

English



FUJITSU Software BS2000

UDS/SQL V2.8

Application Programming

User Guide

Edition March 2016

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH

www.cognitas.de

Copyright and Trademarks

Copyright © 2016 Fujitsu Technology Solutions GmbH.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Contents

1	Preface	11
1.1	Structure of the UDS/SQL documentation	11
1.2	Objectives and target groups of this manual	16
1.3	Summary of contents	17
1.4	Changes since the last edition of the manuals	18
1.5	Notational conventions	20
1.5.1	Warnings and notes	20
1.5.2	Non-SDF notational conventions	20
1.5.3	SDF syntax representation	22
1.6	Sample database	27
2	Overview	29
2.1	The language concept of DML	29
2.2	The range of functions for DML	30
3	Transaction concept	31
3.1	The transaction in multi-DB operation	32
3.2	The transaction in mono-DB operation	33
3.3	Rollback	33
3.4	Page protection	34

4	Currency table	37
<hr/>		
5	DML functions	41
<hr/>		
5.1	Opening and closing transactions	43
5.1.1	Opening a transaction or processing chain (READY)	43
5.1.2	Closing a transaction (FINISH)	45
5.2	Retrieving data	46
5.2.1	Direct access at record type level	48
5.2.1.1	Direct access via the database key (FIND/FETCH-1)	48
5.2.1.2	Direct access via the CALC key (FIND/FETCH-2)	49
5.2.1.3	Direct access via any items (FIND/FETCH-3/7)	50
5.2.2	Sequential access at record type level (FIND/FETCH-4)	55
5.2.3	Access to the CRR (FIND/FETCH-5)	55
5.2.4	Direct access at set level (FIND/FETCH-3/7)	56
5.2.5	Sequential access at set level (FIND/FETCH-4)	59
5.2.6	Access to the CRS (FIND/FETCH-5)	60
5.2.7	Access to the owner of a CRS (FIND/FETCH-6)	60
5.2.8	Sequential access at realm level (FIND/FETCH-4)	61
5.2.9	Access to the CRA (FIND/FETCH-5)	61
5.2.10	Transport the CRU completely or partially into the UWA (GET)	62
5.2.11	Retrieve database key values (ACCEPT-1)	63
5.2.12	Retrieve realm (ACCEPT-2)	64
5.3	Modifying data	65
5.3.1	Store a record in the database and connect it into set occurrences (STORE)	65
5.3.2	Connect a record into a set occurrence (CONNECT)	66
5.3.3	Disconnect existing set relationships (DISCONNECT)	67
5.3.4	Modify the CRU or connect it into another set occurrence (MODIFY)	68
5.3.5	Delete records and their set relationships (ERASE)	69
5.3.6	Correlation between type of set membership and data-modifying statements	70
5.4	Protecting records	71
5.4.1	Activate extended record protection (KEEP)	71
5.4.2	Deactivate extended record protection (FREE)	72
5.5	Testing set memberships in the program (IF)	73
5.5.1	Testing the set membership of the CRU	73
5.5.2	Testing a set occurrence for member records	74

6	Using DML	75
6.1	Structure of the COBOL/CALL DML programs	76
6.1.1	COBOL DML	77
6.1.2	CALL DML	83
6.2	Special features of COBOL DML	86
	Specifying a key for use of the subschema (PRIVACY)	86
	Assigning the subschema and setting up the communication area (DB entry)	86
	COBOL special registers	88
	Transferring a database key value (SET)	88
	Describing error handling routines (USE)	89
	Assigning the COSSD file for compiling a COBOL-DML program	90
6.3	Special features of CALL DML	92
	Testing the structure of the subschema (LOOKC)	94
6.4	Linking, loading and starting a UDS/SQL-TIAM application program	95
6.4.1	Basic aspects	95
	The principle of dynamic loading	95
	Loading the UDS/SQL product modules dynamically	95
	Loading the application-specific data modules SSITAB and PLITAB dynamically	98
	Loading the configuration-specific table module UDSTRTAB dynamically	99
6.4.2	Linking UDS/SQL-TIAM applications	100
6.4.3	Starting a COBOL program	103
6.5	Interoperation in a UDS/SQL-openUTM application	107
	Generating a UDS/SQL-openUTM application	107
	Linking UDS/SQL-openUTM applications	109
	Starting a UDS/SQL-openUTM application	112
	Error codes	113
6.6	Error handling	114
6.6.1	Interrupt handling for UDS/SQL-TIAM applications	114
6.6.2	Database exception conditions	117
	Using special registers (COBOL DML) or the user information area (CALL DML)	117
	Statement codes	118
	Status codes	119
	Combinations of statement codes and status codes	120
	FIND/FETCH status codes	122
6.6.3	CALL DML error handling routine DSCEXT	124
6.7	Translation table for application-specific sorting	126

7	COBOL DML reference section	129
7.1	General rules	130
7.2	ID DIVISION	131
7.3	DATA DIVISION	131
	SUB-SCHEMA SECTION	131
	DB entry	132
7.4	PROCEDURE DIVISION	133
7.4.1	Overview of COBOL DML statements	133
7.4.2	COBOL DML statements	139
	ACCEPT	139
	CONNECT	143
	DISCONNECT	144
	ERASE	146
	FIND/FETCH	148
	FINISH	179
	FREE	179
	GET	180
	IF	181
	KEEP	182
	MODIFY	183
	READY	186
	SET	187
	STORE	189
	USE	194
8	CALL DML reference section	197
8.1	CALL interface	197
8.2	Parameter definitions	198
8.2.1	Rules	199
8.2.2	Format table	201
8.2.3	Format of secondary option (SOPT)	203
8.2.4	Format of user information (UINF)	205
8.2.5	Format of special parameter 1 (SPP1)	209
8.2.6	Format of special parameter 2 (SPP2)	210
8.3	CALL DML calls	211
8.3.1	Overview of the CALL DML functions	211
8.3.2	Functions of CALL DML	226
	Saving currency information (ACCPTC, ACCPTL)	227

	Establishing set connections (CONNEC)	229
	Releasing existing set connections (DISCON)	230
	Deleting records and their set connections (ERASEC)	231
	Retrieval of data (FIND/FTCH)	232
	Concluding processing (FINISC)	247
	Deactivating extended record protection (FREEC)	247
	Transporting a record to the record area (GETC)	248
	Testing database conditions (IFC)	250
	Activating extended record protection (KEEPC)	251
	Modifying records already stored (MODIF1/2)	251
	Preparing for processing (READYC)	254
	Storing records (STORE1/2, STOR1L/2L)	256
8.4	CALL DML Assembler macros	259
	DSCAL	261
	DSCAP	262
	DSCDF	263
	DSCPA	265
8.5	LOOKC function	266
8.5.1	The LOOKC block	268
8.5.2	Description of LOOKC parameters	275
8.5.3	LOOKC tables	278
8.6	Examples using different programming languages	289
9	Testing DML functions using DMLTEST	301
9.1	Introduction	302
9.2	DMLTEST commands	305
	Overview of the DMLTEST commands and general rules	305
	ADD	315
	CONTINUE	316
	DBH	316
	DECLARE	316
	DEFINE	317
	DELETE	317
	DISPLAY	318
	DISPOFF	318
	DO	319
	EDT	321
	END	321
	ESCAPE	321

	EXECUTE	322
	HALT	323
	HELP	323
	LANGUAGE	324
	LEAVE	325
	LIST	326
	MOVE	327
	NEXT	329
	PERFORM	330
	PRINT	331
	PROC	331
	PROFF	332
	PROT	332
	REMARK	332
	RUN	333
	SET	334
	SHOW	336
	SUBSCHEMA	337
	SYSTEM	337
	TRACE	338
	WAIT	339
9.3	The DML statements of DMLTEST	340
9.3.1	Overview of differences between DMLTEST DML and COBOL DML statements . .	340
9.3.2	The DML statements	342
9.4	DMLTEST program execution	359
9.4.1	Interrupts	359
9.4.2	Communication with one or more databases	362
9.5	Error messages	363
10	Appendix	365
<hr/>		
10.1	Status codes	365
	DML status codes	365
	CALL DML status codes	378
10.2	Description of the "MAIL-ORDERS" schema for the sample database	
	SHIPPING	383
10.3	UDS/SQL-openUTM return codes	390
10.4	Additional diagnostic information in openUTM	394

Glossary 401

Abbreviations 443

Related publications 447

Index 453

1 Preface

The **Universal Database System UDS/SQL** is a high-performance database system based on the structural concept of CODASYL. Its capabilities, however, go far beyond those of CODASYL as it also offers the features of the relational model. Both models can be used in coexistence with each other on the same data resources.

COBOL DML, CALL DML and (ISO standard) SQL are available for querying and updating data. COBOL DML statements are integrated in the COBOL language; SQL statements can be used in DRIVE programs or via an ODBC interface.

To ensure confidentiality, integrity and availability, UDS/SQL provides effective but flexible protection mechanisms that control access to the database. These mechanisms are compatible with the openUTM transaction monitor.

The data security concept provided by UDS/SQL effectively protects data against corruption and loss. This concept combines UDS/SQL-specific mechanisms such as logging updated information with BS2000 functions such as DRV (Dual Recording by Volume).

If the add-on product UDS-D is used, it is also possible to process data resources in BS2000 computer networks. UDS/SQL ensures that the data remains consistent throughout the network. Distributed transaction processing in both BS2000 computer networks and networks of BS2000 and other operating systems can be implemented using UDS/SQL together with openUTM-D or openUTM (Unix/Linux/Windows). UDS/SQL can also be used as the database in client-server solutions via ODBC servers.

The architecture of UDS/SQL (e.g. multitasking, multithreading, DB cache) and its structuring flexibility provide a very high level of throughput.

1.1 Structure of the UDS/SQL documentation

The “Guide through the manuals” section explains which manuals and which parts of the manuals contain the information required by the user. A glossary gives brief definitions of the technical terms used in the text.

In addition to using the table of contents, users can find answers to their queries either via the index or by referring to the running headers.

Guide through the manuals

The UDS/SQL database system is documented in five manuals:

- UDS/SQL Design and Definition
- UDS/SQL Application Programming
- UDS/SQL Creation and Restructuring
- UDS/SQL Database Operation
- UDS/SQL Recovery, Information and Reorganization

Further manuals describing additional UDS/SQL products and functions are listed on [page 15](#).

For a basic introduction the user should refer to chapters 2 and 3 of the “[Design and Definition](#)” manual; these chapters describe

- reasons for using databases
- the CODASYL database model
- the relational database model with regard to SQL
- the difference between the models
- the coexistence of the two database models in a UDS/SQL database
- the characteristic features of UDS/SQL

How the manuals are used depends on the user’s previous knowledge and tasks. [Table 1](#) serves as a guide to help users find their way through the manuals.

Examples

A user whose task it is to write COBOL DML programs should look up the column “COBOL/CALL DML Programming” under “User task” in the second line of [table 1](#). There, the following chapters of the “[Design and Definition](#)” manual are recommended:

General information	B = Basic information
Schema DDL	D = Detailed information
SSL	D = Detailed information
Subschema DDL	L = Learning the functions

In the same column the user can also see which chapters of the other manual are of use.

Database administrators who are in charge of database administration and operation will find the appropriate information under the column “Administration and Operation”.

Contents of the five main manuals	User task							
	Design and definition	COBOL/ CALL DML programming	SQL programming	Creation and re-structuring	Administration and operation	Working with openUTM	Working with IQS	Working with UDS-D

Manual UDS/SQL Design and Definition

Preface	B	–	–	–	–	B	B	–
General information	B	B	B	B	B	B	–	–
Designing the database	B	–	–	–	–	–	–	–
Schema DDL	L	D	–	L	L	–	–	–
SSL	L	D	–	L	L	–	–	–
Subschema DDL	L	L	–	L	L	–	–	–
Relational schema	L	–	D	–	–	–	–	–
Structure of pages	D	–	–	D	D	–	–	–
Structure of records and tables	D	–	–	D	D	–	–	–
Reference section	S	–	–	S	–	–	–	–

Manual UDS/SQL Application Programming

Preface	–	B	–	–	–	B	B	–
Overview	–	B	–	–	–	–	–	–
Transaction concept	–	L	–	L	L	D	D	–
Currency table	–	L	–	L	L	–	–	–
DML functions	D	L	–	L	–	–	–	–
Using DML	–	L	–	D	–	–	–	–
COBOL DML reference section	–	L	–	–	–	–	–	–
CALL DML reference section	–	L	–	–	–	–	–	–
Testing DML functions using DMLTEST	–	L	–	–	–	–	–	–

Table 1: Guide through the manuals

(part 1 of 3)

Contents of the five main manuals	User task							
	Design and definition	COBOL/ CALL DML programming	SQL programming	Creation and restructuring	Administration and operation	Working with openUTM	Working with IQS	Working with UDS-D

Manual UDS/SQL Creation and Restructuring

Preface	–	–	–	B	–	B	B	–
Overview	–	–	–	B	B	–	–	–
Database creation	–	–	–	L	–	–	–	–
Defining access rights	–	–	–	L	–	–	–	–
Storing and unloading data	D	–	–	L	–	D	–	–
Restructuring the database	D	–	–	L	–	–	–	–
Renaming database objects	D	–	–	L	–	–	–	–
Database conversion	D	–	–	L	–	–	–	–
Database conversion using BTRANS24	–	–	–	D	–	–	–	–

Manual UDS/SQL Database Operation

Preface	–	–	–	–	B	B	B	–
The database handler	–	–	–	–	L	–	–	D
DBH load parameters	–	–	–	–	L	–	–	D
Administration	–	–	–	–	L	–	–	D
High availability	–	–	–	–	B	–	–	–
Resource extension and reorganisation during live operation	D	–	–	–	B	–	–	–
Saving and recovering a database in the event of errors	D	–	–	D	L	D	–	D
Optimizing performance	–	–	–	–	D	–	–	D
Using BS2000 functionality	–	–	–	–	D	–	–	–
The SQL conversation	–	–	–	–	L	–	–	–
UDSMON	–	–	–	–	D	–	–	–
General functions of the utility routines	–	–	–	–	D	–	–	–
Using IQS	–	–	–	L	D	–	D	–
Using UDS-D	D	D	–	D	D	D	–	D
Function codes of DML statements	–	D	–	–	D	–	–	–

Table 1: Guide through the manuals

(part 2 of 3)

Contents of the five main manuals	User task							
	Design and definition	COBOL/ CALL DML programming	SQL programming	Creation and re-structuring	Administration and operation	Working with openUTM	Working with IQS	Working with UDS-D

Manual**UDS/SQL Recovery, Information and Reorganization**

Preface	–	–	–	–	B	B	B	–
Updating and reconstructing a database	D	–	–	D	L	D	–	–
Checking the consistency of a database	–	–	–	–	L	–	–	–
Output of database information	D	–	–	D	L	–	–	–
Executing online services	D	–	–	D	L	–	–	–
Database reorganization	D	–	–	D	L	–	–	–
Controlling the reuse of deallocated database keys	D	–	–	D	L	–	–	–

Additional Manuals

UDS/SQL Messages	D	D	D	D	D	D	D	D
UDS/SQL System Reference Guide	S	S	–	S	S	S	S	S
IQS	–	–	–	D	D	–	L	–
ADILOS	–	–	–	–	D	–	L	–
KDBS	–	L ¹	–	D	–	–	–	–
SQL for UDS/SQL Language Reference Manual	–	–	D	–	D	–	–	–

Table 1: Guide through the manuals

(part 3 of 3)

¹ only for COBOL-DML

- B provides basic information for users with no experience of UDS/SQL
- L helps the user learn functions
- D provides detailed information
- S provides a reference to syntax rules for practical work with UDS/SQL

Additional notes on the manuals

References to other manuals appear in abbreviated form. For example:

(see the “Application Programming” manual, CONNECT)

advises the user to look up CONNECT in the “Application Programming” manual.

The complete titles of the manuals can be found under “Related publications“ at the back of the manual.

UDS/SQL Messages

This manual contains all messages output by UDS/SQL. The messages are sorted in ascending numerical order, or in alphabetical order for some utility routines.

UDS/SQL System Reference Guide

The UDS/SQL System Reference Guide gives an overview of the UDS/SQL functions and formats.

**SQL for UDS/SQL
Language Reference Manual**

This manual describes the SQL DML language elements of UDS/SQL.

In addition to UDS/SQL-specific extensions, the language elements described include dynamic SQL as an essential extension of the SQL standard.

1.2 Objectives and target groups of this manual

The manual is directed at the programmer of database applications whose job it is to convert problem definitions into statement sequences to be executed on UDS/SQL databases. The resulting program must also take into account the structure of the database and, if necessary, influence this structure. The subschema required for the application must contain all the necessary information.

1.3 Summary of contents

What does this manual contain?

In its introduction, this manual describes the language concept and range of functions of the Data Manipulation Language DML.

The chapters which follow describe the transaction concept and the functioning of the currency table.

In order to assist the user in programming applications, the functions of the various DML statements are explained using COBOL DML as the example. DML application and testing is described in the subsequent chapters. The reference sections are divided as follows:

- COBOL DML, which is integrated into the language scope of the ANSCOBOL compiler COBOL85, and
- CALL DML, which is accessed via the CALL interface of the various programming languages.

Readme file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at <http://manuals.ts.fujitsu.com>. You will also find the Readme files on the Softbook DVD.

Information under BS2000

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

```
SYSRME.<product>.<version>.<lang>
```

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `/SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at <http://manuals.ts.fujitsu.com>.

1.4 Changes since the last edition of the manuals

The main changes introduced in UDS/SQL V2.8 in comparison with Version V2.7 are listed in [table 2](#) below together with the manuals and the sections in which the changes are described. If a specific topic has been dealt with in more than one manual, the manual in which a detailed description appears is listed first. The following codes are used in the “Manual” column for the individual manuals involved:

DES	Design and Definition	DBO	Database Operation
APP	Application Programming	RIR	Recovery, Information and Reorganization
CRE	Creation and Restructuring	MSG	Messages

Topic	Manual	Chapter
UDSMON utility: Improvements concerning transaction time and DB counters		
For output to terminal and output to printer: In the UDS/SQL monitor mask COUNTER, the unit for displaying the AVG TRANSACTION TIME is improved to seconds with milliseconds after the decimal point to enable monitoring of short transactions.	DBO	11
New DISPLAY DBCOUNTERS command in UDSMON for displaying database counters	DBO	11
BSTATUS utility: Limit the TABLE STATISTICS FOR OWNER IN SET		
Improved DISPLAY TABLE FOR OWNER statement to enable limiting the TABLE STATISTICS FOR OWNER IN SET to specific owner records or ranges of records.	RIR	6
New BSTATUS utility routine messages	MSG	3
New BPRECORD utility routine message 2553 in case of value 0 being specified as a start value in RSQ range.	MSG	3
Database Operation: The number of DML statements and I/O operations are counted per database.	DBO MSG	4 2
BOUTLOAD utility: Output in CSV format	CRE MSG	5 3
COPY-RECORD statement: New CSV-OUTPUT operand	CRE	5
New output file format CSV	CRE	5

Table 2: Changes in version V2.8 compared to V2.7

Topic	Manual	Chapter
ONLINE-UTILITY – Reorganize probable position pointers (PPPs)		
New DML REORGPPP - Reorganize PPPs	RIR	8
New SDF statements: SET-REORGANIZE-PPP-PARAMETERS, SHOW-REORGANIZE-PPP-PARAMETERS	RIR	8
New procedure statement REORGPPP	RIR	8
New predefined variables: REORG-PPP-CURRENT, REORG-PPP-LOCKED, REORG-PPP-PAGES	RIR	8
New predefined standard procedure *STDREPPP	RIR	8
New example „Reorganizing pointers“	RIR	8
New status codes with progress information of the online utility REORGPPP and new error codes	APP	10

Table 2: Changes in version V2.8 compared to V2.7



General information

The name BS2000/OSD-BC for the BS2000 basic configuration has changed and from Version V10.0 becomes: BS2000 OSD/BC.

1.5 Notational conventions

This section provides an explanation of the symbols used for warnings and notes as well as the notational conventions used to describe syntax rules.

1.5.1 Warnings and notes

	Points out particularly important information
 CAUTION!	Warnings

1.5.2 Non-SDF notational conventions

Language element	Explanation	Example
<u>KEYWORD</u>	Keywords are shown in underlined uppercase letters. You must specify at least the underlined parts of a keyword.	<u>DATABASE-KEY</u> <u>MANUAL</u>
OPTIONAL WORD	Optional words are shown in uppercase letters without underlining. Such words may be omitted without altering the meaning of a statement.	NAME IS ALLOWED PAGES
<i>variable</i>	Variables are shown in italic lowercase letters. In a format which contains variables, a current value must be entered in place of each variable.	<i>item-name</i> <i>literal-3</i> <i>integer</i>
{ Either or }	Exactly one of the expressions enclosed in braces must be specified. Indented lines belong to the preceding expression. The braces themselves must not be specified.	{ <u>CALC</u> } { <u>INDEX</u> } { <u>VALUE IS</u> } { <u>VALUES ARE</u> }
[optional]	The expression in square brackets can be omitted. UDS/SQL then uses the default value. The brackets themselves must not be specified.	[IS <i>integer</i>] [<u>WITHIN</u> <i>realm-name</i>]

Table 3: Notational conventions

(part 1 of 2)

Language element	Explanation	Example
. . . or , . . .	The immediately preceding expression can be repeated several times if required. The two language elements distinguish between repetitions which use blanks and those which use commas.	<i>item-name</i> , . . . { <u>SEARCH</u> KEY } . . .
. or	Indicates where entries have been omitted for reasons of clarity. When the formats are used, these omissions are not allowed.	<u>SEARCH</u> KEY IS <u>RECORD</u> NAME
␣	The period must be specified and must be followed by at least one blank. The underline must not be specified.	<u>SET</u> <u>SECTION</u> . 03 <i>item-name</i> ␣
Space	Means that at least one blank has to be specified.	<u>USING</u> <u>CALC</u>

Table 3: Notational conventions

(part 2 of 2)

All other characters such as () , . ; “ = are not metacharacters; they must be specified exactly as they appear in the formats.

1.5.3 SDF syntax representation

This syntax description is based on SDF Version 4.7. The syntax of the SDF command/statement language is explained in the following three tables.

Table 4: Metasyntax

Certain characters and representations are used in the statement formats; their meaning is explained in table 4.

Table 5: Data types

Variable operand values are represented in SDF by data types. Each data type represents a specific value set. The number of data types is limited to those described in table 5.

The description of the data types is valid for all commands and statements. Therefore only deviations from table 5 are explained in the relevant operand descriptions.

Table 6: Data type suffixes

The description of the “integer” data type in table 6 also contains a number of items in italics. The italics are not part of the syntax, but are used merely to make the table easier to read.

The description of the data type suffixes is valid for all commands and statements. Therefore only deviations from table 6 are explained in the relevant operand descriptions.

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords. Some keywords begin with <i>*</i> .	OPEN DATABASE COPY-NAME = <i>*NONE</i>
=	The equal sign connects an operand name with the associated operand values.	CONFIGURATION-NAME = <name 1..8>
< >	Angle brackets denote variables whose range of values is described by data types and their suffixes (Tables 5 and 6).	DATABASE = <dbname>
<u>Underscoring</u>	Underscoring denotes the default value of an operand.	SCHEMA-NAME = <i>*STD</i>
/	A slash separates alternative operand values.	CMD = <i>*ALL</i> / <dal-cmd>
(...)	Parentheses denote operand values that initiate a structure.	<i>*KSET-FORMAT(...)</i>

Table 4: Metasyntax

(part 1 of 2)

Representation	Meaning	Examples
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	<pre> USER-GROUP-NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host> </pre>
	A vertical bar identifies related operands within structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	<pre> USER-GROUP-NAME = *ALL-EXCEPT(...) *ALL-EXCEPT(...) NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host> ... </pre>
,	A comma precedes further operands at the same structure level.	,SPACE = <u>STD</u>
list-poss(n):	list-poss signifies that the operand values following it may be entered as a list. If a value is specified for (n), the list may contain no more than that number of elements. A list of two or more elements must be enclosed in parentheses.	NAME = list-poss(30): <subschema-name>

Table 4: Metasyntax

(part 2 of 4)

Data type	Character set	Special rules
alog-seq-no	0..9	1..9 characters
appl	A..Z 0..9 \$,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or \$, #, @ Strings of less than 8 characters are filled internally with underscore characters.
catid	A..Z 0..9	1..4 characters must not start with the string PUB
copyname	A..Z 0..9	1..7 characters, starting with A..Z

Table 5: Data types

(part 1 of 4)

Data type	Character set	Special rules
c-string	EBCDIC characters	1..4 characters Must be enclosed in single quotes; the letter C may be used as a prefix. Single quotes within c-string must be specified twice.
csv-filename	A..Z 0..9 Structure identifier: hyphen	1..30 characters Must be enclosed in single quotes
dal-cmd	A..Z 0..9 hyphen	1..64 characters
date	0..9 Structure identifier: hyphen	Date specification Input format: yyyy-mm-dd yyyy : year; may be 2 or 4 digits long mm : month dd : day
dbname	A..Z 0..9	1..17 characters, starting with A..Z
device	A..Z 0..9 \$,#,@ Structure identifier: hyphen	5..8 characters, starting with A..Z or 0..9 String that can consist of a number of substrings separated by hyphens and which corresponds to a device. In the dialog guidance, SDF shows the permissible operand values. Information as the possible devices can be found in the relevant operand description.
host	A..Z 0..9 \$,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or \$, #, @ Strings of less than 8 characters are filled internally with underscore characters.
integer	0..9,+,-	+ or - may only be the first character.
kset	A..Z 0..9 \$,#,@ Structure identifier: hyphen	1..8 characters String that can consist of a number of substrings separated by hyphens; first character A..Z or \$, #, @ Strings of less than 8 characters are filled internally with underscore characters.
name	A..Z 0..9 \$,#,@	1..8 characters Must not consist only of 0..9 and must not start with a digit

Table 5: Data types

(part 2 of 4)

Data type	Character set	Special rules
realm-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that may consist of a number of substrings by hyphens; first character: A..Z
realmref	0..9	1..3 characters
record-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z In the case of record types with a search key it is recom- mendable to use names with no more than 26 characters, otherwise the set name created implicitly (SYS_...) will be truncated in accordance with the restriction on the name length for sets.
recordref	0..9	1..3 characters
schema-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z
set-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z
structured-name	A...Z 0...9 \$, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A...Z or \$, #, @
subschema-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z

Table 5: Data types

(part 3 of 4)

Data type	Character set	Special rules
time	0..9 Structure identifier: colon	Time-of-day specification Input format: $\left. \begin{array}{l} hh:mm:ss \\ hh:mm \\ hh \end{array} \right\}$ hh, mm, ss: Leading zeros may be omitted
userid	A..Z 0..9 \$,#,@	1..8 characters, beginning with A..Z or \$,#,@ BPRIVACY: Strings of less than 8 characters are filled internally with underscore characters.
volume	A..Z 0..9 \$,#,@	1..6 characters starting with A..Z or 0..9
x-string	Hexadecimal: 00..FF	1..8 characters Must be enclosed in single quotes and prefixed with the letter X. There may be an odd number of characters

Table 5: Data types

(part 4 of 4)

Suffix	Meaning
<i>x..y unit</i>	For the “integer” data type: range specification. <i>x</i> Minimum value permitted for “integer”. <i>x</i> is an (optionally signed) integer. <i>y</i> Maximum value permitted for “integer”. <i>y</i> is an (optionally signed) integer. <i>unit</i> for “integer” only: additional units. The following units may be specified: <i>Mbyte, Kbyte, seconds</i>

Table 6: Data type suffixes

1.6 Sample database

The examples given in this manual refer to the following database:

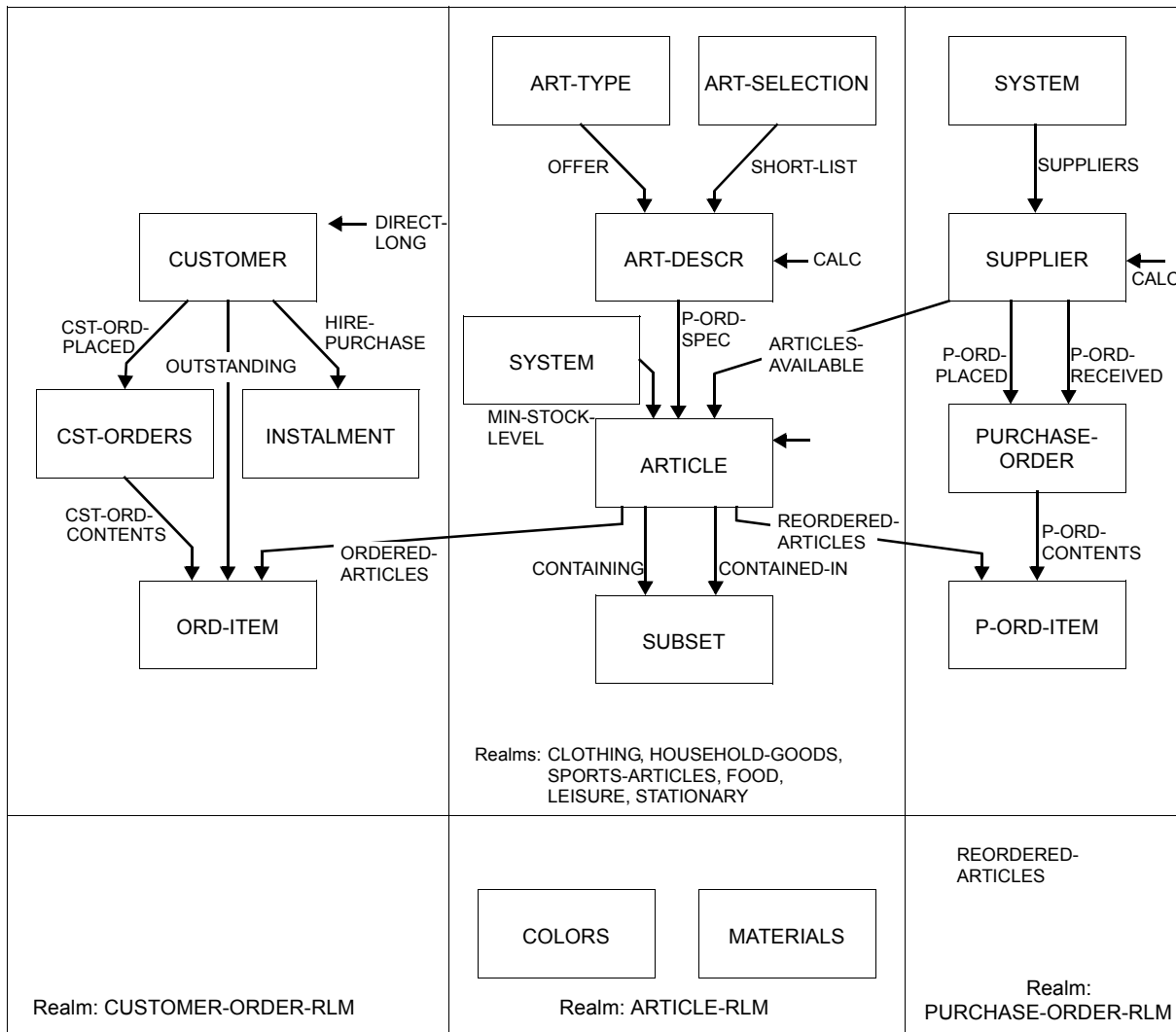


Figure 1: Mail order schema

2 Overview

2.1 The language concept of DML

The language resources available for data processing using files and file systems are not adequate for users wishing to program database applications.

The Data Manipulation Language (DML) has been introduced for the Universal Database System UDS/SQL and conforms to the CODASYL language concept for databases. It is not designed to operate with one specific programming language but can be implemented in conjunction with a number of different programming languages.

DML exists in two different forms:

- COBOL DML is integrated into the language scope of the ANSCOBOL compiler COBOL85 and COBOL2000
- CALL DML can be accessed via the CALL interface of the programming languages Assembler, COBOL, C, FORTRAN, PASCAL and PL/1.

The DML statements take account of the relationships between records as defined in the schema for a database. They execute the changes to the database which have been prepared in the communication area of the program. Other processing of data must be carried out within the application program using the chosen programming language. The communication area in the application program is the UWA (User Work Area).

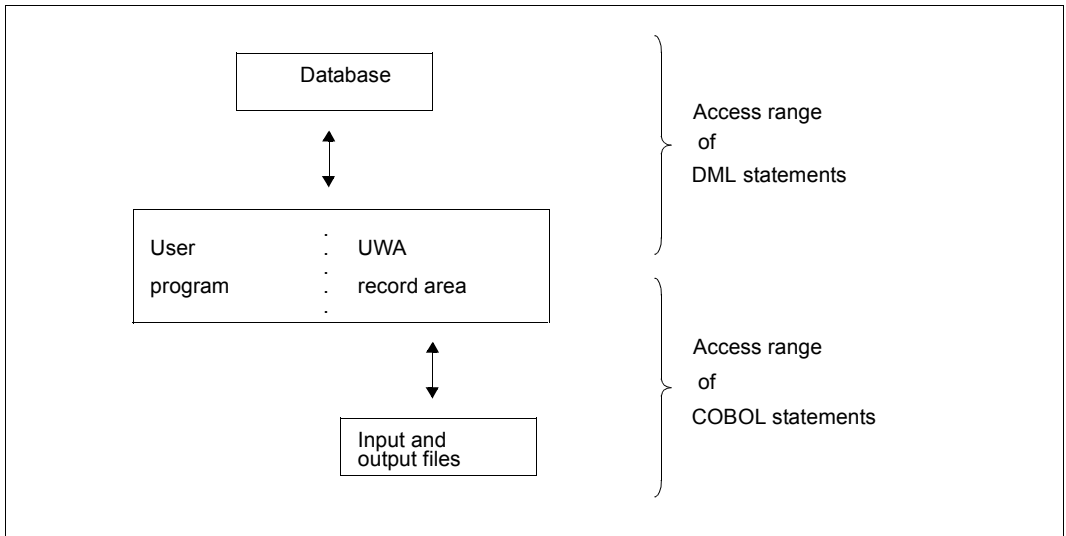


Figure 2: Access range of COBOL and DML statements

2.2 The range of functions for DML

The functions available in DML are:

- opening and closing transactions
- retrieving data
- modifying data
- protecting records
- testing conditions

These functions can only be accessed on the condition that changes made to the database do not adversely affect other users. In UDS/SQL this is taken care of by the transaction concept and the associated recovery concept.

