-record-type null	0503100		X	
FIND CURRENT [record-name] WITHIN area	Current-of-area null or virtual	0503100	X		
	Current-of-area not of correct type (when specified)	0503300		X	
FIND CURRENT [record-name] WITHIN set	Current-of-set null or virtual	0503100			X
	Current-of-set not of correct type (when specified)	0503300		X	
FIND record-name WITHIN set [CURRENT] [USSING]	Current-of-set null	0503100			X
	Set selection failed	0502300		X	X
	UWA illegal decimal data	0505200		X	
	Record not found within occurrence selected	0502400		X	X
	UWA record-description missing	0573660		X	
FIND DUPLICATE WITHIN set USING...	Current-of-set null or virtual	0503100			X
	Current-of-set is the owner or is not of correct type	0503300		X	
	No duplicate found	0502400		X	X

5.10.6 FINISH

Function

Ends the availability of one or more realms to the program.

General Format

```
 ________________________________
|                                |
| FINISH [realm-name-1] ...      |
|________________________________|
```

General Rules

1. Execution of the FINISH statement ends the availability of the realm(s).

2. If realm-name-1,... is specified, the ready state of the realms indicated is terminated. Realm-name-1 may appear only once in a specific FINISH statement.

3. If realm-name-1,... is not specified, the ready state of all the realms defined in the schema is terminated.

4. At the completion of the execution of a FINISH statement, all currency indicators that identify records stored in the affected realms are made null.

5. All affected realms must be in ready state at the beginning of execution of a FINISH statement.

DB-STATUS Values

FUNCTION CATEGORY	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
	None	0000000	-	-	-
	Data base not open	0609100			
	Area not in ready state	0609100	X		

5.10.7 GET

Function

Makes some or all the fields in a data base record available to the program. The object of the statement is the current record of the run-unit.

General Format

```
                 __________________________
                |                          |
Format 1        | GET [record-name]        |
                |__________________________|

                 __________________________
                |                          |
Format 2        | GET {identifier-1} ...   |
                |__________________________|
```

Syntax Rules

1. In Format 1, record-name must be the name of the current record of the run-unit.

2. In Format 2, identifier-1 ... must reference elementary items defined in the record entry for the current record of the run-unit.

General Rules

1. Execution of a GET statement places all or part of the current record of the run-unit in the associated UWA record.

2. In Format 1, the entire contents of the current record are moved to the UWA.

3. In Format 2, only those data items listed will be moved to the UWA record associated with the current record of the run-unit. All other UWA fields will be unchanged.

4. No currency indicator is affected by the GET statement.

DB-STATUS Values

FUNCTION CATEGORY	ERROR CONDITION	DB-REGISTERS			
		DB-STATUS	REALM	RECORD	SET
	None	0000000	-	-	-
	Data base not open	0809100			
	CRU null	0803200			
	CRU not of correct type (when specified)	0803300		X (cru)	
	CRU UWA record-description missing	0873660		X	

5.10.8 DATA BASE CONDITION (IF)

Function

Enables the program to make decisions depending upon the value of a test involving a current record and/or its set memberships. There are two data base conditions:

1. Tenancy condition: status of a record with respect to one or several set occurrences.

2. Membership condition: status of a set occurrence.

NOTE: The "Data-Base Condition" will be referenced, in the rest of the document and statistics/trace/run-time options, as the IF DML function.

General Format

```
                            {OWNER }
Tenancy :      [set-name] {MEMBER}
                            {TENANT}


Membership :    set-name IS [NOT] EMPTY
```

Syntax Rules

The Data Base Condition is a simple condition from a COBOL point of view; as such, it may appear anywhere a simple condition may be used.

The Data Base Condition has a truth value of 'true' or 'false'. This condition cannot be tested when a data-base-exception occurs.

General Rules for the Tenancy Condition

1. The tenancy condition indicates whether or not the current record of the run-unit is presently:

   - the owner of a non-empty occurrence of a specified set-type (or of at least one set-type)

   - or an actual member of a specified set-type (or of at least one set-type)

   - or both.

2. Set-name OWNER is true when the "current-of-run-unit" is the owner of a non-empty occurrence of the set-type specified by "set-name".
   The record-type of the "current-of-run-unit" must be declared in the schema as the owner type of this set-type, otherwise a data-base-exception occurs.

3. OWNER is true when the "current-of-run-unit" is the owner of a non-empty occurrence of at least one set-type. The record-type of the "current-of-run-unit" must appear in the schema as the owner type of at least one set-type, otherwise a data-base-exception occurs.

4. Set-name MEMBER is true when the "current-of-run-unit" is an actual member of an occurrence of the set-type specified by "set-name".
   The record-type of the "current-of-run-unit" must be declared in the schema as a MANUAL OPTIONAL member of this set-type, otherwise a data-base-exception occurs.

5. MEMBER is true when the "current-of-run-unit" is an actual member of an occurrence of at least one set-type. The set-types considered are those for which the record-type of the "current-of-run-unit" is declared in the schema as a MANUAL OPTIONAL member. If no such set-type exists, a data-base-exception occurs.

6. Set-name TENANT is equivalent to:

   - "Set-name OWNER" if the record-type of the "current-of-run-unit" is declared in the schema as owner-type of this set-type.

   - "Set-name MEMBER" if the record-type of the "current-of-run-unit" is declared in the schema as a MANUAL OPTIONAL member-type of this set-type.

     If the record-type of the "current-of-run-unit" is not declared as the owner-type or a MANUAL OPTIONAL member-type of this set-type, a data-base-exception occurs.

7. TENANT is the logical "OR" of "OWNER" and "MEMBER". There must be in the schema set-types for which the record-type of the "current-of-run-unit" is the owner type and/or set-types for which the record-type of the "current-of-run-unit" is a MANUAL OPTIONAL member-type. Otherwise a data-base-exception occurs.

NOTE: The tenancy condition cannot be used to interrogate the "type" data structure described in the schema. It only deals with "occurrences" of the elements of the data structure. For instance the purpose of "IF OWNER" is not to test if the record-type of the "current-of-run-unit" appears in the schema as the owner-type of some set-types; its purpose is to test if the "current-of-run-unit" is currently the owner of at least one non-empty set occurrence.

General Rules for the Membership Condition

1. The membership condition indicates whether or not the occurrence of the specified set-type identified by its current of set-type currently has no members.

2. The current of set-type identifying the set occurrence must not be null or virtual (as a result of an ERASE statement), otherwise a data base exception occurs.

3. Set-name IS EMPTY is true if the set occurrence has no members.

4. When NOT is specified, the result of the test is reversed.

DB-STATUS Values

FUNCTION CATEGORY	ERROR CONDITION	DB-REGISTERS			
		DB-STATUS	REALM	RECORD	SET
	None condition true	0000000	-	-	-
	None condition false	0000000	-	-	-
	Data base not open	0909100			
Tenancy condition	CRU null	0903200			
	CRU type is irrelevant to the condition tested (not owner or OPTIONAL member of the sets)	0903300		X (cru)	
Membership condition	Current-of-set null	0903100			X

5.10.9 MODIFY

Function

Alters the contents of one or more data items in a record and/or changes the set membership of the record. The object of the statement is the current record of the run-unit.

General Format

```
Format 1

       {[record-name]    }
MODIFY {                 } [RETAINING-phrase]
       {{identifier-1}...}


Format 2

                           {ALL             }
MODIFY [record-name] ONLY  {                } MEMBERSHIP
                           { {set-name-1}...}
                          [RETAINING-phrase]


Format 3

       {[record-name]    }            {ALL             }
MODIFY {                 } INCLUDING  {                } MEMBERSHIP
       {{identifier-1}...}            {{set-name-1}...}
                                      [RETAINING-phrase]

RETAINING-phrase Format

                         {MULTIPLE              }
                         { REALM                }
RETAINING CURRENCY FOR   { RECORD               }
                         { { SETS           }   }
                         { {  { set-name }...}  }
```

Syntax Rules

1. Record name, if specified, must be the name of the current record of the run-unit.

2. Identifier-1, ... must reference data items contained within the current record of the run-unit.

3. The current record of the run-unit must be defined in the schema DDL as a member of the set-types referenced by set-name, ..., if any.

General Rules

All Formats

1. If the RETAINING phrase is not specified, the object record becomes, on successful completion, the current record of its realm, the current record of its record-type, and the current record of all set-types in which it is a tenant.

2. If the RETAINING phrase with the optional word REALM is specified, the realm currency indicator is not changed.

3. If the RETAINING phrase with the optional word RECORD is specified, the record-type currency indicator is not changed.

4. If the RETAINING phrase with the optional word SETS is specified, no set-type currency indicators are changed.

5. If the RETAINING phrase with the optional set-name, ... is specified, the set currency indicators for the named set-types are not changed.

6. If the RETAINING phrase with the optional word MULTIPLE is specified then the realm, record-type and set-type currency indicators are not changed.

7. The realm in which the current record of the run-unit is stored must be open in update. Records of sets in which membership is affected by the execution of the MODIFY statement must be stored in realms that are open in update.

8. Records of sets referenced by the set selection criteria must be stored in realms that are in ready state.

Format 1

1. This format replaces the contents of one or more items in the current record of the run-unit.

2. If record-name is omitted, the DBCS replaces the contents of all the items of the current record of the run-unit with the contents of the corresponding items in the UWA.

3. If record-name is specified, the DBCS checks that the current record of the run-unit is of this record-type and performs the action described in rule 2.

4. If identifier-1, ... is specified, the DBCS replaces the contents of the data items specified with the contents of the corresponding items in the UWA.

5. If the data items modified do not satisfy the validity checks defined in the schema DDL for this record-type, the modification is denied and a data-base-exception occurs.

6. If CALC-key items are modified and the MIGRATION IS ALLOWED clause is not specified in the DMCL for this record-type, the record keeps its data-base-key and is logically inserted in the CALC chain corresponding the new CALC-key value.

   If an overflow-link record (OWL) corresponding to the new CALC chain has to be created in the page of the record and if there is no room left to do so, the modification is denied and a data-base-exception occurs.

7. If CALC-key items are modified and the MIGRATION IS ALLOWED clause is specified, the record is moved to the bucket corresponding to the new CALC-key value. The record cannot change areas as a result of a migration.

   Set pointers and currency indicators referencing the moved record are updated to reflect the new position.

8. If modified items are sort-key items in a set (including the DATA-BASE-KEY in case of migration), the set is reordered according to its set ordering criteria.

9. If modified items do not satisfy the no-duplicate controls (CALC-KEY, sort-key, no-duplicate-control-fields), the modification is denied and a data-base-exception occurs.

Format 2

1. Format 2 changes the set membership of the current record of the run-unit for specified set-types.

2. If record-name is specified, the DBCS checks that the current record of the run-unit is of this record-type.

3. If ALL is specified, the affected set-types are those in which the record is an AUTOMATIC MANDATORY member or a MANUAL OPTIONAL member currently connected.

4. If set-name, ... is specified, the affected set-types are those of the list. If the record is defined as a MANUAL OPTIONAL member of one of these set-types, it must be currently connected to this set.

5. For each affected set-type, the set membership is changed as follows:

   - The current record of the run-unit is disconnected from the set occurrence to which it belongs.

   - A new occurrence is selected according to the set selection criteria specified in the schema DDL.

   - The current record of the run-unit is connected to the occurrence selected above. This is done according to the set ordering criteria specified in the schema DDL.

6. If the current record of the run-unit has a LOCATION mode defined as "VIA set-name WITHIN AREA OF OWNER" a modification of the membership in this set will be denied if the new owner in not in the same area as the current record of the run-unit.

7. If data items in the current record of the run-unit do not satisfy the no-duplicate controls (sort-key, no-duplicate-control-fields) in the new set occurrences, the modification is denied and a data-base-exception occurs.

Format 3

1. Format 3 replaces the contents of one or more items in the current record of the run-unit and changes its set membership in one or more sets.

2. The actions performed by the DBCS are the combination of those described for format 1 and format 2.

NOTES:

1. MANUAL OPTIONAL set membership can be changed with a MODIFY statement. If the current record of the run-unit is currently connected to the set, it is treated exactly as an AUTOMATIC MANDATORY set. It is not possible to obtain the effect of a CONNECT with a MODIFY statement. If the record is connected to a set before the MODIFY statement, it will be connected afterwards. If the record is disconnected from a set before the MODIFY statement, it will be disconnected afterwards.

2. If one of several control-fields is to be changed, the one specified will be obtained from the UWA. The others required to execute the function will be obtained from the current record of run-unit on the data base. This means it is possible to change any combination of data items in a record.

3. a) If a data-base-exception occurs during the execution of a MODIFY statement without migration, the data base is restored to the state that it was in before the statement was executed (without using the journals). The currency indicators are not affected and the control returns to the calling program with the appropriate DB-STATUS value.

   b) If a data-base-exception occurs during the execution of a MODIFY statement with migration, but before the updating of the data base has actually begun, then the DBCS proceeds as described above.

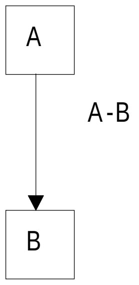
   c) If a data-base-exception occurs during the execution of a MODIFY statement with migration, but after the updating of the data base has actually begun, then the processing depends on the protection level of the data base:

   - If the data base is protected by the BEFORE journal, in BATCH, IOF or TDS environment, the data base is restored to the state that it was in before the statement was executed and control returns to the calling program with the appropriate DB-STATUS value.

   - If the data base is not protected by the BEFORE journal, in a BATCH or IOF environment, the data base is left in an inconsistent state and the step is aborted. The data base must be restored by FILREST or VOLREST.

   - If the data base is protected by the AFTER journal and deferred updates in TDS environment, the data base is restored to the state that is was in at the last commitment point, then the TPR is aborted and not restarted.

Examples



```
RECORD A ...
  LOCATION MODE CALC USING A-1 DUP NOT ...
RECORD B ...
  LOCATION MODE CALC USING B-1 DUP NOT ...
SET A-B ...
  ORDER ... SORTED BY DEFINED KEYS ...
...
   KEY IS ASCENDING B-2
   SET SELECTION IS THRU A-B
    OWNER IDENTIFIED BY CALC-KEY
```

1. Modify CALC-key field in record A.

```
MOVE key-1 TO A-1.
FIND ANY A.
MOVE new-data TO A-1.
MODIFY A-1.
```

2. Modify sort-key in record B without changing occurrences.

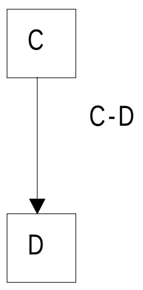
```
MOVE key-2 TO B-1.
FIND ANY B.
MOVE new-sort-key TO B-2.
MODIFY B-2.
```

3. Put a B record under a new A record without changing any fields in the B record.

```
MOVE key-1 TO B-1.
FIND ANY B.
MOVE new-owner-key TO A-1. (selection path for new occurrence)
MODIFY B ONLY A-B.
```

4. Put a B record under a new A record and change the sort-key in the B record.

```
MOVE key-1 TO B-1.
FIND ANY B.
MOVE new-owner-key TO A-1. (selection path for new occurrence)
MOVE new-sort-key TO B-2.
MODIFY B-2 INCLUDING A-B.
```



```
RECORD C ...
  LOCATION MODE CALC USING C-1 DUP NOT ...
RECORD D ...
  LOCATION MODE CALC USING D-1 DUP NOT ...
SET C-D ...
  SET SELECTION IS THRU C-D
     OWNER IDENTIFIED BY
     APPLICATION
```

5. Put a D record under a new C record.

```
MOVE new-owner-key TO C-1. (set selection path for new occurrence)
FIND ANY C.
MOVE key-1 TO D-1.
FIND ANY D RETAINING CURRENCY C-D.
MODIFY D ONLY C-D.
DB-STATUS Values
```

FUNCTION CATEGORY	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
General	None modification	0000000	-	-	-
	Data base not open	1109100			
	CRU null	1103200			
	CRU not of correct type (when specified) or cannot be member of the set specified	1103300		X (cru)	
	Area not ready	1109100	X	X	(X)
	Area not in update	1109200	X	X	(X
	Duplicate data items not allowed (CALC-key or sort-key or other control-field)	1105100	(X)	X	(X)
Data modification	CRU UWA record-description missing	1173660		X	
	Data items do not satisfy the validity checks	1150200		X	
	CALC-key modification cannot be performed without migration	1173640		X	
	Area space exhausted (migration)	1180200	X	X	
Membership modification	Attempt to modify the membership of an OPTIONAL member not currently connected	1108300		X	X
	Set selection failed	1102300		X	X
	Owner of new set occurrence does not respect the AREA OF OWNER condition	1173650	X (owner)	X	X

5.10.10 READY

Function

Prepares one or more realms for processing.

General Format

```
      {                               {EXCLUSIVE     {RETRIEVAL} }}
      {                               {              {UPDATE   } }}
READY { [realm-name-1]...USAGE-MODE IS {SHARED         RETRIEVAL' }}...
      {                               {              {RETRIEVAL} }}
      {                               {[MONITORED]   {UPDATE   } }}
```

General Rules

1. Execution of any READY statement affects entire realms, regardless of the specific format of the statement.

2. At the beginning of the execution of a READY statement, no realm affected by that READY statement may be in ready state.

3. If realm-name-1,... is specified, each realm referenced is prepared for access and placed in ready state. Realm-name-1 may appear only once in a specific READY statement.

4. If realm-name-1,... is not specified, all realms in the data base are prepared for access and placed in ready state.

5. The USAGE-MODE phrase establishes a usage mode for the affected realms and thereby specifies the type of operations permitted. Usage modes remain in effect for the affected realms until a FINISH statement for the affected realms is executed, or until the run-unit that executed the READY statement terminates.

6. The RETRIEVAL phrase permits access to inspect and read the contents of the affected realms.

7. The UPDATE permits both access to, and modification of, the contents of the affected realms.

8. The EXCLUSIVE phrase specifies that the affected realms cannot be affected by concurrent run-units.

9. The SHARED phrase specifies that the affected realms can be read by concurrent run-units using the same usage-mode. The sharing is at the file level in RETRIEVAL mode only.