3 Transaction concept

A transaction consists of a logically complete sequence of DML statements. It begins with READY and ends with FINISH. This sequence of statements is used to process related tasks. A transaction should be as short as possible so as not to block the individual realms and pages of the database(s) for too long. Wherever possible, a transaction should not be longer than 2000 DML statements in batch mode or more than 20 DML statements in an online application. The transaction concept of UDS/SQL ensures that different users can utilize the database(s) without impeding each other.

The transaction concept ensures the logical consistency of the database and the recovery of data.

A transaction is a logically related sequence of DML statements. It is executed either completely or, if there is an error, not at all. This means that the accessed databases are always in a consistent state.

Recovery measures are also performed for each transaction; these can be defined by the user at the start of the transaction to suit the application involved.

When the transaction is opened, the user defines the realms with which he or she wishes to work and the required type of access to the database (USAGE-MODE). The user determines whether other users may work with these realms and is thus able to protect data against unauthorized access.

If the linked-in DBH is used, the protection functions are not relevant since there is only one user.

3.1 The transaction in multi-DB operation

In multi-DB operation, several databases form the multi-DB configuration. A transaction can access all the databases within this configuration.

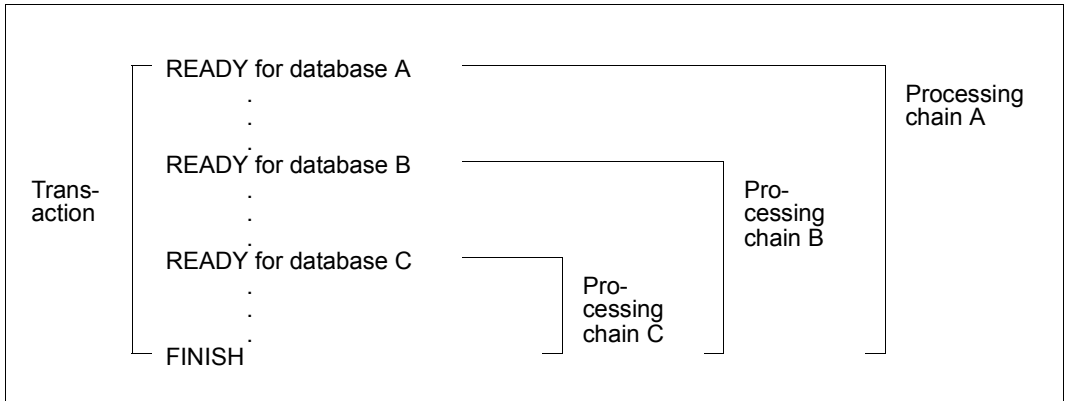


Figure 3: Transaction and processing chains in multi-DB operation

Each database must be opened with a **READY** statement. For this reason a transaction in multi-DB operation includes several **READY** statements. The first **READY** statement opens the transaction and simultaneously opens the first processing chain. Each subsequent **READY** statement opens a further processing chain. A processing chain is a sequence of DML statements addressed to one database within a transaction. All processing chains are concluded by a common **FINISH** statement, which simultaneously terminates the transaction.

In multi-DB operation a transaction consists of several processing chains, in COBOL DML each chain is processed within a separate application module.

In one transaction it is possible to use both processing chains containing COBOL DML and CALL DML statements. However, either COBOL DML statements only or CALL DML statements only can be used within a processing chain.

If there is an error in any **READY** statement of the transaction, the entire transaction is rolled back.

3.2 The transaction in mono-DB operation

In this type of application the single processing chain corresponds to the transaction (see [figure 3](#)). There is only one READY statement per transaction.

3.3 Rollback

Rollback cancels the modifications made by the transaction if, for example, the transaction could not be concluded with FINISH before the end of the program.

In some situations UDS/SQL has to execute a rollback of the entire transaction instead of executing the called DML statement.

This is the case, for example, if two or more transactions are blocking each other by attempting to access blocks, realms or other resources (tables, tasks) which are locked. The term used to describe this situation is “deadlock”. One of the transactions which are causing the deadlock is rolled back.

Transaction rollback can also be caused by:

- serious user errors
- premature termination by the database administrator (DAL command ABORT, CLOSE CALLS)
- input/output errors in realms or log files

A rollback must be recognized by the application program (see [section “Error handling” on page 114](#)). In addition, you can also specify a USE statement with certain database exception conditions and define appropriate recovery routines via the DECLARATIVES of a COBOL DML program (see [section “Database exception conditions” on page 117](#)).

3.4 Page protection

An important consideration in the introduction of database systems is the centralization of data in a database. This also means, however, that concurrent transactions access the database simultaneously.

Whereas the database may be accessed by several users simultaneously for read-only access, only one user at a time may access the data for updates.

The DBH must therefore take protective measures to ensure that the data remains consistent state and to guarantee an error-free execution sequence.

Example

Two programs add a value of 50 each to an item in the database with an initial value of 100. The result will be 200 in this case only if the second program performs its access after the first program has written back the value of 150 into the item. If both programs could read and modify the data at the same time, it is possible that the end result would be 150.

There are two levels of locking that help to maintain consistency:

- locking at the realm level (see [section “Opening a transaction or processing chain \(READY\)” on page 43](#))
- locking at the page level (FIND, FETCH, modifying data, KEEP, FREE); a page is locked as soon as a record of that page is accessed.

Each of these locking levels are, in turn, protected by two types of locks:

- Exclusive locks:
Only one transaction has access to the data.
- Shared locks:
Multiple transactions may access the data at the same time, provided no exclusive lock was set.

The above two locks are used for page protection on the page level as follows:

- On retrieving data (FIND/FETCH), any page that has not been locked is protected against updates from other transactions (shared lock) by the page protection mechanism.
- On updating data, the page in which a record is being updated is protected against all access from other transactions by an exclusive lock. If pages with secondary data (e.g. table entries) are involved, these pages are also locked by the page protection mechanism.

This ensures that deadlocks are prevented in read-only applications.

The page protection can be retained only till the end of the transaction.

In cases where no contending access (for READY EXCLUSIVE) or modification of the database (for READY PROTECTED RETRIEVAL) is possible at the realm level, the page protection routines are deactivated; for READY PROTECTED UPDATE, shared locking is dropped only for updating transactions. The locks then apply only at the realm level in accordance with the specified READY mode (see the READY statement on [page 43](#)).

To prevent concurrently active transactions from impeding each other unnecessarily with long-term locks, it is recommended in interactive operation that you terminate each transaction with FINISH after a few modification operations and then open a new transaction with READY whenever possible.

The pages of the Free Place Administration table (FPA) and the Database Key Translation Table (DBTT) are not subject to page protection. The entries in these tables always refer to a quite specific data page and can only be modified when this data page has been modified and thus locked. Explicit locking of these table entries is therefore superfluous. Different transactions can modify different entries in an FPA or DBTT page at the same time.

4 Currency table

The DBH enters certain current database key values into the currency table. This currency table information can, for example, be used to access the record of a record type whose database key value is stored in the table.

The currency table is always maintained for one processing chain at a time and is reset at FINISH.

A database key value is held in the currency table for each record type, set and realm of a subschema and for the processing chain. The records belonging to these database key values are as follows:

- CRR (Current Record of Record): current record of record type
- CRS (Current Record of Set): current record of set
- CRA (Current Record of Area): current record of realm
- CRU (Current Record of Run-unit): current record of processing chain

A record becomes the current record in the currency table when it is accessed by the DML statements FIND, STORE, ERASE, CONNECT and MODIFY.

The entries in the currency table are also used in the execution of a FIND NEXT statement. FIND NEXT (FIND-4) selects the record logically following the current record of the record type, the set occurrence or the realm.

The structure and value ranges for database key values are discussed in detail in the [“Design and Definition”](#) manual.

All cases where there are individual restrictions with respect to the structure (DATABASE-KEY/DATABASE-KEY-LONG) or value range for a database key value described in the present manual are indicated where appropriate by means of a special note.

Examples

1. The FIND-4 statement is used to select the record following the CRS.
2. The FIND-5 statement is used to reset the currency table information to its previous state.
3. In the FIND-7 statement

FIND PERSONNEL-RECORD WITHIN PERSONNEL-SET CURRENT USING ACTIVITY

the DBH interprets the currency information to determine the set occurrence from which the PERSONNEL-RECORD is to be selected. It checks which record is the current record in the PERSONNEL-SET. The PERSONNEL-RECORD is then selected from the set occurrence to which the current record belongs.

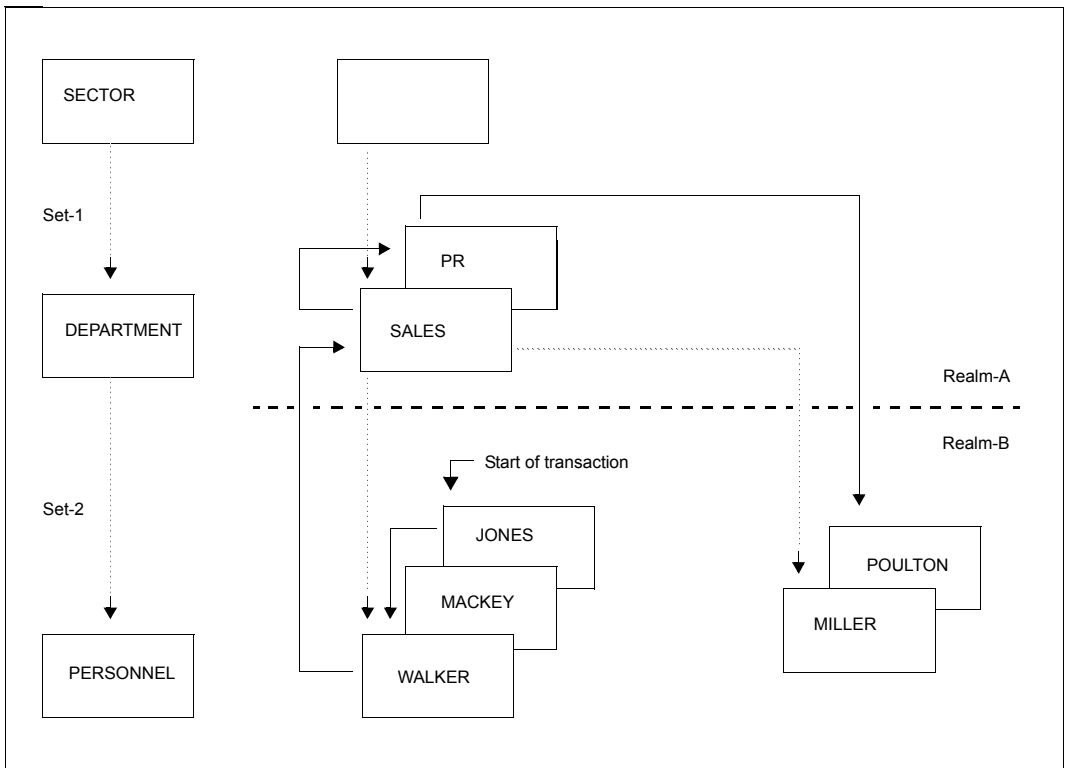


Figure 4: Set occurrences to illustrate the currency table

	Database key of CRU	Database key of CRA		Database key of CRS		Database key of CRR		
		REALM-A	REALM-B	SET-1	SET-2	SECTOR	DEPT.	PERSONNEL
READY	0	0	0	0	0	0	0	0
Retrieval of								
POULTON	POULTON	0	POULTON	0	POULTON	0	0	POULTON
WALKER	WALKER	0	WALKER	0	WALKER	0	0	WALKER
SALES	SALES	SALES	WALKER	SALES	SALES	0	SALES	WALKER
PR	PR	PR	WALKER	PR	PR	0	PR	WALKER
JONES	JONES	PR	JONES	PR	JONES	0	PR	JONES
FINISH	0	0	0	0	0	0	0	0

Table 7: Changes in the currency table

Some DML statements allow the user to specify whether the automatic updating of information in the currency table is to be suppressed. This is done by means of the RETAINING option (see [section "Retrieving data" on page 46](#), for example).

Updates of all currency information apart from that of the CRU can be suppressed.

5 DML functions

The DML functions are divided into five groups:

- opening and closing transactions
- retrieving data
- modifying data
- protecting records
- testing set memberships

This chapter deals with the conversion of user jobs into DML statements. The functions are explained using COBOL DML statements.

The following [table 8](#) shows the COBOL DML statements together with the corresponding CALL DML formats

COBOL-DML	CALL-DML
ACCEPT	ACCPTC, ACCPTL
CONNECT	CONNEC
DISCONNECT	DISCON
ERASE	ERASEC
FETCH1-7	FTCH1, FTCH1L, FTCH2-7
FIND1-7	FIND1, FIND1L, FIND2-7
FINISH	FINISC
FREE	FREEC
GET	GETC
IF	IFC
KEEP	KEEPC
-	LOOKC
MODIFY	MODIF1
-	MODIF2
READY	READYC
SET	-
STORE	STORE1, STOR1L
-	STORE2, STOR2L
USE	-

Table 8: COBOL DML statements and the corresponding CALL DML functions

The SET and USE statements are described in [section “Special features of COBOL DML” on page 86](#). The LOOKC function is presented in [section “Special features of CALL DML” on page 92](#) and explained in detail in [section “LOOKC function” on page 266](#). The syntax rules and formats for COBOL DML and the syntax rules and parameter definitions for CALL DML are described in [chapter “COBOL DML reference section” on page 129](#) and [chapter “CALL DML reference section” on page 197](#), respectively.

5.1 Opening and closing transactions

Processing can be controlled as follows:

- by opening a transaction and specifying usage modes (READY)
- by closing the transaction (FINISH)

5.1.1 Opening a transaction or processing chain (READY)

```
READY[ realm-name, ... ][ USAGE-MODE IS { EXCLUSIVE } { RETRIEVAL }
                                     { PROTECTED } { UPDATE } ]
```

The READY statement opens a transaction or a processing chain (multi-DB mode) for a database. READY defines the usage modes and specifies the realms to be used.

One READY statement must be specified for each database that the user wishes to access. The first READY also opens the transaction.

The realm names specified must refer to realms within the user's subschema.

The usage mode of the realms is defined by USAGE-MODE:

- Access by other processing chains which also wish to work with the realms:
 - EXCLUSIVE: No other processing chain may use these realms simultaneously.
 - PROTECTED: Other processing chains may not modify data simultaneously.
- Access by the user's processing chain:
 - RETRIEVAL: Retrieve data
 - UPDATE: Retrieve and update data

If USAGE-MODE is not specified, USAGE-MODE RETRIEVAL is assumed.

The combinations offered by these two specifications give six possible usage modes for a processing chain, but these are not all compatible with the usage modes of other processing chains.

The following table shows which usage modes are permitted concurrently for one realm in mono-DB operation::

	RETRIEVAL	UPDATE	PROTECTED RETRIEVAL	PROTECTED UPDATE	EXCLUSIVE RETRIEVAL	EXCLUSIVE UPDATE
RETRIEVAL	X ¹	(X) ²	X	(X)	- ³	-
UPDATE	(X)	(X)	-	-	-	-
PROTECTED RETRIEVAL	X	-	X	-	-	-
PROTECTED UPDATE	(X)	-	-	-	-	-
EXCLUSIVE RETRIEVAL	-	-	-	-	-	-
EXCLUSIVE UPDATE	-	-	-	-	-	-

Table 9: Permitted combinations of USAGE MODE for mono-DB operation

¹ X possible

² (X) deadlock possible

³ - not possible

Example

If a realm is already opened with USAGE-MODE UPDATE or EXCLUSIVE RETRIEVAL, it cannot be opened by a further processing chain with READY USAGE MODE PROTECTED UPDATE.

The READY statement is executed only if the defined usage mode matches those of the other processing chains that are accessing the same realms. Otherwise, the system delays the execution until the incompatible processing chains are completed.

Two processing chains are not allowed to concurrently access the same database within one transaction.

This restriction prevents two processing chains of the same user from becoming deadlocked because they have the same access rights at the realm level.

5.1.2 Closing a transaction (FINISH)

FINISH[WITH CANCEL]

A transaction is terminated by means of FINISH.

This simultaneously terminates all open processing chains of this transaction and closes all realms which were opened by these processing chains. When the transaction is completed, the before-images and the currency tables are deleted. The temporary realm is released. Any extension of record protection via KEEP is cancelled.

If the user terminates the transaction with WITH CANCEL, e.g. during a test run of the program, the DBH rolls back all changes made by the processing chains of this transaction. If no RLOG files exist, e.g. where PP LOG=NO, or if the RLOG files are defective, the affected database is flagged as defective and locked for transactions until the database administrator has returned the database to a consistent state, which also includes rolling back previous transactions.

5.2 Retrieving data

There are four DML statements available for retrieving data from the database:

FIND searches for a record in the database, enters its database key value into the currency table, and identifies it as the current record. This record can be directly accessed by UDS/SQL via the database key with the statement sequence ACCEPT, FIND-1 so long as the record is identified in the currency table.

GET delivers the record marked as the current record (CRU) in the currency table to the application program in the User Work Area (UWA).

FETCH combines the functions of FIND and GET, i.e. it not only searches for a record in the database and enters its database key value in the currency table, but also delivers the record to the UWA of the application program.

ACCEPT delivers to the application program the database key value of a record indicated in the currency table or the name of the realm in which the record is stored.

These DML statements are described in the sections below. In the description no differentiation is made between FIND and FETCH. It is taken as read that FETCH additionally delivers the last record accessed to the application program.

Finding a record in the database

$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\} \text{record-selection-expression[} \underline{\text{RETAINING}} \text{ CURRENCY]}$

The *record-selection-expression* enables the user to define how records are to be accessed. The access options are explained in the next chapters.

UDS/SQL identifies a selected record by overwriting all the relevant columns of the currency table with the database key value for the record. The database key value for a previously selected record thus disappears from these columns of the currency table: this latter record can then only be retrieved using a new search procedure in the database.

The RETAINING option may be used to suppress the updating of currency table entries when the user wishes to preserve the database key value of a previous record.

Explicit control of currency table updating

```

RETAINING CURRENCY FOR { MULTIPLE
                        [ REALM][ RECORD][ { SETS
                                                { set-name,... } ] ] }

```

The user can suppress the updating of any currency table entry except the CRU by adding the following specification to the FIND/FETCH statement.

If updating has been suppressed using RETAINING, the currency table can be updated subsequently to the most recent status without having to search for the record again in the database:

```

{ FIND
  FETCH } CURRENT[ RETAINING CURRENCY FOR REALM][ RECORD][ { SETS
                                                                    { set-name,... } ] ]

```

This statement transfers the database key value of the CRU, which always remains current, to the other relevant columns in the currency table.

RETAINING can also be used in this case to specify which columns should be excluded from the update.

5.2.1 Direct access at record type level

With this type of access the contents of an item or combination of items are used to select a record. The collection of records from which the record is to be selected comprises all records of one record type. The items can be any items of the record type.

5.2.1.1 Direct access via the database key (FIND/FETCH-1)

```

{ FIND } [ record-name] DATABASE-KEY IS item-name [OR { PRIOR } ]
{ FETCH }                                     { NEXT } ]

```

Each record has a key which is unique within the database; this is known as the database key value.

If the user transfers a database key value into the item *item-name*, UDS/SQL delivers the associated record. UDS/SQL can also check whether the record belongs to record type *record-name*.

The item *item-name* must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

If you want to search for a record with a database key value containing a REC-REF > 254 and/or an RSQ > 2²⁴-1, *item-name* must be defined with USAGE IS DATABASE-KEY-LONG.

When you specify OR PRIOR/NEXT, the record with the next lowest/next highest database key is made available if no record with the specified database key exists. A special status code then shows that the record with the DATABASE-KEY which was initially specified has not been selected.

5.2.1.2 Direct access via the CALC key (FIND/FETCH-2)

```
{ FIND }
{ FETCH } ANY record-name
```

This FIND statement may only be used if a CALC key is defined for record type *record-name* in the schema (see the “[Design and Definition](#)” manual, LOCATION MODE clause). The user must previously have supplied the items defined as a CALC key in the schema with the CALC key value of the desired record. UDS/SQL accesses the record by converting this value into a relative page number by means of a hash procedure.

When the record type *record-name* is distributed over more than one realm, the user must also previously have supplied the AREA-ID item (see the “[Design and Definition](#)” manual, WITHIN clause) with the name of the realm containing the required record if the record type is not the member record type of a distributable list.

If DUPLICATES ARE ALLOWED is declared in the schema, a CALC key value can belong to several records of the record type. The above statement, however, delivers one record only. For each further record of the record type with the same CALC key value in the same realm, the user must write a statement in the following form:

```
{ FIND }
{ FETCH } DUPLICATE record-name
```

This statement can be used sensibly only if the CALC key of record type *record-name* is defined by DUPLICATES ARE ALLOWED in the schema. UDS/SQL then searches in this record type for a record that is different from the CRR, but possesses the same CALC key value as the CRR and lies within the same realm as the CRR. UDS/SQL uses the associated hash procedure for the search.

The statement must be repeated for each duplicate to be searched for. It cannot be guaranteed that all records which belong to one CALC key value will be found unless the search for the first record is made in each realm which contains records of the record type with

```
{ FIND }
{ FETCH } ANY record-name
```

5.2.1.3 Direct access via any items (FIND/FETCH-3/7)

There are three options available for searching for records using specified item contents:

- You specify item contents from the CRR and search for a duplicate.
- You specify the desired item contents in the UWA and search for a record having these item contents.
- You specify the desired item contents in the UWA or in the statement itself and link these with a search expression in which parentheses, the logical operators AND, OR, NOT and the relational operators >, < and = can be used.

In this case, the item contents need not be specified completely. A mask can be used to edit out the section of an item for which no value is to be specified. UDS/SQL searches for all records which satisfy the search expression and makes them into member records of an implicit dynamic set. If RESULT IN is entered, these sets also become members in an explicit dynamic set.

The dynamic sets can be processed further using DML statements, applying different search criteria, for instance.

The implicit dynamic set can be sorted in ascending or descending order (SORTED BY).

The access types are explained in detail below.

Specifying item contents from the CRR

```

{ FIND }
{ FETCH } DUPLICATE WITHIN record-name USING record-element-name, ...

```

UDS/SQL searches for a record of record type *record-name* which matches the CRR of this record type in the specified record elements.

Specifying item contents in the UWA

$$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\} \text{ record-name } \underline{\text{USING}} \text{ record-element-name, ... } \left[\text{OR } \left. \begin{array}{l} \text{PRIOR} \\ \text{NEXT} \end{array} \right\} \right]$$

UDS/SQL searches for a record of record type *record-name* with record elements that match the values preset in the UWA. The user must supply the desired values for these record elements beforehand in the UWA.

By specifying OR PRIOR/NEXT, the user can cause the prior/next record to be made available if there is no record that matches the specified values. A special status code indicates that the record with the specified values has not been selected. *record-element-name,...* must be defined as SEARCH KEY USING INDEX. The following record is determined by the sort sequence.

If there is more than one record, the above statement only delivers the first record. For each further record, the following statement must be used:

$$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\} \underline{\text{DUPLICATE WITHIN}} \text{ record-name } \underline{\text{USING}} \text{ record-element-name, ...}$$

The same record elements that were used to select the first record must be specified here. This statement is explained above (see [“Specifying item contents from the CRR” on page 50](#)).

Combining item contents into a search expression

```

{ FIND }
{ FETCH }
record-name[ USING search-expression][ RESULT IN set-name-1]
  [ LIMITED BY set-name-2][ TALLYING item-name-1]
  [ SORTED[ { ASCENDING } ] [ { BY } ]
    { DESCENDING } ] [ { ON } ]

record-element-name-1[[,]record-element-name-2]...
[[,][ { ASCENDING } ] [ { BY } ]
  { DESCENDING } ] [ { ON } ]

record-element-name-3[[,]record-element-name-4]...]]]]]]

search-expression ::= { complex-1[ AND complex-2]
                      { complex-2 }

complex-1 ::= [NOT ]condition-1[ { AND } ] [ NOT ] condition-1]...
           { OR }

condition-1 ::= record-element-name-5[ WITH MASK mask] IS
               { EQUAL
                 =
                 GREATER THAN } { item-name-2 }
               { >
                 LESS THAN } { literal-1 }
               { <

complex-2 ::= condition-2[ AND condition-2]...

condition-2 ::= record-element-name-6 IS NEXT
               { GREATER THAN
                 >
                 LESS THAN } { item-name-3 }
               { <

```

This statement can select from the database not just one record, but all records of the record type *record-name* which satisfy the conditions of the search expression.

FIND statements which select only one record from the database use the currency table to store the result of the selection.

When a collection of records is selected, UDS/SQL stores the result of the selection in a dynamic set, making the records selected into member records of the dynamic set. Only the first record found becomes the current record in the currency table. The dynamic set can be used for record selection by further FIND statements.

In *search-expression* the user formulates comparison conditions by specifying item contents or parts of item contents which must occur in the records being searched for. The comparison can be with a value specified in the statement itself or with the contents of an item defined in the COBOL program.

In a *search-expression* of the form *complex-1* AND *complex-2*, all records that satisfy *complex-1* are determined first. This set of records that satisfy *complex-1* is then used as a starting point from which records that match the condition(s) in *complex-2* are selected.

The *condition-2* conditions in *complex-2* are evaluated in order, starting with the left-most condition.

Adding NEXT restricts the selected set of *condition-2* in *complex-2* to those records of the record type *record-name* that contain the value W with the following two characteristics as *record-element-6*:

- W matches the comparison in *condition-2*.
- From all values in the database for *record-element-6* that match the comparison in *condition-2*, W lies closest to the comparison value specified by *literal-2* or *item-name-3*.

- Masking out sections of item contents

Using WITH MASK *mask* the user can define parts of *record-element-name-5* which are to be ignored by UDS/SQL in a comparison condition.

- Inserting all records of the record type into a dynamic set

If no search expression is specified, all records of the record type become member records in a dynamic set.

- Limiting collection of records to be searched to a dynamic set

A dynamic set to which UDS/SQL is to limit the search for records can be named by specifying LIMITED BY *set-name-2*. This means that UDS/SQL only takes into account records which belong to record type *record-name* and are at the same time member records of the dynamic set.

- Counting the records in the collection of records selected

The records contained in the collection of records selected can be counted using TALLYING. UDS/SQL transfers the number of records found to item *item-name-1*

- Sorting the records in the collection of records selected

SORTED BY is used to sort the records contained in the collection of records selected. *record-element-name-1* through *record-element-name-4* must be elements of the *record-name* record type. ASCENDING causes the records to be sorted in ascending order, DESCENDING in descending order. The default for the first entry is ASCENDING and for a repeated entry, the default is the currently valid sorting order. All the sort items must be sorted in the same direction.

- Accessing a duplicate

When selecting a collection of records, UDS/SQL stores the result of the selection in a dynamic set. Only the first record found becomes the current record in the currency table. If no RESULT clause is used, the the following statement must be entered for each further record to be called from the result of the selection:

$$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\} \text{DUPLICATE WITHIN } \textit{record-name}$$

This statement always refers to the last collection of records selected since the dynamic set always contains only the result of the last search expression.

record-name must designate the record type which was processed with the previous search expression. UDS/SQL then delivers the record which follows the CRS in the dynamic set.

- Further processing of the result of the selection

The selection resulting from a search expression is stored by UDS/SQL in an implicit dynamic set. This is a set which cannot be accessed by means of a name.

RESULT IN *set-name-1* can be used to name a dynamic set which is already defined in the schema and is to accept the result of a search expression. The result stored in this dynamic set can be processed further at set level using FIND/FETCH-4 statements.

5.2.2 Sequential access at record type level (FIND/FETCH-4)

With this type of access a record is selected on the basis of the position which it occupies within the logical sequence of all records of one record type.

The sequence is determined by ascending order of database key values.

$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\}$	LAST	$\left. \begin{array}{l} \\ \\ \\ \\ \end{array} \right\} \text{ record-name}$
	FIRST	
	NEXT	
	PRIOR	
	<i>integer</i>	
	<i>item-name</i>	

This option allows the user to select from record type *record-name* either the last or first record, the successor or predecessor of the CRR, or the record whose position corresponds to the specified numeric value.

5.2.3 Access to the CRR (FIND/FETCH-5)

$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\}$	CURRENT	<i>record-name</i>
--	---------	--------------------

This statement is used to access the CRR of record type *record-name* and thus reset the currency information to an earlier status.

The CRR again becomes the current record in all affected columns of the currency table or does so wherever updating has not been suppressed by the RETAINING parameter.

5.2.4 Direct access at set level (FIND/FETCH-3/7)

This type of access is used to select a record via the contents of an item or a combination of items. The collection of records used for the search are all the records of a set occurrence. The items can be any items of the member record type.

UDS/SQL always uses the set occurrence associated with the CRS as the collection of records from which the selection is made.

There are three ways of searching for records using specified item contents:

- Item contents from the CRS are specified, and a search is made for a duplicate (FIND-3)
- The desired item contents are specified in the UWA, and a search is made for a record having these item contents
- The desired item contents are specified in the UWA or in the statement itself, and these contents are linked to a search expression in which parentheses, the logical operators AND, OR and NOT, and the relational operators >, < and = can be used. In this case, the item contents need not be specified completely. A mask can be used to mask out sections of an item for which no value is to be specified. UDS/SQL searches for all records which satisfy the conditions of the search expression and makes them members of an implicit dynamic set. Input of RESULT IN causes the records to also become members of an explicit dynamic set. The dynamic sets can be processed further using DML statements, applying different search criteria, for instance. The implicit dynamic set can be sorted in ascending or descending order (SORTED BY).

These access methods are described in detail below.

Specifying item contents from the CRS

```
{ FIND }
{ FETCH } DUPLICATE WITHIN set-name USING record-element-name,...
```

From the set occurrence which contains the CRS of set *set-name* UDS/SQL selects a member record which is different from the CRS and matches the CRS in the specified record elements.

Specifying item contents in the UWA

```

{ FIND }
{ FETCH } record-name WITHIN set-name[ CURRENT]

                USING record-element-name,...[OR { PRIOR }
                                                    { NEXT } ]

```

From an occurrence of set *set-name* UDS/SQL selects a record with record elements that match the values specified in the UWA. The set occurrence must first be selected and the record elements in the UWA supplied with the desired values. The set occurrences are selected in accordance with the selection method defined for this set in the schema. If the selection method is defined as SELECTION THRU LOCATION MODE OF OWNER, this can be deactivated by specifying CURRENT. The set occurrence can then be selected via the CRS.

set-name must not be a dynamic set.

By specifying OR PRIOR or OR NEXT, the user can have the immediately preceding or following record be selected as a hit if no record that matches the specified values exists. A special status code then indicates that the exact record was not found.

record-element-name,... must be defined as ASCENDING/DESCENDING/SEARCH KEY USING INDEX. The next record is determined by the sort sequence of the key.

If there is more than one record with the specified item contents and OR PRIOR is not used, each subsequent record can be read with the following statement:

```

{ FIND }
{ FETCH } DUPLICATE WITHIN set-name USING record-element-name,...

```

Note that the same record elements used to select the first record must be specified. This statement is explained above (see [“Specifying item contents from the CRS” on page 56](#)).

Combining item contents into a search expression

```

{ FIND }
{ FETCH }
record-name WITHIN setname-1[ CURRENT][ USING search-expression]

[ RESULT IN setname-2][ LIMITED BY set-name-3][ TALLYING itemname]

[ SORTED[ { ASCENDING } ] [ { BY } ]
  { DESCENDING } ]

record-element-name-1[[,]record-element-name-2]...

[[,][ { ASCENDING } ] [ { BY } ]
  { DESCENDING } ]

record-element-name-3[[,]record-element-name-4]...]]]]

search-expression ::= see page 52

```

This statement operates in the same way as direct access using search expressions at record type level (see [page 52](#)). Explanations will therefore be given only for those points in which this type of access differs from access at record type level.

The statement searches through a set occurrence of set *setname-1*, which must be defined beforehand. The set occurrences are selected in accordance with the selection method defined for this set in the schema. If the selection method defined there is SELECTION THROUGH LOCATION MODE OF OWNER, this can be deactivated by specifying CURRENT. The set occurrence can then be selected via the CRS.

- Inserting all member records of the set occurrence into a dynamic set
 - If no search expression is specified, all member records of the set occurrence become member records in a dynamic set.
- Searching through two set occurrences simultaneously
 - LIMITED BY *setname-3* can be used to give the name of a dynamic set to which the search for records by UDS/SQL will be limited. This means that UDS/SQL only takes account of those records which
 - are member records in the selected set occurrence of set *setname-1* and
 - are simultaneously member records in the specified dynamic set.

- Accessing a duplicate

When a collection of records is selected, UDS/SQL stores the results of the selection in a dynamic set. Only the first record found becomes the current record in the currency table. If no RESULT clause is used, the following statement must be entered for each further record to be called from the result of the selection:

$$\left. \begin{array}{l} \{ \text{FIND} \\ \text{FETCH} \} \end{array} \right\} \text{DUPLICATE WITHIN } \textit{set-name}$$

This statement always applies to the most recently selected collection of records since the implicit dynamic set always contains only the result of the last search expression.

set-name must designate the set which was processed with the previous search expression. UDS/SQL then delivers the record following the CRS in the dynamic set.

5.2.5 Sequential access at set level (FIND/FETCH-4)

This type of access is used to select a record on the basis of its position within the logical sequence of all records of a set occurrence. The order is determined in the schema by means of the ORDER clause for this set. In this order the owner record is at the same time the predecessor of the first and the successor of the last member record.

$$\left. \begin{array}{l} \{ \text{FIND} \\ \text{FETCH} \} \left\{ \begin{array}{l} \text{LAST} \\ \text{FIRST} \\ \text{NEXT} \\ \text{PRIOR} \\ \textit{integer} \\ \textit{item-name} \end{array} \right\} \left\{ \begin{array}{l} \textit{record-name} \\ \text{RECORD} \end{array} \right\} \text{WITHIN } \textit{set-name}$$

From the set occurrence which contains the CRS of set *set-name*, the user can select the first or the last record, the predecessor or the successor of the CRS, or the record whose position corresponds to a specified numeric value. If *record-name* is specified, UDS/SQL only selects the record if *record-name* indicates the member record type of the set.

5.2.6 Access to the CRS (FIND/FETCH-5)

```
{ FIND }
{ FETCH } CURRENT[ record-name] WITHIN set-name
```

This gives access to the CRS of set *set-name* and thus resets the currency information to an earlier status. The CRS of set *set-name* again becomes the current record in all relevant columns of the currency table or in those columns where updating has not been suppressed using RETAINING. If *record-name* is specified, UDS/SQL only accesses the CRS if *record-name* indicates the member record type of the set and the CRS is a member record.