10. The MONITORED phrase specifies that the affected realms can be read or modified by concurrent run-units using the MONITORED (use of GAC) usage mode. Sharing is at the page level.

11. The JCL ASSIGN/DEFINE statements must be consistent with the USAGE-MODE defined in the program. Overriding rules allow the user to specify in JCL a sharing mode more restrictive than that specified in the program. For further details see Section VII.

12. When USAGE-MODE IS EXCLUSIVE UPDATE and the BEFORE journal is not used for protection, the realm is placed in the transient state.
    It remains in this state until the execution of a successful FINISH statement.

    If the job step fails before the FINISH execution then the realm will be left in the transient state. The next time the realm is referenced by a READY statement the job step will fail unless the user has asked that this state be ignored (See run-time commands in Section VII).

    If the transient state is ignored, the DBCS records in the IDS labels the time and date of the last time this is done.

    The normal course of action when a job step fails, leaving a realm in the transient state, is to use a restore utility such as FILREST to restore the realm to its original condition.

    The transient state flag can also be reset using the PATCH LABEL command of DBUTILITY.

13. READY statements coded in a TPR are ignored at run-time.

DB-STATUS Values

FUNCTION CATEGORY	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
	None	0000000	-	-	-
	Area already in ready state	1309300	X		

5.10.11 STORE

Function

Causes a record to be stored in the data-base. This statement establishes the current record of the run-unit.

General Format

```
                 [                         {   MULTIPLE          } ]
                 [                         {    REALM            } ]
STORE record-name [ RETAINING CURRENCY FOR {   {SETS          }  } ]
                 [                         {   { {set-name}...}  } ]
                 [                         {    RECORD           } ]
```

General Rules

1. The fields of the UWA record referenced by record-name are moved to the record in the data base. Schema-defined validity checks are carried out.

2. The record referenced by record-name becomes a member of each set-type for which it is an AUTOMATIC member. The sets to which the record is connected are determined by the set selection criteria. The record is connected in accordance with the set ordering criteria of each set-type.

3. The referenced record is established as the owner of an empty set for each set-type in which it is an owner.

4. The STORE statement causes the referenced record to become the current record of the run-unit. The realm-name and record-name of the record are placed in the special registers DB-REALM-NAME and DB-RECORD-NAME respectively.

5. If the RETAINING phrase is not specified, the referenced record becomes the current record of its realm, the current record of its record-type, and the current record of all set-types in which it has been declared to be an owner or AUTOMATIC member.

6. If the RETAINING phrase with the optional word REALM is specified, the realm currency indicator is not changed.

7. If the RETAINING phrase with the optional word RECORD is specified, the record-type currency indicator is not changed.

8. If the RETAINING phrase with the optional word SETS is specified, no set-type currency indicators are changed.

9. If the RETAINING phrase and set-name, ... are specified, the currency indicators of set-name, ... are not changed.

10. If the RETAINING phrase with the optional word MULTIPLE is specified then the realm, record-type and set-type currencies are not changed.

11. The realm in which the record is stored must be open in update. Records of sets of which the stored record is an AUTOMATIC member must be stored in realms that are open in update.

12. Records of sets referenced by the set selection criteria must be stored in realms that are in ready state.

13. The record referenced by record-name remains available to the program in the UWA.

Record Placement

The DBCS assigns a unique data-base-key to the record placed in the database as a result of the execution of the STORE statement. The schema definition and COBOL programmer can specify certain constraints on the assignment of a data-base-key. These constraints depend on the location mode specified for the record and are listed below for the three possible location modes.

DIRECT

If the record has a DIRECT location mode, the contents of the specified data item are used to assign a data-base-key to the new record. The dataitem must contain an area-key or a data-base-key. In the latter case, thearea code is not involved in the area selection and is ignored.

1. A realm is selected, if there is a choice, by means of the schema-specified area identification parameter.

2. If the supplied area-key is within the record range and is not assigned to an existing record, the DBCS will assign it to the new record.

3. If the supplied area-key is within the record range and is already assigned to an existing record, the DBCS searches forward to find an available data-base-key. The first one found is assigned to the new record. If the end of the range is encountered, the search is resumed at the beginning of the range.

4. If the supplied data-base-key is not within the record range, the DBCS returns an exception status.

NOTE: The DIRECT location mode should be avoided as much as possible since it does not lend itself to automatic reorganization by a standard utility.

CALC

If the record has a CALC location mode, the contents of the specified UWA data items are used to assign a data-base-key.

1. A unique realm is selected, if there is a choice, by means of the schema-specified area identification parameter.

2. The location mode data items are used to determine the CALC "bucket" where the record is to be stored. The record is placed in the first page of the bucket or in a subsequent page.

VIA SET

If the record has a VIA SET location mode, the DBCS assigns a data-base-key based on the data-base-key of the owner record occurrence of the set-type specified for the location mode. During the process of assigning a data-base-key and physical location for a VIA SET record, the procedure below is followed:

1. The set selection criteria for the specified set are used to locate the owner record occurrence for the set. This may involve many levels of paths to be traversed before the required owner is found.

2. A realm is selected, if there is a choice, by means of the schema-specified area identification parameter.

3. If the owner record-type of the specified set and the new record-type are in different ranges of the same area, or are in different areas, the new record is placed in its range proportional to the owner occurrence in its range. If both records are in the same range, the new record is placed as near as possible to the owner occurrence

   If the set selection path consists of only one level identified by APPLICATION and if the set order is NEXT or PRIOR, the search for a free data-base-key starts from the position of the current-of-set and not from the position defined above. This optimizes processing time when the set occurrence represents a volatile sequential file spreading over many pages.

NOTES:

1. Since the record stored is not connected to any MANUAL sets, the currency indicators will not be updated for any MANUAL sets. This is true regardless of the presence or absence of any RETAINING phrase.

2. If the location mode is VIA SET and the record to be stored is a MANUAL member of the specified set, the new record is physically placed as if it was going to become a member of the set. Set selection is executed for the set to locate the correct owner occurrence, but the new record is not connected to the set.

3. When a realm is selected for a record, the schema-specified parameter is used. The program executing the STORE statement must put the name of the correct realm in the parameter before executing the STORE statement.

4. Set selection rules for the AUTOMATIC sets and for the set specified in the VIA phrase may require a realm to be selected to locate the entry point record. This occurs when the entry point record is a CALC record that is present in more than one realm. The COBOL program must move the correct realm-name to the AREA-ID parameter before executing the STORE statement.

5. Since there is no guarantee that a DIRECT record was assigned the data-base-key supplied by the program, it is advisable to check this if it is important. The data-base-key actually assigned can be obtained with an ACCEPT statement. This can then be compared to the data-base-key supplied. The data item containing the data-base-key is not altered by the STORE statement, even if the record stored is not assigned the data-base-key supplied.

6. If a data-base-exception occurs during the execution of a STORE statement, the data base is restored to the state it was in before the statement was executed (without using the journals). In addition, currency indicators are not affected by an unsuccessful statement.

DB-STATUS Values

FUNCTION CATEGORY	**ERROR CONDITION**	**DB-REGISTERS**			
		DB-STATUS	**REALM**	**RECORD**	**SET**
	None record stored	0000000	X	X	-
	Data base not open	1509100			
	Data items do not satisfy the validity checks	1505200		X	
	Area-name supplied in AREA-ID is unknown or does not correspond to record-type	1504300		X	
	Area not ready	1509100	X	X	(X)
	Area not in update mode	1509200	X	X	(X)
	Area-key is inconsistent	1504100	X	X	
	Area space exhausted	1580200	(X)	X	
	Duplicate item not allowed (CALC-key or sort-key or other field)	1505100	(X)	X	(X)
	Set selection failed	1502300		X	X
	UWA record-description missing	1573660		X	

5.10.12 USE

Function

Specifies procedures to be executed when the execution of a DML statement results in a data-base-exception condition.

General Format

```
 ________________________
|                        |
| USE FOR DB-EXCEPTION.  |
|________________________|
```

Syntax Rules

1. A USE statement, when present, must immediately follow a section header in the DECLARATIVES part of the PROCEDURE DIVISION and must be followed by a period. The remainder of the section must consist of one or more procedural paragraphs that define the procedure to be used.

2. Only one USE FOR DB-EXCEPTION statement may appear in the DECLARATIVES part.

3. Within a USE procedure, there must be no reference to any non-declarative procedure.

General Rules

1. Appropriate values are placed in the special registers DB-STATUS, DB-REALM-NAME, DB-RECORD-NAME, DB-SET-NAME and DB-DETAILED-STATUS before the execution of a data base exception section.

2. The data-base-exception section is executed whenever the execution of a statement results in a data base exception condition. At the end of the execution of a data base exception section, control is returned to the statement following the statement that caused the exception, unless a STOP RUN is executed in the DECLARATIVES section.

Examples

1. In the absence of any USE FOR DB-EXCEPTION statement, the program must test the contents of DB-STATUS for a non-zero value after each DML function. Successfully completed functions return a zero status code, while unsuccessful functions return a value according to the action taken and the exception experienced. A typical sequence of statements would be:

```
     MOVE "value" TO CALC-CONTROL-ITEM.
     FIND ANY CALC-REC RECORD.
     IF DB-STATUS IS EQUAL TO "0502400" GO TO MISSING-RECORD
ELSE IF DB-STATUS IS NOT EQUAL TO ZERO GO TO OTHER-ERROR.
```

2. A single USE FOR DB-EXCEPTION statement can be used to process all exception conditions. This procedure must distinguish between normal exceptions, such as "End of Set" or "No Record Found To Satisfy Record Selection", and those exceptions that are errors. The procedure has available all the data base special registers, many of which are updated for error conditions, and all data items in the DATA DIVISION of the program. Status flags can be set depending on the type of exception and the identity of records, sets, or realms involved. For example:

```
77    STATUS-FLAG     PIC 9       VALUE 0.
      88 END-OF-PATH              VALUE 1.
      88 MISSING-REC              VALUE 2.
      88 BAD-ERROR                VALUE 3.
PROCEDURE DIVISION.
DECLARATIVES.
ERROR-PROCESSING SECTION.
      USE FOR DB-EXCEPTION.
ERROR-PROCESSING-PARAGRAPH.
      IF DB-STATUS IS EQUAL TO "0502100" GO TO SET
      END-FLAG.
      IF DB-STATUS IS EQUAL TO "0502400" GO TO SET
      MISSING-REC.
      MOVE 3 TO STATUS-FLAG.
      GO TO EXIT-EXIT.
SET-END-FLAG.
      MOVE 1 TO STATUS-FLAG.
      GO TO EXIT-EXIT.
SET-MISSING-REC.
      MOVE 2 TO STATUS-FLAG.
      GO TO EXIT-EXIT.
      .
EXIT-EXIT. EXIT.
END DECLARATIVES.
      .
      FIND NEXT WANTED-RECORD WITHIN THIS-SET.
      IF END-OF-PATH GO TO DONE-WITH-IT.
      .
```

In this example, each data-base-exception causes the USE FOR DB-EXCEPTION section to be invoked. This procedure only checks the status code for certain values and sets condition indicators for reference by the program. It is the program's responsibility to subsequently test for the acceptable, or error, conditions. The USE procedure itself could have detected an error status and caused the program to abort or terminate abnormally.

5.11 SYSTEM FUNCTIONS DEFINING CHECK-POINTS AND COMMITMENT-POINTS

These functions involve the entire step or transaction and not only the data base. Though they are provided by the system rather than by the DBCS, they are mentioned here because of their close connection with the USAGE-MODE specified in READY statements.

5.11.1 Comparison of Check-Points and Commitment-Points

- The concepts of check-point and commitment-point are similar in the sense that they define a run-unit and data base state to which it is possible to return if the next check-point or commitment-unit is not reached succesfully.

- The concepts differ as far as data base sharing is concerned.

  The check-point applies to the case where the data base is reserved for the run-unit from the start (USAGE-MODE EXCLUSIVE) and is not made available to other run-units at each check-point but only at the end of the run-unit. The incidents that may prevent the run-unit from reaching the next check-point are not resource conflicts but only program or system failures. In a well-tested environment these incidents are very unlikely.

  The commitment-point applies to the case where the data base is shared between run-units (USAGE-MODE MONITORED). Data base pages are reserved during the execution of the commitment-unit and released at the next commitment-point. As opposed to program or system failures, conflicts for data base pages are likely and may cause either a temporary wait or a deadlock (for example, deadlocks may be caused by programs in RETRIEVAL mode). Deadlocks are resolved by returning to the last commitment-point and releasing the pages already reserved.

  The interval between two check-points may be of the order of an hour, while the interval between two commitment-points may be of the order of a few seconds.

- The commitment-unit was first introduced in TDS. It has been extended to BATCH or IOF programs which are to access the data base concurrently with each other or with TDS.

  These programs must be split into commitment-units by regular calls to a system function (H_GAC_UCOMIT) which takes a snapshot of the program state and releases the data base pages previously held in exclusive or shared mode.

  If TDS is running concurrently, the frequency of commitment-points in BATCH programs must be high enough to not disturb the response time of the TDS transactions. If only BATCH programs are concurrent, the frequency may be lower but the risk of a wait or deadlock increases with the number of pages locked by a commitment-unit.

  The logic of the commitment-unit itself is not affected by the fact that the usage mode is MONITORED or EXCLUSIVE since it reserves all the data pages required to ensure the consistency of the actions taken during the commitment-unit.

The logic of the BATCH program as a whole, on the contrary, must take into account the fact that a commitment-unit cannot rely on the data base modifications of the previous commitment-unit, since concurrent run-units may have made subsequent modifications in the meantime.

- BATCH programs written in view of concurrent access may sometimes be run in EXCLUSIVE mode. ASSIGN/DEFINE statements allow the user to override the READY options and to suppress the concurrent access at the page level. In this case the system function H_GAC_UCOMIT becomes automatically ineffective, unless a parameter indicates that a check-point is to be taken.

  The latter facility replaces the call to the system function H_CK_UCHKPT. It may be used with every n'th commitment-unit so as to obtain the same frequency as if H_CK_UCHKPT had been coded.

  Program calls to H_CK_UCHKPT are ineffective if they occur when the step is accessing the data base in MONITORED mode.

5.11.2 System Function Defining a Check-Point

A program running in BATCH/IOF environment invokes this function by using:

```
 ___________________________________________
|                                           |
|  CALL "H_CK_UCHKPT" USING CKMODE CKINF    |
|___________________________________________|
```

The parameters (which must be defined by the user) are:

```
01 CKMODE USAGE COMP-2. (output)
01 CKINF  PIC X(32).    (output)
```

For further details, see the System Management Guide.

In TDS there is no check-point.

5.11.3 System Function Declaring a Commitment-Point

A program running in BATCH/IOF environment calls this function by using:

```
 ________________________________________________________
|                                                        |
| CALL "H_GAC_UCOMIT" USING CKMODE CKINF NUMLOCK NOCHKPT|
|________________________________________________________|
```

The parameters are:

```
01 CKMODE USAGE COMP-2. (output)
01 CKINF  PIC X (32).   (output)
01 NUMLOCK COMP-1.      (input)
01 NOCHKPT PIC X.       (input)
```

CKMODE and CKINF have the same meaning and values as for H_CK_UCHKPT.

NUMLOCK defines the maximum number of entries in the lock tableassociated with the starting commitment-unit.
NUMLOCK = -1 requests the system to choose the default value (50).

NOCHKPT specifies whether a check-point is to be taken in the case where no file is in MONITORED mode.
NOCHKPT = " ": the check-point is taken.
NOCHKPT = "Y": no check-point is taken.

In TDS environment, the corresponding function is invoked (when it is not implicit) by the following calling sequence:

```
 ___________________
|                   |
| CALL "H_DFCMIT"   |
|___________________|
```

For further details, see the TDS COBOL Programmer's Guide.

5.12 ACCESS TO THE IDSTRACE FILE

The programmer may call the DBCS function H_IA_UCTRACE to perform the following operations on the IDSTRACE file:

- Write program information onto the IDSTRACE file.
- Open, close the IDSTRACE file.
- Disable, enable the program and/or DBCS write operations onto the IDSTRACE file.

1. The H_IA_UCTRACE write function allows the user to interleave program and DBCS trace data on the same file.

2. The IDSTRACE file is opened by the DBCS at the beginning of a session and closed by the DBCS at the end of the session.

   The H_IA_UCTRACE open, close functions allow the user to open the IDSTRACE file before the first READY of an IDS session and to close it after the last FINISH. In this case the IDSTRACE open and close requests submitted by the DBCS are ignored.

   Figure 5-1 shows a DML program responsible for opening and closing the IDSTRACE file and writing program information interleaved with DBCS trace data.

   A run-unit accessing 2 data bases can collect program and DBCS trace information concerning both data bases on the same IDSTRACE file. The trace-ifn must be the same for the program and the 2 data bases. The IDSTRACE file is opened by the first open request (submitted by the program or one of the 2 IDS sessions) and closed by the last close request.

   Figure 5-2 illustrates a COBOL program PO driving 2 DML program P1, P2, each accessing a different data base. PO is responsible for opening and closing the IDSTRACE file. PO, P1, P2 write program information interleaved with DBCS trace data.

3. The H_IA_UCTRACE disable, enable functions allow the programmer to focus the trace action on a given portion of the program and to apply selection criteria more specific than those available in the TRACE command of the run-unit command language.

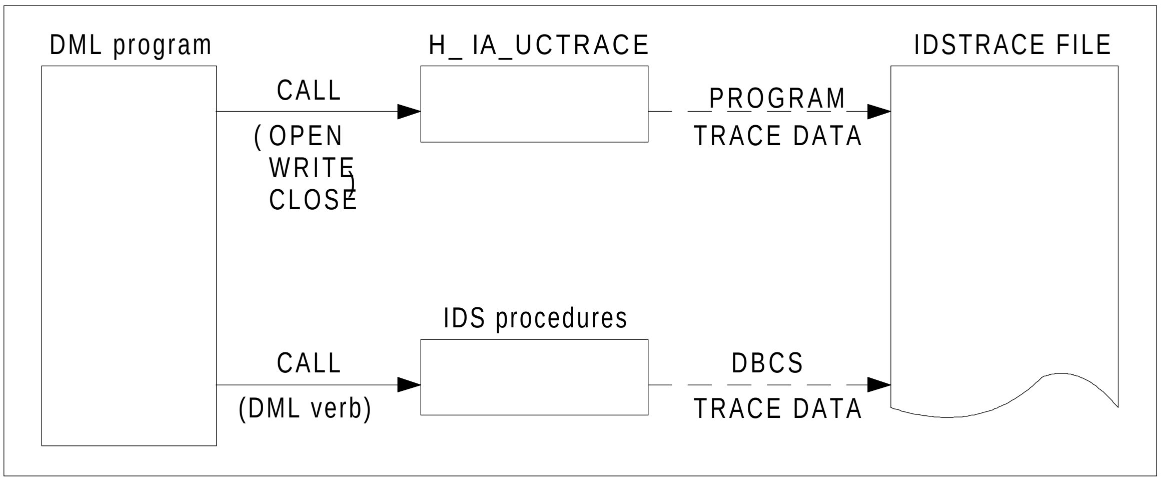


Figure 5-1. User Access to IDSTRACE File (one program)

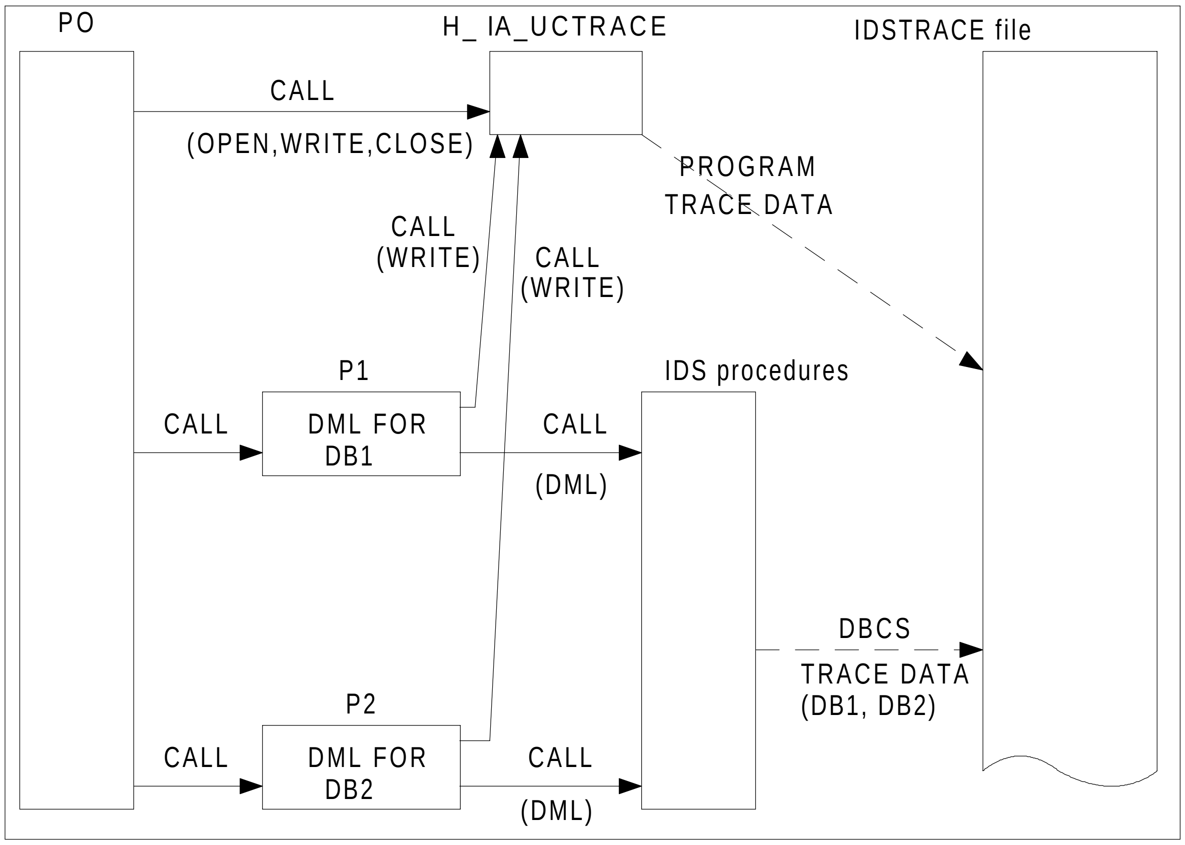


Figure 5-2. User Access to IDSTRACE File (several programs)

Format of Call

```
| CALL "H_IA_UCTRACE" USING TRACEPARAM TRACEIFN  |
|________________________________________________|
```

Parameters

```
01   TRACEPARAM.

   02  TRACEFUNCTION COMP-1. (input)
   02  TRACERETCODE  COMP-1. (output)
   02  TRACELENGTH   COMP-1. (input)
   02  TRACEDATA PIC X(n).   (input)

01   TRACEIFN PIC X(8).      (input)
```

Parameter Descriptions

TRACEFUNCTION: function code.

Values are shown below in decimal.

VALUE	FUNCTION
1	Disable progrm write operations onto IDSTRACE file
2	Disable DBCS write operations onto IDSTRACE file
3	Disable program and DBCS write operations onto IDSTRACE file
4	Enable program write operations onto IDSTRACE file
5	Enable DBCS write operations onto IDSTRACE file
6	Enable program and DBCS write operations onto IDSTRACE file
16	Write onto IDSTRACE file
17	Open IDSTRACE file in OUTPUT mode
18	Open IDSTRACE file in EXTEND mode
19	Close IDSTRACE file

TRACERETCODE: return code.

VALUE	MEANING
0	The function has been successfully executed or rendered dummy by a previous "disable" function (write)
1	The function has not been executed because of incorrect parameters or abnormal conditions

TRACELENGTH: number of characters (n) of TRACEDATA.

TRACEDATA: program data to be written to the IDSTRACE file

NOTE: TRACELENGTH and TRACEDATA are only required when TRACEFUNCTION = 16. They are meaningless otherwise.

TRACEIFN: internal-file-name of the IDSTRACE file.

Rules

1. The disable, enable, write and close functions are not executed if the IDSTRACE file is not currently opened.

2. The open function is not executed if the IDSTRACE file is not currently closed.

3. When the IDSTRACE file is opened, write operations are implicitly enabled.

4. In TDS environment, only the write function is available. Other functions are not executed.

5.13 SIMULATION OF THE CALC HASHING ALGORITHM

This function determines the relative page number within a range where a CALC record will randomize, assuming no page overflow.

Format of Call

```
 ____________________________________________________
|                                                    |
| CALL "H_IA_UCHASH" USING HASHFUNC HASHRANGEDESC    |
|                          HASHKEYDESC HASHDATA      |
|____________________________________________________|
```

Parameters

```
01      HASHFUNC.

   02     HASHRETCODE COMP-1.              (output)
   02     HASHPAGENUMBER COMP-2.           (output)

01      HASHRANGEDESC.                     (input)

   02     HASHNUMPAGE COMP-2.
   02     HASHCALCINTERVAL COMP-1.
   02     HASHINTERVALUNIT COMP-1          VALUE O.

01      HASHKEYDESC.                       (input)

   02     HASHNUMITEM COMP-1 VALUE n.
   02     HASHITEMDESC OCCURS n.

     03     HASHITEMTYPE COMP-1.
     03     HASHITEMLENGTH COMP-2.
     03     HASHITEMSCALE COMP-1.
     03     HASHITEMPTR COMP-2.

01      HASHDATA.                          (input)

        Key items within record
```

Parameter Descriptions

Numeric values are given in decimal.

HASHRETCODE:	return code

VALUE	MEANING
0	Simulation done
1	Incorrect parameter, which may be:
	- HASHNUMPAGE = 0
	- HASHCALCINTERVAL = 0 or > 255
	- HASHINTERVALUNIT < > 0
	- HASHNUMPAGE not multiple of HASHCALCINRERVAL
	- HASHNUMITEM = 0 or > 256
	- total key length > 256
	- invalid HASHITEMTYPE or HASHITEMLENGTH
	- HASHITEMPTR = 0
2	Illegal decimal data in a CALC-key item

HASHPAGENUMBER:	relative page number within the range resulting from the randomization (0 <= HASHPAGENUMBER <= HASHNUMPAGE - 1). This is calculated as follows: the number of pages of the range HASHNUMPAGE IS DIVIDED BY HASHCALCINTERVAL giving the number of buckets as quotient. The CALC-key items described by HASHKEYDESC are retrieved from HASHDATA, concatenated and hashed by the HASH instruction, giving a 32-bit binary value. The latter is extended to 64 bits with zero fill and divided by the number of buckets. The remainder is multiplied by HASHCALCINTERVAL giving HASHPAGENUMBER.
HASHNUMPAGE:	number of pages of the range. It must be a multiple of HASHCALCINTERVAL.
HASHCALCINTERVAL:	number of pages of the CALC INTERVAL.
HASHINTERVALUNIT:	must be zero
HASHNUMITEM:	number of CALC-key items
HASHITEMDESC:	repeating group occuring HASHNUMITEM times and describing each key item.
HASHITEMTYPE, HASHITEMLENGTH:	type and length of the key item.

Data type	**HASHITEMTYPE**	**HASHITEMLENGTH**
UNSIGNED UNPACKED DECIMAL UNSIGNED PACKED DECIMAL UNSIGNED PACKED-2 DECIMAL SIGNED UNPACKED DECIMAL SIGNED PACKED DECIMAL	9 57 25 1 17	Number of decimal digits (sign excluded) 1
SIGNED BINARY	3	Number of binary digits (sign excluded) 15 or 31
CHARACTER	0	Number of characters 1

HASHITEMSCALE:	scale of the key item. 0 for a binary or character item ±p for a decimal item. The scale is not currently used in the hashing algorithm
HASHITEMPTR:	character position (starting at 1) within HASHDATA where the key item begins.
HASHDATA:	record containing the CALC-key items and possibly other items.

Example

Consider a range of 101 pages, a CALC interval of 1 page and a CALC-key composed of 3 items PIC XX, PIC 999, PIC X located at position 8, 10, 13 in the CALC record.

The last three parameters of H_IA_UCHASH are as follows:

```
01   HASHRANGEDESC.

  02   HASHNUMPAGE COMP-2 VALUE 101.
  02   HASHCALCINTERVAL COMP-1 VALUE 1.
  02   HASHINTERVALUNIT COMP-1 VALUE 0.

01   HASHKEYDESC.

  02   HASHNUMITEM COMP-1 VALUE 3.
  02   HASHITEMDESC1.
     03  HASHITEMTYPE1 COMP-1 VALUE 0.
     03  HASHITEMLENGTH1 COMP-2 VALUE 2.
     03  HASHITEMSCALE1  COMP-1 VALUE 0.
     03  HASHITEMPTR1    COMP-2 VALUE 8.
  02   HASHITEMDESC2.
     03  HASHITEMTYPE2 COMP-1 VALUE 9.
     03  HASHITEMLENGTH2 COMP-2 VALUE 3.
     03  HASHITEMSCALE2  COMP-1 VALUE 0.
     03  HASHITEMPTR2    COMP-2 VALUE 10.
  02   HASHITEMDESC3.
     03  HASHITEMTYPE3 COMP-1 VALUE 0.
     03  HASHITEMLENGTH3 COMP-2 VALUE 1.
     03  HASHITEMSCALE3  COMP-1 VALUE 0.
     03  HASHITEMPTR3    COMP-2 VALUE 13.

01   HASHDATA.
```

```
02   RO1-IDENT PIC X(7).
02   RO1-CALC.
   03   RO1-CALC1 PIC XX.
   03   RO1-CALC2 PIC 999.
   03   RO1-CALC3 PIC X.
02   RO1-NUMBER PIC 9(4).
```

6. COBOL/DML Compiling and Linking

6.1 SL AND DD OBJECT USAGE

1. COBOL handles two types of object:

   - SL (Source Language) objects corresponding to the COBOL/DML input source and the COBOL output source

   - DD (Data Description) objects corresponding to the object schema used in the DML processing

2. The COBOL/DML input source can be in:

   - an input-enclosure with the TYPE option COBOL or DATASSF

   - a member of an SL library. The TYPE option of the member must be COBOL, COBOLX or DATASSF.

3. The object schema is a member of a BIN library.

4. Libraries containing the object schema must be assigned statically using the keywords/ifn's DDLIB1, DDLIB2, DDLIB3.

   The member containing the schema whose name is referenced in the SUB-SCHEMA section, is searched in the libraries assigned, in the sequence defined by DDLIB1, DDLIB2, DDLIB3.

5. The object schema used by COBOL may be in the "DDL" state or in the "DDL-DMCL" state.

6. The compiled program is stamped with the name and DDL reference date of the schema used in the processing.

   These name and date will have to match at run-time the name and DDL reference date of the schema loaded for execution.

6.2 COBOL EXTENDED JCL STATEMENT

COBOL can be called using the following specific options of its extended JCL statement to compile a COBOL program with DML statements.

```
COBOL

   DDLIB1=(library-description)
  [DDLIB2=(library-description)]
  [DDLIB3=(library-description)]
       [ {NDDLIST  } ]
       [ {DDLIST   } ]    ;
```

6.2.1 COBOL Specific Parameter Description

DDLIBi	These keywords define the libraries containing the object schema and their search path.
DDLIST NDDLIST	This parameter indicates whether the user wants a listing of the description of all used records. The default value is NDDLIST, meaning no description listing.

6.3 COBOL COMPILATION

Compilation of the COBOL source is performed by the COBOL compiler (COBOL). There are no special IDS/II considerations except that the keyword LEVEL=L64 must be specified in the COBOL JCL statement.

Except for the specific options described above, the COBOL extended JCL statement is unchanged. For further details, consult the COBOL User Guide.

6.4 LINKING

After the successful compilation of the COBOL/DML program, the user must execute LINKER to produce a load-module. This linking procedure is covered for the general case in the LINKER manual, but linkage of programs containing DML may require that a command statement stream is provided to the linkage editor.

The LINKER statement contains the parameter groups

```
{COMFILE= {*input-enclosure-name             }
{         {(sequential-input-file-description)}
{COMMAND=' {command}...'                      }
```

through which LINKER commands are supplied.

The LINKER commands that may be necessary are:

```
 ____________________________________________
|                                            |
|  SEMPOOL=(NUMSN = +p),                     |
|  STACK1=(INITSIZE = 3K, MAXSIZE = 16K),    |
|____________________________________________|
```

The first command is necessary if the number of semaphores required by UFAS is greater than that reserved by default by LINKER. The user is informed of this situation by a message on the JOR at execution of a READY STATEMENT (TASKM, ENTRYOV). The value p should be set to 7 + 4 * number-of-areas. The + sign is mandatory and does not have the same meaning as a space. In order to avoid any computation, the SEMPOOL command may be written as follows:

```
SEMPOOL=(NUMSN = 225),
```

which takes the maximum possible number of semaphores.

The second command is related to the number of stack overflows occurring at execution time. This number must be checked in the JOR, especially when the data base is opened in update. If its value is higher than 200, the STACK1 command must be included to increase to initial size of the stack of ring 1 (default value 2K). If 3K is not sufficient, higher values can be chosen.

6.5 EXAMPLE

An object schema INVENTORY exists in member INVENTORY of library MIDS.BINLIB.

The user wishes to compile a COBOL/DML program named CONTROL17 against this schema. The program has the following outline:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CONTROL17.
        ...
DATA DIVISION.
SUB-SCHEMA SECTION.
DB INVENTORY.
        ...
PROCEDURE DIVISION.
        ...
      READY
        ...
      FINISH
       ...
     END COBOL.
```

The input source is in member CONTROL17 of library MIDS.SLLIB. The compile unit produced will be in library MIDS.CULIB. All the libraries are assumed to be cataloged.

```
$JOB DMLCOB USER = MURIEL PROJECT = LONGEOT;
LIB SL INLIB1=MIDS.SLLIB;

 COBOL SOURCE=CONTROL17
   DDLIB1=MIDS.BINLIB
   CULIB=MIDS.CULIB
  LEVEL=L64 MAP XREF DDLIST;
$ENDJOB;
```

The user now prepares a load-module, without special options, in the library MIDS.LMLIB.

```
$JOB LINK USER=MURIEL PROJECT=LONGEOT;
LIB CU INLIB1=MIDS.CULIB;
LINKER CONTROL17
     OUTLIB=MIDS.LMLIB;
$ENDJOB;
```

After certain runs of the load-module, the JOR indicates that the number of stack overflows is excessively high. The user corrects the situation by relinking with the STACK1 command.