5.2.7 Access to the owner of a CRS (FIND/FETCH-6)

```
{ FIND }
{ FETCH } OWNER WITHIN set-name
```

This statement searches for the owner record in the set occurrence of set *set-name*, which contains the CRS. If the CRS is already the owner record, only the currency table is updated since there is no need to search for the record once more in the database.

5.2.8 Sequential access at realm level (FIND/FETCH-4)

$\left. \begin{array}{c} \{ \text{FIND} \} \\ \{ \text{FETCH} \} \end{array} \right\}$	$\left. \begin{array}{c} \{ \text{LAST} \\ \text{FIRST} \\ \text{NEXT} \\ \text{PRIOR} \\ \text{integer} \\ \text{item-name} \} \right\}$	$\left. \begin{array}{c} \{ \text{record-name} \} \\ \{ \text{RECORD} \} \end{array} \right\}$	$\text{WITHIN } \text{realm-name}$
--	---	--	------------------------------------

This type of access can be used to select a record on the basis of the position which it occupies within the logical sequence of records in the collection of records in which the search is made. The records selected can be:

- for *record-name*: all records in the realm *realm-name* that belong to record type *record-name*. The realm name must be named in the schema in the WITHIN clause of this record type;
- for RECORD: all records of the realm *realm-name*.

In both cases the order of the records is determined by ascending order of database key values.

From the record selection, the user can select the last record, the first record, the next or prior record to the CRA or the record whose position corresponds to a specified numeric value.

5.2.9 Access to the CRA (FIND/FETCH-5)

$\left. \begin{array}{c} \{ \text{FIND} \} \\ \{ \text{FETCH} \} \end{array} \right\}$	$\text{CURRENT[} \text{record-name} \text{] WITHIN } \text{realm-name}$
--	--

This is used to reset the currency information to an earlier status. The CRA of realm *realm-name* again becomes the current record in all relevant columns of the currency table except those where updating has been suppressed using the RETAINING parameter. If *record-name* is specified, UDS/SQL only accesses the CRA if it belongs to the specified record type.

5.2.10 Transport the CRU completely or partially into the UWA (GET)

$$\underline{\text{GET}} \left[\begin{array}{l} \textit{record-name} \\ \textit{record-element-name}, \dots \end{array} \right]$$

This statement makes available in the application program the last record selected from the database - this record always becomes the CRU - by transporting it completely or partially to the UWA.

A selection of record elements can be specified if the complete CRU is not required. In as far as it transports the complete CRU into the UWA, the function is integrated into the FETCH statement. In this case the following applies:

FIND + GET = FETCH

5.2.11 Retrieve database key values (ACCEPT-1)

```
ACCEPT item-name-1 FROM [ { record-name }  
 { realm-name } ] CURRENCY  
 { set-name }
```

This format transfers a database key value from the currency table to *item-name-1*.

The following database key value is transferred to *item-name-1*:

- for *record-name*: database key value of the CRR
- for *realm-name*: database key value of the CRA
- for *set-name*: database key value of the CRS

If none of these names are specified, the database key value of the CRU is supplied.

The item *item-name-1* must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

If the database key value from the currency table contains a REC-REF > 254 and/or an RSQ > 2²⁴-1, *item-name-1* must be defined with USAGE IS DATABASE-KEY-LONG. Otherwise the value 0 is returned in *item-name-1* and the DATABASE-STATUS 15 102 is output.

If the specified current record is not known, the value 0 is likewise returned in *item-name-1*, and the DATABASE-STATUS 15 102 is output.

5.2.12 Retrieve realm (ACCEPT-2)

```
ACCEPT item-name-2 FROM [ { record-name }
                        { set-name }
                        { item-name-3 } ] REALM-NAME
```

This format transfers the name of the realm to which the specified currency information refers to *item-name-2*.

The following realm name is transferred to *item-name-2*:

- for *record-name*: name of the realm to which the CRR belongs
- for *set-name*: name of the realm to which the CRS belongs
- for *item-name-3*: name of the realm containing the record whose database key value is located in *item-name-3*

The item *item-name-3* must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

If you want to search for a record whose database key value contains a REC-REF > 254 and/or an RSQ > 2²⁴-1, *item-name-3* must be defined with USAGE IS DATABASE-KEY-LONG.

If neither *record-name*, *set-name* nor *item-name-3* are specified, the realm name to which the CRU belongs is supplied.

5.3 Modifying data

Data can be modified as follows:

- storing new records and connecting them into set occurrences (STORE)
- connecting records into set occurrences (CONNECT, MODIFY)
- removing records from set occurrences (DISCONNECT, MODIFY)
- modifying item contents (MODIFY)
- erasing records and set relationships (ERASE)

5.3.1 Store a record in the database and connect it into set occurrences (STORE)

```

STORE record-name [ RETAINING CURRENCY FOR { MULTIPLE
[ REALM ] [ RECORD ] [ { SETS
set-name, ... } ] } ]

```

STORE transfers all item contents of record type *record-name* from the UWA into the database. The items must be provided with the desired values beforehand in the UWA. Items of the record type which are not in the record area of UWA are automatically filled with binary zeros by UDS/SQL when they are stored. (The latter is not the case if the records have been stored in compressed form using CALL DML; see STORE2 and STOR2L on [page 258](#)). The record becomes the current record in all relevant columns of the currency table.

In each set for which the record type is defined as owner in the schema, the stored record becomes the owner of a new empty set occurrence. If the set occurrence population for the set was defined as greater than 0 for the set in the SSL, then the associated tables are also set up.

In addition the record is inserted into all sets for which the record type is defined as AUTOMATIC member in the schema. In this case the set occurrences which are to accept the record and where necessary the insertion points within the set occurrences must be defined beforehand.

When the record type is distributed over more than one realm in accordance with the schema, the associated AREA-ID item must also previously have been supplied with the name of the required realm (see the “[Design and Definition](#)” manual, WITHIN clause) if the record type is not the member record type of a distributable list. When a member record of

a distributable list is stored, the AREA_ID is ignored. Free space for new level 0 pages which are to be created in the distributable list is searched for in the current favored realm ("preferred realm").

5.3.2 Connect a record into a set occurrence (CONNECT)

```

CONNECT [ record-name ] TO { set-name-1, ... }
                        { ALL
[RETAINING CURRENCY FOR { set-name-2, ... }
                        { SETS

```

The CONNECT statement operates on the CRU.

If a record type has been defined as a MANUAL member of a set, a record of this record type can only be inserted into this set with a CONNECT statement. The record must already be stored in the database.

The CONNECT statement operates on the following types of set membership of member records:

- MANDATORY MANUAL
A record is inserted as a MANUAL member (since it is a MANDATORY member record, it cannot be removed from the set occurrence using DISCONNECT, but only be transferred to a different set occurrence or deleted by means of ERASE.
- OPTIONAL AUTOMATIC
A record which has already been a member record of an occurrence of the set is inserted again (this AUTOMATIC member record was inserted with STORE and could be removed again with DISCONNECT since it is an OPTIONAL member record).
- OPTIONAL MANUAL
A record is inserted for the first time after a STORE statement (MANUAL member record) or is inserted again as an OPTIONAL member record after a DISCONNECT statement.

In *record-name* the user names the record type with one of the 3 types of set membership given in *set-name-1,...*. The CRU must belong to the records of the specified record type. If no record name is specified the record type is determined from the description of the first set specified.

The set occurrence is selected via the relevant CRS. After execution of the CONNECT statement, the CRU becomes the CRS of each set into which it was inserted. If RETAINING is used, the updating of currency table information is suppressed for *setname-2,....* If SETS is specified, the currency information for all sets involved remains unchanged.

5.3.3 Disconnect existing set relationships (DISCONNECT)

Removing a record from set occurrences

```
DISCONNECT [ record-name ] FROM { set-name,... }
                                     { ALL }
```

This DISCONNECT statement operates on the CRU. It removes the CRU from the set occurrences of all specified sets. The record type in these sets must be optional member record type (i.e. set membership OPTIONAL MANUAL or OPTIONAL AUTOMATIC). The record type *record-name* must match the record type of the CRU.

If no record name is specified, the record type is determined from the description of the first set specified.

The DISCONNECT statement does not modify the currency table information.

Removing all member records from dynamic sets

```
DISCONNECT ALL FROM set-name,...
```

This format operates on all records of *set-name,....*

If all member records are to be removed from a dynamic set, this format of the DISCONNECT statement should be used.

set-name,... must be dynamic sets.

With this format it is also possible to specify result sets of a FIND-7 statement. The DISCONNECT statement does not modify the currency table information.

5.3.4 Modify the CRU or connect it into another set occurrence (MODIFY)

```

MODIFY { record-name
        { record-element-name, ... } } [ { INCLUDING } { ALL
        { ONLY } } { set-name-1, ... } MEMBERSHIP]
      [ RETAINING CURRENCY FOR { SETS
        { set-name-2, ... } } ]
    
```

MODIFY can modify item contents of the CRU or disconnect the CRU from one set occurrence and connect it into another set occurrence of the same set.

If only the first-named function is required, neither ONLY nor INCLUDING may be specified.

If ONLY is specified, UDS/SQL executes the second function only.

If INCLUDING is specified, UDS/SQL executes both functions.

The functions are described separately below.

Modifying item contents of the CRU

```

MODIFY { record-name
        { record-element-name, ... } } [ RETAINING CURRENCY FOR { SETS
        { set-name, ... } } ]
    
```

UDS/SQL transfers, from the UWA to the CRU, either all item contents of record type *record-name* or just the item contents of the specified record elements. The desired contents for the items to be modified must be supplied by the user beforehand in the UWA.

If MODIFY is used to modify key values, UDS/SQL automatically updates all the associated access paths such as hash areas and tables, plus the DBTT. In particular the order of records within a set occurrence may be changed.

The database key value of a record cannot be changed even if the associated database key item is overwritten with a new value.

Connecting the CRU into another set occurrence

```

MODIFY record-name ONLY { ALL
                               { set-name, ... } } MEMBERSHIP
                               [ RETAINING CURRENCY FOR { SETS
                               { set-name, ... } } ]

```

In those sets in which the CRU is already a member of a set occurrence, the CRU can be assigned to another owner record. The CRU can be removed from all its set occurrences and inserted into another set occurrence of the same set or a selection of sets in which this is to occur can be named by the user.

Before the CRU can be reconnected in a set, the user must first define the set occurrence into which the CRU is to be connected and also, if necessary, its position within the set occurrence.

5.3.5 Delete records and their set relationships (ERASE)

```

ERASE record-name [ { PERMANENT
                        { SELECTIVE
                        { ALL
                        } } } MEMBERS ]

```

The ERASE statement operates on the CRU. The record type of *record-name* must be the same as the record type of the CRU.

The ERASE statement removes the CRU from all set occurrences in which it is a member record and deletes it. After the deletion the storage space is available again. The database key value does not become available again until after termination of the transaction.

When a record is deleted its relationship to other records must also be taken into consideration at the same time.

The entry of the CRU in the currency table is deleted.

The other entries in the currency table are marked as deleted. The option of making sequential searches (FIND/FETCH NEXT) is retained.

All record types and set relationships referenced either directly or indirectly via the ERASE statement, must be contained in the subschema. All realms which are named in the WITHIN clause of the record types involved must be present in the subschema.

5.3.6 Correlation between type of set membership and data-modifying statements

Type of set membership	Set membership is established by		Set membership is canceled by		Switches between set occurrences
	STORE	CONNECT	ERASE	DISCONNECT	MODIFY
MANDATORY AUTOMATIC	YES	NO	YES	NO	YES
MANDATORY MANUAL	NO	YES	YES	NO	YES
OPTIONAL AUTOMATIC	YES	YES	YES	YES	YES
OPTIONAL MANUAL	NO	YES	YES	YES	YES

Table 10: Combinations of type of set membership and statements for modifying data

5.4 Protecting records

Pages which are not protected against use by other transactions either at realm level (by USAGE-MODE EXCLUSIVE or PROTECTED RETRIEVAL) or at page level (automatic protection mechanism for updated pages) have the option of data protection at record level:

- activate extended record protection (KEEP)
- deactivate extended record protection (FREE)

Note that the protection for the page containing the CRU (shared lock) is always present.

In order to protect the record beyond this period, however, record protection must be extended by using KEEP.

5.4.1 Activate extended record protection (KEEP)

KEEP

This statement protects the CRU from access by another transaction even when it is no longer the CRU. This KEEP status can only be cancelled by FINISH (termination of transaction) or FREE.

If the realm to which the record belongs has been opened with USAGE-MODE UPDATE, the KEEP statement will create exclusive access rights for the page in which the record (CRU) is located. In other words, information from this page will only be accessible to the transaction involved.

If the realm to which this record belongs has been opened with USAGE-MODE RETRIEVAL, the KEEP statement sets a shared lock (see [page 34](#)) for the page in which the record (CRU) is located. This means that other transactions can only obtain read access to a page.

This page access protection remains in effect until the next FREE or the end of the transaction.

5.4.2 Deactivate extended record protection (FREE)

```
FREE[ ALL]
```

This statement cancels the KEEP status.

If ALL is not specified, the KEEP status is only cancelled for the CRU. It is only possible to work without ALL if the record that was given extended protection by KEEP has been made the CRU again.

If ALL is specified, all the records that were provided with extended protection by KEEP are freed again for other transactions. The normal CRU record protection is, however, retained in both cases.

5.5 Testing set memberships in the program (IF)

5.5.1 Testing the set membership of the CRU

```
IF[ NOT][ setname] {
  OWNER
  MEMBER
  TENANT
} { statement-1 } [ ELSE { statement-2 } ] _
```

The following individual tests are performed:

- with OWNER: whether the CRU possesses member records;
- with MEMBER: whether the CRU is a member record in a set occurrence;
- with TENANT: whether the CRU is the owner of a non-empty set occurrence or a member record in a set occurrence.

If *set-name* is specified, only the named set is tested; dynamic sets may not be specified. If *set-name* is not specified, all sets of the subschema are tested in which the record type of the CRU is owner or member.

The result of the test is “TRUE” if the condition is fulfilled. In this case a branch is made to *statement-1* or NEXT SENTENCE.

If the result is “FALSE”, a branch is made to *statement-2* or NEXT SENTENCE.

If NEXT SENTENCE is specified, a branch is made to the statement following the IF statement.

If a database exception condition arises (see [section “Database exception conditions” on page 117](#)) during execution of an IF statement, the condition “FALSE” is assumed, and a branch is made to *statement-2* or NEXT SENTENCE.

5.5.2 Testing a set occurrence for member records

```
IF setname IS[ NOT] EMPTY { statement-3 } [ ELSE { statement-4 } ]_
```

set-name is used to select the set occurrence via the CRS. A dynamic set may not be specified.

The result of the test is “TRUE” if the condition is fulfilled. In this case a branch is made to *statement-3* or NEXT SENTENCE.

If the result is “FALSE”, a branch is made to *statement-4* or NEXT SENTENCE.

If a database exception condition arises (see [section “Database exception conditions” on page 117](#)) during execution of an IF statement, the condition “FALSE” is assumed, and a branch is made to *statement-4* or NEXT SENTENCE.

6 Using DML

This chapter deals with the special features of COBOL DML and CALL DML and shows their different uses.

The chapter contains a description of the functions in which COBOL DML and CALL DML differ. The relevant formats can be found on pages [129](#) and [197](#).

6.1 Structure of the COBOL/CALL DML programs

COBOL DML program

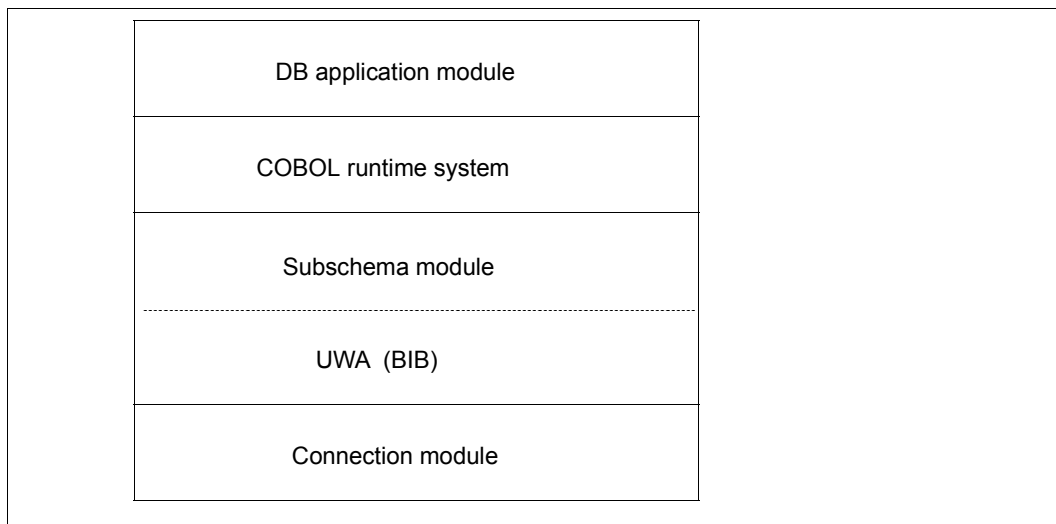


Figure 5: Structure of a COBOL DML program

CALL DML program

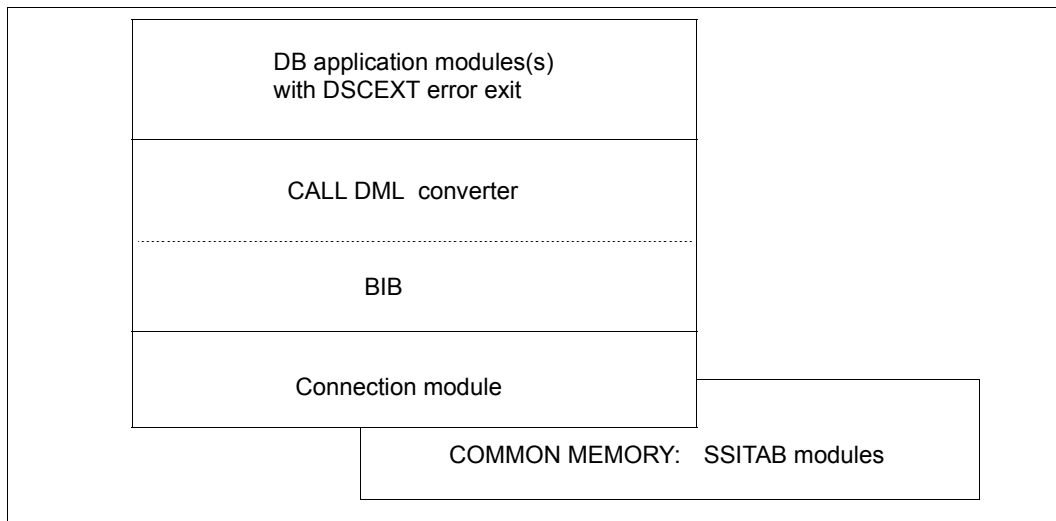


Figure 6: Structure of a CALL DML program

6.1.1 COBOL DML

COBOL DML is used in 3 divisions of a COBOL program:

- the ID DIVISION,
- the DATA DIVISION, and
- the PROCEDURE DIVISION

The DML statements are included in the range of functions of the ANSCOBOL compilers COBOL85 and COBOL2000.

The rules and formats for DML statements can be found in [chapter “COBOL DML reference section” on page 129](#).

Program preparation

The prerequisites for an executable COBOL DML program are:

- One or more application modules
- One or more subschemas which are entered in the COBOL Subschema Directory (COSSD).

Program structure

One or more databases can be used at the same time.

The examples that follow illustrate how to work with a single database (mono-DB operation) and multiple databases (multi-DB operation).

Mono-DB operation

The structure of a DB application program in mono-DB operation can be seen from the illustration below:

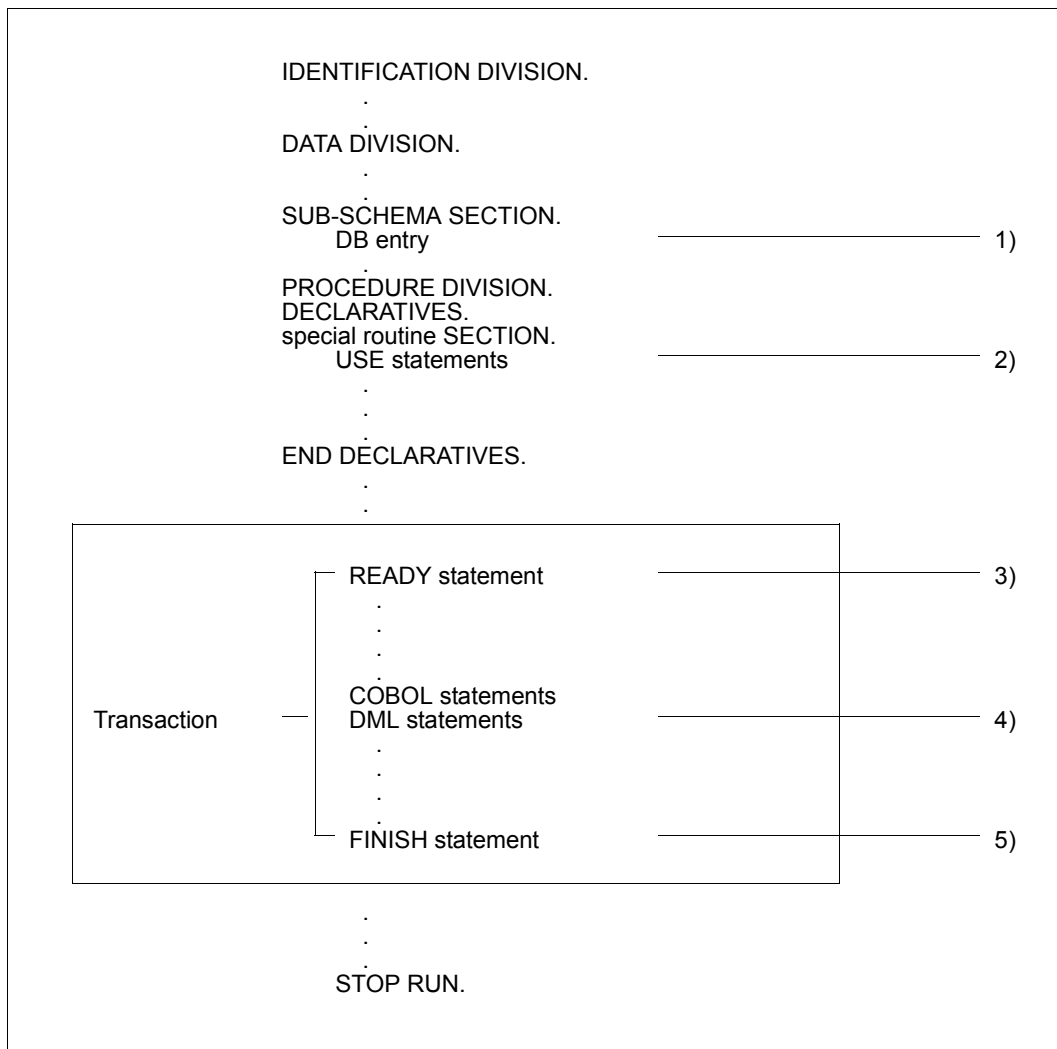


Figure 7: Structure of a COBOL DML program in mono-DB operation

- 1) The DB entry in the SUB-SCHEMA SECTION specifies the name of the subschema which the program is to access. The COBOL compiler uses the DB entry to find the subschema in the COSSD. This subschema is compiled into a module which contains the UWA and the user communication area. The compiler then stores the subschema module in the common area of the COBOL program.

All subschema-related information that is required by the DBH to access the database is thus made available.

- 2) The USE statement can be used to define statement sequences within the DECLARATIVES which will be executed if a DB exception condition occurs.
- 3) The READY statement secures the users' rights of access to the realms required for their processing applications. It represents the start of a transaction.
- 4) DML statements permit the database to be accessed (e.g. STORE, FIND).
- 5) FINISH terminates processing of all resources opened by READY (realms and data pages). This terminates the transaction.

Multi-DB operation

In COBOL DML only one subschema of a database can be specified in the DB entry of the SUB-SCHEMA SECTION of an application module. If there is a requirement to access several databases a separate COBOL module must be generated for each database.

Example 1

A control module is generated to coordinate two DB application modules.

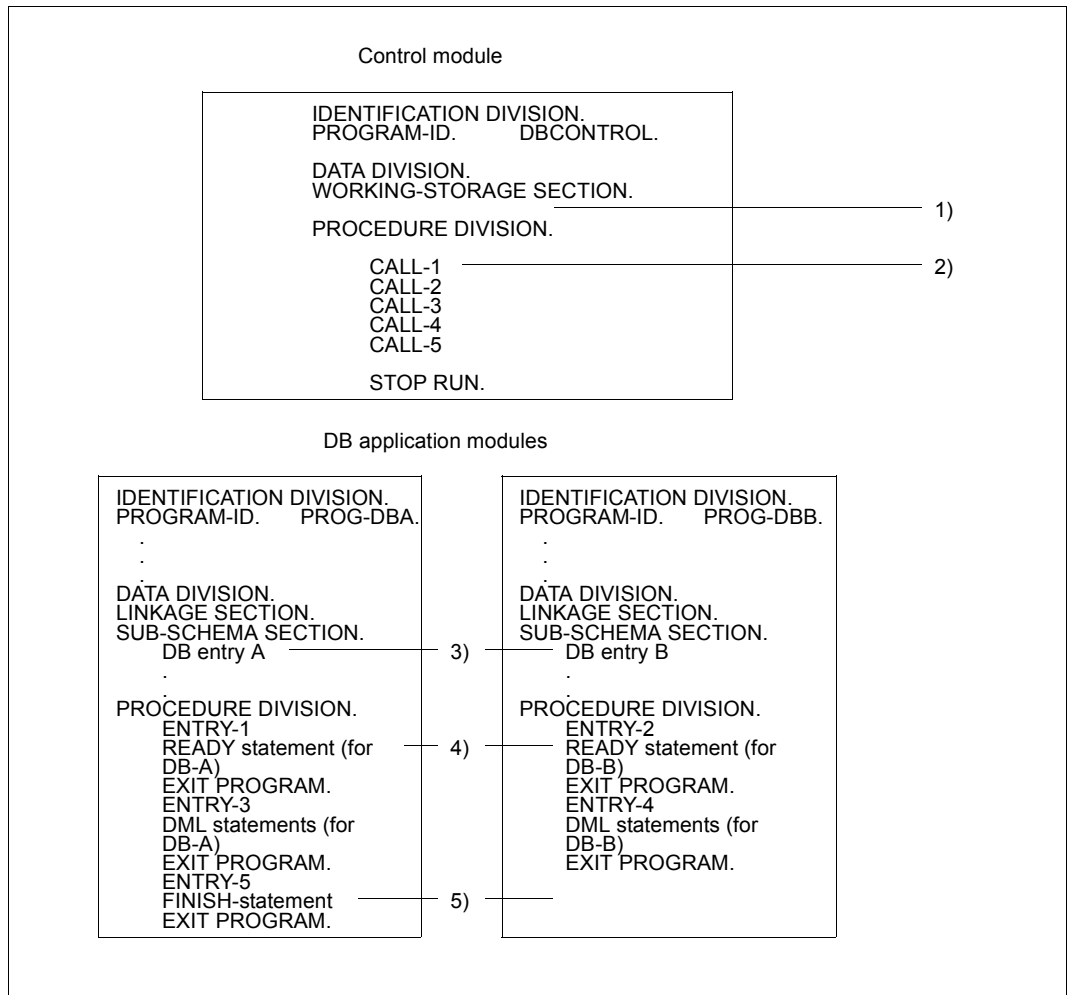


Figure 8: Structure of COBOL DML programs for multi-DB operation

- 1) The control module does not contain a SUB-SCHEMA SECTION.
- 2) The entry points into the DB application modules are defined via CALLs.
- 3) Each module specifies a subschema of a database.
- 4) The first READY statement executed opens a transaction and simultaneously opens a processing chain. Each further READY statement in a different DB application module opens another processing chain.
- 5) In multi-DB operation there is only one FINISH statement; this closes all opened processing chains and at the same time terminates the transaction. It makes no difference from which DB application module this FINISH statement is issued (in this example it is issued by PROG-DBA).

Example 2

The control is integrated in the two application modules.

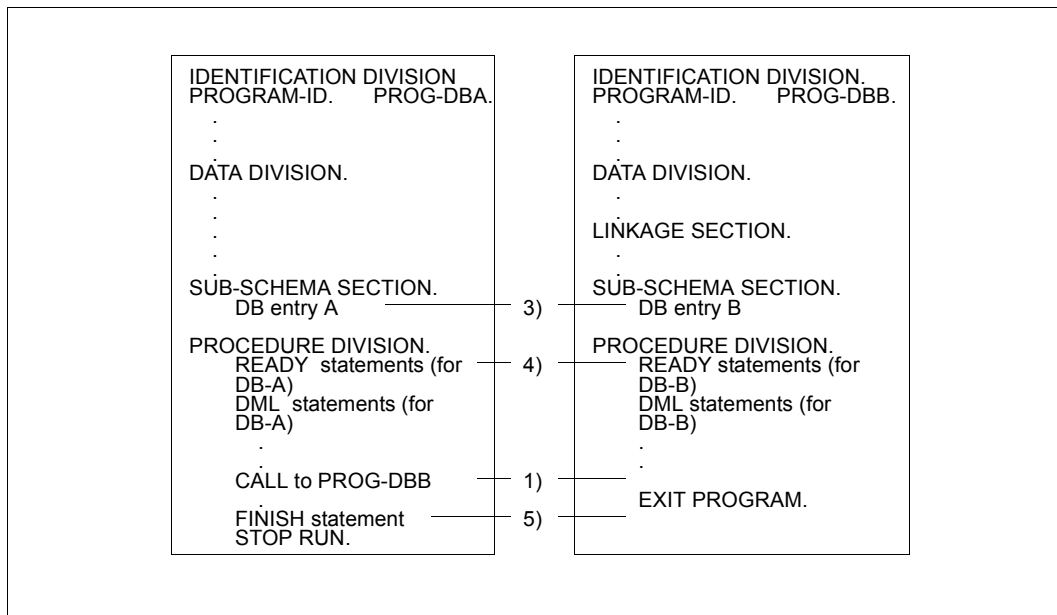


Figure 9: As [figure 8](#) but without control module

1) The second program is called from the first DB application module and assumes control until the EXIT PROGRAM statement returns control to the calling module.

3), 4), 5)

See explanations for “[Example 1](#)” on page 80.

6.1.2 CALL DML

CALL DML makes available DML features in a form which is independent of the programming language used.

The rules of DML - in as far as they are relevant to database access - are also valid for CALL DML. The formulation of CALL DML calls is basically different from that of COBOL DML. The COBOL DML statements which are processed by the COBOL compiler are omitted and are replaced by the subroutine call "CALL" and a number of parameters. These parameters (function code, function option, secondary option, etc.) describe the call and transfer the database name and the data (user information, record area).

This transfer of call information necessitates certain functional differences compared with COBOL DML: CALL DML requires a special converter. The CALL DML converter does not translate the database calls into the form in which they can be processed by the database handler until runtime, when it also checks them for correct syntax and validity.

The parameters of the CALL DML calls themselves, however, also cannot be entered by the user or read in from a file until runtime. Additional support for this process is provided by the LOOKC function, which is not contained in COBOL DML; this function allows access to structure information on the subschema.

CALL DML programs can be executed under openUTM.

The relevant subschema need not be known at the time of CALL DML application program compilation. No functions are required which exceed the normal compiler range.

The transfer areas for records and parameters are defined by the user.

The addresses of the required transfer areas are transferred as part of each CALL DML call; these areas may thus be changed for every call or reversed at the user's discretion.

Generating a CALL DML program

An executable CALL DML program contains the following components:

- one or more application modules,
- UDS/SQL connection modules, and
- one SSITAB module per subschema, created with the BCALLSI utility (see the ["Creation and Restructuring"](#) manual).

Program structure

CALL DML allows more than one database to be accessed from the same module (multi-DB operation). The example below shows the program structure for multi-DB operation.

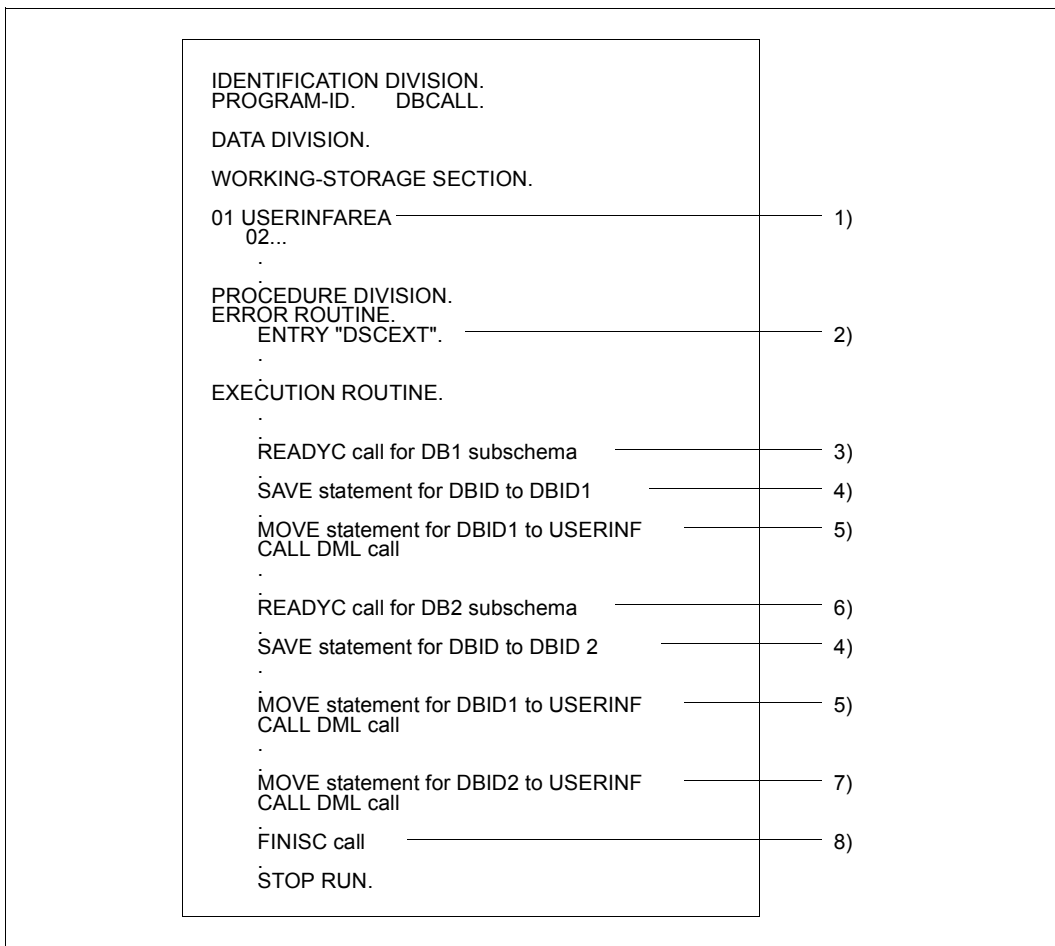


Figure 10: Structure of CALL DML programs for multi-DB operation

- 1) The transfer areas for records and parameters must be defined in the program in CALL DML. The parameter area user-information must be generated in accordance with a preset format.
- 2) The program must contain an error routine with the ENTRY name of DSCEXT (see [section "CALL DML error handling routine DSCEXT" on page 124](#)).
- 3) The CALL with function code name READYC opens the transaction. The first READYC call simultaneously opens the processing chain for the specified subschema of database DB1. A unique database identifier is passed to the user via the user information area.
- 4) The database identifier must be saved in an item.
- 5) The database identifier must be transferred with each CALL DML call (except READYC), since all CALL DML calls are assigned to their respective databases via the database identifier.
- 6) The second READYC call opens access to the database DB2.
- 7) Database DB2 is assigned to the following CALL DML call only when database identifier DBID2 is transferred.
- 8) FINISC terminates the processing of all resources reserved by READYC. This simultaneously terminates the transaction.