```
$JOB RELINK USER=MURIEL PROJECT=LONGEOT;
LIB CU INLIB1=MIDS.CULIB;
LINKER CONTROL17
     OUTLIB=MIDS.LMLIB
     COMMAND='STACK1=(INITSIZE=3K, MAXSIZE=16K) ';
$ENDJOB;
```

7. User Program Execution

7.1 RUN-TIME ENVIRONMENT

7.1.1 Prerequisites

The following operations are prerequisites to the execution of DML programs.

- The schema DDL and DMCL have been translated into an object schema.
- The storage areas have been preallocated against this schema.
- The DML programs have been compiled against this schema and linked.

In order to ensure that the schema referenced at execution is the same as the schema referenced at preallocation and compilation, a certain number of consistency checks are performed at run-time:

- On the first READY that triggers the IDS session, the schema name and DDL reference date recorded in each program by COBOL are compared with the schema name and DDL reference date of the object schema used at execution.
- On each READY, the schema name and DMCL reference date recorded in the label of each area by PREALLOC are compared with the schema name and DMCL reference date of the object schema used at execution.

7.1.2 Execution Modes

The DML programs may be executed in BATCH, IOF and TDS:

- The BATCH environment includes the jobs that either do not activate the telecommunication lines or activate them indirectly through the input and output queues of MCS.

- The IOF environment corresponds to the jobs connected to a terminal under the supervision of the Interactive Operation Facility.

- The TDS environment corresponds to the steps running under the supervision of the Transaction Driven System.

NOTE: The data base utility DBUTILITY is considered as a user step launched by the same type of basic JCL and able to run in either a batch or interactive (IOF) environment.

7.2 OBJECTS REFERENCED AT RUN-TIME

As far as IDS/II is concerned, a user step references four types ofobject:

- Data base object schemas.
- Data base storage areas.
- An optional IDSOPT file containing the run-time commands.
- Optional IDSTRACE files collecting program and DBCS trace information.

Data base object schemas reside in libraries of type BIN. These libraries must be assigned statically using the reserved ifn's DDLIB1, DDLIB2, DDLIB3.

If a schema name recorded in a program of the step is not qualified in the run-time commands, the object schema is searched for loading in the libraries assigned, in the sequence defined by DDLIB1, DDLIB2, DDLIB3.

If a schema name is qualified in a run-time SCHEMA command by the efn of a library or any of the keywords DDLIB1, DDLIB2, DDLIB3, the object schema is searched in the library specified.

If two schemas with identical names are used in the step, they must be stored in two different libraries and at least one must be qualified in the run-time commands.

Object schemas required at execution must be in the "DDL-DMCL" state.

Data base storage areas must be assigned statically using the ifn's defined in the DMCL or redefined in the run-time commands. Only the areas to be READYed need be assigned. If several data bases are referenced in the step, all area ifn's must be different.

- The run-time command file may be:
  - An input-enclosure with the TYPE option DATA or DATASSF.
  - A member of an SL library. The TYPE option of the member must be DATASSF.
  - A sequential file.

The run-time command file, when present, must be assigned statically using the reserved ifn IDSOPT.

The absence of the run-time command file is indicated in the JCL by the absence of an ASSIGN IDSOPT statement. In this case the run-time default options apply.

If several data bases are involved at execution, the group of run-time commands related to a given data base is identified by the first command which must be a SCHEMA command.

- The IDS trace files are standard or permanent SYSOUT files. They are used only when TRACE commands are specified in the run-time command language or when the DML programs call the DBCS function H_IA_UCTRACE.

The ifn used to reference a trace file can be:

- The default ifn IDSTRACE if it is not redefined in the TRACE command.
- The ifn specified in the IDSTRACE clause of the TRACE command.
- The ifn specified in the second parameter of H_IA_UCTRACE.

A static ASSIGN is only needed when the TRACE output is not directed to the standard SYSOUT

In the case of several data bases, the TRACE output can be directed to different SYSOUT files if the ifn's are different or to the same SYSOUT file if the ifn's are identical.

- In a TDS environment:
  - Only one data base is accessible to the TDS step.
  - Any output directed to an IDS trace ifn is automatically redirected to a member of the library assigned to the TDS reserved ifn DBUGFILE.
  - Only secondary programs with DB-DESCRIPTIONS IN LINKAGE can be subject to the "USE procedure" option at TDS generation time.
- The previous considerations are illustrated in Figure 7-1.

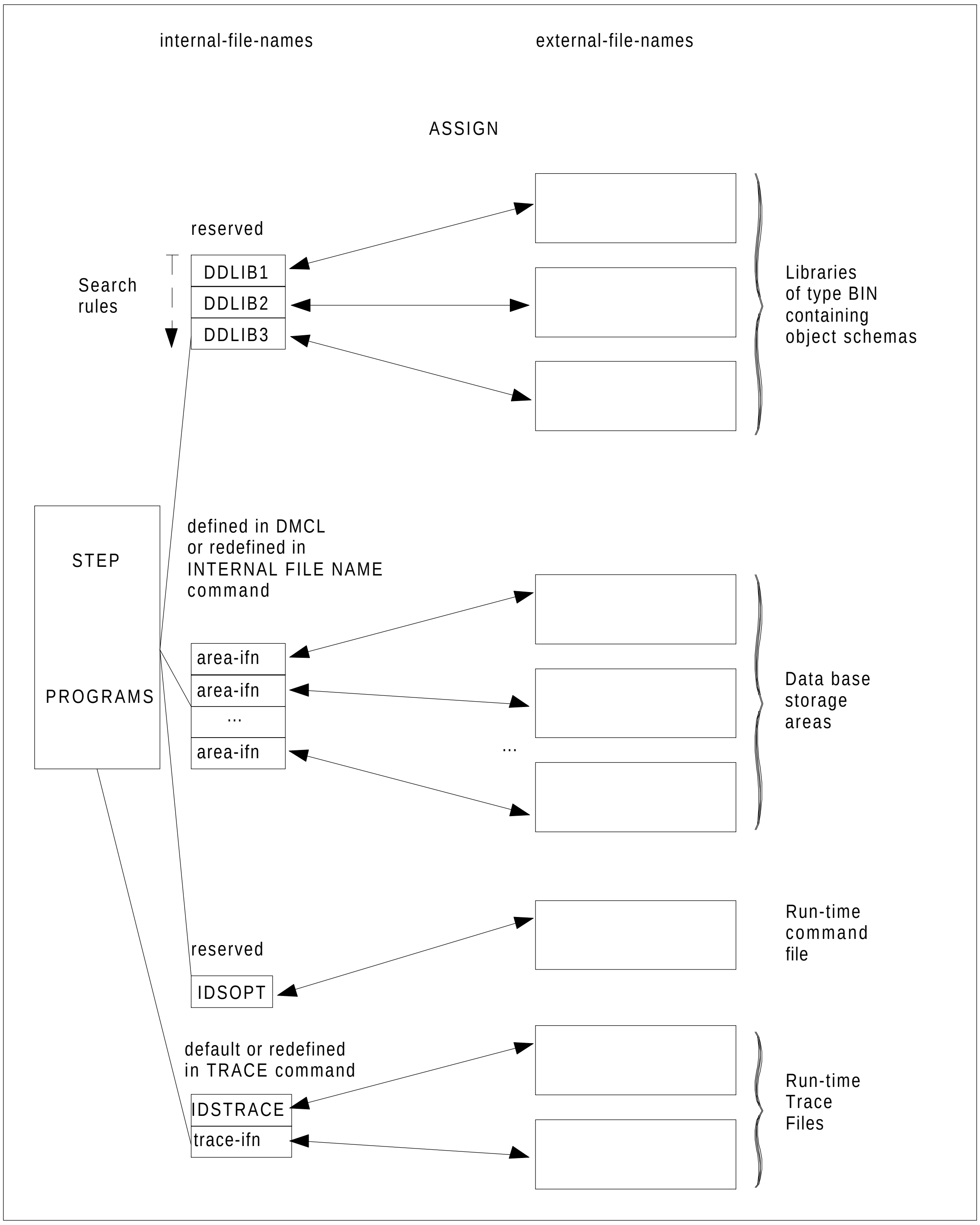


Figure 7-1. Objects Referenced at Run-time

7.3 USER STEP BASIC JCL

A step starting IDS programs uses the basic JCL:

```
STEP   user-load-module   {step-parameter}...[REPEAT] ;
  [ SIZE...;]
    ASSIGN  DDLIB1  library-description ;
  [ ASSIGN  DDLIB2  library-description ; ]
  [ ASSIGN  DDLIB3  library-description ; ]
   { ASSIGN  area-ifn  area-efn  assign-parameters ; }
   {     [ DEFINE  area-ifn  define-parameters ; ]   }...
   [ ASSIGN IDSOPT   {*input-enclosure-name   } ;  ]
                     {input-file-description  }    ]
   [ ASSIGN idstrace-ifn simple-file-description ;]...
   [ other statement for non-IDS file ; ]...
ENDSTEP ;
```

7.3.1 Parameter Description

REPEAT	This parameter is mandatory if the step runs in concurrent access environment (GAC) or if it activates the checkpoint function.
SIZE	NBBUF(total number of buffers) and POOLSIZE (total size of UFAS pool) must be specified if default values are insufficient.
DDLIBi	These reserved ifn's reference the libraries containing the object schemas and define their search path.
area-ifn	Internal-file-name of a storage area, as defined in the DMCL. An ASSIGN statement is required for each area involved in the processing. It specifies the area usage mode (SHARE, ACCESS parameters) to confirm or override the usage mode specified in the READY. A DEFINE statement may be necessary to specify the JOURNAL and READLOCK parameters. Under TDS the READLOCK parameter must not be specified.
IDSOPT	This reserved ifn references the file containing the run-time commands, if any.
idstrace-ifn	This ifn references a SYSOUT file onto which trace information is written. The default ifn is IDSTRACE. It can be redefined in the TRACE command of the run-time command language. Under TDS the trace ifn must not be DBUGFILE. The ASSIGN statement is only required if the trace file is not the standard SYSOUT.

7.3.2 Example

A schema has the following DMCL entries:

```
SCHEMA NAME IS MRK-TECH-PUB ...
AREA NAME IS SOFTWARE-MANUALS
   AREA INTERNAL FILE NAME IS SOFTPBS ...
AREA NAME IS HARDWARE-MANUALS
   AREA INTERNAL FILE NAME IS HARDPBS ...
```

The schema library and the two areas are catalogued with the external-file-names DB.BINLIB, DB.SOFT, DB.HARD respectively.

Run-time statistics are required and the DBCS has to trace onto the standard SYSOUT the STORE statements that involve at least 10 input-output transfers between the buffers and the data base.

If the areas are used in MONITORED mode and protected by the BEFORE and AFTER journals, the step enclosure may be coded as follows:

```
 __________________________________________________________
|                                                          |
| STEP ...REPEAT;                                          |
| ASSIGN  DDLIB1  DB.BINLIB;                               |
| ASSIGN  SOFTPBS DB.SOFT SHARE = MONITOR ACCESS = WRITE;  |
| DEFINE  SOFTPBS JOURNAL = BOTH;                          |
| ASSIGN  HARDPBS DB.HARD SHARE = MONITOR ACCESS = WRITE;  |
| DEFINE  HARDPBS JOURNAL = BOTH;                          |
| ASSIGN  IDSOPT *RUNCOM;                                  |
| ENDSTEP;                                                 |
|                                                          |
| $INPUT RUNCOM;                                           |
| SCHEMA NAME IS MRK-TECH-PUB.                             |
| STATISTICS WITH TIMING.                                  |
| TRACE STORE                                              |
|    THRESHOLD IS 10 I-O                                   |
|    PRINT CURRENT FIELDS STATISTICS TIMING.               |
| $ENDINPUT;                                               |
|__________________________________________________________|
```

7.4 CONCURRENT ACCESS AND AREA PROTECTION

7.4.1 Data Base Sharing

- Three main concepts are involved in the sharing of a data base:
  - <u>The resource to be shared</u>: this is the data base area at a global level or the area page at a lower level.
  - <u>The user of a resource</u>: this is the BATCH/IOF step or the TDS transaction.
  - <u>The time period during with a user is granted access to a resource</u>: is the time frame delimited by the ASSIGN-DEASSIGN functions at the file level, or a part of a commitment-unit at the page level.
- A data base is composed of several areas. If different users access different areas at the same time, they are not considered as sharing the data base in the context of the following discussion.
- <u>Three sharing modes are available at file level</u>:
  - <u>EXCLUSIVE</u>: the area is dedicated to one user who can perform RETRIEVAL or UPDATE operations.
  - <u>SHARED</u>: the area is shared by several users who perform only RETRIEVAL operations.
  - <u>MONITORED</u>: the area is shared by several users who perform RETRIEVAL or UPDATE operations. For these users, access control takes place at the page level.

When a user is granted access to an area, he remains a tenant of thisresource until DEASSIGN.

- <u>Three sharing modes are available at the page level</u> (when the sharing mode is MONITORED at file level):
  - <u>EXCLUSIVE</u>: the page is dedicated to only one user who can perform RETRIEVAL or UPDATE operations.
  - <u>SHARED</u>: the page is shared by several users who perform only RETRIEVAL operations.
  - <u>STATISTICAL</u>: access control is disabled; a copy of the disk page is made available to any user as soon as his request is issued. Only RETRIEVAL operations are allowed in this mode.

When a user is granted access to a page, he remains a tenant of this resource until the end of the commitment-unit.

7.4.2 Area Usage Modes

- In the preceding paragraph, the sharing has been considered from the data base point of view. In this paragraph, it is considered from the user point of view.

When a user requests access to an area, he indicates the usage intended for this area. The usage includes the sharing mode at file level (EXCLUSIVE, SHARED, MONITORED) and the access mode (RETRIEVAL, UPDATE).
When sharing at the file level is MONITORED, the usage mode includes the sharing and access at page level.

- Figure 7-2 illustrates the various possible usages and the different places in the system where they can be specified.

- The first column lists the generic usage modes as defined above.

- The second column relates to the specification of the usage mode in the DML program. It can be seen that only the sharing and access modes at the file level can be specified in the READY statement.

The logic of the program must be adapted to the usage mode selected (splitting into commitment-units in MONITORED mode).

- The third column relates to the specification of the usage mode in the JCL of BATH/IOF step.

The SHARE, ACCESS parameters of the ASSIGN statement define the sharing and access modes at the file level. The READLOCK parameter of the DEFINE statement defines the sharing mode of pages accessed for retrieval; pages accessed for update are always reserved in EXCLUSIVE mode.

File Management takes into account the ASSIGN statement (and not the READY statement) when evaluating the condition for granting access to the area. Therefore, the ASSIGN statement must indicate either the same usage mode as that of the READY statement or a more restrictive usage mode: EXCLUSIVE instead of SHARED or MONITORED, SHARED instead of MONITORED, UPDATE instead of RETRIEVAL. The overriding rules are dealt with in the next paragraph.

GENERIC USAGE-MODE		DML READY	TERMINOLOGY (FILE)	BATCH/IOF JCL ASSIGN JCL DEFINE	TERMINOLOGY (FILE) (PAGE)	TDS JCL TDS	TERMINOLOGY ASSIGN (FILE) GENERATION (PAGE)
FILE LEVEL	PAGE LEVEL	FILE LEVEL	PAGE LEVEL	FILE LEVEL	PAGE LEVEL	FILE LEVEL	PAGE LEVEL
EXCLUSIVE RETRIEVAL		EXCLUSIVE RETRIEVAL		SHARE=NORMAL ACCESS=SPREAD			
EXCLUSIVE UPDATE		EXCLUSIVE UPDATE		SHARE=NORMAL ACCESS=WRITE			
SHARED RETRIEVAL		SHARED RETRIEVAL		SHARED=NORMAL ACCESS=READ			
MONITORED RETRIEVAL	EXCLUSIVE RETRIEVAL	MONITORED RETRIEVAL	cannot be specified	SHARE=MONITOR ACCESS=READ	READLOCK=EXCL	SHARE=MONITOR ACCESS=READ SPREAD ALLREAD	default EXCLUSIVE RETRIEVAL
	SHARED RETRIEVAL		cannot be specified		READLOCK= NORMAL		TRANSACTION ... SHARED READ
	STATISTICAL RETRIEVAL		cannot be specified		READLOCK=STAT		TRANSACTION SUPPRESS CONCURRENT CONTROL
MONITORED UPDATE	EXCLUSIVE RETRIEVAL	MONITORED UPDATE	cannot be specified	SHARE=MONITOR ACCESS=WRITE	READLOCK=EXCL	SHARE=MONITOR ACCESS= WRITE SPWRITE	default EXCLUSIVE RETRIEVAL
	EXCLUSIVE UPDATE		always EXCLUSIVE UPDATE		always EXCLUSIVE UPDATE		always EXCLUSIVE UPDATE
	SHARED RETRIEVAL		cannot be specified		READLOCK= NORMAL		TRANSACTION ... SHARED READ

Figure 7-2. Area Usage-mode Designation in Various Contexts

The necessity to specify the usage mode at the JCL level has the advantages:

- Job Management is able to avoid deadlocks at the file level.
- Information concerning the usage mode at the page level can be supplied.
- The EXCLUSIVE mode at the file level may be defined in the JCL for a program whose logic corresponds to the MONITORED mode. This avoids the overhead of the access control at the page level each time the program may run in this EXCLUSIVE mode.

The necessity of specifying the usage mode at the JCL level has the disadvantages:

- The user is led to supply the same information twice, when the usage mode is the same in the program and in the JCL.
- The ASSIGN/DEASSIGN time period may comprise several READY/FINISH sessions with different usage modes. The ASSIGN, DEFINE statements must specify the most restrictive of these usage modes.

  The fourth column relates to the specification of the usage mode in a TDS environment. An added complexity is due to the fact that:

- the TDS step as a whole is a user with respect to the other BATCH/IOF/TDS steps.
- the TDS step corresponds to several users represented by the transactions in progress.

The usage mode at the file level is defined in the JCL of the TDS step and applies to all transactions.

The EXCLUSIVE mode is not relevant since transactions are normally concurrent.

The SHARED mode is not supported for TDS-controlled files.

For the MONITORED mode, it is possible to specify that an area:

- is reserved in exclusive mode with respect to other steps but is shared by the transactions. Strangely enough, this refinement of the sharing mode is indicated by the ACCESS parameter (SPREAD, SPWRITE)

- is shared by other steps as well as by the transactions (READ, WRITE).

The usage mode at the page level is defined in the TDS generation phase and may be different between transactions. The READLOCK parameter of the DEFINE statement must not be specified.

7.4.3 ASSIGN/READY Usage Mode Overriding Rules

The JCL ASSIGN statement can override the program READY statement. The overriding rules are enforced by both File Management and the DBCS.

Figure 7-3 shows the JCL ASSIGN SHARE and ACCESS parameters that can be specified depending on the READY USAGE-MODE parameter.

When the combination of parameters is not allowed, the intersection of a row and a column contains a hyphen.

When the combination of parameters is allowed, the intersection contains the usage that is finally retained.

SPREAD, ALLREAD and SPWRITE are not mentioned in the chart for the sake of clarity.

If the EXCLUSIVE RETRIEVAL mode defined in READY is actually desired, the ASSIGN must specify