6.2 Special features of COBOL DML

Specifying a key for use of the subschema (PRIVACY)

```
[PRIVACY. PRIVACY KEY FOR COMPILE IS literal.]
```

If a lock against access to the subschema is defined by the PRIVACY LOCK FOR COMPILE clause in the subschema being used, an appropriate key, called a PRIVACY KEY, must be specified in the COBOL program during compilation. The PRIVACY KEY clause is given in the IDENTIFICATION DIVISION.

Assigning the subschema and setting up the communication area (DB entry)

The DB entry is entered in the Data Division of the COBOL program as part of the SUBSCHEMA SECTION.

```
DB subschemaname WITHIN schemaname.
```

The DB entry specifies which subschema is to be used by your COBOL DML program.

This entry is evaluated both during compilation and when executing the program.

The subschema determines the record area, which not exceed 65 535 bytes in length. The actual length of the record area is, in turn, essentially determined by the length of the record types contained in the underlying subschema. The record area holds one record of each record type in the subschema and also includes the IMPLICITLY-DEFINED-DATA-NAMES, i.e. the area for implicitly-defined items of the database (e.g. the AREA-ID items of WITHIN clauses of the schema).



Record areas of up to 65 535 bytes in length can be used with COBOL2000 or COBOL85 as of V2.2C21, provided the statement SUBSCHEMA FORM IS OLD was not specified during compilation of the subschema (see the “[Creation and Restructuring](#)” manual).

During compilation, the user work area (UWA) is generated. The contents of this UWA are a component of the application program and depend on the subschema called. Before the COBOL DML program is compiled with the COBOL compiler, the database to be processed must be assigned with the link name DATABASE:

```
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname
```

The following areas are always generated in the UWA:

- SYSTEM-COMMUNICATION-LOCATIONS (contains the COBOL special registers; see [page 88](#))
- RECORD AREA
- PRIVACY RECORD (contains the items for the BPRIVACY locks):

The PRIVACY RECORD need not be filled by the COBOL program; existing entries, if any, are ignored by UDS/SQL. The PRIVACY data is transferred to UDS/SQL by openUTM or BS2000 outside the COBOL program.

All items can be referenced in a COBOL program by using the names generated in the UWA.

If several modules are linked together to form a program, each module can contain one DB entry.

The DB entry leads to the execution of a start routine which calls the Database Handler. At application program runtime when the READY statement is executed, the name of the subschema is taken from the DB entry and transferred to the DATA Base Handler which then loads a corresponding Subschema Information Area SSIA. The SSIA is the internal representation of the subschema used by the DBH. All SSIAs are stored in the Database Directory (DBDIR).

Besides the SSIAs, each database also has its own Schema Information Area (SIA). This SIA can be accessed by the DBH at any time in order to obtain the schema information required to execute DML statements.

COBOL special registers

The COBOL special registers are storage areas generated by the COBOL compiler. These special registers are only set up once for each DB entry. The user can access the values of the special registers in the COBOL DML program.

The registers have the following formats:

DATABASE-REALM-NAME	PIC X(30)
DATABASE-RECORD-NAME	PIC X(30)
DATABASE-SET-NAME	PIC X(30)
DATABASE-STATUS	PIC 9(5)

The DBH modifies the values of the COBOL special registers following DML statements and enters the current values (see [“Using special registers \(COBOL DML\) or the user information area \(CALL DML\)” on page 117](#)).

Transferring a database key value (SET)

```
SET item-name-1,... TO item-name-2
```

This statement transfers a database key value into one or more items. All items used must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG. The contents of *item-name-2* are transferred to *item-name-1*,...

More information on transferring database key values with the SET statement can be found in [chapter “COBOL DML reference section”](#) under [“SET” on page 187](#).

Describing error handling routines (USE)

```
USE FOR DATABASE-EXCEPTION[ ON { OTHER  
                                { literal-1,... } ]+.
```

The USE statement initiates routines for handling database exception conditions (see [section “Database exception conditions” on page 117](#)). These routines may only be specified in the DECLARATIVES of a COBOL DML program.

Database exception conditions do not occur only as a result of errors. The DBH fills the COBOL special registers after every COBOL DML statement. By checking the DATABASE-STATUS with the USE statement, you could, for example, also determine whether or not a duplicate record has been found. You can program specific routines depending on the type of database exception condition (see [table 15 on page 119](#)).

Different USE statements can be defined to control the program. This can be achieved by two methods:

- by interrogating the database exception conditions that occur when a DML statement cannot be completed successfully
- by using a specific exception condition as a switch for the program logic.

The DECLARATIVES of a COBOL DML program are executed after each DML statement that has created a database exception condition. This helps to reduce the programming overhead.

Assigning the COSSD file for compiling a COBOL-DML program

When a COBOL-DML application program is compiled, the COBOL compiler must read the COSSD file of the database concerned. You must consequently assign the COSSD file to the COBOL compiler in one of the following ways:

- Assignment via LINK-NAME=UDSCOSSD
- Assignment via LINK-NAME=DATABASE

Assignment via LINK-NAME=UDSCOSSD

This procedure is only supported by the COBOL2000 compiler Version 1.4 or higher.

The COBOL compiler is explicitly assigned the COSSD file using the command

```
/ADD-FILE-LINK      LINK-NAME=UDSCOSSD, -
/                   FILE-NAME=[:catid:][$userid.]dbname.COSSD
```

Here `:catid:` and `$userid` are the catalog ID and user ID under which the COSSD file is cataloged. If `:catid:` or `$userid` is not specified, the file name is completed according to the standard rules of BS2000.

The COSSD file must be cataloged under the name specified in the command as a COSSD file is not searched for at another position when an error occurs.



This procedure is essential if multiple COSSD files with the corresponding database name exist in all catalogs which can be accessed locally from the user ID.

If a UDS/SQL pubset declaration exists, it is **not** taken into account.

Example of a command sequence:

```
/ADD-FILE-LINK LINK-NAME=UDSCOSSD,FILE-NAME=dbname.COSSD
/START-COBOL2000-COMPILER -
/SOURCE=cobolsource, -
/COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=LLM), -
/MODULE-OUTPUT=*LIBRARY(LIBRARY=library-1,ELEMENT=element)
```

Assignment via LINK-NAME=DATABASE

This procedure is supported by all COBOL2000 and COBOL85 compilers.

The COBOL compiler is notified of the database name using the command

```
/SET-FILE-LINK      LINK-NAME=DATABASE, -
/                  FILE-NAME=[:catid:][$userid.]dbname
```

If a `:catid:` is specified in the SET-FILE-LINK command, it is ignored. The COBOL compiler then searches for a COSSD file with the name `dbname.COSSD` in all catalogs which can be accessed locally from the user ID which was specified explicitly in the SET-FILE-LINK command or which was supplemented by BS2000.



This procedure can be used only if only one COSSD file with the relevant database name exists in all catalogs which can be accessed locally from the user ID.

If a UDS/SQL pubset declaration exists, it is **not** taken into account.

If an assignment for LINK=UDSCOSSD also exists, only the procedure for LINK=UDSCOSSD is used.

Example of a command sequence:

```
/SET-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=dbname
/START-COBOL2000-COMPILER -
/SOURCE=cobolsource, -
/COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=LLM), -
/MODULE-OUTPUT=*LIBRARY(LIBRARY=library-1, ELEMENT=element)
```

Error handling

If the COBOL compiler cannot access the COSSD file, it issues the message ERROR ACCESSING SUB-SCHEMA.

This message is also output if, in the event of assignment via LINK-NAME=DATABASE, the file exists more than once in different pubsets. In this case use specific assignment of the COSSD file via LINK-NAME=UDSCOSSD.

6.3 Special features of CALL DML

Conversion at program runtime

The CALL DML calls are not converted into the form required by the DBH until program runtime. Nor are they checked for validity until then. This means that CALL DML is much more flexible in use since it is also possible to generate the calls according to the data entered during the program run.

Abbreviated record format

The strict subschema record format can be replaced by an abbreviated record format. Then space must be provided in the program only for the required items. The abbreviated record format is possible for the calls GETC, MODIF2, STORE2 and STOR2L; the format is accessed via the secondary option parameter VAR. This only affects the length of the record in the database if the record is defined with COMPRESSION ALL (see the “[Design and Definition](#)” manual, Compression).

IMPLICITLY-DEFINED-DATA-NAMES

In the Schema DDL, item names are declared for items which are designed to transfer data to the DBH outside the records. These include the database key item of the clause LOCATION MODE DIRECT/DIRECT-LONG and the AREA-ID item of the WITHIN clause (see the “[Design and Definition](#)” manual).

CALL DML does not refer to these declared names; a special parameter is available for the transfer of the corresponding data (special parameter 2: implicitly defined data area in FIND/FETCH and STORE).

Sets with SET SELECTION THROUGH LOCATION MODE OF OWNER

Sets defined using the SET OCCURRENCE SELECTION clause THRU LOCATION MODE OF OWNER are handled like sets defined using the CURRENT OF SET clause.

User information area

Communication with the DBH is via the user-information parameter area, which must be generated in a fixed format and must be specified in each CALL. Examples of its contents are items for the special registers, the database status, transfer items for a database key value and the counter for the TALLYING function of FIND7A/FTCH7A.

Variable items

CALL DML allows variable items to be processed without problems as long as a suitable programming language is used, i.e. it allows variable length records to be stored. A binary value must be supplied by the user for the length item associated with the variable item.

Subschema information

The maximum permitted length for the record area (RECA) defined by the subschema in CALL DML is 65 535 bytes. The actual length of the record area is essentially determined by the length of the record types contained in the underlying subschema. The record area holds one record of each record type in the subschema and also includes the IMPLICITLY-DEFINED-DATA-NAMES, i.e. the area for implicitly-defined items of the database (e.g. the AREA-ID items of WITHIN clauses of the schema).

If the statement SUBSCHEMA FORM IS OLD was specified when compiling the subschema, the maximum permitted length for the corresponding record area is 61 328 bytes.

The description of the subschema used must be made available to CALL DML in the form of special modules; these have a different basic structure from the subschema modules used for COBOL DML. These modules are designated SSITAB modules (Subschema Information Table) modules. They are generated by utility routine BCALLSI (see the [“Creation and Restructuring”](#) manual).

A CALL DML application program can call various subschemas. Subschema names must be unique in the first 6 bytes to guarantee that the SSITAB modules cannot be confused.

Only elementary items can be accessed via an item name. Indexed items cannot be named in CALL DML. This is a programming consideration but does not involve any restriction on the range of functions when compared with COBOL DML.

Testing the structure of the subschema (LOOKC)

If the user wishes to change the parameter list in CALL DML calls, precise information about the subschema is required.

The following information can be obtained by using:

CALL DML, LOOKC...

a name	<ul style="list-style-type: none"> – one or more elements – the owner/member of a set
a realm	<ul style="list-style-type: none"> – realms in which records of a specific record type can be stored
a record type	<ul style="list-style-type: none"> – one or more record types – the owner/member record type of a specific set
a set	<ul style="list-style-type: none"> – one or more sets – one or more sets in which a specific record type is an owner or member record type
an item	<ul style="list-style-type: none"> – one or more items of a specific record type
a key	<ul style="list-style-type: none"> – one or more keys of a specific set – one or more keys of a set in which a specific item is contained – one or more keys in which a specific item is contained
the items of a key	<ul style="list-style-type: none"> – one or more items of a specific key

Table 11: Structure information obtained by LOOKC

6.4 Linking, loading and starting a UDS/SQL-TIAM application program

6.4.1 Basic aspects

The principle of dynamic loading

Only one “version-independent” connection module (UDSLNKI, UDSLNLK or UDSLNKA) is permanently linked into the UDS/SQL application program for connection to the independent or linked-in DBH (see also [section “Linking UDS/SQL-TIAM applications” on page 100](#)). All other connection modules of the product which are required for the execution of a UDS/SQL application as well as application-specific data modules (SSITAB, PLITAB) are dynamically loaded by the DBL (dynamic binder loader) at the time of execution. This concept has the advantage that UDS/SQL applications do **not** have to be relinked when the UDS/SQL version is changed. If longer-term relinking becomes necessary, this can be done later and successively for individual applications.

Before the start of the application program, the libraries containing the product and data modules to be dynamically loaded must be available or assigned. The various dynamic-loading strategies (and thus assignment techniques) are described below, arranged according to the product and data modules.

Loading the UDS/SQL product modules dynamically

In the standard case, UDS/SQL is installed with IMON and managed centrally in the Software Configuration Inventory (SCI). Several product versions can be installed concurrently and several UDS/SQL subsystem versions can be preloaded. The BS2000 command SELECT-PRODUCT-VERSION is available for assigning the product library from which the modules are to be dynamically loaded and for selecting the appropriate subsystem version:

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=product,VERSION=version,SCOPE=*TASK
```

product

Name of the delivery unit (e.g. UDS-SQL or UDS-D)

version

The version number specified should always include the release status and correction status (e.g. 02.8A00). The version number specified with SELECT-PRODUCT-VERSION does not need to be permanently assigned in each user procedure, but can be set to a centrally maintained value by means of a job variable or S variable.

In the “[Database Operation](#)” manual, section “Using multiple UDS/SQL versions concurrently”, you will find examples of start procedures which include the variable structuring of the version number specified via SELECT-PRODUCT-VERSION with the aid of job variables.

When calling the application program into which the version-independent connection module for the **independent** DBH is linked, you select the version of UDSBCCON, the version-dependent connection module to be dynamically loaded, with SELECT-PRODUCT-VERSION.

When calling the application program into which the version-independent connection module for the **linked-in** DBH is linked, you select the version of LCCONCT, the version-dependent connection module to be dynamically loaded, with SELECT-PRODUCT-VERSION.

The actual UDS/SQL coding in the version appropriate for the connection module is used as a preloaded subsystem or is dynamically loaded from the corresponding SYSLNK library of the product UDS/SQL.

The abbreviation “SCI” is used for this dynamic-loading procedure in the error messages of the product.

The SELECT-PRODUCT-VERSION command is also recommended when for long periods only one UDS/SQL version is used on the system.

If SELECT-PRODUCT-VERSION is not specified, with the first dynamic-loading operation in the version-independent module an attempt is made to load from any existing private installation (see [page 97](#)). If no private installation is found, the standard version is loaded from the SCI.

Using UDS-D:

When using UDS-D (only possible with the independent DBH), you should use an additional SELECT-PRODUCT-VERSION command to specify the UDS-D version that corresponds to the UDS/SQL version if UDS-D is used as a subsystem (see the Release Notice if applicable). Here too it is recommended that you also use SELECT-PRODUCT-VERSION when for long periods only one UDS-D version is used on the system.

Private installation

If the product UDS/SQL is not managed in the SCI throughout the system, the modules can be dynamically loaded from libraries of a private installation.

The libraries from which the modules of the product UDS/SQL and, if applicable, the products UDS-D and UDSKDBS are to be dynamically loaded in the desired version, are explicitly assigned with the ADD-FILE-LINK command. The following link names are available for the assignments:

\$UDSLIB	Modules of the product UDS/SQL (independent and linked-in DBH)
\$UDSDLIB	Modules of the product UDS-D (only with independent DBH)
\$UDSKLIB	Modules of the product KDBS (only with linked-in DBH)

The abbreviation “\$UL” is used for this procedure in the error messages of the product.

For reasons of compatibility, the use of a UDS.MODLIB, possibly with TASKLIB assignment, continues to be supported for dynamically loading the modules if the products are not provided via SCI or via the link names \$UDSLIB, \$UDSDLIB and \$UDSKLIB. The abbreviation “TSK” is used for this procedure in the error messages of the product.

The SELECT-PRODUCT-VERSION commands for the products UDS/SQL and, if applicable, UDS-D might also be required for a private installation for the following reasons:

In parallel with a privately installed product version, UDS/SQL and UDS-D subsystems might also be preloaded. The SELECT-PRODUCT-VERSION command ensures that the modules loaded dynamically from the private installation establish a connection to the correct subsystem version.

Loading the application-specific data modules SSITAB and PLITAB dynamically

In the case of CALL DML application programs, the library from which the SSITAB modules are to be dynamically loaded at execution time must be known. With UDSKDBS applications, the library from which the PLITAB modules are to be dynamically loaded must also be known.

The data modules do not need to be stored in a single library (as in previous UDS/SQL versions), but can be kept in various libraries, e.g. in a separate one for each database.

The library(ies) are assigned via link name assignments with the ADD-FILE-LINK command.

SSITAB module:

1. Firstly, the library that was assigned with the link name \$UDSSSI is searched.
2. If the SSITAB modules are kept in more than one library, e.g. in a separate one for each database, the other libraries with the link names BLSLIB00 through BLSLIB99 can be assigned.

Assignment with the link name \$UDSSSI and if appropriate also with BLSLIB nn is the standard procedure that we recommend. The abbreviation “\$UL” is used for this procedure in the error messages of the product.

3. If the link name \$UDSSSI is not used or the dynamic-loading operation was unsuccessful, for compatibility reasons a UDS.MODLIB library in the runtime ID or a library assigned with the SET-TASKLIB command is searched. The abbreviation “TSK” is used for this procedure in the error messages of the product.

PLITAB module:

The library that was assigned with the link name \$UDSPLEX is searched firstly. The rest of the search in the libraries assigned with the link name BLSLIB nn or in a UDS.MODLIB library is performed in the same way as for the SSITAB modules.

Loading the configuration-specific table module UDSTRTAB dynamically

If a translation table (UDSTRTAB) is to be used by the DBH for user-specific sorting of character items, the library from which the UDSTRTAB module is to be dynamically loaded must be known. Firstly, a library in the configuration user ID that was assigned with the link name \$UDSKONF is searched. Because this table is used by the DBH, a library need only be assigned when linked-in applications are started.

For further details, see [section “Translation table for application-specific sorting” on page 126](#).

6.4.2 Linking UDS/SQL-TIAM applications

This section describes how to link a UDS/SQL-TIAM application program. For details of how to link a UDS/SQL-openUTM application, see [page 109](#).

For a UDS/SQL-TIAM application to be executed, the connection to the UDS/SQL DBH must be established. For this purpose, UDS/SQL provides connection modules which are linked into the UDS/SQL application.

Linking version-independent connection modules into the UDS/SQL application

UDS/SQL provides the version-independent connection modules UDSLNKI, UDSLNKL and UDSLNKA which are linked into the UDS/SQL-TIAM application. These version-independent connection modules contain the branch points for the processing of the DML statements and dynamically load the required version-dependent connection modules (UDSBCCON or LCCONCT).

[table 12](#) shows which version-independent connection modules are linked in and which version-dependent connection modules are dynamically loaded from them, depending on the DBH used. If the UDSLNKA module is linked in, you must specify which DBH variant is to be used with the MODIFY-JOB-SWITCHES command before calling the application program.

DBH used	Linked-in module	Dynamically loaded module
Independent DBH	UDSLNKI	UDSBCCON
	UDSLNKA Job switch 28 = ON	
Linked-in DBH	UDSLNKL	LCCONCT
	UDSLNKA Job switch 28 = OFF	

Table 12: Version-independent and version-dependent connection modules

If only the version-independent connection modules are linked into the UDS/SQL application, the UDS/SQL application can be used unchanged with a new UDS/SQL version or with a UDS/SQL correction version, if the system environment is otherwise the same. You can thus use the functional extensions of a new UDS/SQL version without having to relink the application program immediately.

Technically, the version-dependent connection modules can be statically linked into the program instead of the version-independent ones (see [table 12](#), under “Dynamically loaded module”). However, this is not the method we recommend, as it requires relinking whenever you change the UDS/SQL version.

For special cases, the connection modules contain 'weak externs'. If these address references are not resolved, this does not prevent the application from starting:

- The resolving of DSCEXT is required for the error handling of CALL DML calls (see [section “CALL DML error handling routine DSCEXT” on page 124](#)).
- You can ignore any message indicating that UDS@UNR# is not resolved.

Example

The following example shows the linking of a UDS/SQL application (COBOL DML) with the independent DBH.

```
//START-BINDER
//START-LLM-CREATION INTERNAL-NAME=module
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=library-1
, ELEMENT=element)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=$.udssyslnklib
, ELEMENT=UDSLNKI)
//RESOLVE-BY-AUTOLINK LIBRARY=crtesyslnk
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=library-2, ELEMENT=module)
//END
```

Handling name conflicts

If, during linking of applications which use a version-independent UD_{SLNK}_x connection module, the entries of the UD_{SLNK}_x module remain visible, name conflicts may occur during further dynamic loading of the version-dependent UDS/SQL modules.

To resolve these name conflicts it is generally sufficient to use the START-EXECUTABLE-PROGRAM command with the specification DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD)) at the start of the application (see also the START-EXECUTABLE-PROGRAM command in [section “Starting a COBOL program” on page 103ff](#)). If a specific application does not allow this “general” specification at application startup, the UDS-specific name conflicts can be handled selectively during linking with the linkage editor BINDER:

1. The entries of the version-independent module UD_{SLNK}_x (\$UNIASE, \$UNIBASE, DML, DMLTRACE, KDBSFITA, KKDS, KLDS, LINDA, SQLUDS) are masked during linking with the BINDER. This variant is suitable for applications in which all UDS/SQL calls are made from one load unit.

Example

```
/START-BINDER
...
//MODIFY-SYMBOL-VISIBILITY SYMBOL-NAME=( $UNIASE, $UNIBASE, DML -
//, DMLTRACE, KDBSFITA, KKDS, KLDS, LINDA, SQLUDS), VISIBLE=*NO()
...
//END
```

2. The entries of the version-independent module UD_{SLNK}_x (\$UNIASE, \$UNIBASE, DML, DMLTRACE, KDBSFITA, KKDS, KLDS, LINDA, SQLUDS) are renamed specifically for each application during linking with the BINDER. This variant is suitable for applications in which the UDS/SQL calls are made from several load units. In each load unit, only the respective entries provided or referenced there are renamed; the remaining entries are masked as in point 1.

Example

```
/START-BINDER
...
//RENAME-SYMBOLS SYMBOL-NAME=SQLUDS, NEW-NAME=UDSSQL -
//, SYMBOL-OCCURRENCE=*PAR(OCCURRENCE-NUMBER=*ALL)
...
//END
```

6.4.3 Starting a COBOL program

You can use the following DBH variants:

With	independent DBH	linked-in DBH
Mono-DB	X	X
Multi-DB	X	X
<i>open</i> UTM	X	-

Table 13: DBH variants

1. Starting an application program with the independent DBH

```
[/MODIFY-JOB-SWITCHES ON=28] _____ (1)
```

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version _____ (2)
[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]
```

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME={ dbname } - (3)
{ configuration-name }
```

```
[/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-library-1] _____ (4)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=SSITAB-library-nn]
```

```
[/ADD-FILE-LINK LINK-NAME=$UDSPLEX,FILE-NAME=PLITAB-library-1] _____ (5)
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=PLITAB-library-nn]
```

```
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=library-2, _____ (6)
ELEMENT=modul),DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD))
```

[*Application program parameters*]

- (1) If the connection module UDSLNKA was linked into the application program, job switch 28 must be set to ON so that the independent DBH is used (the version-dependent module UDSBCCON is dynamically loaded).

- (2) It is recommended that you generally specify which UDS/SQL version is to be used with the SELECT-PRODUCT-VERSION command, as a number of UDS/SQL versions could be installed concurrently and a number of versions of the UDS/SQL subsystem could be preloaded with IMON in the Software Configuration Inventory (SCI).
When using UDS-D, you should use an additional SELECT-PRODUCT-VERSION command to specify the UDS-D version that corresponds to the UDS/SQL version if UDS-D is used as a subsystem.
For further details, see [“Loading the UDS/SQL product modules dynamically” on page 95](#).
- (3) For the execution of a UDS/SQL application program you must make the UDS/SQL configuration known to the relevant application: With the SET-FILE-LINK command you assign the configuration file (FILE-NAME=*configuration-name*) via the link name DATABASE. In mono-DB operation the configuration name can also match the name of the database used (FILE-NAME=*dbname*). If FILE-NAME=*dbname*, the usage mode SHARED-RETRIEVAL for the database is only possible if you specify the load parameter DBNAME (see the [“Database Operation”](#) manual).
- (4), (5) In the case of CALL DML application programs, the library from which the SSITAB modules are to be dynamically loaded at execution time must be assigned with the ADD-FILE-LINK command. With KDBS applications the library from which the PLITAB modules are to be dynamically loaded must also be assigned.
For further details, see [“Loading the application-specific data modules SSITAB and PLITAB dynamically” on page 98](#).
- (6) The application program is started with the START-EXECUTABLE-PROGRAM command.
The preset value NAME-COLLISION=*STD in RUN-MODE=*ADVANCED should not be changed, as this may lead to name conflicts in dynamically loadable entries in applications which were linked with the BINDER (see also [“Handling name conflicts” on page 102](#)).

2. Starting an application program with the linked-in DBH

Commands which affect the control of the DBH itself are not described below or only very briefly. A detailed description of these commands can be found in the “[Database Operation](#)” manual, section “Commands for starting DBH”.

```
[/MODIFY-JOB-SWITCHES OFF=28] _____ (1)
```

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version _____ (2)
```

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME= $\left. \begin{array}{l} \textit{dbname} \\ \textit{configuration-name} \end{array} \right\}$  - (3)
```

```
[/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-library-1] _____ (4)
```

```
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=SSITAB-library-nn]
```

```
[/ADD-FILE-LINK LINK-NAME=$UDSPLEX,FILE-NAME=PLITAB-library-1] _____ (5)
```

```
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=PLITAB-library-nn]
```

```
[/ADD-FILE-LINK LINK-NAME=$UDSKONF,FILE-NAME=UDSTRTAB-library] _____ (6)
```

```
/CREATE-FILE FILE-NAME=confname.DBSTAT,SUPPRESS-ERR=*FILE-EXISTING (7)
```

```
/CREATE-FILE FILE-NAME=confname.DBSTAT.SAVE,SUPPRESS-ERR=*FILE-EXISTING
```

```
[/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=DBH-parameter-file] _____ (8)
```

```
[Further DBH-specific commands] _____ (9)
```

```
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=library-2, _____ (10)
```

```
ELEMENT=modul),DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD))
```

```
[Application program parameters]
```

- (1) If the connection module UDSSLNKA was linked into the application program, job switch 28 must be set to OFF so that the linked-in DBH is used (the version-dependent module LCCONCT is loaded dynamically).
- (2) It is recommended that you generally specify which UDS/SQL version is to be used with the SELECT-PRODUCT-VERSION command, as a number of UDS/SQL versions could be installed concurrently and a number of versions of the UDS/SQL subsystem could be preloaded with IMON in the Software Configuration Inventory (SCI).

For further details, see “[Loading the UDS/SQL product modules dynamically](#)” on page 95.

- (3) As when starting an application with the independent DBH, the UDS/SQL configuration must be made known. See under (3) on [page 104](#).
- (4), (5) In the case of CALL DML application programs, the library from which the SSITAB modules are to be dynamically loaded at execution time must be assigned with the ADD-FILE-LINK command. With KDBS applications the library from which the PLITAB modules are to be dynamically loaded must also be assigned.
For further details, see [“Loading the application-specific data modules SSITAB and PLITAB dynamically” on page 98](#).
- (6) If a translation table (UDSTRTAB) is to be used for user-specific sorting of character items, the library from which the UDSTRTAB module is to be dynamically loaded must be known. Firstjy, a library in the configuration user ID that was assigned with the link name \$UDSKONF is searched. For further details, see [section “Translation table for application-specific sorting” on page 126](#).
- (7) The DB status files must be created if they do not already exist. These files are created with a 4-Kbyte page format and may be placed on different volumes.
If the DB status files are to reside on private disk, they must be created using the following command:
- ```
/CREATE-FILE FILE-NAME=confname.DBSTAT[.SAVE]
, SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn
, DEVICE-TYPE=device[, SPACE=...])
, SUPPRESS-ERRORS=*FILE-EXISTING
```
- (8) The link name PPFIL is used to assign a file from which the linked-in DBH is to read in the load parameters. The load parameters can also be in a PLAM library element or in a procedure file. The assignment commands required for this can be found in the [“Database Operation”](#) manual, section “Commands for starting DBH”.
- (9) Further optional commands for controlling the linked-in DBH are described in detail in the [“Database Operation”](#) manual, section “Commands for starting DBH”.
- (10) The application program is started with the START-EXECUTABLE-PROGRAM command.  
The preset value NAME-COLLISION=\*STD should not be changed as this may lead to name conflicts in dynamically loadable entries in applications which were linked with the BINDER (see also [“Handling name conflicts” on page 102](#)).

## 6.5 Interoperation in a UDS/SQL-openUTM application

### Generating a UDS/SQL-openUTM application

Generating a UDS/SQL-openUTM application involves the same steps as generating an openUTM application.

Information on generating a UDS/SQL-openUTM application is provided in the openUTM manual “[Generating Applications](#)”.

The individual steps are:

- Defining the application configuration and generating the KDCROOT module with the KDCDEF utility routine
- Compiling the KDCROOT main routine
- Compiling the application program units
- Linking the modules to the application program

### KDCDEF statement DATABASE

To enable the interoperation of openUTM and UDS/SQL, it is also essential that the database connection be generated in the KDCROOT main routine with the DATABASE statement:

```
DATABASE [ENTRY=entry][, LIB=LOGICAL-ID(SYSLNK)][, TYPE=type]
```

ENTRY=*entry*

indicates the ENTRY name of the database;

The user enters the ENTRY name for the UDS/SQL interface used:

- \$UNIBASE for COBOL DML (default value)
- DML for CALL DML
- KKDS for KDBS
- SQLUDS for SQL

LIB=LOGICAL-ID(SYSLNK)

The connection module is loaded dynamically from the library entered as the IMON installation path for SYSLNK. A UDS/SQL version assigned via SELECT-PRODUCT-VERSION or, by default, the highest UDS/SQL version is used here. Further options for assigning a value to the LIB parameter are described in the openUTM manual “[Generating Applications](#)”.

TYPE=*type*

identifies the type of the database system.

Here you enter UDS. If, in a openUTM transaction, you are using both the SQL and a CODASYL interface, you may also enter DB (see [“Two UDS/SQL configurations in one UDS/SQL openUTM application” on page 113](#)).

### Compiling the KDCROOT main routine

When compiling KDCROOT, you must assign UDS/SQL and openUTM macro libraries.

*Example of compilation with ASSEMBH*

```
/START-ASSEMBH
//COMPILE SOURCE=rootname,MACRO-LIB=(utmmacrolib -
// ,udssyslib),MODULE-LIB=rootlibrary)
//END
```

## Linking UDS/SQL-openUTM applications

You can choose from three different variants when linking a UDS/SQL-openUTM application:

- The UDSCON connection module is loaded dynamically by openUTM
- The UDSCON connection module is loaded dynamically by UDS/SQL
- The UDSCON connection module is permanently linked in

These three variations will be explained in the following:

### The UDSCON connection module is loaded dynamically by openUTM

The UNRESOLVED EXTRNS: UDSCON message of the BINDER can be ignored in this case. openUTM loads the UDS/SQL connection module UDSCON dynamically. The UDSCON connection module contained in the UDS/SQL subsystem can be used together by all applications. When a new correction version is used, the UDS/SQL-openUTM application does not have to be relinked.

#### *Example*

```

/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=module
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=root-library
, ELEMENT=KDCROOT-module)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=library
, ELEMENT=(prog-unit-1,prog-unit-2,...prog-unit-n))
//RESOLVE-BY-AUTOLINK LIBRARY=library
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=library -
// .ELEMENT=element,OVERWRITE=YES)
//END

```

### The UDSCON connection module is loaded dynamically by UDS/SQL

The version-independent module UDSCNUV is linked into the UDS/SQL openUTM application. The reference to UDSCON contained in the KDCROOT module must be renamed to UDSCNUV. UDSCNUV then loads the UDS/SQL connection module dynamically (see also [section “Loading the UDS/SQL product modules dynamically” on page 95](#)). The UDSCON connection module contained in the UDS/SQL subsystem can be used together by all applications. When a new correction version is used, the UDS/SQL-openUTM application does not have to be relinked. The module in which UDSCNUV was loaded cannot be shared.

#### *Example*

```

/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=modul
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=root-library -
// ,ELEMENT=KDCROOT-module)
//RENAME-SYMBOLS SYM-NAM=UDSCON,SYM-TYP=ALL,NEW-NAME=UDSCNUV
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=library -
// ,ELEMENT=(prog-unit-1,prog-unit-2,...prog-unit-n))
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=udssyslnklib -
// ,ELEMENT=UDSCNUV)
//RESOLVE-BY-AUTOLINK LIBRARY=library
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=library -
// ,ELEMENT=element,OVERWRITE=YES)
//END

```

### The UDSCON connection module is permanently linked in



Every time you change UDS/SQL versions, even correction versions, the program must be relinked. You do not need to load the UDSCON connection module when starting the UDS/SQL openUTM application. The UDSCON connection module contained in the UDS/SQL subsystem cannot be used.

#### *Example*

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=module
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=root-library -
// ,ELEMENT=KDCROOT-module)
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=library -
// ,ELEMENT=(prog-unit-1,prog-unit-2,...prog-unit-n))
//INCLUDE-MODULES MODULE-CONTAINER=*LIB(LIBRARY=udssyslnklib -
// ,ELEMENT=UDSCON)
//RESOLVE-BY-AUTOLINK LIBRARY=library
//RESOLVE-BY-AUTOLINK LIBRARY=(utmsyslnklib,crtesyslnklib[.PARTIAL-BIND] -
// ,utmsyslnksplrtslib)
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=library -
// ,ELEMENT=element,OVERWRITE=YES)
//END
```

You will find additional information on linking an UDS/SQL openUTM application in the “[Generating Applications](#)” openUTM manual and on linking the CRTE in the “[CRTE \(BS2000\)](#)” manual.