```
SHARE=NORMAL and ACCESS= {WRITE }
                         {SPREAD}
```

PROGRAM READY JCL ASSIGN		USAGE MODE				
		EXCLUSIVE		SHARED	MONITORED	
		RETRIEVAL	UPDATE	RETRIEVAL	RETRIEVAL	UPDATE
NORMAL	READ	-	-	SHARED RETRIEVAL	SHARED RETRIEVAL	-
	WRITE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
MONITOR	READ	-	-	-	MONITORED RETRIEVAL	-
	WRITE	-	-	-	MONITORED RETRIEVAL	MONITORED UPDATE
ONEWRITE	READ	-	-	-	SHARED RETRIEVAL	-
	WRITE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE	SHARED RETRIEVAL	SHARED RETRIEVAL	EXCLUSIVE UPDATE
FREE	READ					
	WRITE					

Figure 7-3. Area Final Usage-Mode (File level) After Overriding of DML READY by JCL ASSIGN

7.4.4 Area Protection by Journals

- Two modes of protection are provided to restore a data base area, or part of it, to a known stable state corresponding to a consistency-point: beginning of step, check-point, commitment-point, end of step.

The consistency-point is termed local if it relates to a given user;other users may still be working on other parts of the area.

The consistency-point is termed global if it relates to all users; everyuser has reached a local consistency-point and the data base area isinactive.

- The area protection is based on the assumption that the pages modified by a user between two consistency-points are held in exclusive mode.

- The BEFORE JOURNAL, by recording the state of the pages before their modification, allows the system to restore these pages to the state they were in at a previous consistency-point.

This method is used to recover from program or system failures or to solve deadlock situations.

- The AFTER JOURNAL, by recording the state of the pages after their modification, allows the system to restart from a save copy of the area, corresponding to a global consistency-point, and to re-apply all the modifications that occurred until a subsequent consistency-point.

This method is used to recover from hardware errors causing the physical destruction of the data base area. It is therefore recommended that the AFTER journal not reside on the same volumes as those of the data base.

In TDS environment, the AFTER journal may also be used in conjunction with the DEFERRED UPDATES mechanism to replace the BEFORE journal protection, insofar as there are enough buffers to accommodate all the pages modified during the commitment-unit.

- The use of the BEFORE or AFTER journal is optional or mandatory depending on the area usage mode and the environment.

If the access mode is RETRIEVAL, journals are not used.

If the usage is EXCLUSIVE UPDATE in BATCH/IOF steps, both journals are optional.

If the usage mode is MONITORED UPDATE in BATCH/IOF steps, the BEFORE journal is mandatory and the AFTER journal optional.

If the usage mode is MONITORED UPDATE in a TDS step, either the BEFORE journal or the AFTER journal with DEFERRED UPDATES must be activated.
When the BEFORE journal is used, the AFTER journal is optional.

If several steps update the same data base, whether sequentially or concurrently, is is recommended that they activate the area protection in a consistent way. This is particularly true for the AFTER journal when it serves for the protection against media errors.

The DBA has the possibility to impose a uniform and permanent protection level by specifying JOURNAL options in the catalog.

7.4.5 Consistency of Area Protection Levels for the Same BATCH/IOF User

If it is important that a given area be accessed with the same protection level by several users, it is also important that a given user access the different areas of the data base with consistent protection levels.

The system itself performs no consistency check on the protection levels of different files. In IDS/II the areas are part of the same logical structure and the DBCS checks at least that, if a return to a consistency point is required, it is possible to rollback all the areas currently opened.

Figure 7-4 illustrates the constraints on the usage of the different areas opened for update by the same user in the BATCH/IOF environment. In the TDS environment, the areas are mandatorily in MONITORED mode.

In any event, the DBA may enforce the consistency of protection levels between areas by specifying the same JOURNAL options in the catalog.

Areas Already in READY State READY request for new area		EXCLUSIVE UPDATE		MONITORED UPDATE
		no BEFORE	BEFORE	always BEFORE
EXCLUSIVE UPDATE	no BEFORE	Y (ROLLBACK impossible)		-
	BEFORE	-		Y (ROLLBACK possible)
MONITORED UPDATE	always BEFORE			

Figure 7-4. Consistency of Area Protection Levels for the Same User In BATCH/IOF Environment

7.5 RUN-TIME COMMAND LANGUAGE

The syntax rules and descriptive notations are those described in Section III for the DDLPROC command language.

7.5.1 Reserved words

ABNORMAL	MODLAB
ACCEPT	NAME (NAMES)
ALL	NO
ANY	NON-NULL
ARE (IS)	NOT
AREA (REALM)	NUMBER
AT	OCCURRENCE
BUFFER (BUFFERS)	OF
CANCEL	OPEN
CHECK	OUTPUT
CLEANPOINT	PAGE
COMMITMENT	POINTERS
CONNECT	POOL
CURRENCIES	PRINT
CURRENT	PROGRAM
DB-REGISTERS	PUTPAGE
DB-STATUS	READY
DDLIB1	REALM (AREA)
DDLIB2	RECORD
DDLIB3	RELPAGE
DISCONNECT	RESTART
DUMP	ROLLBACK
ERASE	SAVE
EXCEPT	SCHEMA
EXTEND	SEGMENT (SEGMENTS)
FIELDS	SET
FILE	SOURCE
FIND	STARTDBS
FINISH	STATE
FOR	STATEMENT (STATEMENTS)
GET	TATISTICS
GETLAB	STORE
GETPAGE	STRUCTURE
GETPTR	SWITCH
HEXADECIMAL (HEXA)	SYNCHRO
IAC	SYSTEM
IDSTRACE	TERMDBS
IF	THRESHOLD
IGNORE	THROUGH (THRU)
IN (OF)	TIMING
INCONSISTENT	TRACE
INPUT	TRANSIENT
INTERNAL	UNLOAD
IS (ARE)	WARNING
I-O	WHEN
LINE	WITH
MODIFY	

7.5.2 Types of Command

There are eleven types of command. The group of commands that correspond to a given schema is identified by the first command, which must be a SCHEMA command.

The run-time command language format is as follows.

```
[ SCHEMA command                      ]
[                                     ]
[   [IGNORE TRANSIENT command]        ]
[                                     ]
[   [POOL command]                    ]
[                                     ]
[   [BUFFERS command]                 ]
[                                     ]
[   [SAVE CURRENCIES command]         ]
[                                     ]
[   [SYNCHRO command]                 ]   ...
[                                     ]
[   [NO WARNING command]              ]
[                                     ]
[   [STATISTISC command]              ]
[                                     ]
[   [TRACE command]                   ]
[                                     ]
[   [CHECK PAGE command]              ]
[                                     ]
[   [INTERNAL FILE NAME command]      ]
```

The commands will be describe in detail.

7.5.3 SCHEMA Command

Function

To identify the schema to which subsequent commands apply and to provide information for the loading of this schema.

Format

```
                             [        {library-efn} ]
SCHEMA NAME IS schema-name [{IN}   {DDLIB1      } ]
                             [{OF}   {DDLIB2      } ]
                             [        {DDLIB3      } ]
        [FOR PROGRAM program-name]
        [NUMBER OF SEGMENTS IS integer] .
```

Rules

1. ach time a main program starts an IDS session, the DBCS searches the run-time command file, looking for a SCHEMA command whose schema-name corresponds to that recorded in the program.

2. If the PROGRAM clause is not specified, the search for the SCHEMA command stops as soon as it identifies the right schema-name.

3. If the PROGRAM clause is specified, the DBCS further checks that "program-name" corresponds to the name of the program starting the IDS session. The search continues if the names do not match.

   This option is required to differentiate 2 schemas with identical names used by 2 different programs of the step.

4. If the search for the SCHEMA command fails, all default options are applied.

5. If the library qualification is absent, the schema is searched for loading in the libraries assigned, in the sequence defined by DDLIB1, DDLIB2, DDLIB3.

6. If the library qualification is present, the schema is loaded from the library specified.

7. If the SEGMENTS clause is not specified, the schema is loaded in the number of segments indicated in the PRINT STORAGE DESCRIPTION report of DDLPROC.

8. If the SEGMENTS clause is specified, the schema is loaded in the number of segments specified, provided that it is less than the default number indicated by DDLPROC.

   This option is useful if the segment vacant entries of the step are a limited resource. In TDS environment, it is recommended that 1 segment be specified.

7.5.4 IGNORE TRANSIENT Command

Function

To ignore the transient or inconsistent state of some realms when they are readied.

Format

```
              TRANSIENT
IGNORE                            STATE OF {realm-name}... .
              INCONSISTENT
```

Rules

1. If TRANSIENT is specified, the realms of the list are made available to the program even if they are in the TRANSIENT state at READY time.

   The TRANSIENT condition occurs when a previous step updating the realms in exclusive mode without the BEFORE journal protection has terminated normally or abnormally without finishing these realms.

   The TRANSIENT condition indicates that a part of the data base is presumably inconsistent. The check of this condition at READY time is a safeguard against the further use of the realms.

   The normal way to recover from this situation is to restore the realms from save copies. Another way is to run DBUTILITY to ascertain that the data base is consistent, to patch local inconsistencies if any, and to use the RESET TRANSIENT clause of the PATCH LABEL command.

   However, if the user is certain that the faulty step left the realms in a consistent state, as regards both IDS pointers and user data, he may use the IGNORE TRANSIENT command. The fact that the TRANSIENT state is ignored is recorded in the IDS label.

2. If INCONSISTENT is specified, the realms of the list are made available to the program even if they are found in the INCONSISTENT state at READY time.

   The INCONSISTENT condition occurs when a previous IDS session has detected and recorded in the IDS label a data base inconsistency. The inconsistency is recorded even if the realms are open in retrieval mode (except when SHARE=MONITOR and READLOCK=STAT, in which case the inconsistency may be only apparent). The check of this condition at READY time is a safeguard against the further use of the realms.

   The normal way to recover from this situation is to restore the realms from consistent save copies. Another way is to patch the inconsistent part of the data base by using the PATCH command of DBUTILITY after analysis of the DBCS error message reporting the inconsistency, to validate the patch and if the validation is successful, to use the RESET INCONSISTENT clause of the PATCH LABEL command.

However, if the user is certain that the program to be run will not be working on the part of the data base that is inconsistent, he may use the IGNORE INCONSISTENT command. The fact that the INCONSISTENT state is ignored is recorded in the IDS label.

3. If the IGNORE command is omitted, then the program is aborted if it attempts to ready realms which are either in the TRANSIENT or INCONSISTENT state.

4. If the IGNORE command is present, the TRANSIENT and/or INCONSISTENT state of the involved areas will be ignored but not reset.

7.5.5 BUFFER POOL Command

Function

To specify whether the areas of the data base share a buffer pool.

Format

```
 _____________________________________________
|                                             |
| BUFFER POOL NAME IS buffer-pool-name.       |
|_____________________________________________|
```

Syntax rules

1. The buffer-pool-name must be at most four characters long.

2. If the blank name is used then it must be in the form ' ' (that is, blanks enclosed by apostrophes).

3. Non-blank names may contain letters, digits, the minus sign and the underscore. They must be delimited by apostrophes if they do not conform to the rules for the formation of DDL names.

General rules

1. If the POOL command is absent, then the default name is that specified in the DMCL. If the POOL command is present, it overrides the DMCL.

2. If the blank name is specified, there is no buffer pool and the NUMBER OF BUFFERS command gives the number of buffers reserved for each area (that is, each area has its own set of buffers).

3. If a non-blank name is specified, there is a buffer pool and the NUMBER OF BUFFERS command gives the total number of buffers that will be shared by all areas.

4. The purpose of the pool of buffers is to reduce the total buffer space requirements when several areas are in the READY state. When a DML program accesses one area for a certain length of time:

   - If there is no buffer pool, it cannot use more buffer space than that reserved for this area. The space reserved for the other areas is not available.

   - If there is a buffer pool, the total pool space may be used for this area. When the program starts working on a different area, the pool space is assigned to the buffers of the new area.

5. In a TDS environment, this command is ignored.

7.5.6 NUMBER OF BUFFERS Command

Function

To specify the number of buffers reserved for each area or for the buffer pool.

Format

```
 ___________________________________
|                                   |
| NUMBER OF BUFFERS IS integer .    |
|___________________________________|
```

Rules

1. If the BUFFERS command is absent, the default number of buffers is that specified in the DMCL. If the BUFFERS command is present, it overrides the DMCL.

2. The minimum value that can be specified is 3.

3. If there is a non-blank buffer-pool-name then the value specified is the number of buffers to be shared amongst all areas. In the case where different areas have different page sizes, the largest value will be chosen as the buffer size.

4. If there is a blank buffer-pool-name then the value specified is the number of buffers to be reserved for each area.

5. The number of buffers to choose at run-time is a compromise between the buffer space requirements and the performance requirements.

   If a program works for a certain time on a limited set of pages (cluster) the number of buffers can be chosen to accommodate all these pages.

   If a program accesses pages in a completely random manner, then increasing the number of buffers will not improve the total performance.

6. In TDS environment, this command is ignored.

7.5.7 SAVE CURRENCIES Command

Function

To request that currency indicators be not nulled across commitment-points in a BATCH/IOF environment.

Format

```
 ____________________________________
|                                    |
| SAVE CURRENCIES AT COMMITMENT .    |
|____________________________________|
```

Rules

1. If the SAVE CURRENCIES command is omitted, the currency indicators are nulled at each commitment-point and the data base pages to which they correspond are unlocked by the run-unit.

2. If the SAVE CURRENCIES command is specified, the currency indicators keep their values across commitments, but the data base pages to which they correspond are unlocked by the run-unit at the commitment-point.

   This option should be used only if conventions exist between applications, which ensure that the records referenced by the currency indicators of a run unit are not erased or replaced by concurrent run-units.

   If this condition is not met, the DBCS may abort the run-unit on encountering an inconsistency between the data base and the currency indicators.

3. In TDS environment this command is ignored. The currency indicators are always nulled and the pages unlocked at the commitment-point.

   If a record data-base-key is to be kept across commitments, it must be saved by an ACCEPT statement in the TRANSACTION STORAGE before the end of a commitment unit, and restored by a FIND DB-KEY statement in the next commitment unit.

4. This command does not apply to check-points. In non-concurrent access environment, currencies are left unchanged across check-points.

7.5.8 SYNCHRO Command

Function

To disable the asynchronous page reading occurring during the sequential search of an area.

Format

```
 ____________
|            |
| SYNCHRO .  |
|____________|
```

Rules

1. If this command is omitted, each FIND ... WITHIN realm-name triggers the look-ahead mechanism which initiates the transfer of the page following (or preceding) the page currently needed. The transfer can occur asynchronously while the DBCS or the program is processing the current page, thus reducing the total elapsed time.

2. If this command is specified, the look-ahead mechanism is disabled.

3. This command may be useful in the case where the extra buffer needed for the asynchronous read must be freed before the page it contains is actually used.

   For example, if a program, using a pool of 4 buffers, searches an area sequentially and, for each record found, searches a set occurrence spreading over 3 pages in another area, then the total number of pages involved in the processing of the occurrence is 5 (4 for the owner and members and 1 for the page read in advance). As there are only 4 buffers, the page read in advance for the owner will be released during the processing of the members and will have to be read again for the next occurrence, thus degrading the performance. In this case it is recommended to use the SYNCHRO command or to augment the number of buffers or to suppress the pool in order to insulate the buffers of the two areas.

7.5.9 NO WARNING Command

Function

To prevent the DBCS from sending warning messages to the JOR.

Format

```
 _______________
|               |
| NO WARNING .  |
|_______________|
```

Rules

1. If the NO WARNING command is omitted, the warning messages prepared by the DBCS in the DB-DETAILED-STATUS register are sent to the JOR (Job Occurence Report).

2. If the NO WARNING command is specified, the warning messages are not sent to the JOR.

7.5.10 STATISTICS Command

Function

To request the DBCS to gather statistical and timing information during the IDS/II session.

Format

```
 ____________________________
|                            |
| STATISTICS [WITH TIMING] . |
|____________________________|
```

Rules

1. If the STATISTICS command is specified without the TIMING option, the information is reported in the JOR (Job Occurrence Report) at IDS/II session end:

   - The number of buffers used for the pool or for each area.
   - The number of program calls to each category of DBCS function: ACCEPT, CONNECT, ...
   - The average number of SYSTEM functions called by each category of DBCS function. The SYSTEM functions in question are limited to UFAS (Unified File Access System) and GAC (General Access Control).
   - The average number of page transfers performed by each category of DBCS function (I/O on journals are not included).
   - The number of calls to each category of DBCS sub-function. The sub-functions are DIRECT PLACEMENT (for the storage of a DIRECT or VIA record), CALC PLACEMENT, SET SELECTION, and SET INSERTION.
   - The average number of SYSTEM functions called by each category of DBCS sub-function.
   - The average number of page transfers performed by each category of DBCS sub-function.
   - The number of calls to each category of SYSTEM function.
   - The average number of page transfers performed by each category of SYSTEM function.
   - The total number of calls to DBCS functions.
   - The total number of calls to SYSTEM functions.
   - The total number of page transfers.

2. If the STATISTICS command is specified with th TIMING option, the information is also reported:

   - The average elapsed time for each category of DBCS function.
   - The average processor time for each category of DBCS function. This time includes the processor time spent in the operating system functions called.
   - The total DBCS elapsed time.
   - The total DBCS processor time.

3. If the STATISTICS command is omitted, no statistical information is collected during the IDS session.

4. Figure 7-5 shows an example of the run-time session statistics with the TIMING option.

NOTES: 1. Non-integral values are rounded to the closest printable value. For example, an average number of I-0s reported as "0.02" means that the actual value v is such that 0.015 < v < = 0.025.

2. The total number of I-Os reported by the DBCS may differ slightly from the number of I-0 CONNECT displayed on the JOR; this is because OPEN and CLOSE I-0 are not available to the DBCS.

3. The SYSTEM statistics are intended for the field engineers in case performance problems require investigation.

4. For the interpretation of the statistics,see the related paragraph at the end of this section.