## Starting a UDS/SQL-openUTM application

```

.
.
[01 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version]
[02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]
.
.
[03 /ADD-FILE-LINK LINK-NAME=$UDSLIB,FILE-NAME=udslib]
[04 /ADD-FILE-LINK LINK-NAME=$UDSDLIB,FILE-NAME=udsdlib]
[05 /ADD-FILE-LINK LINK-NAME=$UDSKLIB,FILE-NAME=udskdbslib]
.
.
06 .UDS DATABASE=confname1
[07 .DB..DATABASE=confname2]

```

01), 02)

Because different versions of the subsystems UDS-SQL and/or UDS-D can be preloaded concurrently, you should always specify the product version you want with SELECT-PRODUCT-VERSION. Here it is possible to load the respective subsystem versions you require on DBH startup using job variables (see the “[Database Operation](#)” manual, section “Using multiple UDS/SQL versions concurrently”). If you do not specify SELECT-PRODUCT-VERSION, the subsystem with the highest version will be used. If, in the case of UDS/SQL, no corresponding subsystem version is preloaded and neither UDSCON nor UDSCNUV is permanently linked, the connection module UDSCON is loaded dynamically from the UDS/SQL library specified in the KDCDEF statement DATABASE (LIB=LOGICAL-ID(SYSLNK)).

03), 04), 05)

These assignments of libraries of a private installation need only be entered if the UDS/SQL version used is not available in the SCI and CALL-DML (\$UDSLIB) is used with UDS-D (\$UDSDLIB) or UDSKDBS (\$UDSKLIB).

06), 07)

Start parameters that are passed from UTM to UDS/SQL.

In the start parameters you may only specify configuration names (*confname1*, *confname2*) without an ID.

You need only specify the start parameter .DB when you have generated a database statement using TYPE=DB. The two blanks after .DB are mandatory.

For further information on starting a UDS/SQL-openUTM application, see the „[Generating Applications](#)” manual.



*Example*

```

/SET-LOGON-PARAMETERS
/ADD-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=SYSLOG
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=02.8A00
/ASSIGN-SYSDTA TO=*SYSCMD
/START-EXE-PROG FROM-FILE=*LIB(LIBRARY=LIB.TEST, ELEMENT=AUFABW) -
/ ,DBL-PAR=(LOADING=(LOAD-INFORMATION=REFERENCES)-
/ ,ERROR-PROC=UNRESOLVED-EXTRNS=DELAY))
.UTM START FILEBASE = KDCFILE.AUFABW
.UTM START STARTNAME=MANUAL.STARTEN
.UTM END
.UDS DATABASE=DBHSQL
.FHS MAPLIB=LIB.TEST
END
/EXIT-JOB

```

**Two UDS/SQL configurations in one UDS/SQL openUTM application**

It is possible in a single UDS/SQL openUTM application to process SQL queries as well as CODASYL-/CALL-DML queries with UDS/SQL in a single openUTM transaction. When this is done, UDS/SQL is generated as type UDS and as type DB (TYPE = DB) in the openUTM application.

*Requirements*

- When assembling the main routine KDCROOT, the UDS/SQL macro library (SYSLIB.UDS-SQL.*nnn*) must be assigned. It contains the KDCDBU macro as well as the KDCDBD macro.
- SQL queries and CODASYL-/CALL-DML queries may only be mixed in a transaction when they are passed in their entirety to different UDS/SQL configurations.

The ENTRY name SQLUDS may only be specified in the definition with type UDS.

**Error codes**

In a UDS/SQL-openUTM application, runtime information for diagnostics purposes is stored in a special area (DB-DIAGAREA or DB record; see the [“Messages, Debugging and Diagnostics in BS2000”](#) manual). This diagnostic information is version-specific. The codes involved are output by openUTM.

See also [section “UDS/SQL-openUTM return codes” on page 390](#) and [section “Additional diagnostic information in openUTM” on page 394](#).

## 6.6 Error handling

### 6.6.1 Interrupt handling for UDS/SQL-TIAM applications

The interrupt handling discussed in this section is only applicable to UDS/SQL-TIAM applications; error handling for UDS/SQL-openUTM applications is dealt with on [page 113](#). The interrupt handling facilities provided by UDS/SQL through the routine SCSXUSER prevent the operating system from aborting the program in the case of defined STXIT event classes. In addition, users can define their own interrupt routines.

#### UDS/SQL interrupt handling with SCSXUSER (UDS-STXIT)

The UDS-STXIT routine SCSXUSER handles the following causes of interrupts (see also the “[Executive Macros](#)” manual):

- program error (program check)
- interrupt via interval timer
- end of program runtime
- unrecoverable program error
- message for the program (INFORM-PROGRAM)
- BREAK/ESCAPE
- abnormal end (ABEND)
- program termination

If certain STXIT interrupt events occur, vital measures for the entire database system are carried out, e.g. rollback of the affected transaction.

The UDS STXIT routine SCSXUSER is automatically included in the STXIT parallelism concept.

You can also define your own STXIT routine for additional interrupt handling. This STXIT routine should make use of the STXIT parallelism concept and should end with EXIT CONTINU=YES to ensure that the execution of the SCSXUSER routine is not prevented. If the STXIT event class “message for the program” occurs with the INFORM-PROGRAM command, SCSXUSER (UDS STXIT) checks to see if the message is intended for UDS/SQL. If this is the case, SCSXUSER is terminated with EXIT CONTINU=NO. The following STXIT routines for this event are not executed.

In the case of the BREAK/ESCAPE interrupt, the program runs to a defined interrupt point. The registers then show the status at the time of the interrupt. The program can be continued with TRACE or RESUME.

The use of UDS-STXIT via SCSXUSER has the following advantages:

- From the viewpoint of the system as a whole:  
Open transactions that are not terminated due to an illegal exit or error in the application program or the omission of a concluding FINISH statement are rolled back. The transaction channels occupied as a result of the interrupt (see the “[Database Operation](#)” manual, DBH load parameter TRANSACTION) thus become available (each incomplete transaction reduces the number of transactions that can be conducted concurrently by 1).
- From the viewpoint of view of the application program involved:  
The rollback with SCSXUSER enables the application program to easily skip a transaction with the aid of an independent STXIT routine (in the event of data errors, for example); if it is no longer possible to work on the database, the application program can continue to perform operations that are independent of the database.

If SCSXUSER cannot be loaded dynamically, the connection module outputs a corresponding warning but continues with the processing. In private installations (cf. [page 97](#)) it is possible to operate UDS/SQL-TIAM applications without the UDS/SQL interrupt handling. Applications which are linked via openUTM with UDS/SQL do not work with SCSXUSER. For these applications, openUTM takes over the interrupt handling.

## User-programmed interrupt handling

Apart from the interrupt handling by UDS/SQL with SCSXUSER, you can also define your own STXIT routines for any desired event classes by using the STXIT macro and the STXDNEW operand.

These STXIT routines are terminated with the EXIT macro and the operands CONTINU=YES, TERM=NO.

### *Exceptions*

- STXIT routines for the event classes “program error” and “program check” should be terminated with the EXIT macro, CONTINU=YES, TERM=(STEP) under STXIT parallelism.
- A STXIT routine for the event class “message for the program” should be terminated with the EXIT macro and the operands CONTINU=YES, TERM=NO if a message intended for that routine has been processed.



### **CAUTION!**

DML statements cannot be sent to the DB in STXIT routines, since the UDS STXIT SCSXUSER always attempts to terminate the transaction. If STXIT routines from application programs and other systems which have not been adapted to STXIT concurrency are run, there is no guarantee that all STXIT routines for each event will start. Transaction channels may also remain locked and thus not be available for the rest of the session section.

Messages and dumps may be initiated from other systems, since all the STXIT routines for a given interrupt event are normally activated when the event occurs.

## 6.6.2 Database exception conditions

### Using special registers (COBOL DML) or the user information area (CALL DML)

- DATABASE-REALM-NAME  
This gives the name of the realm associated with the database exception condition when a DML statement cannot be terminated successfully.
- DATABASE-SET-NAME  
This gives the name of the set associated with the database exception condition when a DML statement cannot be terminated successfully.
- DATABASE-RECORD-NAME  
This gives the name of the record associated with the database exception condition when a DML statement cannot be terminated successfully.
- DATABASE-STATUS  
This register holds the database status indicator, which is given a new value whenever a new DML statement is executed. The two left-most characters contain the statement code, and the three right-most characters the status code.  
A list of which status codes belong to which database exception conditions can be found in [section “Status codes” on page 365](#).  
If the execution of any DML statement ends with a database exception condition, the statement code shows which DML statement has caused the exception condition, and the status code shows which exception condition has occurred. If a DML statement was executed successfully, both the statement code and the status code are set to the value zero.  
This rule does not apply to exception conditions detected by the UDS/SQL connection module. In this case, the statement code always contains the value zero (“00”).

The special registers DATABASE-REALM-NAME and DATABASE-RECORD-NAME are also used for FIND and STORE statements if no database exception condition has occurred. They contain the name of the realm involved and the record involved; DATABASE-STATUS then has the value zero.

The contents of special registers DATABASE-REALM-NAME, DATABASE-SET-NAME and DATABASE-RECORD-NAME cannot be interpreted if the DBH cannot recognize any realm, record or set names because the DML statement has not provided information on these.

## Statement codes

When COBOL DML is used, the statement code can be obtained from special register DATABASE-STATUS. In the case of CALL DML the statement code is stored in the first two positions of the result item in the user-information parameter.

| Statement code | Statements of                                                 |                                                                                                                          |
|----------------|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
|                | COBOL DML                                                     | CALL DML                                                                                                                 |
| 01             | CONNECT                                                       | CONNEC                                                                                                                   |
| 02             | DISCONNECT                                                    | DISCON                                                                                                                   |
| 03             | ERASE                                                         | ERASEC                                                                                                                   |
| 04             | FIND/FETCH                                                    | { FIND1/FTCH1, FIND1L/FTCH1L<br>FIND2/FTCH2<br>FIND3/FTCH3<br>FIND4/FTCH4<br>FIND5/FTCH5<br>FIND6/FTCH6<br>FIND7A/FTCH7A |
| 05             | FINISH                                                        | FINISC                                                                                                                   |
| 06             | FREE                                                          | FREEC                                                                                                                    |
| 07             | GET                                                           | GETC                                                                                                                     |
| 08             | IF                                                            | IFC                                                                                                                      |
| 09             | KEEP                                                          | KEEPC                                                                                                                    |
| 10             | MODIFY                                                        | { MODIF1<br>MODIF2                                                                                                       |
| 14             | STORE                                                         | { STORE1, STOR1L<br>STORE2, STOR2L                                                                                       |
| 15             | ACCEPT                                                        | ACCPYC, ACCPTL                                                                                                           |
| 16             | SET                                                           | -                                                                                                                        |
| 25             | -                                                             | LOOKC                                                                                                                    |
| 00             | For all exception conditions entered by the connection module |                                                                                                                          |

Table 14: Statement codes and associated functions

The UDS online utility (see the [“Recovery, Information and Reorganization”](#) manual) uses statement code 13 for its specific DMLs.

## Status codes

Table 15 shows the correlation between UDS/SQL behavior and status codes:

|                                             | Status codes               |                  |                                                |                |           |                  |
|---------------------------------------------|----------------------------|------------------|------------------------------------------------|----------------|-----------|------------------|
|                                             | 000                        | 001              | 018, 113,<br>122, 218                          | ≠ 000<br>READY | 200       | any<br>other     |
| <b>DML statement successfully completed</b> | yes                        | yes              | no                                             | no             | so far    | no               |
| <b>Transaction aborted by UDS/SQL</b>       | no <sup>1)</sup>           | no <sup>1)</sup> | yes                                            | yes            | no        | no <sup>1)</sup> |
| <b>Contents of record area UWA ...</b>      | corresponding to statement | the next record  | undefined because transaction has been aborted |                | unchanged | unchanged        |

Table 15: Meaning of the status codes

<sup>1</sup> Unless FINISH WITH CANCEL is executed.

The complete list of status codes can be found on pages [365](#) (COBOL DML) and [378](#) (CALL DML).

## Combinations of statement codes and status codes

| Status code                                   | Statement code |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
|-----------------------------------------------|----------------|----|----|----|-----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|
|                                               | 00             | 01 | 02 | 03 | 04                                | 05 | 06 | 07 | 08 | 09 | 10 | 12 | 13 | 14 | 15 | 16 | 25 |   |
| 001<br>010                                    |                |    |    |    | see<br>table 17<br>on<br>page 122 |    |    |    |    |    |    |    | X  |    |    |    |    |   |
| 011<br>012<br>013<br>018<br>020               |                | X  | X  | X  |                                   |    |    | X  | X  | X  | X  | X  | X  | X  | X  | X  |    | X |
| 021<br>022<br>023<br>024<br>027<br>028<br>029 |                | X  |    |    |                                   |    |    |    | X  |    | X  |    | X  |    | X  |    |    |   |
| 031<br>032<br>033                             |                | X  | X  | X  |                                   |    |    | X  | X  |    | X  |    |    |    | X  |    |    |   |
| 042<br>043<br>044                             |                | X  | X  | X  |                                   |    |    |    |    | X  |    | X  | X  | X  | X  | X  |    |   |
| 051                                           |                | X  |    |    |                                   |    |    |    |    |    | X  |    |    |    | X  |    |    |   |
| 071<br>072                                    |                |    |    | X  |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 081<br>082<br>083                             |                | X  | X  |    |                                   |    |    |    |    |    | X  |    |    |    |    |    |    |   |
| 091<br>092<br>093<br>099                      |                | X  | X  | X  |                                   |    |    | X  |    | X  |    | X  | X  | X  | X  | X  |    |   |
| 101<br>102<br>103                             |                |    |    |    |                                   |    | X  |    |    |    |    |    |    |    |    | X  | X  |   |
| 113                                           | X              | X  | X  | X  |                                   |    | X  | X  | X  |    | X  | X  | X  | X  | X  | X  |    | X |

Table 16: Combinations of statement codes and status codes

(part 1 of 3)



| Status code | Statement code |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
|-------------|----------------|----|----|----|-----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|
|             | 00             | 01 | 02 | 03 | 04                                | 05 | 06 | 07 | 08 | 09 | 10 | 12 | 13 | 14 | 15 | 16 | 25 |   |
| 122         | X              | X  | X  | X  | see<br>table 17<br>on<br>page 122 | X  | X  | X  | X  | X  | X  | X  | X  | X  |    |    | X  |   |
| 123         |                |    |    |    |                                   |    |    |    |    |    |    | X  | X  |    |    |    |    | X |
| 124         |                |    |    |    |                                   |    |    |    |    |    |    | X  | X  |    |    |    |    |   |
| 131         | X              |    |    |    |                                   |    |    |    |    |    |    |    | X  | X  |    |    |    |   |
| 132         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  | X  |    |    |    |   |
| 134         | X              | X  | X  | X  |                                   |    | X  | X  | X  | X  | X  | X  |    | X  | X  | X  |    | X |
| 136         | X              | X  | X  | X  |                                   |    | X  | X  | X  | X  | X  |    |    | X  | X  | X  |    | X |
| 137         | X              |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 141         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  | X  |    |    |    |   |
| 142         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  | X  |    |    |    |   |
| 144         | X              | X  | X  | X  |                                   |    |    | X  | X  | X  | X  | X  |    | X  | X  |    |    |   |
| 145         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  | X  |    |    |    |   |
| 146         | X              | X  | X  | X  |                                   |    |    | X  | X  | X  | X  | X  | X  | X  | X  | X  |    |   |
| 151         | X              |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 152         | X              |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 154         | X              |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 155         | X              |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 161         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 162         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 163         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 164         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  |    |    |    |    |   |
| 165         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  |    |    |    |    |   |
| 166         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  |    |    |    |    |   |
| 167         |                |    |    |    |                                   |    |    |    |    |    |    |    | X  |    |    |    |    |   |
| 183         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 184         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 191         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 192         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 193         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 194         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 195         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 197         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 198         |                |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
| 200         | X              |    |    |    |                                   | X  |    |    |    |    |    |    |    |    |    |    |    |   |
| 201         | X              |    |    |    |                                   | X  |    |    |    |    |    |    |    |    |    |    |    |   |
| 218         | X              | X  | X  | X  |                                   |    | X  | X  | X  | X  | X  | X  |    | X  | X  |    | X  |   |

Table 16: Combinations of statement codes and status codes

(part 2 of 3)

| Status code | Statement code |    |    |    |                                   |    |    |    |    |    |    |    |    |    |    |    |    |   |
|-------------|----------------|----|----|----|-----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|
|             | 00             | 01 | 02 | 03 | 04                                | 05 | 06 | 07 | 08 | 09 | 10 | 12 | 13 | 14 | 15 | 16 | 25 |   |
| 781         |                |    |    |    | see<br>table 17<br>on<br>page 122 |    |    |    |    |    |    |    | X  |    |    |    | X  |   |
| 782         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 783         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 784         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 785         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 786         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 789         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    | X |
| 802         |                | X  |    |    |                                   |    |    |    |    |    | X  |    |    | X  |    |    |    |   |
| 804         |                |    |    |    |                                   |    |    |    |    |    |    |    |    | X  |    |    |    |   |
| 805         |                | X  | X  | X  |                                   |    |    |    |    | X  | X  | X  | X  | X  |    |    |    |   |
| 888         |                |    |    |    |                                   |    |    |    |    |    | X  |    |    | X  |    |    |    |   |
| 898         |                |    |    |    |                                   |    |    |    |    |    | X  |    |    | X  |    |    |    |   |
| 899         |                |    |    |    |                                   |    | X  |    |    |    | X  |    |    | X  |    |    |    |   |
| 901         |                | X  | X  | X  |                                   |    | X  | X  |    |    | X  | X  | X  | X  |    |    |    |   |
| 950         |                |    |    |    |                                   |    |    |    |    |    | X  | X  | X  |    |    |    |    |   |
| 954         |                |    |    |    |                                   |    |    |    |    |    | X  | X  | X  |    |    |    |    |   |

Table 16: Combinations of statement codes and status codes

(part 3 of 3)

## FIND/FETCH status codes

| Status code | Formats of the FIND/FETCH record selection expression |   |   |   |   |   |   |
|-------------|-------------------------------------------------------|---|---|---|---|---|---|
|             | 1                                                     | 2 | 3 | 4 | 5 | 6 | 7 |
| 001         |                                                       |   |   |   |   |   | X |
| 018         | X                                                     | X | X | X | X | X | X |
| 020         | X                                                     | X | X | X | X | X | X |
| 021         |                                                       | X | X | X |   |   |   |
| 023         |                                                       |   |   |   |   |   | X |
| 024         | X                                                     | X |   | X |   |   | X |
| 027         |                                                       |   | X |   |   |   | X |
| 028         | X                                                     |   |   |   |   |   |   |
| 029         |                                                       |   |   | X | X |   |   |
| 031         |                                                       | X | X | X | X | X | X |
| 032         |                                                       |   |   |   | X |   |   |
| 033         |                                                       |   |   |   | X |   |   |
| 042         | X                                                     |   | X | X |   |   | X |
| 043         |                                                       | X |   |   |   |   | X |
| 071         |                                                       | X | X | X | X |   |   |

Table 17: Combinations of statement code 04 and status codes

(part 1 of 2)

| Status code | Formats of the FIND/FETCH record selection expression |   |   |   |   |   |   |
|-------------|-------------------------------------------------------|---|---|---|---|---|---|
|             | 1                                                     | 2 | 3 | 4 | 5 | 6 | 7 |
| 04...       |                                                       |   |   |   |   |   |   |
| 091         | X                                                     | X | X | X | X | X | X |
| 101         |                                                       |   |   | X |   | X | X |
| 102         | X                                                     |   |   |   |   |   |   |
| 103         | X                                                     | X | X | X | X | X | X |
| 113         | X                                                     | X | X | X | X | X | X |
| 122         | X                                                     | X | X | X | X | X | X |
| 134         | X                                                     | X | X | X | X | X | X |
| 136         | X                                                     | X | X | X | X | X | X |
| 144         | X                                                     | X | X | X | X | X | X |
| 146         | X                                                     | X | X | X | X | X | X |
| 183         |                                                       |   |   |   |   |   | X |
| 184         |                                                       |   | X |   |   |   | X |
| 191         |                                                       |   |   |   |   |   | X |
| 192         |                                                       |   |   |   |   |   | X |
| 193         |                                                       |   | X | X |   |   | X |
| 194         |                                                       |   |   |   |   |   | X |
| 195         |                                                       |   |   |   |   |   | X |
| 197         |                                                       |   | X |   |   |   |   |
| 198         |                                                       |   | X |   |   |   |   |
| 218         | X                                                     | X | X | X | X | X | X |
| 805         |                                                       |   |   |   |   |   | X |
| 901         | X                                                     | X | X | X | X | X | X |

Table 17: Combinations of statement code 04 and status codes

(part 2 of 2)

### 6.6.3 CALL DML error handling routine DSCEXT

The CALL DML error handling routine DSCEXT is only applicable to UDS/SQL-TIAM applications.

If the user information parameter has no identifier, i.e. if USINF\* or UINF1\* is omitted or is not in the correct position in the user information area, the program is not able to communicate with UDS/SQL, and no database status can be transferred. To specifically deal with this eventuality, the user must define an error routine with ENTRY name DSCEXT in the program.

If the DSCEXT error exit has not been defined, status code C91 is output. The program is not able to execute DML functions.

If the user information parameter is not detected during program runtime, the following message appears at the operator console:

```
UDS0283 UDS USER ERROR: USERINF PARAM WRONG OR MISSING
```

and a branch is made to error exit DSCEXT.

If the user has neither provided the user information area nor this error exit, the additional message:

```
UDS0284 UDS USER ERROR: NO DSCEXT-ROUTINE DEFINED
```

is output, and the program is aborted.

Thus, CALL DML does not use the return address in register 14 in either case (see [chapter "CALL DML reference section" on page 197](#)). The CALL DML call is not executed.

The DSCEXT routine is not required in openUTM application programs and is not supported.

In the case of a branch to the DSCEXT error exit, it is rarely of any use to continue processing, since a programming error exists. This may be typically due to:

- an invalid user information area definition
- invalid input of parameter or transfer addresses in register 1
- inadvertent overwriting of the user information area during the program run

It is not necessary to output a message and to enter a FINISC in case of an open transaction. UDS/SQL outputs an appropriate message and generates a FINISH WITH CANCEL if the program is terminated while a transaction is open.

The DSCEXT routine can be defined in its own module if it is to be used in several CALL DML programs. In that case it must be linked into the programs.

#### Example

DSCEXT error in a COBOL program:

```
DSCEXT.
 ENTRY "DSCEXT".
 DISPLAY "DSCEXT=ERROR EXIT" UPON T.
 STOP RUN.
```

## 6.7 Translation table for application-specific sorting

If you want a different sorting of character items than the one preset in UDS/SQL (in accordance with the EBCDI code), you can create a translation table. This table is valid for all databases in the UDS session.

This sort sequence only applies to sorting the result set from a search and not to data storage or when determining data contents.

National items (Unicode: UTF-16, PICTURE N, USAGE NATIONAL) and numeric items are not affected by any existing translation table.

### Creating the translation table

You create an Assembler module with the name UDSTRTAB and enter it in any library of the configuration user ID. This library must be assigned before the start of the independent or linked-in DBH (see “Assigning a translation table” on [page 128](#)).

To ensure that the alphabetical order is unambiguous, the user must check that no printable characters are duplicated. UDS/SQL does not check for this. Duplicated characters lead to an ambiguous sort sequence.

An ambiguous sort sequence may, however, be desired. For example, in the case of German vowels with umlauts (see example below).

There are several ways of creating a translation table:

#### 1. Character conversion

This is based on the same translation table (X'000102 ... FF'). The programming overhead is not very great if there are only a few conversions.

#### *Example*

The table only recognizes the 26 letters (X'C1' through X'E9') and Ö as X'8C'. Ö appears in the table between O and P.

```
UDSTRTAB START
 TITLE ' UDS-TRANSLATION-TABLE '
TAB DC 256AL1 (*-TAB) Create identical translation
* table
 ORG TAB+X'8C'
 DC X'D7' 'Ö' -> 'P'
*
 ORG TAB+X'D7' 'P' -> 'Q' 'Q' -> 'R'
 DC X'D8D9DA8C' 'R' -> X'DA' X'DA' -> 'Ö'
 END
```

## 2. Entering the desired alphabetical order

*Example*

No distinction is made between uppercase and lowercase letters during the sort operation. German umlauts following the corresponding letter without umlauts, e.g. **Ä** after **A**, **ß** after **s**.

```
UDSTRTAB START
 TITLE ' UDS-TRANSLATION-TABLE'
TAB DC x'00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F'
 x'10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F'
 x'20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F'
 x'30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F'
 x'40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F'
 x'50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F'
 x'60 61 62 63 64 65 66 E3 68 69 6A 6B 6C 6D 6E 6F'
 x'70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F'
 x'80 C1 C3 C4 C5 C6 C7 C8 C9 CA CA C2 D7 E6 CE CF'
 x'90 D1 D2 D3 D4 D5 D6 D8 D9 DA DA DB DC DD DE DF'
 x'AO A1 E2 E4 E5 E7 E8 E9 EA EB AA C2 D7 E6 AE AF'
 x'BO B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF'
 x'CO C1 C3 C4 C5 C6 C7 C8 C9 CA CA CB CC CD CE CF'
 x'DO D1 D2 D3 D4 D5 D6 D8 D9 DA DA DB DC DD DE DF'
 x'EO E1 E2 E4 E5 E7 E8 E9 EA EB EA EB EC ED EE EF'
 x'FO F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF'
 END
```

The blanks in the table above have only been included to make the example easier to read. They should not be included in the original translation table.

### Assigning the translation table

Before the independent DBH or the linked-in application program is called, the library from which the self-generated UDSTRTAB module is to be dynamically loaded must be assigned.

1. The UDSTRTAB module can be stored in the configuration user ID in a PLAM library of any name. The link name \$UDSKONF is available for assigning this library with the ADD-FILE-LINK command.

```
/ADD-FILE-LINK LINK-NAME=$UDSKONF,FILE-NAME=UDSTRTAB-library
```

This is the standard method that we recommend.

2. If the link name \$UDSKONF is not used or the dynamic loading according to method 1 was not successful, for compatibility reasons the system checks whether a library called UDS.MODLIB exists in the configuration user ID. It then loads dynamically from this library or from one assigned with SET-TASKLIB.
3. If the UDSTRTAB module could not be dynamically loaded by methods 1 and 2, the UDSTRAB module from the SYSLNK library of the UDS/SQL product managed in the SCI (EBCDIC sort sequence) is used.

If the UDSTRTAB module cannot be dynamically loaded, the DBH uses the EBCDIC sort sequence internally.



---

## 7 COBOL DML reference section

This section describes the COBOL DML statements and the relevant syntax rules.

The user specifies COBOL DML statements in COBOL programs. These statements are compiled by the COBOL compiler COBOL85 or COBOL2000. The same rules and the same reserved words that apply to COBOL statements also apply to COBOL DML statements (see the „[COBOL Compiler, Reference Manual](#)“).

For processing databases with a page length of 4KB or 8KB you require COBOL2000 or COBOL85 as of version 2.2C21 if the application program needs to be recompiled. This also applies to databases with a page length of 2048 bytes if the application program contains subschemata that were **not** compiled with the DDL compiler option “SUBSCHEMA FORM IS OLD” (see the “[Creation and Restructuring](#)“ manual).

## 7.1 General rules

When a format is used, you must replace the following variables by a current value. Three categories of variables can be distinguished:

| Variable                                                                                                  | Current value                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>schema-name</i><br><i>subschema-name</i><br><i>realm-name</i><br><i>set-name</i><br><i>record-name</i> | All names must originate from the subschema.                                                                                                                                                                 |
| <i>record-element-name</i><br><i>item-name</i>                                                            | The context of the DML statement shows whether the name has to originate from the SUB-SCHEMA SECTION or whether it can belong to another section.<br>The names cannot be indexed.                            |
| <i>literal</i>                                                                                            | The literal must be enclosed in quotation marks (or in apostrophes in the case of COBRUN QUOTE1) (see the “ <a href="#">COBOL Compiler, User’s Guide</a> ”). These are not part of the value of the literal. |
| <i>integer</i>                                                                                            | Is composed of a maximum of 15 digits                                                                                                                                                                        |

Table 18: Metavariables of COBOL-DML

You can write the COBOL program in any COBOL reference format using the COBOL-DML statements. In the free form reference format the previous column conventions no longer apply.

## 7.2 ID DIVISION

If the subschema is protected by a PRIVACY LOCK FOR COMPILE clause, the appropriate PRIVACY KEY must be specified in the COBOL DML program.

---

```
[PRIVACY. PRIVACY KEY FOR COMPILE IS literal.]
```

---

This entry must follow the PROGRAM-ID ENTRY:

*literal* must be one of the keywords assigned within the IDENTIFICATION DIVISION of the subschema.

## 7.3 DATA DIVISION

### SUB-SCHEMA SECTION

The Data Division of a COBOL program which uses COBOL DML must contain a section entitled SUB-SCHEMA SECTION.

---

```
SUB-SCHEMA SECTION.
```

---

This section must be the last section of the Data Division. It contains the DB clause.

The DB clause names a subschema and ensures that all areas which are necessary for communication with the DBH are reserved within the UWA (User Work Area).

The UWA consists of the following areas:

- SYSTEM-COMMUNICATION-LOCATIONS with the COBOL special registers
- the record area with
  - a separate area for each record type in the subschema
  - the IMPLICITLY-DEFINED-DATA-NAMES (area for items not belonging to record types)
- PRIVACY-RECORD with BPRIVACY locks

## DB entry

---

DB *subschema-name* WITHIN *schema-name\_*

---

*subschema-name*

must be the name of a subschema of the specified schema and must be known to the database system

*schema-name*

must be the name of a schema known to the database system.

The DB clause must occur once in each module which contains DML statements.

COBOL DML programs which are composed of various modules linked together can specify in their DB clauses either one subschema or different subschemas belonging to either one schema or different schemas.

As soon as the first COBOL module with DB clause is processed, communication with the DBH is opened even if no DML statement is called.

## 7.4 PROCEDURE DIVISION

The examples for DML statements refer to a sample database. The schema for this database is illustrated in the Appendix on [page 383](#).

### 7.4.1 Overview of COBOL DML statements

| Statement                                                                                                                                                                                                                                                                                                                                                                                    | Function                                                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Format 1:</p> <pre>ACCEPT <i>item-name-1</i> FROM [ { <i>record-name</i>                            { <i>set-name</i>                            { <i>realm-name</i> } } ] CURRENCY</pre> <p>Format 2:</p> <pre>ACCEPT <i>item-name-2</i> FROM [ { <i>record-name</i>                            { <i>set-name</i>                            { <i>item-name-3</i> } } ] REALM-NAME</pre> | <p>the database key value of the CRR, CRS, CRA or CRU into item <i>item-name-1</i></p> <p>Transfers</p> <p>the name of the realm in which the CRR, CRS, CRU or record belonging to a specified database key value is stored</p> |
| <pre>CONNECT [ <i>record-name</i> ] IO { <i>set-name-1,...</i>                            { ALL } [RETAINING CURRENCY FOR { <i>set-name-2,...</i>                            { SETS } ]</pre>                                                                                                                                                                                                | <p>Inserts the CRU into the set occurrence</p>                                                                                                                                                                                  |
| <p>Format 1:</p> <pre>DISCONNECT [ <i>record-name</i> ] FROM { <i>set-name,...</i>                                   { ALL } }</pre> <p>Format 2:</p> <pre>DISCONNECT ALL FROM <i>set-name,...</i></pre>                                                                                                                                                                                     | <p>Removes the CRU from set occurrences</p> <p>Removes all records from dynamic sets</p>                                                                                                                                        |
| <pre>ERASE <i>record-name</i> [ { PERMANENT                        { SELECTIVE                        { ALL } } MEMBERS ]</pre>                                                                                                                                                                                                                                                              | <p>Deletes the CRU from the database, where necessary with associated member records</p>                                                                                                                                        |

Table 19: COBOL DML statements (overview)

(part 1 of 6)

| Statement                                                                                                                                                                                                                                                                                                                                        | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> { FIND } { FETCH } } record-selection-expression [ RETAINING CURRENCY FOR        { MULTIPLE       { [REALM] [ { SETS       { set-name,... } ] [RECORD] } ]         </pre>                                                                                                                                                                  | <p>Selects one or more records from the database depending on the format of the record selection expression, makes the selected record into the CRU and, if on corresponding RETAINING entry was made, into the</p> <ul style="list-style-type: none"> <li>- CRR,</li> <li>- CRS in all sets in which it is owner or member,</li> <li>- CRA in the realm in which it is stored</li> </ul> <p>FETCH: additionally transfers the selected record into the UWA of the application program</p> |
| <p>Formats of the record selection expression</p> <p>Format 1:</p> <pre> [record-name] DATABASE-KEY IS item-name [OR { PRIOR { NEXT } ]         </pre> <p>Format 2:</p> <pre> { ANY { DUPLICATE } } record-name         </pre> <p>Format 3:</p> <pre> DUPLICATE WITHIN { record-name { set-name } [USING record-element-name,...]         </pre> | <p>Access via the database key</p> <p>Access via CALC key (hash procedure)</p> <p>Access to a record which matches the CRR or CRS in specific item contents or access to a record which satisfies the conditions of a previous search expression (FIND/FETCH-7)</p>                                                                                                                                                                                                                        |

Table 19: COBOL DML statements (overview)

(part 2 of 6)

| Statement                                                                                                                                                                                                                                                                                                                                                                                   | Function                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Format 4:</b></p> <pre> { LAST   FIRST   NEXT   PRIOR   integer   item-name } { record-name } [ WITHIN { set-name }   { RECORD } { realm-name } ]                     </pre> <p><b>Format 5:</b></p> <pre> CURRENT [ record-name ] [ WITHIN { set-name }   { realm-name } ]                     </pre> <p><b>Format 6:</b></p> <pre> OWNER WITHIN set-name                     </pre> | <p>Access to the first or last record, to the prior or next record to the CRR, CRS or CRA, or to a record whose position within a collection of records corresponds to a numeric value to be specified The collection or records can be a record type, a set occurrence, a realm or an intersection of sets of a record type within a realm.</p> <p>Access to the CRR, CRS, CRA or CRU</p> <p>Access to the owner record of a CRS</p> |

Table 19: COBOL DML statements (overview)

(part 3 of 6)

| Statement                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Function                                                                                                               |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <p><b>Format 7:</b></p> <pre> record-name[ WITHIN set-name-1[ CURRENT]]  (   USING record-element-name-1,...[ OR PRIOR/NEXT]   [ USING search-expression]   [ RESULT IN set-name-2]   [ LIMITED BY set-name-3]   [ TALLYING item-name-1]    [ SORTED[ { ASCENDING } ][ { BY } ]     { DESCENDING } ][ { ON } ]    record-element-name-2[[,]record-element-name-3]...    [[,][ { ASCENDING } ][ { BY } ]     { DESCENDING } ][ { ON } ]    record-element-name-4[[,]   record-element-name-5]...)]  search-expression ::= { complex-1[ AND complex-2] }                     { complex-2 }  complex-1 ::= [ NOT] condition-1                  { AND }                 { OR } [ NOT] condition-1)...  complex-2 ::= condition-2[ AND condition-2]...  condition-1 ::= record-element-name-6[ WITH MASK mask]                  { EQUAL                 { GREATER THAN } { item-name-2 }                 { &gt;                 { LESS THAN } { literal-1 }                 { &lt;  condition-2 ::= record-element-name-7 IS NEXT                  { GREATER THAN                 { &gt;                 { LESS THAN } { item-name-3 }                 { &lt; </pre> | <p>Access to records via freely selected items or counting and buffering of records found and searches using masks</p> |

Table 19: COBOL DML statements (overview)

(part 4 of 6)



| Statement                                                                                                                                                                                                                                                                                                                                                                                                    | Function                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>FINISH[ WITH CANCEL]</code>                                                                                                                                                                                                                                                                                                                                                                            | Terminates a transaction and releases locked realms and blocks                                          |
| <code>FREE[ ALL]</code>                                                                                                                                                                                                                                                                                                                                                                                      | Terminates the effect of the KEEP status                                                                |
| <code>GET[ { record-name<br/>record-element-name,... } ]</code>                                                                                                                                                                                                                                                                                                                                              | Makes available the CRU or individual items of the CRU in the record area of the UWA                    |
| <p><b>Format 1:</b></p> <code>IF[ NOT][ set-name] { OWNER<br/>MEMBER<br/>TENANT }</code><br><br><code>{ statement-1 } [ ELSE { statement-2 } ]</code><br><code>{ NEXT SENTENCE } [ ELSE { NEXT SENTENCE } ]</code> <p><b>Format 2:</b></p> <code>IF set-name IS[ NOT] EMPTY</code><br><br><code>{ statement-1 } [ ELSE { statement-2 } ]</code><br><code>{ NEXT SENTENCE } [ ELSE { NEXT SENTENCE } ]</code> | Tests set memberships in the program                                                                    |
| <code>KEEP</code>                                                                                                                                                                                                                                                                                                                                                                                            | Protects the CRU from access by other transactions until a FREE statement or the end of the transaction |
| <code>MODIFY { record-name<br/>record-element-name,... }</code><br><br><code>[ { INCLUDING } { SETS<br/>ONLY } { set-name-1,... } MEMBERSHIP ]</code><br><br><code>[ RETAINING CURRENCY FOR { ALL<br/>set-name-2,... } ]</code>                                                                                                                                                                              | Modifies item contents of the CRU and/or inserts it into another set occurrence within a set.           |
| <code>READY[ realm-name,... ]</code><br><br><code>[ USAGE-MODE IS [ { EXCLUSIVE } { RETRIEVAL }<br/>{ PROTECTED } { UPDATE } ] ]</code>                                                                                                                                                                                                                                                                      | Opens a transaction or a processing chain                                                               |

Table 19: COBOL DML statements (overview)

(part 5 of 6)

| Statement                                                                                                                                                                                                                                               | Function                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>SET <i>item-name-1</i>,...<i>IO item-name-2</i></code>                                                                                                                                                                                            | Transfers the contents of a database key item into one or more database key items                                                                                                                                                                                                                             |
| <code>STORE <i>record-name</i> [ RETAINING CURRENCY FOR</code><br><br>$\left\{ \begin{array}{l} \text{MULTIPLE} \\ \text{[REALM]} \left[ \begin{array}{l} \text{SETS} \\ \text{set-name, ...} \end{array} \right] \text{[RECORD]} \end{array} \right\}$ | Transfers a record from the UWA into the database as a new record<br><br>Inserts the new record into all sets for which its record type is defined in the schema as AUTOMATIC member<br><br>Sets up a new set occurrence for each set for which the record type is defined in the schema as owner record type |
| <code>USE FOR DATABASE-EXCEPTION [ ON { OTHER<br/> <i>literal</i>, ... } ]</code>                                                                                                                                                                       | Defines sequences of instructions to be executed if a DML statement is terminated with a database exception condition                                                                                                                                                                                         |

Table 19: COBOL DML statements (overview)

(part 6 of 6)

## 7.4.2 COBOL DML statements

### ACCEPT

The ACCEPT statement returns the contents of the specified currency information or the realm name belonging to a current record. There are two formats:

#### Format 1:

In Format 1, ACCEPT determines the database key value of the CRR, CRS, CRA or CRU and returns it in a COBOL item of type USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

---

```
ACCEPT item-name-1 FROM { record-name
 { set-name
 { realm-name } }] CURRENCY
```

---

#### *item-name-1*

must have been defined with a USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG clause.

If the CRR, CRS, CRA or CRU contains a database key value with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1, *item-name-1* must be defined with USAGE IS DATABASE-KEY-LONG. Otherwise, UDS/SQL transfers the value 0 to *item-name-1* and outputs the DATABASE-STATUS 15102.

#### *record-name, set-name, realm-name*

If one of these names is specified, the database key value of the relevant current record is supplied to *item-name-1*.

If none of the names is specified, the database key value of the CRU is supplied to *item-name-1*. This variant of the statement requires no DBH contact; it is thus especially fast.

If the specified current record is not known, the value zero is supplied to *item-name-1*.

*Example 1*

```

WORKING-STORAGE SECTION.
01 DB-KEY USAGE IS DATABASE-KEY.
01 DB-KEY-RED REDEFINES DB-KEY.
 02 REC-REF PIC X.
 02 RSQ PIC XXX.
01 REC-REF-NO.
 02 FILLER PIC X.
 02 REC-REF PIC X.
01 REC-REF-NO-RED REDEFINES REC-REF-NO.
 PIC S9(4) COMP.

01 REC-SEQ-NO.
 02 FILLER PIC X.
 02 RSQ PIC XXX.
01 REC-SEQ-NO-RED REDEFINES REC-SEQ-NO.
 PIC S9(9) COMP.

01 DB-KEY-OUTPUT.
 02 REC-REF PIC 9(3).
 02 RSQ PIC 9(8).
*
 FIND 4 ARTICLE.
 ACCEPT DB-KEY FROM CURRENCY.
 MOVE REC-REF OF DB-KEY-RED TO REC-REF OF REC-REF-NO.
 MOVE RSQ OF DB-KEY-RED TO RSQ OF REC-SEQ-NO.
 MOVE REC-REF-NO-RED TO REC-REF OF DB-KEY-OUTPUT.
 MOVE REC-SEQ-NO-RED TO RSQ OF DB-KEY-OUTPUT.
 DISPLAY "DB-KEY = " DB-KEY-OUTPUT UPON TERMINAL.

(OUT) DB-KEY = 00900000004

```

A DATABASE-KEY item is a 4-byte binary item that is used for storing database key values with a REC-REF  $\leq 254$  and an RSQ  $\leq 2^{24}-1$ . In the case of ACCEPT, UDS/SQL fills the DATABASE-KEY item as follows: on executing ACCEPT, the first byte of the item contains the record reference number (REC-REF), and the other 3 bytes contain the record sequence number (RSQ). The DATABASE-KEY item must thus be split into a REC-REF part and an RSQ part. Since COBOL does not recognize binary items of 1 or 3 bytes, the record reference number must first be stored in a binary item of half-word length, and the record sequence number in a binary item of word length. The two values are then converted into binary representation. The record type ARTICLE has the record reference number 9. The fourth record of this record type has the record sequence number 4.

*Example 2*

```

WORKING-STORAGE SECTION.
01 DB-KEY USAGE IS DATABASE-KEY-LONG.
01 DB-KEY-RED REDEFINES DB-KEY.
 02 REC-REF PIC S9(4) COMP.
 02 FILLER PIC S9(4) COMP.
 02 RSQ PIC S9(9) COMP.
01 DB-KEY-OUTPUT.
 02 REC-REF PIC 9(5).
 02 RSQ PIC 9(10).
*
 FIND 4 ARTICLE.
 ACCEPT DB-KEY FROM CURRENCY.
 MOVE REC-REF OF DB-KEY-RED TO REC-REF OF DB-KEY-OUTPUT.
 MOVE RSQ OF DB-KEY-RED TO RSQ OF DB-KEY-OUTPUT.
 DISPLAY "DB-KEY = " DB-KEY-OUTPUT UPON TERMINAL.

(OUT) DB-KEY = 0000900000000004

```

A DATABASE-KEY-LONG item is an 8-byte binary item that is used to store database key values.

In the case of ACCEPT, the DATABASE-KEY-LONG item is filled by UDS/SQL with a database key value as follows:

- Bytes 1-2: record reference number REC-REF)
- Bytes 3-4: not used
- Bytes 5-8: record sequence number (RSQ)

It is therefore necessary to split the DATABASE-KEY-LONG item into a REC-REF part and an RSQ part.

**Format 2:**

Format 2 of ACCEPT determines the realm to which the record specified in the FROM clause belongs.

---

```
ACCEPT item-name-2 FROM { record-name
 set-name
 item-name-3 }]REALM-NAME
```

---

*item-name-2*

must be an alphanumeric item.

*item-name-3*

must have been defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

If you want to define the associated realm for a record containing a database key value with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1, *item-name-3* must be defined with USAGE IS DATABASE-KEY-LONG.

*record-name, set-name, item-name-3*

If *record-name* is specified, the name of the realm to which the CRR of record type *record-name* belongs is returned in *item-name-2*.

If *set-name* is specified, the name of the realm to which the CRS of the set *set-name* belongs is returned in *item-name-2*.

If *item-name-3* is specified, the name of the realm to which the record with the database key value given in *item-name-3* belongs is returned in *item-name-2*.

If none of the names are specified, the name of the realm to which the CRU belongs is returned in *item-name-2*.

If the specified record does not exist or if the record has been deleted, blanks are returned in *item-name-2*.

## CONNECT

The CONNECT statement ensures that the CRU is made a member of one or more sets in which its record type is an optional member (i.e. MANDATORY MANUAL, OPTIONAL AUTOMATIC or OPTIONAL MANUAL member).

---

```
CONNECT [record-name] TO { set-name-1, ... }
 { ALL }

[RETAINING CURRENCY FOR { set-name-2, ... }]
 { SETS }
```

---

### *record-name*

If *record-name* is specified, the record type *record-name* must be declared as an optional member of the specified set(s). The CRU must belong to this record type. If *record-name* is not specified, the member record type is determined from the description of the first set. The same checks are undertaken for this record type as for a specified record name (this rule does not apply if ALL is specified).

### *set-name-1*,...

If this parameter is specified, the CRU is inserted into the corresponding set occurrence of each set name specified. The set occurrence is determined by the CRS. If the set is a dynamic set, the CRU must have the same record type as the current member records of this set. It should not represent the result set of a FIND-7 statement.

**ALL** If ALL is specified, the record type *record-name* must be declared as the member record type of at least one set which belongs to the subschema. The specified record is inserted into a set occurrence of all sets in which its record type is an optional member record type and of which it is not a member at the time. Dynamic sets cannot be used with ALL.

### RETAINING

This specification suppresses the modification of the currency information for *set-name-2*,.... If SETS is specified, the currency information for all sets involved remains unmodified.

RETAINING automatically applies to all sets not involved in the CONNECT statement.

READY USAGE-MODE UPDATE is always required for CONNECT statements, unless only dynamic sets have been specified. In the latter case, READY USAGE-MODE RETRIEVAL is also allowed.

## DISCONNECT

The DISCONNECT statement removes the CRU from one or more set occurrences, provided that its record type is declared as optional member of the specified set (Format 1).

Format 2 of DISCONNECT allows all records to be removed from dynamic sets.

### Format 1:

---

```
DISCONNECT[record-name] FROM { set-name, ... }
 { ALL }
```

---

#### *record-name*

If *record-name* is specified, the record type *record-name* must be declared as an optional member of the specified set(s) and must match the CRU.

If *record-name* is not specified, the member record type is determined from the description of the first set. The same tests are performed for this record type as for a specified record name (this rule does not apply if ALL is specified).

#### *set-name*,...

If this option is specified, the CRU is removed from the corresponding set occurrence of each specified set name if it is an OPTIONAL member.

if the set involved is a dynamic set, the CRU must have the same record type as the current member records of this set. The set must not be a result set of a FIND7 statement.

#### ALL

If this option is specified, *record-name* is removed from all sets of the subschema in which it is currently a member if it is an OPTIONAL member record.

READY USAGE-MODE UPDATE is always required for this format.

The CRS is retained, as is the sequential search option (FIND/FETCH-4).



**Format 2:**

---

DISCONNECT ALL FROM *set-name*,...

---

*set-name*,...

the sets specified must be dynamic sets; it is permitted to specify the result sets of a FIND-7 statement.

This format can also be used if the subschema employed is opened with READY USAGE-MODE RETRIEVAL.

## ERASE

The ERASE statement erases the CRU from the database and removes it from all set occurrences in which it was a member. In addition, the ERASE statement can

- erase all MANDATORY member records of the CRU or
- all records which are OPTIONAL members in set occurrences in which the record involved is the owner can be removed from these set occurrences and erased if required.

---

```
ERASE record-name [{ PERMANENT
 { SELECTIVE }
 ALL } MEMBERS]
```

---

### *record-name*

The subschema to which this record belongs must contain:

- all record types of which records are erased or removed from set occurrences as a result of the ERASE statement;
- all sets in which any record to be erased is either the owner or a MANDATORY member;
- all sets from which a record has to be removed;
- the realm to which the record to be erased belongs;
- all realms named in the WITHIN clauses of any record type which is a member record type of the record to be erased.

If *record-name* only is specified, the CRU is erased only if it does not possess any current member records.

### PERMANENT

Specifying this parameter

- erases the CRU,
- erases the MANDATORY member records of the CRU and
- removes the OPTIONAL member records of the CRU from the set occurrence.

If MANDATORY member records are also owner records of other set occurrences which are not empty, the ERASE statement operates under the same conditions on the member records of these set occurrences.

**SELECTIVE**

Specifying this parameter erases the following records:

- the CRU,
- the MANDATORY member records of the CRU and
- the OPTIONAL member records of the CRU if they do not have a different owner in another set. Otherwise, the OPTIONAL member records are removed from the set occurrence only.

If member records are also owner records of other set occurrences which are not empty, the ERASE statement operates under the same conditions on the member records of these set occurrences.

**ALL** Specifying this parameter erases the CRU and all its member records.

If member records are also owner records of other set occurrences which are not empty, the ERASE statement operates under the same conditions on the member records of these set occurrences.

If PERMANENT, SELECTIVE or ALL is specified, all realms of the subschema must be opened with READY USAGE-MODE EXCLUSIVE UPDATE without realm names being specified.

The entries in the currency table are marked as deleted. The sequential search option (FIND/FETCH-4) is retained.

Depending on the BMODTT setting the database key value of a deleted record can always be reused ("REUSE") after the end of the deleting transaction, or it remains locked ("KEEP").

## FIND/FETCH

The FIND statement selects a record from the database and makes it current record in all relevant columns of the currency table in which updating has not been suppressed. The columns involved are the columns of the associated record type, of all sets in which the record is owner or member and of the realm in which the record is stored. If the record found is a member of a dynamic set, the CRS involved is not updated unless the dynamic set is specified explicitly as a WITHIN set.

In one case the FIND statement selects a collection of records and makes its first record current record in all relevant columns of the currency table.

The FETCH statement implements all the functions of the FIND statement. In addition it transfers the new CRU into the UWA.

This additional function of FETCH is taken as read in the following text and will not be referred to explicitly henceforth.

---


$$\left. \begin{array}{l} \text{FIND} \\ \text{FETCH} \end{array} \right\} \text{ record-selection-expression} [ \text{RETAINING CURRENCY FOR} \\ \left. \begin{array}{l} \text{MULTIPLE} \\ \left[ \text{REALM} \right] \left[ \text{RECORD} \right] \left[ \left. \begin{array}{l} \text{SETS} \\ \text{set-name, ...} \end{array} \right\} \right] \end{array} \right\} ]$$


---

### *record-selection-expression*

Specifies the way in which the user wishes to access records. There are 7 formats of the record selection expression and these are explained on the following pages.

### RETAINING

Must be used either with MULTIPLE or with at least one of the options REALM, RECORD, SETS or *set-name,...* This enables the user to explicitly suppress the updating of the currency table in those columns where it is wished to preserve the database key value of the previous record.

If RETAINING... is not specified, the record found becomes

- the CRA of the realm in which it is stored,
- the CRR of its record type and
- the CRS of all sets in which it is the owner or a member.

The currency tables of the dynamic sets are not updated unless the set is specified explicitly as a WITHIN set.

**MULTIPLE**

Suppresses the updating of the entire currency table except for the CRU.

**REALM**

Suppresses the updating of the CRA.

**RECORD**

Suppresses the updating of the CRR.

**SETS**

Suppresses the updating of all CRSs.

*set-name,...*

Suppresses the updating of the CRSs of the specified sets.

**Formats of the record selection expression:****Format 1:**


---

```
[record-name]DATABASE-KEY IS item-name [OR {PRIOR
NEXT }]
```

---

*record-name*

The statement is only executed if the database key value specified in item *item-name* belongs to this record type *record-name*. UDS/SQL then selects the record belonging to the database key value from the database.

*item-name*

Must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG. The user must transfer the database key value of the desired record into this item beforehand.

If you want to search for a record containing a database key value with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1, *item-name* must be defined with USAGE IS DATABASE-KEY-LONG.

**OR PRIOR**

When OR PRIOR is specified, the record with the next lowest database key is supplied if no record with the specified database key exists.

The DATABASE-STATUS 04001 then shows that the record which was actually selected was not found.

In this case the DATABASE-STATUS 04024 shows that neither a record with the specified database key nor one with a lower database key exists.

OR PRIOR/NEXT can be used in COBOL2000 V1.5 or higher.

**OR NEXT**

When OR NEXT is specified, the record with the next highest database key is supplied if no record with the specified database key exists.

The DATABASE-STATUS 04001 then shows that the record which was actually selected was not found.

In this case the DATABASE-STATUS 04024 shows that neither a record with the specified database key nor one with a higher database key exists.

OR PRIOR/NEXT can be used in COBOL2000 V1.5 or higher.

*Example 1*

```

WORKING-STORAGE SECTION.
01 DB-KEY USAGE IS DATABASE-KEY.
01 DB-KEY-RED REDEFINES DB-KEY.
 02 REC-REF PIC X.
 02 RSQ PIC XXX.
01 RECORD-REF-NO.
 02 FILLER PIC X.
 02 REC-REF PIC X.
01 REC-REF-NO-RED REDEFINES REC-REF-NO.
 PIC S9(4) COMP.

01 REC-SEQ-NO.
 02 FILLER PIC X.
 02 RSQ PIC XXX.
01 REC-SEQ-NO-RED REDEFINES REC-SEQ-NO.
 PIC S9(9) COMP.

*
 FETCH-1.
 MOVE 9 TO REC-REF-NO-RED.
 MOVE 15 TO REC-SEQ-NO-RED.
 MOVE REC-REF OF RECORD-REF-NO TO REC-REF OF DB-KEY-RED.
 MOVE RSQ OF REC-SEQ-NO TO RSQ OF DB-KEY-RED.
 FETCH DATABASE-KEY IS DB-KEY; DISPLAY "DESIGNATION = "
 DESIGNATION OF ARTICLE UPON TERMINAL.

(OUT) DESIGNATION = BITTER

```

A DATABASE-KEY item is a 4-byte binary item that is used for storing database key values with a REC-REF  $\leq 254$  and an RSQ  $\leq 2^{24}-1$ . A database key value must be stored in this item in such a way that the first byte of the item contains the record reference number (REC-REF) and the remaining 3 bytes the record sequence number (RSQ). This makes it necessary to divide the DATABASE-KEY item into a REC-REF and an RSQ. Since COBOL does not recognize binary items of 1 or 3 bytes, the record reference number must first be stored in a binary item of half-word length, and the record sequence number in a binary item of word length. Both values are converted to binary representation in the process.

The output shows the 15th record of the 9th record type. This is the record designated BITTER and the record type ARTICLE.

*Example 2*

```

WORKING-STORAGE SECTION.
01 DB-KEY USAGE IS DATABASE-KEY-LONG.
01 DB-KEY-RED REDEFINES DB-KEY.
 02 REC-REF PIC S9(4) COMP.
 02 FILLER PIC S9(4) COMP.
 02 RSQ PIC S9(9) COMP.
 *
FETCH-1.
 MOVE 9 TO REC-REF OF DB-KEY-RED.
 MOVE 0 TO FILLER OF DB-KEY-RED.
 MOVE 15 TO RSQ OF DB-KEY-RED.
 FETCH DATABASE-KEY IS DB-KEY; DISPLAY "DESIGNATION = "
 DESIGNATION OF ARTICLE UPON TERMINAL.

(OUT) DESIGNATION = BITTER

```

A DATABASE-KEY-LONG item is a 8-byte binary item for storing database key values. Each such value must be stored in the DATABASE-KEY-LONG item as follows:

- Bytes 1-2: record reference number (REC-REF)
- Bytes 3-4: must contain the value 0
- Bytes 5-8: record sequence number (RSQ)

It is therefore necessary to split the DATABASE-KEY-LONG item into three parts: a REC-REF, a filler and an RSQ.



**Format 2:**

---

|           |   |                    |
|-----------|---|--------------------|
| ANY       | } | <i>record-name</i> |
| DUPLICATE |   |                    |

---

**ANY**

Selects a record of record type *record-name* associated with a pre-specified CALC key value, where this value is converted via a hash procedure into a relative page number. The user must first supply the items defined as CALC key in the schema (see the “[Design and Definition](#)” manual, LOCATION MODE clause) with the CALC key value of the desired record.

**DUPLICATE**

Is only allowed when the CALC key for record type *record-name* is defined in the schema as DUPLICATES ARE ALLOWED. UDS/SQL then searches in this record type for a record which is different from the CRR, but possesses the same CALC key value as the CRR and is located within the realm of the CRR. UDS/SQL uses the appropriate hash procedure for the search.

*record-name*

Must designate a record type which is defined in the schema as LOCATION MODE IS CALC (see the “[Design and Definition](#)” manual, LOCATION MODE clause).

When the record type *record-name* is distributed over more than one realm, the user must also previously have supplied the AREA-ID item (see the “[Design and Definition](#)” manual, WITHIN clause) with the name of the realm containing the required record if the record type is not the member record type of a distributable list.

If DUPLICATES ARE ALLOWED is declared in the schema, a CALC key value can belong to more than one record of the record type. All records belonging to a CALC key value of the record type within a realm will be obtained if the first record is searched for using ANY and all subsequent records with DUPLICATE.

*Example*

```
FETCH-2
 FETCH 9 ARTICLE-DESCR; DISPLAY "ART-NO. = " ART-NO OF
 ARTICLE-DESCR UPON TERMINAL.
 MOVE "CLOTHING" TO RLM-SELECTION-3.
 PERFORM READ-DUPLICATES UNTIL DATABASE-STATUS = 04021.
 DISPLAY "NO FURTHER DUPLICATE PRESENT" UPON TERMINAL.
 FETCH ANY ARTICLE-DESCR; DISPLAY "ART-NO. = " ART-NO OF
 ARTICLE-DESCR UPON TERMINAL.
 PERFORM READ-DUPLICATES UNTIL DATABASE-STATUS = 04021.
 DISPLAY "NO FURTHER DUPLICATE PRESENT" UPON TERMINAL.
 GO TO FETCH-3.
READ-DUPLICATES
 FETCH DUPLICATE ARTICLE-DESCR; IF DATABASE-STATUS IS NOT
 = 04021 DISPLAY "ART-NO. = " ART-NO OF ARTICLE-DESCR
 UPON TERMINAL.

(OUT) ART-NO. = 000005
(OUT) NO FURTHER DUPLICATE PRESENT
(OUT) ART-NO. = 000001
(OUT) ART-NO. = 000002
(OUT) ART-NO. = 000003
(OUT) ART-NO. = 000004
(OUT) ART-NO. = 000005
(OUT) NO FURTHER DUPLICATE PRESENT
```

In this case all article descriptions whose CALC key value matches the 9th article description are to be selected from the database. The example shows that all duplicates will be found only if the search is started with FETCH ANY.

**Format 3:**


---

```

DUPLICATE WITHIN { record-name } [USING record-element-name, ...]
 { set-name }

```

---

*record-name*

If USING is not specified, the collection of records in which the search is made is the result of the most recent search expression in the program.

*record-name* may then only be specified if the search expression has been programmed at record type level, i.e. in the form:

```

{ FIND }
{ FETCH } } record-name [USING search-expression]

```

(see FIND/FETCH-7 on [page 163](#), ).

*record-name* must then be the record name used there and UDS/SQL delivers a record from the result of the selection. A new check is made on the search expression so that only the currently valid record selected will be output even if changes have been made in the meantime by transactions running in parallel. If USING is specified, the collection of records in which the search is made is the named record type. UDS/SQL selects from the record type a record which matches the CRR in the items which have been defined by *record-element-name*,....

*set-name*

If USING is not specified, the collection of records in which the search is made is the result set of the most recent search expression in the program.

*set-name* may then only be specified if the search expression has been programmed at set level, i.e. in the form:

```

{ FIND }
{ FETCH } } record-name WITHIN set-name [USING search-expression]

```

(see FIND/FETCH-7 on [page 163](#)).

*set-name* must then be the set name used there and UDS/SQL delivers a record from the result of the selection. The search expression is also checked again. If USING is specified, the collection of records in which the search is made is the set occurrence which contains the CRS of the named set. From this set occurrence UDS/SQL selects a set which matches the CRS in the items defined by *record-element-name*,....

*record-element-name*,...

Must be specified whenever no reference is made to a preceding statement in the form:

```
{ FIND }
 { FETCH } [USING search-expression]
```

The record elements in which the record searched for is to match the CRR of record type *record-name* or the CRS of set *set-name* must be listed. The CRS must be a member record.

The record elements must belong to record type *record-name* or to the member record type of set *set-name*.

If a record element or a combination of record elements is specified which is defined as a key in the schema, UDS/SQL uses any available direct access paths. These direct access paths can be as follows:

- at record type level:  
Record SEARCH key table or hash procedure for SEARCH key
- for the primary key at set level:  
Pointer array, list or sort key table
- for a secondary key at set level:  
Set SEARCH key table or hash procedure

A prerequisite is that the key is defined in the schema as DUPLICATES ARE ALLOWED. If other record elements are specified, UDS/SQL searches sequentially through the record type.

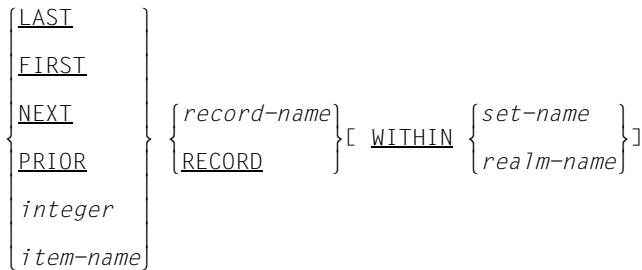
If the user wishes to read only those records which are selected as the result of a search expression, then a sequence of FIND-3 statements is required.

*Example*

```
FETCH-3.
 FETCH 2 PURCHASE-ORDER; DISPLAY "ORDER NO. = " P-ORD-NO
 " DATE = " P-ORD-DAY P-ORD-MTH P-ORD-YEAR UPON
 TERMINAL.
 FETCH DUPLICATE WITHIN P-ORD-PLACED USING P-ORD-MTH,
 P-ORD-YEAR; DISPLAY "P-ORD-NO = " P-ORD-NO
 " DATE = " P-ORD-DAY P-ORD-MTH P-ORD-YEAR UPON
 TERMINAL.

(OUT) ORDER-NO. = 7995 DATE = 100182
(OUT) ORDER-NO. = 8540 DATE = 030182
```

The second record of record type ORDER is dated 10.1.82.  
An order from the same month is then output.

**Format 4:**

**LAST** Delivers the last record in the collection. The collection is defined by *record-name*, RECORD, *set-name* and *realm-name*.

**FIRST** Delivers the first record.

**NEXT** Delivers the next record

- of the CRS when *set-name* is specified
- of the CRA when *realm-name* is specified
- of the CRR when WITHIN is omitted.

**PRIOR**

Delivers the prior record

- of the CRS when *set-name* is specified
- of the CRA when *realm-name* is specified
- of the CRR when WITHIN is omitted.

*integer*

Must be a positive (=standard) or negative signed integer. 0 is not allowed.

UDS/SQL delivers the record whose position corresponds to the specified numeric value. UDS counts the position:

- forward beginning at the first member record in the case of positive integers,
- backward beginning at the last member record in the case of negative integers.

An entry of 1 therefore corresponds to FIRST, whereas -1 corresponds to LAST. Negative values are not allowed if the set is defined in the SSL as MODE IS CHAIN without LINKED TO PRIOR.

*item-name*

Must be a numeric item which contains a signed decimal integer. The rules given above for *integer* also apply here.

*record-name*

Defines the named record type as the collection of records in which the search is to be made when WITHIN is not specified. The order is determined by ascending order of database key value.

## RECORD

Must not be used without WITHIN.

*set-name*

Defines the set occurrence which contains the CRS of the named set as the collection of records. The order is determined by the ORDER clause defined for this set in the schema. The owner record in this sequence is prior to the first member record.

If *record-name* is specified, it must designate the member record type of this set.

As FIND7 is not permitted with RESULT IN and SORTED, the set cannot be a sorted result set.

*realm-name*

If *record-name* is specified, the realm name must come from the DDL WITHIN clause of this record type. The record collection then consists of the records which belong to record type *record-name* and are located in realm *realm-name*. Otherwise the realm name must come from a DDL WITHIN clause of any record type belonging to the subschema, and the collection then consists of the records of the realm.

In both cases the sequence of the records is defined by ascending order of database key value.

*Example*

```

FETCH-4.
 FETCH FIRST SUPPLIER; DISPLAY SUPPL-NAME UPON TERMINAL.
 FETCH 2 PURCHASE-ORDER WITHIN P-ORD-PLACED; DISPLAY
 "P-ORD-NO = " ORDER-NO. " DATE = " P-ORD-DAY
 P-ORD-MTH P-ORD-YEAR UPON TERMINAL.

(OUT) MONA FASHIONS
(OUT) ORDER-NO. = 7995 DATE = 100182

```

The first record of record type SUPPLIER is output firstly. The second record is then selected from the set occurrence of this supplier in the set P-ORD-PLACED.

**Format 5:**


---

```

CURRENT[record-name] [WITHIN { set-name }
 { realm-name }]

```

---

*record-name*

Initiates access to the CRR of this record type and thereby resets the currency information to the status of the CRR if it is not followed by a WITHIN specification: the CRR again becomes the current record in all relevant columns of the currency table or in those columns where updating has not been suppressed with RETAINING.

*set-name*

Initiates access to the CRS of this set and thereby resets the currency information to the status of the CRS: the CRS again becomes the current record in all the relevant columns of the currency table or in those columns where updating has not been suppressed using RETAINING.

If *record-name* is specified, it must designate the member record type of the set. UDS/SQL then executes the statement only if the CRS belongs to this record type, i.e. if the CRS is not the owner record.

*realm-name*

Initiates access to the CRA of this realm and thereby resets the currency information to the status of the CRA: the CRA again becomes the current record in all relevant columns of the currency table or in those columns where updating has not been suppressed using RETAINING.

If *record-name* is specified, the realm name must come from the DDL WITHIN clause of this record type. UDS/SQL then executes the statement only if the CRA belongs to this record type. Otherwise the realm name must originate from the DDL WITHIN clause of any other record type belonging to the subschema.

If FIND CURRENT is used in the program without other specifications, the currency information is updated to the status of the CRU. It only makes sense to use this statement if RETAINING has been used previously.



*Example*

```
 FETCH-5.
 FIND FIRST SUPPLIER.
 FIND NEXT SUPPLIER RETAINING CURRENCY FOR SUPPLIERS.
 FETCH CURRENT SUPPLIER RETAINING CURRENCY FOR SUPPLIERS;
 DISPLAY SUPPL-NAME UPON TERMINAL.
 FETCH CURRENT SUPPLIER WITHIN SUPPLIERS; DISPLAY
 SUPPL-NAME UPON TERMINAL.

(OUT) BRAKSPEARS BREWERY
(OUT) MONA FASHIONS
```

The example shows the effect of the currency information and the **RETAINING** specification.

The first record output is the CRR of record type **SUPPLIER**. The second record output originates from the same record type, but is the CRS in set **SUPPLIERS**, in which **SUPPLIER** is member record type.

**Format 6:**


---

OWNER WITHIN *set-name*

---

*set-name*

Must not designate a SYSTEM set, since this cannot have an owner record.  
 UDS/SQL accesses the owner of the CRS. If the CRS itself is the owner record, this means that the CRS is accessed.