```
|   **** STATISTICS * V40.0 * PROGRAM:TG02 * SCHEMA:TG02-SH ****                      |
|   BUFFER POOL:YES * NUMBER OF BUFFERS:4                                             |
|   DBCS STATISTICS:                                                                  |
|            ************************************************************************ |
|            *  DBCS FUNCTION    *   NUMBER   *AVERAGE NB OF* AVERAGE NB  * AV. ELAPSED* AV. |
|            *                   *  PROCESS   *             *             *            *    |
|            *     CATEGORY      *  OF CALLS  *SYSTEM CALLS *  OF I/O     * TIME (MS)  * TIME|
|            *                   *     (MS)   *             *             *            *    |
|            *-------------------*------------*-------------*-------------*------------*-----|
|            * ACCEPT            *        431 *        0.00 *        0.00 *        0.1 *    |
|            *                   *        0.1 *             *             *            *    |
|            * FIND DIRECT       *        184 *        0.82 *        0.16 *       21.3 *    |
|            *                   *        1.5 *             *             *            *    |
|            * FIND CURRENT      *         26 *        0.31 *        0.00 *       10.9 *    |
|            *                   *        0.8 *             *             *            *    |
|            * FIND F/L/I AREA   *          2 *       58.00 *       20.00 *      533.5 *    |
|            *                   *       94.6 *             *             *            *    |
|            * FIND N/P SET      *        396 *        0.76 *        0.14 *       20.9 *    |
|            *                   *        1.7 *             *             *            *    |
|            * FIND SELECT USING *         33 *        2.18 *        0.64 *       71.2 *    |
|            *                   *        5.3 *             *             *            *    |
|            * FINISH            *          1 *        1.00 *        -.-- *      948.0 *    |
|            *                   *       47.1 *             *             *            *    |
|            * GET               *        264 *        0.00 *        0.00 *        0.2 *    |
|            *                   *        0.2 *             *             *            *    |
|            * MODIFY MEMBERSHIP *         33 *        8.30 *        1.00 *      143.1 *    |
|            *                   *       11.1 *             *             *            *    |
|            * READY             *          1 *        0.00 *        -.-- *     2165.3 *    |
|            *                   *      234.9 *             *             *            *    |
|            * STORE             *        155 *        6.34 *        1.23 *      128.2 *    |
|            *                   *       10.3 *             *             *            *    |
|            ************************************************************************ |
|            *********************************************************               |
|            * DBCS SUBFUNCTION  *   NUMBER   *AVERAGE NB OF* AVERAGE NB  *              |
|            *     CATEGORY      *  OF CALLS  *SYSTEM CALLS *  OF I/O     *              |
|            *-------------------*------------*-------------*-------------*              |
|            * SET SELECTION     *        200 *        1.53 *        0.29 *              |
|            * SET INSERTION     *        167 *        2.29 *        0.20 *              |
|            * DIRECT PLACEMENT  *        125 *        2.00 *        0.64 *              |
|            * CALC PLACEMENT    *         30 *        2.70 *        1.63 *              |
|            *********************************************************               |
|   SYSTEM STATISTICS:                                                                |
|            *******************************************                             |
|            *  SYSTEM FUNCTION  *   NUMBER   * AVERAGE NB  *                            |
|            *     CATEGORY      *  OF CALLS  *  OF I/O     *                            |
|            *-------------------*------------*-------------*                            |
|            * CHECKCI           *         36 *        0.00 *                            |
|            * FREECI            *        734 *        0.00 *                            |
|            * GETCI             *        734 *        0.51 *                            |
|            * SETCI             *        401 *        0.00 *                            |
|            *******************************************                             |
|   TOTAL NUMBER OF DBCS CALLS   : 1526                                               |
|   TOTAL NUMBER OF SYSTEM CALLS : 1905                                               |
|   TOTAL NUMBER OF I-C          : 371                                                |
|   TOTAL DBCS ELAPSED TIME (MN) : 0.729                                              |
|   TOTAL DBCS PROCESS TIME (MN) : 0.062                                              |
|_____________________________________________________________________________________|
```

Figure 7-5. Run-time Session Statistics.

7.5.11 TRACE Command

Function

To activate the tracing of DML statements.

The trace function includes:

- an "external" trace, intended to help the end user follow the execution of a DML program. He can select the types of DML function to be traced, restrict their tracing to certain conditions and choose the amount of information to be printed.

- an "internal" trace, intended to help Field Engineering to check the IDS-UFAS-GAC interface or investigate performance problems. The user need not be concerned with the internal trace.

General Format

```
TRACE clause

  [WHEN RECORD clause]

  [WHEN AREA clause]

  [WHEN DB-STATUS clause]

  [WHEN THRESHOLD clause]

  [WHEN PROGRAM clause] ...

  [PRINT clause]

  [IDSTRACE clause]

  [OPEN clause]

  [IAC clause] .
```

Rules

1. Each clause will be described separately .

2. When the "internal" trace is not activated, the tracing condition is:

```
        TRACE-function-list-condition

    AND RECORD-condition

    AND REALM-condition

    AND DB-STATUS-condition

    AND THRESHOLD-condition

    AND (PROGRAM-1-condition [OR PROGRAM-2 condition]...)
```

3. When the "internal" trace is activated, the tracing condition becomes:

```
TRACE-function-list-condition

AND (PROGRAM-1-condition [OR PROGRAM-2-condition] ...) since
the other conditions cannot be evaluated before the end of the
DML function.
```

7.5.11.1 TRACE Clause

Function

To specify the DML functions to be traced.

Format

```
         ALL

      {               ACCEPT         }
      {               CONNECT        }
      {               DISCONNECT     }
      {               ERASE          }
      {               FIND           }
TRACE { [{ANY EXCEPT}] GET            }   STATEMENTS
      { [{NOT       }] IF             }
      {               MODIFY         }
      {               READY          }
      {               STARTDBS       }
      {               STORE          }
      {               TERMDBS        }
```

Rules

1. If ALL is specified then all DML functions are traced. These include the DML statements proper and two other functions:

   - STARTDBS (START Data Base Session) is the initialization function triggered by the first READY of an IDS session.

   - TERMDBS (TERMINATE Data Base Session) is the termination function triggered by the last FINISH of an IDS session.

2. The IF DML function is the IDS/II generic name for the "Data Base Condition" function.

3. If a list of DML functions is specified and NOT (or ANY EXCEPT) is not coded, then only the functions of the list are traced.

4. If a list of DML functions is specified and NOT (or ANY EXCEPT) is coded, then all functions except those in the list are traced.

7.5.11.2 WHEN RECORD Clause

Function

To restrict the trace action to operations at the completion of which thecurrent record of the run-unit belongs to a list of record-types.

Format

```
 ______________________________________________
|                                              |
| WHEN RECORD NAME IS [NOT] {record-name} ... |
|______________________________________________|
```

Rules

1. If NOT is absent, the list is made up of the record-types specified.

2. If NOT is present, the list is made up of all the record types of the schema except those specified.

3. If this clause is omitted, then operations are traced whatever the type of the current record of the run-unit.

7.5.11.3 WHEN AREA Clause

Function

To restrict the trace action to operations at the completion of which the area of the current record of the run-unit belongs to a list of areas.

Format

```
      {AREA }
WHEN  {     }  NAME IS [NOT] {area-name}...
      {REALM}
```

Rules

1. If NOT is absent, the list is made up of the areas specified.

2. If NOT is present, the list is made up of all the areas of the schema except those specified.

3. If this clause is not specified, then operations are traced whatever the area of the current record of the run-unit.

7.5.11.4 WHEN DB-STATUS Clause

Function

To restrict the trace action to operations at the completion of which DB-STATUS contains certain non-null values.

Format

```
WHEN DB-STATUS IS  {NON-NULL}
                   {ABNORMAL}
```

Rules

1. If NON-NULL is specified, then the DB-STATUS values are all those different from "0000000".

2. If ABNORMAL is specified, then the DB-STATUS values are all those different from "0000000" "0502100" (end of set or end of realm) and "0502400" (record not found).

3. If this clause is omitted, then operations are traced independently of the DB-STATUS value.

7.5.11.5 WHEN THRESHOLD Clause

Function

To restrict the trace action to operations which incur a user-defined minimum number of physical I/O operations (page transfers).

Format

```
 _________________________________
|                                 |
| WHEN THRESHOLD IS integer I-O   |
|_________________________________|
```

Rules

1. "Integer" specifies the minimum number of I/O operations.

2. If this clause is omitted then the trace action is independent of the number of I/O operations incurred by the DML function being traced.

7.5.11.6 WHEN PROGRAM Clause

Function

To restrict the trace action to specified occurrences of DML statements coded at specified source line numbers in specified programs.

Format

```
WHEN PROGRAM NAME IS program-name

  [ SOURCE LINE NUMBER IS integer-1 [THRU integer-2]  ]
  [        {OCCURRENCE IS integer-3 [THRU integer-4]] ]...
```

1. When specified, "integer-1" must be less than "integer-2" and "integer-3" must be less than "integer-4".

2. The specified line numbers ("integer-1", "integer-2") refer to the internal line numbers affected by the COBOL compiler when compiling the specified program.

3. If the SOURCE LINE phrase is not specified, the trace action takes place when the DML function is coded in the program whose name is referenced.

4. If the SOURCE LINE phrase is specified without the OCCURRENCE option, the trace action takes place when the DML function is coded, in the program specified, at a source line whose number:

   - Equals integer-1, when the THRU phrase is absent.

   - Belongs to the range defined by integer-1 and integer-2, when the THRU phrase is present.

5. If the SOURCE LINE phrase is specified with the OCCURRENCE option, the trace action is limited to:

   - The n'th occurrence (n=integer-3) of a DML function within the specified line range, if the THRU phrase is absent.

   - The n'th through p'th occurrences (n=integer-3, p=integer-4) of a DML function within the specified line range, if the THRU phrase is present.

   The occurrence counter related to a line range is set to 0 at the beginning of an IDS session (BATCH/IOF environment) or at the beginning of a TPR (TDS environment). It is incremented by 1 each time all other trace conditions (TRACE-function-list, RECORD, ...,PROGRAM, SOURCE LINE) are satisfied. In particular, if different DML functions satisfying the conditions are executed sequentially within the same line range, the occurrence counter is incremented for each of them.

6. If several SOURCE LINE phrases are coded, the line ranges must not overlap.

7. If several PROGRAM clauses are specified, they must reference different program-names.

8. If no PROGRAM clause is specified, then operations are traced whatever the program where they are coded.

7.5.11.7 PRINT Clause

Function

To specify the amount of information to be printed for each function traced.

Format

```
      [            [ FIELDS [IN HEXADECIMAL]  ]    ]
      [ CURRENT    [                          ]    ]
      [            [ SYSTEM POINTERS          ]    ]
      [                                            ]
      [ [   {AREA } ]                              ]
      [ [   {REALM} ]                              ]
      [ [           ]    CURRENCIES                ]
PRINT [ [   RECORD  ]                              ]
      [ [   SET     ]                              ]
      [ DB-REGISTERS                               ]
      [ STATISTICS                                 ]
      [ TIMING                                     ]
```

Rules

1. If CURRENT is specified, information on the current record of the run-unit at the completion of the traced function is displayed:

    - If FIELDS is specified, the data fields of the record are displayed in accordance with their schema attributes, or in hexadecimal if HEXADECIMAL is coded.

    - If POINTERS is specified, the set pointers of the record are displayed.

    - If neither FIELDS nor POINTERS is specified, both data fields and set pointers are displayed.

2. If CURRENCIES is specified, the contents of currency indicators at the completion of the traced function are displayed:

    - If AREA/REALM is coded, the area currency indicators are displayed.

    - If RECORD is coded, the record-type currency indicators are displayed.

    - If SET is coded, the set-type currency indicators are displayed.

    - If neither AREA nor RECORD nor SET is coded, all currency indicators are displayed.

3. If DB-REGISTERS is specified, the contents of the data base registers at the completion of the traced function are displayed.

4. If STATISTICS is specified, information on the SYSTEM functions activated by the DBCS during the traced function is displayed.

5. If TIMING is specified, the process-time and elapsed-time of the traced function are displayed.

6. If PRINT is specified alone then the maximum information is printed out.

7. If the PRINT clause is omitted, only the function name and the data-base-key of the current record of the run-unit are displayed.

7.5.11.8 Interpretation of the TRACE Listing

A sample TRACE output listing is shown in Figure 7.6. The following comments specify the meaning of the non-self-explanatory symbols.

First Line:

On the first line are listed the program-name and source internal line number of the traced statement, the category of the DBCS function, the current record of the run-unit, the name of the schema currently used and the current time at completion of the function.

If the current-of-run-unit (CRU) at completion of the statement is non-null, CRU is followed by the letter P (for Present) and the data-base-key of the CRU. The latter is given in hexadecimal form and in interpreted decimal form (A: area code number, P:

page number, L: line number). AN and RN supply the area-name and record-name of the CRU.

If the current-of-run-unit is null, CRU is followed by the letter N (for Null) only.

Pointers:

The CRU pointers, if present, are given in their order of appearance in the record pointer zone. They are displayed in:

- The hexadecimal form which shows how many bytes are used.
- The interpreted decimal form which indicates the area code number (for a global pointer), the page and line number.

Complementary information is supplied by the letters:

```
L => local pointer
G => global pointer
E => the record is owner of an empty set occurrence
D => the record is an optional member currently disconnected
N => NEXT pointer
P => PRIOR pointer
O => OWNER pointer
```

Fields:

- Offsets of the fields in the record data zone are given in decimal and hexadecimal.
- The type of the fields is indicated by the symbols:

```
CHAR   for CHARACTER.
FB15,  FB31 for SIGNED BINARY 15 and 31.
UPK    for UNSIGNED UNPACKED DECIMAL
PK     for UNSIGNED PACKED DECIMAL
PK2    for UNSIGNED PACKED-2 DECIMAL
UPKS   for SIGNED UNPACKED DECIMAL
PK     for SIGNED PACKED DECIMAL.
```

- When fields are displayed in hexadecimal, an EBCDIC interpretation is also provided.

Currencies:

A currency indicator may have the states:

- Present => letter P.
- Null => the corresponding indicator is not displayed.
- Virtual => letter V. The related DBK represents:
  - For a current-of-realm, the DBK of the erased record.
  - For a current-of-set, the DBK of the next record of the erased or disconnected record.

AN and RN give the area-name and record-name of the record.

Names:

The area, record and set names are right-truncated to 17 characters.

```
**********************************************************************************************
*****PGID:TG11                           LINE:9080  **FIND N/P SET      **AT                  *
*                                      08:55:24.917**SCH:TG11-SH          **CRU N            *
**CURRENT OF REALM  TG11-A00           V DBK:000001A1 A:0    P:26       L:1   AN:TG11-A00    *
*                                      RN:R02                                                *
**CURRENT OF RECORD R01                P DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      RN:R01                                                *
**CURRENT OF SET    S01                V DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      RN:R01                                                *
*                   S02                P DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      AN:R01                                                *
**DBSTATUS:0502100 AN:                               RN:                             SN:S01  *
**TIMING                                     PROCESS:2.3               ELAPSED:2.3           *
*                                      (MILLISECS)                                           *
**STATISTICS              SYSTEM FUNCTION:  NUMBER  :  AVERAGE         DBCS SUBFUNCTION:     *
*                                       NUMBER  :  AVERAGE                                   *
*                            CATEGORY    : OF CALLS : I-O COUNT            CATEGORY    : OF  *
*                            CALLS : I-O COUNT                                               *
*                         ---------------+----------+-------------     ----------------+-----*
*                         FREECI         :        1 :        0.00                            *
*                         GETCI          :        1 :        0.00                            *
**********************************************************************************************
*****PGID:TG11                           LINE:9I30  **FIND N/P SET      **AT                  *
*                                        08:55:24.949**SCH:TG11-SH          **CRU P          *
*                                    CRU DBK:00000030 A:0    P:3        L:0   AN:TG11-A00    *
*                                    RN:R02                                                  *
**HEADER  STATE:ACTIVE    CODE:2     SIZE:46    STATUS:0   NEXT CALC-RECORD L:-   HASH       *
*                                    CHECK:E448 OWNER OWL RANK:0                             *
**POINTERS MEMBER:S01                            L   N:    0180 A:-    P:24       L:0        *
*                                                AN:TG11-A00                                 *
*                                                    P:    0180 A:-    P:24       L:0        *
*                                                AN:TG11-A00                                 *
                                                     O:    0180 A:-    P:24       L:0        *
*                                                AN:TG11-A00                                 *
**FIELDS      0 0000:R02-IDENT                                         CHAR  === R02         *
*             7 0007:R02-CH                                            CHAR   C:             *
             10 000A:R02-C                                             UPKS  +3              *
*            12 000C:R02-CALCH                                         CHAR   CALC:          *
*            18 0012:R02-CALC                                          UPKS  +5327           *
*            22 0016:R02-SH                                            CHAR   S:             *
*            25 0019:R02-S                                             UPKS  +3              *
*            27 0018:R02-TRAILER                                       CHAR   +++            *
**CURRENT OF REALM  TG11-A00           P DBK:00000030 A:0    P:3        L:0   AN:TG11-A00    *
*                                      RN:R02                                                *
**CURRENT OF RECORD R01                P DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      RN:R01                                                *
                    R02                P DBK:00000030 A:0    P:3        L:0   AN:TG11-A00    *
*                                      RN:R02                                                *
**CURRENT OF SET    S01                V DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      RN:R01                                                *
*                   S02                P DBK:00000180 A:0    P:24       L:0   AN:TG11-A00    *
*                                      AN:R01                                                *
**DBSTATUS:0000000 AN:TG11-A00                       RN:R02                          SN:S01  *
**TIMING                                     PROCESS:2.3               ELAPSED:2.3           *
*                                            (MILLISECS)                                     *
**STATISTICS              SYSTEM FUNCTION:  NUMBER  :  AVERAGE         DBCS SUBFUNCTION:     *
*                            NUMBER  :  AVERAGE                                              *
*                           CATEGORY   : OF CALLS : I-O COUNT  CATEGORY : OF CALLS : I-O COUNT
*                         ---------------+----------+-------------     ----------------+------
                          FREECI         :        1 :        0.00                            *
                          GETCI          :        1 :        0.00                            *
**********************************************************************************************
```

Figure 7-6. Run-time External Trace Listing

7.5.11.9 IDSTRACE Clause

Function

To redefine the internal-file-name of the IDSTRACE file.

Format