*Example*

```

FETCH-6.
 FETCH CURRENT PURCHASE-ORDER WITHIN P-ORD-PLACED; DISPLAY
 "P-ORD-NO = " ORDER-NO. " DATE = " P-ORD-DAY
 P-ORD-MTH P-ORD-YEAR UPON TERMINAL.
 FETCH OWNER WITHIN P-ORD-PLACED; DISPLAY "SUPPLIER-NAME = "
 SUPPL-NAME UPON TERMINAL.

(OUT) ORDER-NO. = 7995 DATE = 100182
(OUT) SUPPLIER-NAME = MONA FASHIONS

```

Here the owner of order 7995 in set P-ORD-PLACED is output.

**Format 7:**

*record-name* [ WITHIN *set-name-1* [ CURRENT] ]

|                                                                                 |                                                                                  |   |
|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------|---|
| }                                                                               | <u>USING</u> <i>record-element-name-1</i> , ... [ <u>COR</u> { <u>PRIOR</u> } ]  | } |
|                                                                                 | [ <u>USING</u> <i>search-expression</i> ] [ <u>RESULT</u> IN <i>set-name-2</i> ] |   |
|                                                                                 | [ <u>LIMITED</u> BY <i>set-name-3</i> ] [ <u>TALLYING</u> <i>item-name-1</i> ]   |   |
|                                                                                 | [ <u>SORTED</u> [ { <u>ASCENDING</u> } ] [ { <u>BY</u> } ]                       |   |
|                                                                                 | [ { <u>DESCENDING</u> } ] [ { <u>ON</u> } ]                                      |   |
|                                                                                 | <i>record-element-name-2</i> [ [ , ] <i>record-element-name-3</i> ] ...          |   |
| [ [ , ] [ { <u>ASCENDING</u> } ] [ { <u>BY</u> } ]                              |                                                                                  |   |
| [ { <u>DESCENDING</u> } ] [ { <u>ON</u> } ]                                     |                                                                                  |   |
| <i>record-element-name-4</i> [ [ , ] <i>record-element-name-5</i> ] ... ] ... ] |                                                                                  |   |

*search-expression* ::= { *complex-1* [ AND *complex-2* ] }  
 { *complex-2* }

*complex-1* ::= [ NOT ] *condition-1* [ { AND } ] [ NOT ] *condition-1* ...  
 { OR }

*condition-1* ::= *record-element-name-6* [ WITH MASK *mask* ] IS

|   |                                 |                        |
|---|---------------------------------|------------------------|
| } | [ <u>NOT</u> ] { <u>EQUAL</u> } | }                      |
|   | =                               |                        |
|   | { <u>GREATER THAN</u> }         |                        |
|   | >                               |                        |
|   | { <u>LESS THAN</u> }            |                        |
| < |                                 |                        |
|   |                                 | }                      |
|   |                                 | { <i>item-name-2</i> } |
|   |                                 | { <i>literal-1</i> }   |

*complex-2* ::= *condition-2* [ AND *condition-2* ] ...

*condition-2* ::= *record-element-name-7* IS NEXT

|   |                                        |                        |
|---|----------------------------------------|------------------------|
| } | [ <u>NOT</u> ] { <u>GREATER THAN</u> } | }                      |
|   | >                                      |                        |
|   | { <u>LESS THAN</u> }                   |                        |
|   | <                                      |                        |
|   | { <u>LESS THAN</u> }                   |                        |
| < |                                        |                        |
|   |                                        | }                      |
|   |                                        | { <i>item-name-3</i> } |
|   |                                        | { <i>literal-2</i> }   |

*record-name*

Defines the named record type as the collection in which the search is to be made when WITHIN is not specified.

*set-name-1*

Defines a set occurrence of the named set as the record collection. The required set occurrence is selected by specifying CURRENT.

*record-name* must designate the member record type of this set. In a dynamic set it is the record type whose records are located within the dynamic set.

## CURRENT

See [“Selecting a set occurrence” on page 169](#).

*record-element-name-1*

Must designate record elements of record type *record-name*. Here the user lists those elements of the record being searched which must match the preset values given in the UWA. The record elements must therefore be supplied with the desired values beforehand in the UWA. *record-element-name-1* can also be a group item name.

## OR PRIOR

Delivers the record with the preset key value given in the UWA. If no such record exists, the OR PRIOR clause returns the immediately preceding record of the set occurrence in accordance with the sort sequence. This sort sequence is defined by *record-element-name-1,...*, where *record-element-name-1,...* must be the ASCENDING KEY or DESCENDING key of the set *set-name-1*. Any violation of these conditions will be acknowledged by UDS/SQL at runtime with the DATABASE-STATUS 04101.

If there are duplicate records for a key value, UDS/SQL returns the one with the highest record sequence number (RSQ). Consequently, a FIND-7 OR PRIOR statement and a sequence of FIND-4 PRIOR statements could be used to determine all duplicates.

If the preset key value given in the UWA is lower than the smallest existing key value in the set occurrence, UDS/SQL will return the DATABASE-STATUS 04024.

## OR NEXT

Delivers the record with the preset key value given in the UWA.

If no such record exists, the OR NEXT clause returns the immediately following record of the set occurrence in accordance with the sort sequence. This sort sequence is defined by *record-element-name-1,...*, where one of the following conditions must be satisfied:

- *record-element-name-1,...* is the ASCENDING KEY, DESCENDING KEY or a SEARCH KEY USING INDEX of the set *set-name-1*.
- *record-element-name-1,...* is a SEARCH KEY USING INDEX of the record type *record-name*.

Otherwise, UDS/SQL will return the DATABASE-STATUS 04101 at runtime.

If there are duplicate records for a key value, UDS/SQL returns the one with the lowest record sequence number (RSQ). In other words, you can find all duplicates by using

- a FIND-7 OR NEXT statement and a sequence of FIND-4 NEXT statements or
- a FIND-7 OR NEXT statement and a sequence of FIND-3 statements.

If the preset key value given in the UWA is greater than the highest existing key value in the set occurrence, UDS/SQL will return the DATABASE-STATUS 04024.

*search-expression*

Specifies comparison conditions to be satisfied by the item contents in the records being searched.

The following special features can be utilized here:

- Storage of the entire set of records selected:  
UDS/SQL searches not just for one record, but transfers into a dynamic set all records which satisfy the conditions of the search expression. The first record found becomes the current record in the currency table.
- Not-equal as comparison condition:  
UDS/SQL can select all records which satisfy a not-equal condition (*condition-1*) or just the record which most closely matches the not-equal condition (*condition-2*).
- Logical operators AND, OR, NOT as comparison conditions
- Masks
- The option of presetting a value either in the UWA (*item-name-2/-3*) or in the statement itself (*literal-1/-2*)
- Storing selected records in a set (RESULT)
- Limiting the search to an intersection of two sets of records (LIMITED)
- Counting selected records (TALLYING)
- Sorting selected records (SORTED BY).

The subschema used must contain a temporary realm.

## USING

If USING is not specified, UDS/SQL interprets this as a search expression which is satisfied by all records of the record collection.

### *condition-1*

Delivers all records which satisfy the comparison condition. Negative item contents and und negative comparison contents are allowed only when the relational operator is "EQUAL".

A sequence of more than one *condition-1* is evaluated by UDS/SQL in accordance with the following rules: NOT has a higher priority than AND and AND has a higher priority than OR.

This sequence of priorities can be changed if logical operations which UDS/SQL is to evaluate as a single expression are placed in parentheses.

### *record-element-name-6*

Designates a record element of record type *record-name* which is to contain a preset value. It must not contain a variable item. Group items may also be used.

If UDS/SQL is not to check the entire contents of the record element, parts of the record can be suppressed using a mask.

The preset value is specified in *item-name-2* or *literal-1*.

### *mask*

Masks out part of *record-element-name-6*. *mask* must be defined in exactly the same way as *record-element-name-6*; otherwise it is automatically converted into the form of *record-element-name-2*.

Each byte of the mask must possess the value LOW VALUE or HIGH VALUE. The positions of *record-element-name-6* which are defined as LOW VALUE in the mask are not evaluated by UDS/SQL during comparison. The same positions in *item-name-2* must also be occupied by LOW-VALUE.

### *item-name-2*

Designates a record element or an item in the WORKING-STORAGE SECTION which contains the preset value for *record-element-name-6*. It must be defined in exactly the same way as *record-element-name-6* if WITH MASK is not used. If WITH MASK is used, only the length must correspond to *record-element-name-6*.

*item-name-2* must be provided with the desired value beforehand.

### *literal-1*

Is the value which is preset for *record-element-name-6* in the statement.

*literal-1* must have exactly the same length as *record-element-name-6*.

*condition-2*

Delivers all records that satisfy the comparison condition. Not-equal conditions, negative item contents and negative comparison contents are not allowed.

After *condition-1* in *complex-1* has been evaluated firstly, all *condition-2* conditions in *complex-2* are evaluated in order, starting with the left-most condition.

Adding NEXT restricts the selected set of *condition-2* in *complex-2* to those records of the record type *record-name* that contain the value W with the following two characteristics as *record-element-6*:

- W matches the comparison in *condition-2*.
- From all values in the database for *record-element-6* that match the comparison in *condition-2*, W lies closest to the comparison value specified by *literal-2* or *item-name-3*.

*record-element-name-7*

Designates a record element of record type *record-name* which is to contain a preset non-negative value. It must not contain a variable-length item. Group items may also be used. The value is preset in *item-name-3* or *literal-2*.

*item-name-3*

Designates a record element or an item in the WORKING-STORAGE SECTION which contains the preset value for *record-element-name-7*. It must be defined in exactly the same way as *record-element-name-7*.

The desired non-negative value for *item-name-3* must be provided beforehand in the UWA.

*literal-2*

Is the value which is preset for *record-element-name-7* in the statement. *literal-2* must be exactly the same length as *record-element-name-6* and must not be negative.

## RESULT

Is required if UDS/SQL is to store the selected records in an explicit dynamic set which can be accessed using other FIND/FETCH-4 statements. In this way the search for records can be programmed so that several hierarchy levels of the data structure are processed in all, i.e. the items used for the complete search can belong to different record types (complex query).

*set-name-2*

Must designate a dynamic set.

**LIMITED**

May only be specified if *set-name-1* does not designate a dynamic set.

LIMITED limits the collection of records to those records which

- belong to the record type *record-name* or to the selected set occurrence of the set *set-name-1*
- and are also contained in set *set-name-3*.

*set-name-3* must not be a sorted dynamic set, since it is not possible to create the intersection of a selected set (hit list) and a sorted dynamic set.

If *set-name-3* is an unsorted dynamic set, the SORTED clause (see below) is not allowed, since the intersection of a selected set and a dynamic set cannot be sorted.

*set-name-3*

Must designate a dynamic set.

**TALLYING**

Counts the records which satisfy the conditions of the search expression and stores the number in item *item-name-1*.

*item-name-1*

Must designate a numeric item.

**SORTED...**

May not be specified if *set-name-3* (see LIMITED clause) is a dynamic set.

The SORTED clause sorts the records that satisfy the search expression. If a user-specific translation table has been added to the UDS.MODLIB, the sort sequence defined therein will be used.

ASCENDING causes the records to be sorted in ascending order.

DESCENDING causes the records to be sorted in descending order.

*record-element-name-2* through *record-element-name-5*

Must identify records of the record type *record-name*.

This is a list of the record elements by which the records are to be sorted. The user can determine the priority of the sort items by means of the order of the record elements.

*record-element-name-2...5* may also be the name of a group item. The records are sorted according to type providing a subschema of UDS/SQL V2.0 or later exists which was not compiled using "SUBSCHEMA FORM IS OLD". Group items are handled in the same way as a character item. If nothing is entered, ascending is the default.

The default value for the first entry is ASCENDING, for repeated entries, the currently valid sort direction. All the sort items must, however, be sorted in the same direction.



**Selecting a set occurrence**

A set which is not a SYSTEM set generally possesses more than one set occurrence so that it is necessary first to specify the occurrence which is to be searched.

If CURRENT is specified, the set occurrence is determined by the CRS. The same applies if the selection method given in the schema for the set occurrences of this set is SELECTION THRU CURRENT OF SET.

If CURRENT is not specified and the selection method defined in the schema is SELECTION THRU LOCATION MODE OF OWNER, then only the key value which belongs to the owner record of the desired set occurrence must be transferred into the UWA.

The key is:

- a database key if the owner record type in the schema is defined with LOCATION MODE IS DIRECT or LOCATION MODE IS DIRECT-LONG.
- a CALC key if the owner record type is defined in the schema with LOCATION MODE IS CAL.

UDS/SQL selects the set occurrence by accessing the relevant user record by means of the database key or via the hash procedure and automatically making it CRS.

Table 20 gives an overview of how set occurrences must be selected.

| <div style="text-align: center;"> <b>Schema definition</b><br/><br/> <b>FIND statement</b> </div> |     | SET OCCURRENCE SELECTION IS THRU                                                                                 |                                                                                                                |                                                         |
|---------------------------------------------------------------------------------------------------|-----|------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
|                                                                                                   |     | CURRENT OF SET                                                                                                   | LOCATION MODE OF OWNER                                                                                         |                                                         |
|                                                                                                   |     |                                                                                                                  | LOCATION MODE IS<br><br>{ DIRECT<br>DIRECT-LONG }<br><br>{ <i>item-name</i><br><i>identifier</i> }             | LOCATION MODE IS<br><br>CALC USING <i>item-name</i> ... |
| <b>without CURRENT</b>                                                                            | CRS | Database key value in the item <i>item-name</i> or <i>identifier-1</i> in UWA or items named in the ALIAS clause | CALC key value in items <i>item-name</i> ,... or <i>identifier-2</i> in UWA or items named in the ALIAS clause |                                                         |
| <b>with CURRENT</b>                                                                               | CRS | CRS                                                                                                              | CRS                                                                                                            |                                                         |

Table 20: Selection of set occurrences

### Utilizing direct access paths

Direct access paths can be used to considerably accelerate the execution of a search order. Such direct access paths are:

- primary keys at record type level (LOCATION MODE IS CALC)
- primary keys at set level (ASCENDING/DESCENDING KEY); excluding the case ORDER IS SORTED without the addition INDEXED (only possible with MODE IS CHAIN)
- secondary keys at record type and set levels (SEARCH KEY USING INDEX/ CALC).

In order to ensure that a key in *record-element-name-1*,... is evaluated, the following must be observed with FIND/FETCH-7:

- If WITHIN *set-name-1* is not specified:  
The key is defined at record type level.
- If WITHIN *set-name-1* is specified:  
The key is defined in the set *set-name-1*.

In order to ensure that a key in *search-expression* is evaluated, the following must be observed with FIND/FETCH-7:

- Without WITHIN *set-name-1* or with WITHIN *set-name-1*, where *set-name-1* is a SYSTEM set of the MANDATORY AUTOMATIC member type:  
The key is defined at the record type level or in any SYSTEM set of the MANDATORY AUTOMATIC member type whose member record type is the underlying record type
  - With WITHIN *set-name-1*, where *set-name-1* is not a SYSTEM set of the MANDATORY AUTOMATIC member type:  
The key is defined in *set-name-1*.
- Direct access with FIND-7 USING *record-element-name-1*,...

If the key consists of several items, then *record-element-name-1*,... must exactly match the items and their sequence in the compound key.

If a compound key is identical to a group item in the subschema, then the group item name can be specified instead of *record-element-name-1*. The group item is then treated as an item with TYPE=CHARACTER. If *record-element-name-1* is the primary key at record type level, the table search is not supported. In this case, it is better to use FIND-2.

- Direct access with FIND-7 USING *search-expression*

With UDS/SQL V2.0 and below, a condition within a search expression could only be evaluated by means of an existing direct access path (“index”) if the relevant record element name is the elementary item of a single key or if it is a data group whose elementary items comprise exactly the items of a compound key.

In the case of the data group you had the option of declaring elementary items of the data group as irrelevant with regard to the condition by entering mask characters in the items, but the problems of data replication and data independence remained (for example, items must appear several times in the record if they occur in several data-group keys).

In UDS/SQL V2.2 and higher, only the relevant items are named in the search expression. A condition is evaluated while retaining direct access even if, as a data group or an elementary item, it only comprises part of the items of the underlying compound key.

Mask conditions combined with “AND” and “OR” which refer to one and the same index table are fully evaluated via direct access paths in UDS/SQL V2.2 and higher.

In UDS/SQL V2.0 and below, with "AND" only one of the conditions was evaluated via index table and the ensuing superset was processed further sequentially; with "OR" the the table search was carried out without restriction of the search area.

*Application scenarios for the use of compound keys*

The formulation of search queries by means of compound keys offers the following advantages:

- Data replication becomes superfluous because items need only appear once in the record
- Additional search keys can be avoided through clever selection of the items in compound keys
- Compound keys allow different item types, while data groups prohibit certain item types (data groups are handled like items of type PIC(X) and cannot therefore contain any non-printable numeric data types)
- Masks are not needed because irrelevant data can be omitted from the search condition

Evaluation by means of compound keys, therefore, generally leads to shorter records and ensures data independence on the part of the application programs. This means that less maintenance and modification is required, especially since, when there are changes in the data structure, data groups require more maintenance even without replication items.

For applications in the SQL environment which work with an ODBC server, compound keys offer ideal support because the ODBC interface does not support data groups, and the use of compound keys represents a considerable improvement here.

The following examples illustrate the use of the compound key in UDS/SQL V2.2 and higher. All are based on the record type CUST. C-INDUST denotes the industry, C-ZIP the zip code and C-VOLUME the total order volume of the customer:

RECORD NAME IS CUST

|              |    |                 |                |
|--------------|----|-----------------|----------------|
| UDS/SQL V2.0 | 02 | C-INDUST-ZIP    |                |
|              | 03 | C-INDUST        | PIC X(20)      |
|              | 03 | C-ZIP           |                |
|              | 04 | C-REGION        | PIC 9          |
|              | 04 | C-ZIPREST       | PIC 9999       |
|              | 02 | C-REGION-INDUST |                |
|              | 03 | C-REGION-1      | PIC 9          |
|              | 03 | C-INDUST-1      | PIC X(20)      |
|              | 02 | C-VOLUME-REGION |                |
|              | 03 | C-VOLUME        | PIC 9(9)       |
|              | 03 | C-REGION-2      | PIC 9          |
| UDS/SQL V2.2 | 02 | C-INDUST        | PIC X(20)      |
| and higher   | 02 | C-REGION        | PIC 9          |
|              | 02 | C-ZIPREST       | PIC 9999       |
|              | 02 | C-VOLUME        | TYPE DECIMAL 9 |

C-REGION-1 and C-REGION-2 are replications of C-REGION. C-INDUST-1 is a replication of C-INDUST.

C-VOLUME was declared in UDS/SQL V2.0 as a printable numeric value because C-VOLUME is to be processed as part of a data group with mask characters. In UDS/SQL V2.2 and higher, data groups and mask characters can be dropped if compound keys are used, and the item type can thus be freely selected.

*Example 1*

The task is to evaluate the customers on the one hand by industry and on the other by region. To ensure an efficient evaluation run, two search keys are defined each time. Data replication can be omitted in UDS/SQL V2.2 and higher.

**UDS/SQL V2.0:**

```
SEARCH KEY IS C-INDUST, C-REGION, C-ZIPREST USING INDEX
SEARCH KEY IS C-REGION-1, C-INDUST-1 USING INDEX

FETCH CUST USING C-INDUST-ZIP ='ELECTRCL*' *search for customers of an
industry*
FETCH CUST USING C-REGION-INDUST ='7*'
```

The '\*' character stands for mask characters at the end of an identifier. With respect to COBOL DML the statement is syntactically incorrect.

**UDS/SQL V2.2 and higher:**

```
SEARCH KEY IS C-INDUST, C-REGION, C-ZIPREST USING INDEX
SEARCH KEY IS C-REGION, C-INDUST USING INDEX *no replications*

FETCH CUST USING C-INDUST ='ELECTRCL'
FETCH CUST USING C-REGION = 7
```

*Example 2*

The task is to determine the 'major customers' of a region. UDS/SQL V2.0 offered different options for this, which either required data replication or could only be partly processed via the index table. For the comparison, only the two variants without data replication are considered:

**UDS/SQL V2.0:**

```
SEARCH KEY IS C-VOLUME USING INDEX

FETCH CUST USING C-VOLUME >=10000 AND C-REGION = 8
```

**or with search keys**

```
SEARCH KEY IS C-VOLUME USING INDEX
SEARCH KEY IS C-REGION USING INDEX

FETCH CUST USING C-VOLUME >=10000 AND C-REGION = 8
```

In the first case, only C-VOLUME can be evaluated via the index table; the superset thus formed is searched sequentially for C-REGION = 8.

In the second case, interim sets of hits are created and cut with respect to C-VOLUME and C-REGION. This solution is disadvantageous for update functions because an additional search key table has to be maintained.

UDS/SQL V2.2 and higher:

```
SEARCH KEY IS C-VOLUME, C-REGION USING INDEX
FETCH CUST USING C-VOLUME >=10000 AND C-REGION = 8
```

*Use of compound keys in search expressions with the Boolean operators AND and OR*

Compound keys only show their full effect within search expressions which contain either only the Boolean operator AND or only OR.

If both operators are mixed in a search expression, the AND components ('AND group') and the OR components (OR group) are evaluated separately.

All the following examples are based on the search keys defined here:

```
SEARCH KEY IS C-INDUST, C-REGION, C-ZIPREST USING INDEX
SEARCH KEY IS C-REGION, C-INDUST USING INDEX
SEARCH KEY IS C-VOLUME, C-REGION USING INDEX

FETCH CUST USING C-INDUST = 'ELECTRCL'
 AND (C-REGION = 2 OR C-REGION = 8)
```

C-INDUST and C-REGION cannot be evaluated in the same run.

The assignment of the search criteria to index tables is carried out according to the rules described below if the search criterion can be selected from more than one index table as a key item.

For this purpose we distinguish between "main key items" and "secondary key items". Main key items form the first key item within a compound key, Secondary key items occur as the second to the nth key item. It may happen that a search criterion is the main key item in one index table and the secondary key item in the other; e.g. C-INDUST and C-REGION in the first two search keys in the example refer alternately to the main key item and the secondary key item. The case where a key item is the main key item in multiple index tables is not dealt with; the selection of the index table is then uncertain.

*AND groups*

The search conditions are processed in the stipulated sequence.

Once the first main key item is found, the system first checks whether the search expression contains secondary key items which match it (i.e. key items which belong to the same index table) before searching for further main key items.

*Examples:*

```
FETCH CUST USING C-INDUST = 'ELECTRCL' AND C-REGION = 2
```

C-INDUST is the first main key item to be found. Although C-REGION is also a main key item (in a different index table), C-INDUST is assigned as the secondary key item.

```
FETCH CUST USING C-REGION > 7 AND C-INDUST = 'ELECTRCL'
AND C-VOLUME > 100000
```

C-REGION is the first main key item to be found. Although C-INDUST is also a main key item (in a different index table), C-REGION is assigned as the secondary key item. C-VOLUME is selected as the second main key item.

```
FETCH CUST USING C-INDUST NOT= 'ELECTRCL' AND C-REGION = 2
```

C-INDUST is actually the first main key item to be found. However, because "NOT=" is a very unselective condition, it is not acknowledged as a main key item; instead, C-REGION is selected in the other search key and C-INDUST is assigned to it as the secondary key item.

A compound CALC key can only be evaluated within an AND group, and even then only if the main and all secondary key items (in any order) are listed with the "=" operator without a mask.

Once the search conditions have been assigned to index or CALC tables, the order in which the tables are evaluated is determined:

1. Search condition as single key or data-group key with "=" operator
2. Search condition as single key or data-group key with "<" or ">" operator, or as main key item in a compound key in conjunction with an "=", "<" or ">" operator together with the associated secondary key items
3. All other search conditions

*OR groups*

If an OR group exists, the search conditions are processed in the order in which they occur. Only the main key items are selected for selective table access. Secondary key items are also evaluated via table access, but without restriction of the search area.

Example:

```
FETCH CUST USING C-INDUST = 'ELECTRCL' OR C-REGION = 2
```

Both items are main key items of two different index tables and are selected as such.

*Restrictions*

In the following cases, UDS/SQL cannot utilize or can only partially utilize the direct access facility even if a table exists:

- A key condition is logically ORed with a non-key condition.
  - The key refers to a CALC key and one of the following conditions is fulfilled:
    - *record-element-name-6* is masked
    - the key is *record-element-name-7* (NEXT)
    - *condition-1* or the relational operator is negated (NOT)
    - the relational operator is not “EQUAL”.
  - The key is a masked list sort key and the mask does not begin with HIGH VALUE (LOW VALUE mask).
  - The key is a DESCENDING KEY.
  - The key is part of *complex-2* (NEXT) and one of the following conditions is fulfilled:
    - *complex-2* consists of more than one condition
    - the search expression consists of *complex-1* and *complex-2*.
- Sequential search using single sweep
- DIRECT-CALC records are read in the order of their physical position, i.e. using single sweep.

Single sweep is only used with FIND3/7 at record type level, while the search with FIND FIRST and FIND NEXT is carried out using the DBTT.



### Handling of dynamic sets if status code $\neq$ 000

If a FIND/FETCH-7 USING *search-expression* does not find any records (status code = 024), the implicit dynamic set and any dynamic sets specified explicitly are subsequently empty. If a FIND/FETCH-7 USING *search-expression* (status code  $\neq$  000 and  $\neq$  024) is not successful, the implicit dynamic set and any dynamic sets specified explicitly are reset to their status prior to FIND/FETCH-7.

#### Example 1

```

FETCH-7-RECORD-ELEMENT-NAME.
 MOVE 1 TO P-ORD-MTH.
 MOVE 82 TO P-ORD-YEAR.
 FETCH PURCHASE-ORDER USING P-ORD-MTH, P-ORD-YEAR;
 DISPLAY "ORDER-NO. = " P-ORD-NO " O-DATE = " P-ORD-DAY
 P-ORD-MTH P-ORD-YEAR UPON TERMINAL.
 PERFORM READ-DUPLICATE-ORDER UNTIL DATABASE-STATUS = 04021.
 DISPLAY "NO FURTHER DUPLICATES" UPON TERMINAL.
 GO TO FETCH-7-SEARCH-EXPRESSION.
READ-DUPLICATE-ORDER.
 FETCH DUPLICATE WITHIN PURCHASE-ORDER USING P-ORD-MTH, ORDER-
 YEAR;
 IF DATABASE-STATUS IS NOT = 04021 DISPLAY "ORDER NO. = "
 P-ORD-NO" O-DATE = " P-ORD-DAY P-ORD-MTH P-ORD-YEAR
 UPON TERMINAL.

(OUT) ORDER-NO. = 0003 O-DATE = 150182
(OUT) ORDER-NO. = 7995 O-DATE = 100182
(OUT) ORDER-NO. = 8540 O-DATE = 030182
(OUT) ORDER-NO. = 7854 O-DATE = 030182
(OUT) NO FURTHER DUPLICATES PRESENT

```

The first FETCH statement selects a record which has the month and year corresponding to the date given in the UWA.

A subsequent sequence of FETCH DUPLICATE outputs all other orders which are dated with the same month and year.

A FETCH DUPLICATE statement requires a USING specification.

*Example 2*

```
WORKING-STORAGE SECTION.
01 END-CONDITION PIC XX.
 88 END VALUE IS "YES".
*
FETCH-7-SEARCH-EXPRESSION.
 FIND ARTICLE-DESCR USING DESIGNATION OF ARTICLE-DESCR IS
 = "SUMMER DRESS WITH JACKET" RESULT IN RESULTSET.
 FETCH ARTICLE WITHIN ORDER-SPECS USING PRICE IS < 300
 AND SIZE IS = 36 AND CURR-STOCK IS > MIN-STOCK;
 PERFORM READ-DUPLICATE-ARTICLE UNTIL DATABASE-STATUS = 04021.
 PERFORM READ-DUPLICATE-ARTICLE-DESCR UNTIL END.
 GO TO MODIFY-REC-ELEMENT.
READ-DUPLICATE-ARTICLE-DESCR
 FIND NEXT RECORD WITHIN RESULTSET; IF DATABASE-STATUS
 = 04021 MOVE "YES" TO END-CONDITION ELSE PERFORM
 READ-DUPLICATE-ARTICLE UNTIL DATABASE-STATUS = 04021.
READ-DUPLICATE-ARTICLE
 FETCH DUPLICATE WITHIN ORDER-SPECS; IF DATABASE-STATUS
 IS NOT = 04021 DISPLAY ARTICLE UPON TERMINAL.
```

This is an example of a complex search query: the search for article records first uses items of record type ARTICLE-DESCR and buffers the selected records in a dynamic set. Then each set occurrence of a selected record is searched for item contents of member record type ARTICLE.

## FINISH

The FINISH statement terminates a transaction and releases reserved realms and pages.

---

FINISH[ WITH CANCEL]

---

### WITH CANCEL

If this option is specified, the DBH performs a rollback to cancel all changes made by this transaction.

In openUTM programs a FINISH(without CANCEL) is delayed until PEND.

## FREE

The FREE statement terminates the extended record protection provided by KEEP for one or more records:

---

FREE[ ALL]

---

**ALL** If ALL is specified, all records which were given additional protection by KEEP are released for use by other transactions and the normal CRU lock is maintained.  
If ALL is not specified, only the additional record protection for the CRU is released.

## GET

The GET statement transports the entire CRU or part of the CRU into the UWA.

---

$$\underline{\text{GET}} \left[ \begin{array}{l} \textit{record-name} \\ \textit{record-element-name}, \dots \end{array} \right]$$

---

*record-name*

must designate the record type of the CRU. UDS/SQL then transports the entire CRU into the UWA.

*record-element-name,...*

must originate from the record type of the CRU. This entry is used to select specific record elements of the CRU whose contents UDS/SQL is to transport into the UWA.

If neither *record-name* nor *record-element-name,...* is specified, UDS/SQL transports the entire CRU into the UWA.

**IF**

The IF statement tests set membership in the program. There are two formats:

**Format 1:**


---

```
IF[NOT][set-name] { OWNER } { statement-1 } [ELSE { statement-2 }]_
 { MEMBER } { NEXT SENTENCE }
 { TENANT }
```

---

*set-name*

must not be a dynamic set; limits the test to the specified set.

If *set-name* is not specified, all sets of the subschema in which the record type of the CRU is an owner or member record type are tested.

**OWNER**

Tests whether the CRU is the owner record of a non-empty set occurrence.

**MEMBER**

Tests whether the CRU is a member record in a set occurrence.

**TENANT**

Tests whether the CRU is the owner record of a non-empty set occurrence or a member record in a set occurrence.

**Format 2:**


---

```
IF set-name IS[NOT] EMPTY { statement-1 } [ELSE { statement-2 }]_
 { NEXT SENTENCE }
 { NEXT SENTENCE }
```

---

*set-name*

Uses the CRS to select the set occurrence and checks if it is empty. *set-name* must not be a dynamic set.

If the CRS is a member record that has been deleted or disconnected, status code 031 is issued.

*statement-1* through *statement-2*

May be any COBOL or DML statements.

If a database exception condition arises, "FALSE" is detected, and a branch is made to *statement-2* or NEXT SENTENCE.

## KEEP

The KEEP statement protects the CRU from updating accesses by other processing chains and transactions even though another record has become the CRU.

---

### KEEP

---

The KEEP statement in a transaction means that the CRU is locked to other transactions until a FREE or FINISH statement is given.

The KEEP statement always locks the page in which the CRU is located. In a RETRIEVAL transaction, this page is locked to updating, in an UPDATE transaction to any type of access.

## MODIFY

The MODIFY statement can:

- replace all item contents or a selection of item contents of the CRU by values from the UWA.
- connect the CRU to another set occurrence within a set and
- make the CRU into the CRS of all sets in which its position is changed by the user. The CRA and CRR are not changed.

---

```

MODIFY { record-name } [{ INCLUDING } { ALL } MEMBERSHIP]
 { record-element-name, ... } [{ ONLY } { set-name-1, ... }
 [RETAINING CURRENCY FOR { SETS }
 { set-name-2, ... }]

```

---

### *record-name*

Must designate the record type of the CRU. If ONLY is not specified, UDS/SQL transfers all item contents of the items belonging to the record type from the UWA to the CRU. The desired values for the items must be specified by the user beforehand in the UWA.

### *record-element-name*,...

Must designate record elements from the record type of the CRU. This record type must not contain variable-length items. The contents of the record elements are to be transported from the UWA into the CRU. The desired values for the record elements must be provided beforehand by the user. If key values are changed, UDS/SQL automatically updates all the associated tables. This means particularly that the sequence of records within a set occurrence can change. The database key value of a record cannot be modified.

### INCLUDING

Is only allowed if the CRU is contained within at least one set occurrence. INCLUDING modifies the item contents of the CRU and connects the CRU into other set occurrences. The sets involved are specified by ALL or *set-name*,.... The user must first select from these sets the set occurrences and, if required, the positions within the set occurrences into which the CRU is to be connected. The selection can also be defined using SET-OCCURRENCE SELECTION THRU LOCATION MODE OF OWNER (see also FIND-7 on page 163, STORE (Selecting a set occurrence) on page 191, and STORE (Specifying the insertion point within the set occurrence) on page 192).

**ONLY** May not be used in combination with *record-element-name,...* and is only allowed if the CRU is contained in at least one set occurrence.

ONLY leaves the item contents of the CRU unchanged and connects the CRU into other set occurrences of the set involved. For selection of the set occurrence refer to INCLUDING above.

**ALL** Changes the CRU in all sets in which it belongs to a set occurrence. This does not apply to dynamic sets.

*set-name-1,...*

Must designate a set to which the CRU of a set occurrence belongs. This entry is used to list the sets in which the CRU is to be assigned to a new owner record. Specification of a dynamic set is ignored.

**RETAINING**

Can be used to suppress updating of the CURRENCY table in those cases where the database key value of the previous record is to be retained.

RETAINING automatically applies to all sets not involved in the MODIFY statement. Therefore specification of a dynamic set has no effect.

**SETS** Suppresses the updating of all CRSs.

*set-name-2,...*

Suppresses the updating of the CRSs of the specified sets.



*Example 1*

```

MODIFY-RECORD-ELEMENT.
 FETCH FIRST SUPPLIER; DISPLAY "SUPPLIER-NAME =
 "SUPPL-NAME
 "STREET-NO. = " SUPP-STREET-NO
 UPON TERMINAL.
 MOVE 83 TO SUPP-STREET-NO.
 MODIFY SUPP-STREET-NO.

(OUT) SUPPLIER-NAME = MONA FASHIONS STREET NO. = 1

```

The output shows a supplier and the street number to be modified. In this example MODIFY modifies street number 1 to 83.