```
 ___________________________________________
|                                           |
| IDSTRACE INTERNAL FILE NAME IS ifn        |
|___________________________________________|
  ‾‾‾‾‾‾‾‾          ‾‾‾‾
```

Rules

1. "ifn" must conform to the rules for the formation of internal-file-names as defined in the JCL Reference Manual.

   In addition, "ifn" must be different from:

   ```
   IDSOPT
   PRTFILE
   COMFILE
   SLLIBa
   DDLIBb
   SAVEc
   ```

   where a,b,c are decimal digits or spaces.

2. If this clause is omitted, the default IDSTRACE is assumed.

7.5.11.10 OPEN Clause

Function

To specify the OPEN mode of the IDSTRACE file.

Format

```
        {OUTPUT}
OPEN    {      }
‾‾‾‾    {EXTEND}
```

Rules

1. If OUTPUT is specified, trace information is written starting from the beginning of the file. Previously recorded information is overwritten.

2. If EXTEND is specified, trace information is written starting from the end of the file.

   This option is ineffective if the trace file is SYS.OUT.

3. If this clause is omitted, OPEN OUTPUT is assumed.

7.5.11.11 IAC Clause

Function

To activate the "internal" trace which displays information on certain system calls performed by the DBCS during the execution of a DML function. IAC stands for Integrated Access Component.

Format

```
     [          FIELDS [IN HEXADECIMAL]1  ]
     [                                 ]
     [          SYSTEM POINTERS        ]
IAC  [ WITH                            ]
     [         [INPUT ]                ]
     [         [      ] PAGE           ]
     [         [OUTPUT]                ]
```

Rules

1. If only IAC is coded, the trace is limited to the internal system calls with their parameters, return codes, process-time and elapsed-time.

2. If the supplementary phrase FIELDS is coded, then for each storage record involved in an internal call, the data field values are made available. They are displayed in accordance with their schema attributes, or in hexadecimal if HEXADECIMAL is specified.

3. If the supplementary phrase POINTERS is coded, then for each storage record involved in an internal call, the set pointers are made available.

4. If the supplementary phrase PAGE is coded, pages accessed and/or modified by the DBCS are printed in hexadecimal.

   - If INPUT is specified, pages requested from Buffer Management are printed before use (internal function GETCI, CHECKCI).

   - If OUTPUT is specified, modified pages are printed before their release to Buffer Management (internal function FREECI).

   - If neither INPUT nor OUTPUT is specified, pages requested from Buffer Management and modified pages released to Buffer Management are printed.

5. If the IAC clause is omitted, the "internal" trace is inhibited.

NOTE: When the IAC clause is specified, the RECORD, AREA, DB-STATUS and THRESHOLD clauses are ignored in the evaluation of the tracing condition.

7.5.11.12 TRACE Command Example

Let us consider the case of a run-unit composed of three programs, as shown in Figure 5.2. The main program PO is a COBOL driver (no DML); the second program P1 accesses the data base described by the schema DB1; the third program P2 accesses the data base described by the schema DB2.

User and IDS trace information is gathered on the same IDSTRACE file.

Suppose that the IDS trace is limited to the FINDs of records EMPLOYEE in the realm REGION-3 of the first data base and to the STOREs of records EMPLOYEE in the second data base. Information to be printed includes the record data fields in hexadecimal and the data base registers. The run-time commands are specified as follows:

```
SCHEMA NAME IS DB1.
TRACE FIND
   RECORD IS EMPLOYEE
   REALM IS REGION-3
   PRINT CURRENT FIELDS IN HEXADECIMAL DB-REGISTERS.

SCHEMA NAME IS DB2.
TRACE STORE
   RECORD IS EMPLOYEE
   PRINT CURRENT FIELDS IN HEXADECIMAL DB-REGISTERS.
```

7.5.12 CHECK PAGE Command

Function

To check the structure of the pages retrieved or modified by the DBCS.

Format

```
       [INPUT ]
CHECK  [      ]  PAGE STRUCTURE .
       [OUTPUT]
```

Rules

1. If this command is specified with the INPUT keyword, the complete structure of pages requested from Buffer Management is validated before use.

2. If this command is specified with the OUTPUT keyword, the complete structure of modified pages is validated before their release to Buffer Management.

3. If this command is specified with both INPUT and OUTPUT keywords or with neither, the complete structure of retrieved and modified pages is validated.

4. If this command is not specified, no validation occurs.

5. The page structure validation is the same as that performed by the VALIDATE PAGE command of DBUTILITY.

6. When inconsistency is detected in the page structure, an error message and the page contents are printed on the JOR and IDSTRACE file, if any, and the step (or transaction) is aborted.

7. This command is intended for the investigation of the cause of data base inconsistencies. It should not be used in normal conditions since it consumes processor time.

7.5.13 INTERNAL FILE NAME Command

Function

To redefine the internal-file-names to be used at run-time in the ASSIGN/DEFINE statements referencing the specified storage areas.

Format

```
                   {      {AREA }                     }
INTERNAL FILE NAME { FOR  {     }  realm-name IS ifn }... .
                   {      {REALM}                     }
```

Rules

1. "ifn" must conform to the rules for internal-file-names defined in the JCL Reference Manual.

   In addition, "ifn" must be different from:

   ```
   IDSOPT
   IDSTRACE
   PRTFILE
   COMFILE
   SLLIBa
   DDLIBb
   SAVEc
   ```

   where a,b,c are decimal digits or spaces.

2. If no "ifn" in given for a realm of a data base, the default implicitly or explicitly defined in the DMCL is used.

3. There must be no duplicates among the internal-file-names of a schema.

   If several data bases are accessed in the same step and if the internal-file-names are not unique among the different schemas, the user must ensure the uniqueness of each internal-file-name using the INTERNAL FILE NAME command for the relevant schemas.

7.6 INTERPRETATION OF JOR MESSAGES

7.6.1 Message Format

Except for the banners, statistics and syntax error messages, the DBCS formats its JOR messages as follows:

```
DDMx...PGID : y...SLNZ : z...SCH : w...ORG: {aa bbbb}...{ERR}:u...Text
                                                        {WNG}
```

DDM DDM	stands for Data Description and Manipulation. "x..." is a 5-digit decimal number. It is the key identifying the DBCS message text.
PGID	"y..." is a 30-character string containing the program identificaton as specified in the PROGRAM-ID clause. When this information is not available or is irrelevant, it is replaced by asterisks.
SLN	"z..." is a 7-digit decimal number identifying the source internal line number where the statement involved is coded When this information is not available or is irrelevant, it is replaced by 0.
SCH	"w..." is a 30-character string containing the schema name characterizing the data base on which the current DBCS function has been executed.
ORG	This field provides information on the DBCS module that originated the message. "aa" is a 2-digit decimal number which identifies a DBCS function. This function may be external (such as STORE) or internal (such as SELECT). The possible function codes are presented later in this section. "bbbb" is a 4-digit decimal number which identifies a point in the procedure code of the DBCS function. This identification depends on the function and type of error and can be used only in conjunction with the listing. It is therefore reserved for Field Engineering. The group aa bbbb may be repeated to indicate the stack of DBCS internal functions called when the error was detected and reported. The maximum depth is 6.
ERR/WNG ERR	indicates a fatal error, WNG indicates a warning, "u..." is a 4-digit decimal number. It is the code of the error. It is equal to the message key when one message corresponds to one error. It is equal to or greater than the message key when one message covers a whole category of errors. The error codes are given later in this section.

Text	This is the plain text reporting the error. Though it is made as explicit as possible, complementary information may be found in the Error Messages and Return Codes Pocket Guide.

NOTE: The DBCS messages described above for the JOR can also be sent with the same format to the IDSTRACE file, if any, and to the SYSOUT and TERMINAL reports of the data base utilities or placed in the DB-DETAILED-STATUS register of the DML programs (warnings only).

7.6.2 DBCS Function Codes

Table 7-1 shows the possible DBCS function codes appearing in JOR messages. Values are given in decimal. Several categories can be distinguished:

- Codes 00=>24	correspond to external DML statements
- Codes 25=>29	correspond to the prologs generated in COBOL programs.
- Codes 30=>34	correspond to the external or internal functions related to the initialization and termination of a session or commitment-unit.
- Codes 35=>44	correspond to external functions activated by the data base utilities.
- Codes 45=>49	correspond to internal functions appearing as "sub-functions" in the statistics of an IDS session.
- Codes 50=>79	correspond to other internal functions such as those calling UFAS/GAC (60 64) or the JOURNAL (75 79).
- Codes 80=>99	correspond to the compatibility layer that converts the 1C/1D generated code of DML statements into the 1E generated code.

Table 7-1. DBCS Function Codes in JOR Messages

Function	Code	Function	Code	Function	Code	Function	Code
	00	UCINIT0	25	CHECK	50	WAJN	75
ACCEPT	01	UCINIT2	26	REPORT	51	OPAJN	76
CONNECT	02	UCINIT	27	ABTERM	52	CLAJN	77
DISCONNECT	03		28	TRACE	53	NOTBJN	78
ERASE	04		29	PAGCHK	54	ROLLBJN	79
FIND	05	STARTDBS	30	GETRCALC	55		80
FINISH	06	TERMDBS	31	ECOSEM	56	IACCEPT	81
	07	ROLLBACK	32		57	ICONNEC	82
GET	08	RESTART	33		58	IDISCON	83
IF (DB Condition)	09	CLEANPOINT	34		59	IERASE	84
	10	GETPR	35	GETCI	60	IFIND	85
MODIFY	11	SWITCH	36	FREECI	61	IFINISH	86
	12	PUTPAGE	37	SETCI	62		87
READY	13	GETPAGE	38	LOKCI	63	IGET	88
	14	RELPAGE	39	CHCKCI	64	IIF	89
STORE	15	GETLAB	40	UNLCI	65		90
	16	MODLAB	41	INICMT	66	IMODIFY	91
	17	READCMT	42		67		92
	18	MODOPT	43		68	IREADY	93
	19	FREELOCK	44		69		94
	20	SELECT	45		70	ISTORE	95
	21	INSERT	46	PGREORG	71	IINIT1	96
	22	PLACEDIR	47		72	IINIT2	97
	23	PLACECALC	48		73		98
	24		49		74		99

7.6.3 DBCS Error Code Categories

Table 7-2 shows the possible error code categories. Values are given in decimal.

The first category corresponds to interface problems with the system functions UFAS/GAC/JOURNAL/TDS. It reports system return codes which the DBCS is unable to interpret or which prevent any further action.

The second category corresponds to the analysis of the run-time command language. It includes errors related to the syntax itself and errors due to the IDSOPT environment.

The third category corresponds to the errors which occur when the schema is loaded (incorrect ASSIGN, mismatch between schema names or dates, etc.) or when the areas are readied (incorrect ASSIGN, transient state, etc.). The fourth category corresponds to the tag and format consistency checks that the DBCS performs continuously when accessing its run-time control structures.

The next five categories correspond to the consistency checks that the DBCS performs continuously when putting together and interpreting the control structures prepared before execution: schema, program, data base.
As it is often difficult to determine which of the three elements is faulty, the classification of an error in one of the five categories may be arbitrary and not reflect the actual cause of the error.

The last category corresponds to warning codes.

Table 7-2. DBCS Error Code Categories in JOR Messages

Category	**Sub-category**	**Error Code**
UFAS/GAC/JOURNAL/TDS interface	UFAS/GAC	0 -<=256> 31
	JOURNAL	32 -<=256> 39
	TDS	40 -<=256> 47
	Others	48 -<=256> 63
Run-time command language analysis	Syntax analysis	64 -<=256> 351
	Others	352 -<=256> 383
Session or area initialization	Schema loading	384 -<=256> 415
	Area opening	416 -<=256> 447
	Others	448 -<=256> 511
System consistency checks	Control structure tags	512 -<=256> 535
	Control structure formats (release compatibility)	536 -<=256> 559
	TDS	560 -<=256> 575
	Others	576 -<=256> 767
Data base inconsistency		768 -<=256> 895
DML inconsistency		896 -<=256> 1023
Schema inconsistency		1024 -<=256> 1151
Schema/data base inconsistency		1536 -<=256> 1791
Schema/DML inconsistency		1792 -<=256> 1919
Warnings		3584 -<=256> 3839

7.7 INTERPRETATION OF STATISTICAL AND TIMING INFORMATION

1. Given the same DML program and the same initial state of the data base, some of the cumulated results of an IDS session must be interpreted in the context of the execution environment:

   - The number of calls to the DBCS functions is constant.

   - The number of internal calls to the UFAS/GAC functions is greater in MONITORED mode than in EXCLUSIVE or SHARED mode.

   - The number of I/0s may vary according to the buffer pool options and the number of buffers used.

   - Suppose that the program searches a sorted set occurrence that fits into four pages. If the number of buffers is at least 4, the four pages are loaded into main memory on the first reference and then the transfers between the pages and the UWA become main memory "move" operations, so long as the program keeps working on this set occurrence. If the number of buffers is less than 4, and the sorted order does not correspond to the data-base-key order, the program changes pages frequently, requiring an I/0 to replace a page by another in one of the buffers.

   - Furthermore, due to the buffer management strategy that defers the rewriting of modified pages, page transfers may be put down to the account of a statement that was not responsible for them. For example, a "FIND DB-KEY IS..." statement, which needs at most one transfer, may be attributed an extra transfer if it is first necessary to free a buffer by rewriting onto disk the modified page it contained.

     This is particularly true in TDS environment with DEFERRED UPDATES, where all the I/0s related to the rewriting of modified pages are performed by the COMMIT function.

   - In Batch/IOF environment, the figures reported in the charts do not include the I/Os of unsuccessful commitment-units, that is, those performed between the last commitment-point (or check-point) preceding an abort-point and this abort-point.

     In the case of a step restart after a deadlock or any other incident, the JOR UFAS and I/O CONNECT statistics correspond to the activity of the last restarted portion of the step, while the DBCS statistics correspond to the activity of the whole step (excluding the aborted phases, as mentioned above).

   - In TDS environment, the figures reported in the charts include the I/Os of unsuccessful commitment units.

- The elapsed times are meaningful in a monoprogramming environment without output writer. They may be influenced by the number of stack overflows if they occur too frequently. (See Section VI for the appropriate LINKER command.)

  Theoretically, the process times are not affected by the multiprogramming environment, unless contention for main memory results in a great number of missing segments.

  As with the I/O counts, the elapsed and process times are influenced by the buffer pool chosen and the number of buffers used. They are also influenced by the sharing mode and the presence or absence of journals.

  The figures reported in the charts do not include the time spent by the external trace function at the end of a statement. If the internal trace is triggered, the time necessary for the output of SYSTEM calls information is incorporated in the elapsed and process times, which makes them meaningless.

  The figures reported in the charts include the time spent by the CHECK PAGE function when the latter is activated in the run time options.

  In TDS environment, the times reported for the TDS session involve all the transactions of the session:

- The total process time represents the sum of the process times of the transactions. There is no overlapping between these process times.

- The total elapsed time also represents the sum of the elapsed times of the transactions. As these transactions run in parallel during the TDS session, the total sum may far exceed the elapsed time of the TDS session itself.

2. In order to obtain averages as meaningful as possible, DML statements have been divided into a number of categories greater than the number of DML verbs. For instance, a "FIND DB-KEY IS..." statement, which requires at most one transfer, is not placed in the same category as a "FIND record-name WITHIN set-name USING ..." statement, which requires the traverse of the set selection path and the search of the selected set occurrence.

   Nevertheless this categorization on the basis of the external format does not give complete precision. For example, a "FIND FIRST WITHIN realm-name" statement will take 40 ms to retrieve a record in the first page of a full area and perhaps 40s to realize that another area is empty. Similarly a STORE statement will take 40 ms to store an owner-only CALC record and perhaps 15 seconds to store a record which is an AUTOMATIC member of 10 sorted sets. It is clear that, in those cases, the average of the I/0 count and elapsed and process times within the category is meaningless.

3. The previous considerations apply to the statistics and timing information displayed on the JOR at the end of an IDS session, as well as to the statistics and timing information displayed on IDSTRACE at the end of a traced DML statement.

A. DDL, DMCL, DML Syntax

SCHEMA DDL SKELETON					
Entry		**Subentry of entry**		**Clause of entry or subentry**	
Number	**Name**	**Number**	**Name**	**Number**	**Name**
	SCHEMA	-	-	1	SCHEMA
1 to 2048	AREA	-	-	1	AREA
1 to 2048	RECORD	1	RECORD	1	RECORD
				1	LOCATION
				1	WITHIN
				0 to 240	CHECK
		0 to 2048	DATA	1	DATA-BASE-DATA-NAME
				0 or 1	TYPE
				0 or 1	OCCURS
				0 or 1	CHECK
0 to 2048	SET	1		1	SET
				1	OWNER
				1	ORDER
		1 to 204	MEMBER	1	MEMBER
				1	INSERTION
				0 to 240	DUPLICATES
				0 or 1	KEY
				1	SELECTION

```
 ______________________________
|                              |
|  Schema DDL - Schema Entry   |
|______________________________|
```

```
 ______________________________
|                              |
| SCHEMA NAME IS schema-name.  |
|______________________________|
```

```
 ______________________________
|                              |
| Schema DDL - Area Entry      |
|______________________________|
```

```
 ______________________________
|                              |
| AREA NAME IS area-name.      |
|______________________________|
                 Schema DDL - Record Subentry

RECORD NAME IS record-name-1

                  {DIRECT data-base-parameter-1              }
                  {                                          }
LOCATION MODE IS  {CALC USING {data-base-identifier-1}...    }
                  {     DUPLICATES ARE [NOT] ALLOWED         }
                  {VIA set-name-1 SET                        }

         {{ANY} AREA        }                                     }
WITHIN   {{area-name-1}  ...  } [AREA-ID IS data-base-parameter-2] }
         {AREA OF OWNER                                            }
[CHECK IS condition]... .

              Schema DDL - Data Subentry


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

[        {{              {UNPACKED}  }                                  } ]
[        {{[UNSIGNED]    {[PACKED]}  }                                  } ]
[        {{              {PACKED-2}  } DECIMAL integer-1 [ , integer-2 ]} ]
[        {{                          }                                  } ]
[        {{  SIGNED      {UNPACKED}  }                                  } ]
[        {{              {[PACKED]1}  }                                  } ]
[TYPE IS {                                                              } ]
[        {                                                              } ]
[        { [SIGNED]      BINARY  {15}                                   } ]
[        {                       {31}                                   } ]
[        {  CHARACTER integer-3                                         } ]

 [OCCURS integer-4 TIMES]

 [CHECK IS VALUE [NOT] {literal-1 [THRU literal-2]...] .

                    Schema DDL - Set Subentry


SET NAME IS set-name-1

OWNER IS record-name-1


ORDER IS PERMANENT INSERTION IS

   {FIRST                                                          }
   {                                                               }
   {LAST                                                           }
   {                                                               }
   {NEXT                                                           }
   {                                                               }
   {PRIOR                                                          }
   {                                                               }
   {         { WITHIN RECORD-TYPE                            }     }  .
   {         {                                               }     }
   {         { BY DEFINED KEYS                               }     }
   {         {                                               }     }
   {SORTED   {   [RECORD-TYPE SEQUENCE IS {record-name-2]...]}     }
   {         {                                               }     }
   {         {                 {FIRST      }                 }     }
   {         {    DUPLICATES ARE {LAST       }                 }     }
   {         {                 {NOT ALLOWED}                 }     }
 _________________________________
|                                 |
| Schema DDL - Member Subentry    |
|_________________________________|

               Schema DDL - Member Subentry

 MEMBER IS record-name-1

 {INSERTION IS AUTOMATIC RETENTION IS MANDATORY}
 {                                             }
 {INSERTION IS MANUAL RETENTION IS OPTIONAL    }

 [DUPLICATES ARE NOT ALLOWED FOR {data-base-identifier}...]...

[       {ASCENDING }         {data-base-identifier-2}        ]
[KEY IS {          }         {RECORD-TYPE           }        ]
[       {DESCENDING}         {DATA-BASE-KEY         }        ]
[                                                            ]
[     [ [ASCENDING ]         {data-base-identifier-3} ]      ]
[     [ [DESCENDING]         {RECORD-TYPE           } ] ...  ]
[     [                      {DATA-BASE-KEY         } ]      ]
[
[                            {FIRST      } ]                 ]
[       [DUPLICATES ARE      {LAST       } ]                 ]
[                            {NOT ALLOWED} ]                 ]


SET SELECTION [FOR   set-name-1] IS

   THRU set-name-2 OWNER IDENTIFIED BY

   { APPLICATION                                            }
   {
   { DATA-BASE-KEY [EQUAL TO data-base-parameter-1]         }
   {
   {CALC-KEY [ data-base-identifier-4 EQUAL TO              }
   {
   {           {data-base-identifier-5} ]                   }
   {           {                      } ] ...               }
   {           {data-base-parameter-2 } ]                   }
   {                                                        }
   {        [AREA-ID EQUAL TO data-base-parameter-3]        }

 [THEN THRU set-name-3 WHERE OWNER IDENTIFIED BY               ]
 [                                                             ]
 [   {data-base-identifier-6 [EQUAL TO                         ]
 [             {data-base-identifier-7} ] }                    ]
 [             {                      } ] }  ...               ]
 [             {data-base-parameter-4 } ] }                    ] ... .
```

DMCL SKELETON

Entry		**Clause of entry**	
Number	**Name**	**Number**	**Name**
1	SCHEMA	1	SCHEMA
		0 or 1	AREAS
		0 or 1	GLOBAL
		0 or 1	NO LOCAL
		0 or 1	POOL
		0 or 1	BUFFERS
1 to 2048	AREA	1	AREA
		0 or 1	AREA FILE
		1	NUMBER-OF-PAGES
		1	LINES-PER-PAGE
		1	PAGE-SIZE
		0 or 1	CALC-INTERVAL
		0 or 1	LOCAL
0 to 2048	RECORD	1	RECORD
		0 to 2048	RANGE
		0 or 1	MIGRATION
0 to 2048	SET	1	SET
		0 or 1	NO LOCAL

```
 ______________________
|                      |
| DMCL - Schema Entry  |
|______________________|
```

```
 ___________________________________________________
|                                                   |
| SCHEMA NAME IS schema-name                        |
|                                                   |
|      [EXTEND NUMBER OF AREAS TO integer]          |
|                                                   |
|     [EXTEND GLOBAL POINTERS TO integer BYTES]     |
|                                                   |
|     [NO LOCAL POINTERS]                           |
|                                                   |
|     [BUFFER POOL NAME IS buffer-pool-name]        |
|                                                   |
|     [NUMBER OF BUFFERS IS integer].               |
|___________________________________________________|
                          DMCL - Area Entry

AREA NAME IS area-name

   [AREA INTERNAL FILE NAME IS ifn]
                              [            {HIGH} ]
   NUMBER-OF-PAGES IS integer [ OPTIMIZE   {    } ]
                              [            {LOW } ]

   NUMBER OF LINES-PER-PAGE IS integer

   PAGE-SIZE IS integer BYTES

  [CALC-INTERVAL IS integer PAGES]

  [ EXTEND LOCAL POINTERS TO integer BYTES]   .

                        DMCL - Record Entry

RECORD NAME IS record-name

[                             {integer PAGES FROM PAGE integer} ]
[ RANGE [WITHIN area-name] IS {                               } ]
[                             {PAGE integer THRU integer      } ]
[                                                               ]
[                   [           {HIGH} ]                        ]
[                   [OPTIMIZE   {    } ]                        ]
[                   [           {LOW } ]                        ]

  [MIGRATION IS ALLOWED]   .

  ____________________
 |                    |
 | DMCL - Set Entry   |
 |____________________|

  ___________________________
 |                           |
 | SET NAME IS set-name      |
 |                           |
 |     [NO LOCAL POINTERS] . |
 |___________________________|
                  DML in DATA DIVISION

DATA DIVISION .

SUB-SCHEMA SECTION .

DB schema-name .

[                    {WORKING-STORAGE     }               ]
[ DB-DESCRIPTIONS IN {                    } SECTION .     ]
[                    {LINKAGE             }               ]

[ {            {ALL                     }   }    ]
[ {RECORDS ARE {                        }   }    ]
[ {            {[NOT] {record-name} ... }   }    ]
[ {                                         }    ]
[ {            {ALL                     }   }  . ]
[ {REALMS ARE  {                        }   }    ]
[ {            {[NOT] {realm-name} ...  }   }    ]
   ...
WORKING-STORAGE SECTION .
 ...
level number identifier USAGE IS DB-KEY .
 ...
LINKAGE SECTION .
 ...
level number identifier USAGE IS DB-KEY .
 ...

 _____________________________
|                             |
| DML in PROCEDURE DIVISION   |
|                             |
|     (Declarative Part)      |
|_____________________________|
 _____________________________________________________________
|                                                             |
|  USE FOR DB-EXCEPTION . (first USE section in DECLARATIVES) |
|_____________________________________________________________|
```

DML in PROCEDURE DIVISION

(Non-Declarative Part)

```
                         [realm-name ]
ACCEPT ident-1 FROM      [record-name]   CURRENCY
        (DB-KEY)         [set-name   ]

                          [record-name]
ACCEPT ident-2 FROM       [set-name   ]   REALM-NAME .
        (PIC X(30))       [ident-3    ]
                          [(DB-KEY)   ]

                                  {NEXT }
ACCEPT ident-4 FROM set-name      {PRIOR}
        (DB-KEY)                  {OWNER}

ACCEPT ident-5 FROM realm-name LINES-PER-PAGE .
        (COMP-1)

ACCEPT ident-6 FROM realm-name MINIMUM-DB-KEY
        (DB-KEY)            [OF record-name] .

ACCEPT ident-7 FROM realm-name NUMBER-OF-PAGES
        (COMP-2)            [OF record-name] .


CONNECT [record-name] TO set-name [RETAINING-phrase] .


DISCONNECT [record-name] FROM set-name .


ERASE [record-name] [ALL MEMBERS] .
```

DML in PROCEDURE DIVISION

(Non-Declarative Part)

```
FIND [record-name] DB-KEY IS identifier [RETAINING-phrase] .
                             (DB-KEY)

      {ANY      }
FIND  {         }  record-name [RETAINING-phrase] .
      {DUPLICATE}

      {NEXT      }
      {PRIOR     }
      {FIRST     }                      {realm-name}
FIND  {LAST      } [record-name] WITHIN {          }
      {integer   }                      {set-name  }
      {identifier} [RETAINING-phrase] .
      {(COMP-2)  }

                           [        {realm-name} ]
FIND CURRENT [record-name] [ WITHIN {          } ]
                           [        {set name  } ]
                 [RETAINING-phrase] .

FIND OWNER WITHIN set-name [RETAINING-phrase] .

FIND record-name WITHIN set-name [CURRENT] [USING {identifier} ... ]
                                           [RETAINING-phrase] .

FIND DUPLICATE WITHIN set-name USING identifier ...
                 [RETAINING-phrase] .
```

```
FINISH [realm-name] ... .

GET [record-name] .
GET {identifier} ... .

                 {OWNER }
[NOT] [set-name] {MEMBER}
                 {TENANT}
set-name IS [NOT] EMPTY

       {[record-name   ]}
MODIFY {                } [RETAINING-phrase] .
       {{identifier} ...}

                            {ALL           }
MODIFY [record-name] ONLY   {              } MEMBERSHIP
                            {{set-name} ...}
                                      [RETAINING-phrase] .

       {[record-name]   }           {ALL           }
MODIFY {                } INCLUDING {              } MEMBERSHIP
       {{identifier} ...}           {{set-name} ...}
                                        [RETAINING-phrase] .


READY {[realm-name] ... USAGE-MODE IS    }
      {                                  }
      { {EXCLUSIVE       {RETRIEVAL} }   }
      { {                {UPDATE   } }   }
      { {SHARED           RETRIEVAL  }   } ... .
      { {                {RETRIEVAL} }   }
      { {[MONITORED]     {         } }   }
      { {                {UPDATE   } }   }

STORE record-name [RETAINING-phrase] .

                                      {MULTIPLE  }         }
RETAINING-phrase:                     {REALM     }         }
                  RETAINING CURRENCY FOR {RECORD    }         }
                                      {{SETS     }    }    }
                                      {{{set-name} ...}    }
```

B. Differences Between GCOS64 Release 1E and GCOS 7

This appendix summarizes the differences between the two software releases. Differences between releases 1D and GCOS7 can be obtained by concatenation of the differences between releases 1D and 1E and the current appendix contents.

The differences specific to each IDS/II component are presented in the same order as the sections which appear in the manual.

B.1 NEW FEATURES IN PREALLOC

- The TRACK option in the SPLIT parameter enables the user to specify the starting disk address of each extent in terms of cylinder and track numbers.

Compatibility

1. There is no change in the internal format of the data base. 1E areas are accepted under GCOS7 without conversion.

2. Areas preallocated under GCOS7 may be used under 1E.

B.2 NEW FEATURES IN COBOL/DML COMPILING AND LINKING

1. COBOL programs containing IDS/II DML statements are compiled only by the COBOL compiler. The DMLPROC preprocessor is thus made obsolete.

   In order for the COBOL compiler to retrieve the data base Schema, new keywords (DDLIB1, DDLIB2, DDLIB3) have been introduced in the COBOL extended JCL statement.

2. The use of the new option DDLIST when compiling a COBOL/DML program, provides the user with a listing of the description of all records used.

3. DML statements are no longer restricted to the main text of the program. They may be the subject of COPY statements.

4. READY and FINISH statements may appear in any COBOL/DML program, i.e. in secondary programs (DB-DESCRIPTIONS IN LINKAGE) as well as main programs (DB-DESCRIPTIONS IN WORKING-STORAGE). There remain no restrictions on which type of COBOL/DML program (main or secondary) is executing the first READY or the last FINISH function.

5. The "IF" DML statement becomes a "Data Base Condition" and as such may be used anywhere a simple condition may appear.

6. Some automatic conversions, as far as data type and data length are concerned, are provided whenever possible. For example, in the statement

   ```
   ACCEPT identifier FROM realm-name LINES-PER-PAGE
   ```

   the identifier no longer needs to be declared as COMP-1.

7. The cross-references produced by the COBOL compiler provide more detailed and precise information on DML usage.

8. When linking programs containing DML statements, the VACSEG command is no longer necessary since the LINKER considers all unaffected segments as vacant.

   To avoid an abort with the message VMM ENTRYOV while executing a module linked under release 1E, the user may relink it under GCOS7 without using the VACSEG command.

Compatibility

1. Since the DMLPROC preprocessor is obsolete, JCL compiling COBOL/DML programs has to be modified:

   - to suppress the DMLPROC step,

   - to include new COBOL keywords.

2. Load-modules linked under release 1E are executable unchanged under GCOS7.

3. Compiled programs produced by DMLPROC and COBOL under release 1E may be linked with COBOL/DML programs compiled under GCOS7 by the COBOL compiler.

4. When compiling a COBOL/DML program, the COBOL compiler stops its search for the schema object as soon as it finds a member with the required name (i.e. the schema name) in the search path (DDLIB1, DDLIB2, DDLIB3).
   It is therefore advisable that the search path does not contain binary objects with the same name as the schema which are not schema objects.

5. Some syntactical and semantic rules are enforced:

   - IDS/II statements should not be written in Area A,

   - usage DB-KEY will be checked when it is compulsory; for example, in the statement "ACCEPT identifier FROM ... CURRENCY",

   - permissible attributes on data with DB-KEY usage will be checked,

   - IDS/II tables and registers will not be writable,

   - DB-CXT, DB-PARAMETERS and DB-REGISTERS can only be explicitly referenced in:

   ```
   . PROCEDURE DIVISION USING...
   . CALL... USING...
   ```

6. Implicit contextual qualification of field-names is not permitted.

7. All records referred to in a COBOL/DML program must be listed, explicitly or implicitly, in the SUB-SCHEMA section.

8. If neither RECORDS nor REALMS is specified in the SUB-SCHEMA section, all the record-types of the data base are made available to the program.

B.3 NEW FEATURES IN USER PROGRAM EXECUTION

1. Secondary COBOL/DML program (i.e. with DB-DESCRIPTIONS IN LINKAGE) which are compiled under GCOS7 may be objects of the "USE procedure-name" clauses in the TDS generation. If so, they need not be linked to the TPRs chich call them.

2. The program identification, used in IDSOPT and IDSTRACE, is no longer limited to 12 characters.

3. JOR messages contain the identification of the schema being used in order to be more precise when executing in a multi-data base environment.

4. The output line length of IDSTRACE has been limited to 120 characters, to avoid truncation under TDS.

5. When STARTDBS (see TRACE clause in section 7) is triggered by a secondary program, the corresponding trace refers to the main program as PGID and to an undefined line (LINE:***).

6. A new command (INTERNAL FILE NAME...) has been introduced as a run-time option. This enables the user to redefine internal-file-names which are used in the ASSIGN/DEFINE statements to reference specified storage areas. Thus duplication of internal-file-names in a multi-data-base environment is avoided.

Compatibility

- The line numbers used in run-time options and TRACE editions are no longer the source external line numbers; they refer to the COBOL generated internal line numbers.

Technical publication remarks form

Title :	DPS7000/XTA NOVASCALE 7000 IDS/II Reference Manual Database Products: IDS-II

Reference N° :	47 A2 11UD 00	**Date:**	**February 1984**

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please include your complete mailing address below.

NAME : ______________________ Date : __________

COMPANY : ______________________

ADDRESS : ______________________

____

Please give this technical publication remarks form to your BULL representative or mail to:

Bull - Documentation D$^{ept.}$
1 Rue de Provence
BP 208
38432 ECHIROLLES CEDEX
FRANCE
info@frec.bull.fr

Technical publications ordering form

To order additional publications, please fill in a copy of this form and send it via mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone:	+33 (0) 2 41 73 72 66
FAX:	+33 (0) 2 41 73 70 66
E-Mail:	srv.Duplicopy@bull.net

CEDOC Reference #	**Designation**	**Qty**
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
[_ _] : The latest revision will be provided if no revision number is given.		

NAME: ______________________________ Date: __________

COMPANY: ______________________________

ADDRESS: ______________________________

____

PHONE: ____________________ FAX: ____________________

E-MAIL: ______________________________

For Bull Subsidiaries:
Identification: ______________________________

For Bull Affiliated Customers:
Customer Code: ______________________________

For Bull Internal Customers:
Budgetary Section: ______________________________

For Others: Please ask your Bull representative.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 11UD 00