*Example 2*

```

MODIFY-OWNER-ASSIGNMENT.
 FETCH OWNER WITHIN P-ORD-PLACED;
 DISPLAY "SUPPLIER-NAME = "
 SUPPL-NAME "ADDRESS = " SUPPL-STREET " "
 SUPP-STREET-NO" SUPPL-TOWN UPON TERMINAL.
 MOVE 2 TO SUPPL-NO.
 MOVE "BRAKSPEARS BREWERY" TO SUPPL-NAME.
 FIND ANY SUPPLIER.
 FIND CURRENT PURCHASE-ORDER RETAINING CURRENCY
 FOR MULTIPLE.
 MODIFY PURCHASE-ORDER ONLY P-ORD-PLACED MEMBERSHIP.
 FETCH OWNER WITHIN P-ORD-PLACED;
 DISPLAY "SUPPLIER-NAME = "
 SUPPL-NAME "ADDRESS = " SUPPL-STREET " "
 SUPP-STREET-NO " "SUPPL-TOWN UPON TERMINAL.

(OUT) SUPPLIER-NAME = MONA FASHIONS ADDRESS = 83 GUILDFORD STREET
 CHERTSEY
(OUT) SUPPLIER-NAME = BRAKSPEARS ADDRESS = 3 KING STREET
 BREWERY HENLEY-UPON-THAMES

```

An order which was assigned by mistake to supplier MONA FASHIONS is transferred to the set occurrence of supplier BRAKSPEARS BREWERY.

## READY

The READY statement opens a transaction or a processing chain and readies one or more realms for processing.

---

```
READY[realm-name,...][USAGE-MODE IS [{EXCLUSIVE} {RETRIEVAL}
 {PROTECTED} {UPDATE }]]
```

---

*realm-name*,...

The specified realms of the subschema are opened.

If *realm-name*,... is not specified, all realms belonging to the subschema are opened including the temporary realm.

USAGE-MODE

This defines the usage mode for the realms involved and thereby specifies the type of access allowed.

EXCLUSIVE

No other processing chains may use the realms concurrently.

PROTECTED

Concurrent updates by other processing chains are not permitted.

RETRIEVAL

Retrieve data

UPDATE

Retrieve and update data

If you do not specify USAGE-MODE, USAGE-MODE RETRIEVAL is assumed.

## SET

The SET statement transfers the contents of a database key item to one or more database key items.

---

```
SET item-name-1,... TO item-name-2
```

---

*item-name-1*,...

must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

*item-name-2*

must be defined with USAGE IS DATABASE-KEY or USAGE IS DATABASE-KEY-LONG.

In order to transfer database key values with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1 correctly, *item-name-2* and all items listed under *item-name-1*,... must be defined with USAGE IS DATABASE-KEY-LONG.

The following cases must be distinguished when transferring database key values:

- *item-name-2* and all items listed under *item-name-1*,... are of type USAGE IS DATABASE-KEY:  
In this case, the SET statement transfers the value of *item-name-2* to all items listed under *item-name-1*,... on a one-to-one basis.  
There is no change in the DATABASE-STATUS if the transfer is completed correctly.
- *item-name-2* is of type USAGE IS DATABASE-KEY, and at least one of the items listed under *item-name-1*,... is of type USAGE IS DATABASE-KEY-LONG:  
The SET statement transfers DATABASE-KEY values to DATABASE-KEY-LONG values as follows:
  - The record reference number (REC-REF) of 1-byte length is copied right-aligned into the corresponding 2-byte area of the receiving item.
  - The record sequence number (RSQ) of 3-byte length is copied right-aligned into the corresponding 4-byte area of the receiving item.

A DATABASE-STATUS of 00000 is output if the transfer is completed correctly.

- *item-name-2* and all items listed under *item-name-1*,... are of type USAGE IS DATABASE-KEY-LONG:  
In this case, the SET statement transfers the value of *item-name-2* to all items listed under *item-name-1*,... on a one-to-one basis.  
A DATABASE-STATUS of 00000 is output if the transfer is completed correctly.

- *item-name-2* is of type USAGE IS DATABASE-KEY-LONG; and at least one of the items listed under *item-name-1,...* is of type USAGE IS DATABASE-KEY:  
The SET statement transfers DATABASE-KEY-LONG values to DATABASE-KEY values as follows:
  - The record reference number (REC-REF; originally of 2-byte length) is copied into the corresponding 1-byte area of the receiving item.
  - The record sequence number (RSQ, originally of 4-byte length) is copied into the corresponding 3-byte area of the receiving item.

If a data loss is detected at runtime due to the truncation of leading positions in the record reference number and/or record sequence number (in cases where they are too large), the SET statement transfers the database key value 0 to the corresponding receiving item, and UDS/SQL reports DATABASE-STATUS 00102.

Note that the SET statement is not aborted, i.e. all other transfers are completed. A DATABASE-STATUS of 00000 is output if the database key value is transferred to all receiving items correctly.

## STORE

The STORE statement:

- transfers a record from the UWA into the database as a new record,
- inserts the new record into all sets for which its record type is defined in the schema as AUTOMATIC member.
- sets up a new set occurrence with the associated tables for each set for which the record type of the new record is defined in the schema as owner record type and for which the set occurrence population is defined in the SSL as greater than 0 and
- makes the new record into the current record in all the relevant columns of the currency table in which updating has not been suppressed using RETAINING. The columns involved are all those of the associated record type, of all sets in which the new record is member or owner and of the realm in which the record is stored.

A prerequisite is that the subschema includes the following as a minimum:

- all items named in the LOCATION MODE clause of the associated record type,
- all sets for which the record type is declared AUTOMATIC member in the schema and
- all items which are needed for correct selection of the owner records of these sets if the selection method is defined as SELECTION THRU LOCATION MODE OF OWNER.

---

STORE *record-name*

$$\left[ \text{RETAINING CURRENCY FOR } \left\{ \begin{array}{l} \text{MULTIPLE} \\ \left[ \text{REALM} \right] \left[ \text{RECORD} \right] \left\{ \begin{array}{l} \text{SETS} \\ \text{set-name, ...} \end{array} \right\} \right] \end{array} \right\} \right]$$


---

*record-name*

Designates the record to be stored. The items in this record must be supplied with the desired values beforehand in the UWA. Items of the record type which are not contained in the subschema are automatically filled with binary zeros when they are stored by UDS/SQL.

### RETAINING

Must be used either with MULTIPLE or with at least one of the parameters REALM, RECORD, SETS or *set-name,...*

RETAINING automatically applies to all sets not involved in the STORE statement. Therefore specification of a dynamic set has no effect.

## Selecting a realm

If more than one realm is provided in the schema in order to accept records of this record type, a specific realm must be defined:

1. The realm is uniquely defined if the record type is member in a set and the schema defines PLACEMENT OPTIMIZATION for the record type or MODE IS LIST for the set. In this case UDS/SQL stores the record
  - in the realm of the associated owner record or
  - in the realm of the DETACHED WITHIN clause in the set entry of SSL or
  - in the first realm of the DDL WITHIN clause of the member record type, even if the user selects another realm in accordance with the following procedure. When a record which is a member of a distributable list is stored, a free page is searched for in the DBH in the preferred realm if needed. In this case any realm specification in the AREA-ID item is ignored.
2. In other cases the user must define the realm by transferring the desired realm name into the item defined in the WITHIN clause of the schema as AREA-ID item for this record type (see the “[Design and Definition](#)” manual, WITHIN clause). UDS/SQL ignores this specification if the realm has been defined as in 1).

## Assigning database key values

Each record which is to be stored is assigned a database key value by the DBH. After a storing transaction has been canceled, the database key values which have been assigned are generally released again, but are initially not assigned again when records are stored. Only when the end of the DBTT has been reached are the values assigned again from the start of the DBTT in accordance with an activated online DBTT extension.

If there is provision in the schema DDL for the assignment of database key values by the user, there will be an item defined there to accept database key values (see the “[Design and Definition](#)” manual, LOCATION MODE clause). This item must be supplied with the desired value before a record is stored. The database key value of each record could thus be used as an information carrier (of the customer number, for example) that provides quick and direct access to a record using FIND-1.

Database key values that are assigned by the user are handled by UDS/SQL as follows:

- UDS/SQL independently determines the record reference number (REC-REF) associated with the record type *record-name*; the record reference assigned by the user is ignored. In other words, UDS/SQL always assigns the desired database key value correctly, even if an invalid record reference number is specified.

- UDS/SQL will ignore the record sequence number (RSQ) specified by the user and independently assign an available RSQ in the following cases:
  - If the user assigns a record sequence number with a value of 0.
  - If the record sequence number given by the user is greater than the current number of lines in the database key translation table (DBTT) for the record type *record-name* (see the “[Design and Definition](#)” manual). Assigning a higher record sequence number therefore does not result in online DBTT extension.
  - If there is already a record for the record type *record-name* with the record sequence number assigned by the user.

If the record type *record-name* is not defined in the schema with LOCATION MODE IS DIRECT or LOCATION MODE IS DIRECT-LONG, the database key values for records of the record type *record-name* are assigned by UDS/SQL.

ACCEPT can be used to check whether the actual database key value matches the specification given (see the examples for FETCH-1 and ACCEPT, starting on [page 151](#) and [page 139](#), respectively).

### Selecting a set occurrence

In each set in which the record type *record-name* is AUTOMATIC member, the stored record is automatically inserted into a set occurrence. The owner records of the record to be stored must therefore be selected beforehand. This must be done by using the selection methods defined for these sets in the schema (see the “[Design and Definition](#)” manual, SET OCCURRENCE SELECTION clause). Depending on the schema definition, either the set occurrence is defined by the CRS or the database key or CALC key value of the associated owner record must be transferred into the UWA. UDS/SQL then automatically selects the set occurrence by accessing the owner record via the database key or the hash procedure. [table 21](#) summarizes how a set occurrence is selected.

| Schema definition                             | SET OCCURRENCE SELECTION IS THRU |                                                                                                                |                                                                                                                |
|-----------------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
|                                               | CURRENT OF SET                   | LOCATION MODE OF OWNER[ ALIAS FOR { <i>item-name</i><br><i>identifier-1</i> }<br>IS <i>identifier-2</i> ]...   |                                                                                                                |
|                                               |                                  | LOCATION MODE IS                                                                                               |                                                                                                                |
|                                               | { DIRECT<br>DIRECT-LONG }        | { <i>item-name</i><br><i>identifier-1</i> }                                                                    | CALC USING <i>item-name</i> ,...                                                                               |
| Set occurrence is determined by the user via: | CRS                              | Database key value in the item <i>item-name</i> or <i>identifier</i> in UWA or items named in the ALIAS clause | CALC key value in items <i>item-name</i> ,... or <i>identifier-2</i> in UWA or items named in the ALIAS clause |

Table 21: Selection of a set occurrence

### Specifying the insertion point within the set occurrence

In the selected set occurrence the stored record is given the location which corresponds to the order defined for the set in the schema (see the “[Design and Definition](#)” manual, ORDER clause). The cases in which the insertion location depends on user specifications are shown in [table 22](#) below.

| Schema definition                              | ORDER IS                                                      |                                                                     |
|------------------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------------|
|                                                | NEXT/PRIOR<br>SET OCCURRENCE SELECTION IS THRU CURRENT OF SET |                                                                     |
| Insertion point is determined by the user via: | CRS                                                           | Key value in items <i>item-name</i> ,... in the record to be stored |

Table 22: Determining the insertion point within a set occurrence

In the case SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER, the CRS has no meaning; the insertion point for ORDER IS NEXT or PRIOR is, just as for ORDER IS FIRST or LAST, at the beginning or end, respectively, of the set occurrence.



*Example*

```
STORE-ARTICLE.
 MOVE "CLOTHING" TO RLM-SELECTION-4.
 MOVE 4 TO ART-NO OF ARTICLE-DESCR.
 FIND ARTICLE-DESCR USING ART-NO OF ARTICLE-DESCR.
 MOVE "MONA FASHIONS" TO SUPPL-NAME.
 FIND SUPPLIER USING SUPPL-NAME.
 MOVE 8 TO COL-NO OF ARTICLE.
 MOVE 36 TO SIZE.
 MOVE 3721 TO ART-NO-AVAIL.
 MOVE 5 TO COL-NO-AVAIL.
 MOVE 285 TO PRICE..
 STORE ARTICLE.
```

The item designated by color number 8, size 36 etc. is stored in the CLOTHING realm and at the same time inserted into the set occurrences of article description 4 and supplier MONA FASHIONS.

## USE

The USE statement defines command sequences to be executed if a DML statement is terminated with a database exception condition.

---

```

USE FOR DATABASE-EXCEPTION[ON { OTHER
 { literal-1,... } }]+

```

---

**ON** This specification enables the user to use several USE statements in a program. The specified literals must be unique for each USE statement.

If a USE statement is specified without ON, it must be the only one within a program.

If ON is omitted or ON OTHER is specified, the command sequence associated with the USE statement is executed on all database exception conditions.

### OTHER

may only be used once within a program.

must be specified if not all database exception conditions were specified by *literal-1,....*

*literal,...*

Each *literal* must represent a database exception condition. However, no *literal* may be '00000'.

The USE statement may only be specified in the DECLARATIVES of a COBOL program.

It must be the first statement immediately following the section header. The associated section must contain the command sequences to be executed due to the USE statement.

The command sequences must remain within the DECLARATIVES and must not be referenced by other sections of the COBOL program. Within the command sequence the command sequence of another USE statement may only be addressed by means of PERFORM.

If a database exception condition occurs, the current values are stored in the special registers. If one DML statement causes several exception conditions, the contents of the special registers refer to that which was last recognized.

*Example*

```

 .
 .
 .
PROCEDURE DIVISION.
*
DECLARATIVES.
*
PROTECT SECTION.
 USE FOR DATABASE-EXCEPTION ON "12901",
 "12950",
 "12951",
 "12952",
 "12953",
 "12954",
 "12955".
M1. DISPLAY "*** INVALID PRIVACY-INPUT! ***" UPON T.
 DISPLAY "*** DATABASE-STATUS = " DATABASE-STATUS " ***"
 UPON T.
 STOP RUN.
*
NOT-FOUND SECTION.
 USE FOR DATABASE-EXCEPTION ON "04024".
M2. MOVE HIGH-VALUE TO NOTE.
*
REST SECTION.
 USE FOR DATABASE-EXCEPTION ON OTHER.
M3. DISPLAY "***** ERROR OCCURRED! *****" UPON T.
 DISPLAY "DATABASE-STATUS = " DATABASE-STATUS UPON T.
 DISPLAY "RECORD TYPE = " DATABASE-RECORD-NAME UPON T.
 DISPLAY "SET = " DATABASE-SET-NAME UPON T.
 DISPLAY "REALM = " DATABASE-REALM-NAME UPON T.
 FINISH WITH CANCEL.
 STOP RUN.
*
END DECLARATIVES.
*
*
*
PROGRAM SECTION.
*
LEAD.
 .
 .
 .

```



---

## 8 CALL DML reference section

This chapter describes the DML CALL interface and the CALL DML calls with the associated parameter definitions and formats.

### 8.1 CALL interface

There are two CALL interface variants:

- The first 8 characters of the names of database objects must be unique. This applies to realm, set and record names within a subschema and the names of items within a record. This restriction must be taken into account when the database is designed. This variant is referred to below as (CALL8).
- No such tight restriction with regard to the uniqueness of names must be observed by the second CALL interface variant. This variant can be used for any database and is referred to below as (CALL30).

The CALL interface can be used in all programming languages which observe the following register conventions when calling a subroutine:

Register 15: Branch destination

Register 14: Return address

Register 1: Address of the parameter area

This is the case for FORTRAN and COBOL.

The branch destination is the same for each DML call. The address is >DML<.

The addresses in the parameter list (Register 1) are listed contiguously. They point to locations in the programs containing the appropriate parameter information. In COBOL or FORTRAN programs the compiler generates the parameter list when the standard CALL is used.

For examples in different programming languages refer to [page 289](#).

## 8.2 Parameter definitions

CALL DML recognizes the following parameters:

1. Function code (FCOD)
2. Function option (FOPT)
3. Secondary option (SOPT)
4. User information (UINF)
5. Record name (RECN)
6. Set name (SETN)
7. Realm name (RLMN)
8. Item name (ITMN)
9. Record area (RECA)
10. Special parameter 1 (SPP1)
11. Special parameter 2 (SPP2)
12. Special parameter 3 (SPP3)

The function code and user information parameters are used in each CALL DML call. Which of the remaining parameters are used depends on the function in combination with the function option and secondary option. The contents of the first three parameters thus determine which parameter items are read or written by the CALL DML converter. These parameters have to be taken from the descriptions of the individual calls or from the respective overview.

Only the addresses of the parameter items have to be transferred; their length is determined by the CALL DML rules, by the subschema or by the syntax of the item contents (lists of names, search expression).

Each parameter has a fixed location in the parameter list of the call and in the corresponding address list. It is therefore necessary, to specify the names of items for unused parameters too, even though the contents of these items are not significant for the execution of the DML call. The CALL DML converter only interprets the addresses of the parameter items that are used.

This rule allows the same CALL to call all possible DML statements. Only the contents of the parameter items used have to be modified accordingly before execution of the CALL instruction.

The information contained in parameter items must be stored in a precisely defined format so that it can be correctly interpreted by both the CALL DML converter and the application program.

The formats of the individual parameter items have been made as uniform as possible for ease of use. Deviations from the standard formats are explained in detail in the description of the CALL calls and in the format table.

For a description of the secondary option, user information, and the special parameters 1 and 2, refer to pages [203](#) to [209](#).

The syntax and format of search expressions are described under FIND7A/FTCH7A.

## 8.2.1 Rules

The user may omit parameters which would normally be specified at the end of the parameter list provided these parameters are not used by the CALL.

- The first parameter is always the function code (FCOD).
- The order specified above must be adhered to.
- Up to the last parameter used no parameter item may be left out.
- Optional parameters ([ ]) count as used parameters. If no other value is specified for these parameters, the associated items must be filled by the requisite number of blanks.
- Parameters identified in the overview by a hyphen or for which there is no entry are among the parameters not used.
- The contents of unused parameter items are not relevant.
- Each parameter is subject to formatting rules with regard to the contents of the parameter item; these rules may depend on the function and must be adhered to.
- Any name can be chosen for the parameter items.
- Indexed items cannot be name, i.e. individually selected, in CALL DML . Processing of indexed items requires the complete subschema format of the record.
- The names of realms, sets, records and items can occur at various positions in the CALL DML parameters.  
In the (CALL8) variant, the names specified must be 8 characters long, in the (CALL30) variant, they must be 30 characters long. If necessary the names must be padded with blanks.  
The user selects the variant he or she wishes to use via the UINF parameter item end of user information (see [page 205](#)):
  - (CALL8) variant: “USINF\*”
  - (CALL30) variant: “UINF1\*”

- CALL DML recognizes two formats for the transfer of names: the single name format for specifying a single name and the name list format for specifying several names.

- Single name format

*Example for (CALL8) variant:*

```
OF-SET ---> OF-SET
```

- Name list format

The names are separated by commas and enclosed in parentheses.

*Example for (CALL8) variant:*

```
(INCREDB)
(EXAMPLE1 , EXAMPLE , EXAMP , EX)
```



## 8.2.2 Format table

| Parameter             | Contents                                                 | Length for variant |          | Format                                                                          |
|-----------------------|----------------------------------------------------------|--------------------|----------|---------------------------------------------------------------------------------|
|                       |                                                          | (CALL8)            | (CALL30) |                                                                                 |
| 1. FCOD               | -                                                        | 6                  |          | Keyword                                                                         |
| 2. FOPT               | -                                                        | 6                  |          | Keyword                                                                         |
| 3. SOPT               | RET, VAR                                                 | 6/≥2               |          | Fixed format/ free format, see <a href="#">page 203</a>                         |
|                       | RET, RES, LMS, TAL<br>SOA/SOD, NOW                       | 12/≥2              |          |                                                                                 |
| 4. UINF               | DBH communication                                        | 126                |          | see <a href="#">page 205</a>                                                    |
| 5. REC N              | Record-name                                              | 8                  | 30       | Single name format                                                              |
|                       | Record-name or .....                                     | 8                  | 30       | Single name format or blanks                                                    |
|                       | RECORD                                                   | 8                  | 30       | RECORD.../RECORD.....                                                           |
| 6. SET N              | set name                                                 | 8                  | 30       | Single name format                                                              |
|                       | 1-25 set names                                           | 10-226             | 32-776   | Name list format                                                                |
| 7. RLMN               | Realm name                                               | 8                  | 30       | Single name format                                                              |
|                       | Item name in the secondary<br>option SOA or SOD          | 8                  | 30       | Single name format                                                              |
|                       | 1-25 realm-names                                         | 10-226             | 32-776   | Name list format                                                                |
|                       | 1-25 item names in the<br>secondary option SOA or<br>SOD | 10-226             | 32-776   | Name list format                                                                |
| 8. ITMN               | Item name                                                | 8                  | 30       | Single name format                                                              |
|                       | 1-25 item names                                          | 10-226             | 32-776   | Name list format                                                                |
|                       | Search expression                                        | -                  |          | see <a href="#">page 245</a>                                                    |
| 9. RECA <sup>1)</sup> | Complete record                                          | -                  |          | Subschema format                                                                |
|                       | Selection of items without<br>VAR option                 | -                  |          | Item positions in record area as<br>subschema format                            |
|                       | Selection of items using<br>VAR option                   | -                  |          | Record format corresponds to<br>item name list, length = sum of<br>item lengths |

Table 23: CALL DML formats

(part 1 of 2)

| Parameter | Contents                                                                                                                                             | Length for variant |          | Format                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|----------|-----------------------------------------|
|           |                                                                                                                                                      | (CALL8)            | (CALL30) |                                         |
| 10. SPP1  | RETAINING without set names                                                                                                                          | 9                  |          | see <a href="#">page 209</a>            |
|           | RETAINING with set names                                                                                                                             | 17-235             | 39-785   |                                         |
|           | Subschema (READYC)                                                                                                                                   | 30                 |          | Full subschema name, padded with blanks |
| 11. SPP2  | Integer (FIND4/FTCH4)                                                                                                                                | 4                  |          | Binary integer $\neq 0$                 |
|           | Result set name (FIND7A/FTCH7A)                                                                                                                      | 8                  | 30       | Single name format                      |
|           | Implicitly defined data:<br>DB key of LOCATION<br>MODE clause and/or realm name of WITHIN clause<br>(STORE/STOR1L,<br>STORE2/STOR2L,<br>FIND2/FTCH2) | 42                 |          | see <a href="#">page 210</a>            |
| 12. SPP3  | Limited set name (FIND7A/FTCH7A)                                                                                                                     | 8                  | 30       | Single name format                      |

Table 23: CALL DML formats

(part 2 of 2)

<sup>1</sup> When the record area is used to return data, it is normally overwritten also if the DML statement FTCH or GET has been aborted due to an error (status  $\neq 000$ ). However, if the database status begins with C, P or S, the record area remains unchanged.

### 8.2.3 Format of secondary option (SOPT)

The following secondary functions can be selected with this parameter:

|         |                             |                   |
|---------|-----------------------------|-------------------|
| RET     | Retaining                   |                   |
| VAR     | Abbreviated record format   |                   |
| RES     | Result set                  | } FIND7A/FTCH7A   |
| LMS     | Limited by dynamic set      |                   |
| TAL     | Tallying                    |                   |
| SOA/SOD | Sorted ascending/descending |                   |
| NOW     | No Wait                     | READYC, FIND/FTCH |

This parameter is always optional. If more than one secondary function is available for selection, then it is possible to select any combination of these functions. If none of the available functions are to be used, the parameter item must be blank-filled (6-12) or be given blanks enclosed in parentheses.

The functions 'retaining', 'result set' and 'limited-by-dynamic-set' are linked to the use of additional parameters, namely special parameters 1 (RET), 2 (RES) and 3 (LMS). These parameters must only be specified if the corresponding secondary option has been specified. Otherwise it is not necessary to enter these parameters in the parameter list of the CALL DML statement provided they come at the end of the list.

Two methods of representation are available for the transfer of the secondary option symbols in the parameter item "secondary option". Both methods of representation are equally suitable for generating or modifying the parameter item "secondary option" at program runtime.

Future extensions to the secondary options will only be accepted in the free-format method in order to ensure the upward compatibility of "secondary option" parameter items in application programs.

### Fixed-format secondary option

The “secondary option” parameter has a fixed length (6 or 12 characters). Each symbol has a fixed position within the parameter. If a function is not required, blanks must be entered at the corresponding point.

Syntax:

- all formats except FIND7A/FTCH7A      {RET}{VAR}
- FIND7A/FTCH7A                            {RET}{RES}{LMS}{TAL}

*Examples*

1.             or RET   or   VAR or RETVAR
2. 12 blanks or RETRESLMSTAL or   RESLMS   etc.

### Free-format secondary option

The “secondary option” parameter does not have a fixed length. It begins in the first byte of the parameter item with an open parenthesis and ends with a close parenthesis. The function symbols can be in any sequence and at any location but may not be repeated. They can follow in immediate succession or be separated by any number of blanks and commas.

Syntax:

- $$\text{param} ::= ( \{ [ \left\{ \begin{array}{c} \text{sym} \\ \text{---} \end{array} \right\} ] [ \left\{ \begin{array}{c} \text{---} \\ \text{---} \end{array} \right\} ] \dots )$$
- $$\text{sym} ::= \text{RET/VAR/RES/LMS/TAL} / \left\{ \begin{array}{c} \text{SOA} \\ \text{SOD} \end{array} \right\} / \text{NOW}$$

*Examples*

1. ( ) or ( , , , , )
2. (RET) or (RET, , , )
3. (RETRESLMS) or ( , , , RES, RET, LMS)

The examples in any one line above are identical in meaning; they were deliberately chosen to illustrate the format options.

## 8.2.4 Format of user information (UINF)

The parameter user-information is used by all CALL DML calls. It identifies a transaction to the DBH. In multi-DB applications it also selects the database which is to be accessed by the DML call.

Details of which user-information data must be saved can be found in the following section.

No changes may be made by the user anywhere in the entire parameter area during the execution of a CALL DML call. Only one DML call can of course be processed at a time for a single transaction. This constraint also applies to program routines which are outside the normal execution sequence of a transaction (error routine, STXIT).

It is recommended that the parameter item for user information should be aligned on a word boundary, but this is not a requirement. The length of the item is 126 bytes.

### User information area

| Contents                | Input | Output | Length | Displ. | Type                            |
|-------------------------|-------|--------|--------|--------|---------------------------------|
| System Communication    |       |        |        |        |                                 |
| Locations               |       | X      | 30     | 0      | character                       |
| - realm name            |       | X      | 30     | 30     | character                       |
| - record name           |       | X      | 30     | 60     | character                       |
| - set name              |       |        |        |        |                                 |
| Result items            |       |        |        |        |                                 |
| - statement code        |       | X      | 2      | 90     | character                       |
| - status code           |       | X      | 3      | 92     | character                       |
| Empty item              |       |        | 1      | 95     | binary                          |
| DATABASE-KEY            | X     | X      | 4      | 96     | binary                          |
| Counter                 |       | X      | 4      | 100    | binary                          |
| Empty item              |       |        | 7      | 104    | binary                          |
| DB identifier           | X     | X      | 1      | 111    | binary                          |
| DATABASE-KEY-LONG       | X     | X      | 8      | 112    | binary                          |
| End of user information | X     |        | 6      | 120    | character:<br>USINF*/<br>UINF1* |

Table 24: User information area

## Rules

### *Input*

- DATABASE-KEY** must be prefilled with a database key value of type DATABASE-KEY for the following statements:
- FIND1/FTCH1
  - ACCPTC with secondary option RLMDBK.
- In all other cases, the contents of the item are not significant as input.
- The database key value must be entered into the item as follows:
- Byte 1: record reference number (REC-REF)
  - Bytes 2-4: record sequence number (RSQ)
- Database key values with a REC-REF > 254 and/or an RSQ >  $2^{24}-1$  cannot be passed in this item.
- Empty items** must be deleted with binary zeros or blanks before each READYC call.
- DB identifier** must be deleted with binary zeros before each READYC call. Following the READYC call, the item will contain the correct DB identifier. This identifier must be supplied again with every CALL DML call (except for: FINISC) in order to identify its associated processing chain and thus the database.
- DATABASE-KEY-LONG** must be prefilled with a database key value of type DATABASE-KEY-LONG for the following statements:
- FIND1L/FTCH1L
  - ACCPTL with secondary option RLMDBK
- In all other cases, the contents of the item are not significant as input.
- The database key value must be entered into the item as follows:
- Bytes 1-2: record reference number (REC-REF)
  - Bytes 3-4: binary 0
  - Bytes 5-8: record sequence number (RSQ)

## End of user information

This must be occupied by the character string USINF\* if the (CALL8) variant is being used or UINF1\* if (CALL30) is being used. This identifier identifies the user-information parameter item to ensure that the database status can be returned to the application program.

If the CALL DML converter does not detect the keyword USINF\* or UINF1\*, it attempts to branch to the error exit DSCEXT, which must be defined as an entry address (ENTRY) in each CALL DML application program (see [page 124](#)). The DML call is not executed in this case.

The DSCEXT routine is not serviced in openUTM application programs.

*Output*

## System Communication Locations

The functions of the items correspond to those in COBOL DML.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Realm name   | Result item for ACCPTC or ACCPTL; with function options RLM... the CALL DML converter stores the found realm name in this item.                                                                                                                                                                                                                                                                                                                                                                    |
| Status code  | In addition to the status codes which are common to COBOL DML, CALL DML also recognizes other status codes (see <a href="#">page 378</a> ).<br><br>In a CALL DML status code the “System Communication Locations“ items contain no significant entries.                                                                                                                                                                                                                                            |
| DATABASE-KEY | The database key value is returned when ACCPTC is specified with function option DB-KEY and DBK...<br><br>The database key value is returned as follows: <ul style="list-style-type: none"> <li>– Byte 1: record reference number (REC-REF)</li> <li>– Bytes 2-4: record sequence number (RSQ)</li> </ul> <p>UDS/SQL cannot return database key values with a REC-REF &gt; 254 and/or an RSQ &gt; 2<sup>24</sup>-1 in this item and therefore reports the DATABASE-STATUS 15102 in such cases.</p> |
| Counter      | Following the execution of a FIND7A/FTCH7A with function option ...FST or ...SEX (search expression) and secondary option TAL (tallying function), the counter contains the total number of records in the retrieved collection which satisfy the search conditions.                                                                                                                                                                                                                               |

**DATABASE-KEY-LONG**

The database key value is returned when ACCPTL is specified with function option DB-KEY and DBK...

The database key value is returned as follows:

- Bytes 1-2: record reference number (REC-REF)
- Bytes 3-4: binary 0
- Bytes 5-8: record sequence number (RSQ)



## 8.2.5 Format of special parameter 1 (SPP1)

If you want to use RETAINING, you must specify the desired functions in special parameter 1. The following entries are possible (similar to COBOL DML):

```

MULTIPLE_ RETAINING CURRENCY FOR MULTIPLE
RLM RETAINING CURRENCY FOR REALM
REC RETAINING CURRENCY FOR RECORD
SET RETAINING CURRENCY FOR SETS
STNsetname RETAINING CURRENCY FOR set-name-1,...
STEsetname RETAINING CURRENCY FOR ALL SETS EXCEPT set-name-1,...

```

### Format

These symbols are transferred in a fixed format (at least 9 bytes):

| Bytes 1...9                                                                                                                                                | Bytes 10...235 for (CALL8) variant<br>Bytes 10...785 for (CALL30) variant |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <pre> M U L T I P L E _ R L M R E C S E T R L M R E C _ _ _ _ _ _ R E C S E T R L M _ _ _ S E T R L M _ _ _ _ _ _ _ _ R E C _ _ _ _ _ _ _ _ S E T   </pre> |                                                                           |
| <pre> R L M R E C S T N R L M _ _ _ S T N _ _ _ R E C S T N _ _ _ _ _ S T N   </pre>                                                                       | <pre> } {set-name } {set-name-1,... }   </pre>                            |
| <pre> R L M R E C S T E R L M _ _ _ S T E _ _ _ R E C S T E _ _ _ _ _ S T E   </pre>                                                                       | <pre> } {set-name } {set-name-1,... }   </pre>                            |

Table 25: Transfer format of special parameter 1

Set names can only be specified for the key STN or STE. A maximum of 25 set names may be specified. If STN or STE is not contained in bytes 7 to 9, bytes 10 to 235 or bytes 10 to 785, as appropriate, are not interpreted. The function STE is only available in CALL DML.

## 8.2.6 Format of special parameter 2 (SPP2)

If you want to use the STORE1/STOR1L or STORE2/STOR2L functions with the function option IMPDAT, you must specify the following in special parameter 2:

- database key value (if LOCATION MODE DIRECT or LOCATION MODE DIRECT LONG has been defined for the record type involved) and/or
- realm name (if more than one realm name has been specified in the WITHIN clause of the record type involved and if the record type is not the member record type of a distributable list)

If you want to use the FIND2/FTCH2 functions with the function option ANYIMP, you must specify the name of the realm in which the record is to be found in special parameter 2.

### Format

| Bytes 1...4                                                                                         | Bytes 5...34                                                                                                                                    | Bytes 35...42                                                                                            |
|-----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Database key value of type DATABASE-KEY (relevant for STORE1, STORE2);<br><br>if not used: binary 0 | Realm name in full length, padded with blanks (relevant for FIND2/FTCH2 as well as STORE1/STOR1L and STORE2/STOR2L);<br><br>if not used: blanks | Database key value of type DATABASE-KEY-LONG (relevant for STOR1L, STOR2L);<br><br>if not used: binary 0 |

Table 26: Transfer format of special parameter 2

## 8.3 CALL DML calls

### 8.3.1 Overview of the CALL DML functions

Information in response to invoked CALL DML functions can be found in the shaded columns.

| 1. FCOD | 2. FOPT                                                                      | 3. SOPT | 4. UINF                                                                                                 | 5. RECN                                                  | 6. SETN                                       | 7. RLMN                                  | 8. ITMN |
|---------|------------------------------------------------------------------------------|---------|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------|-----------------------------------------------|------------------------------------------|---------|
| ACCPTC  | DB-KEY<br>DBKREC<br>DBKRLM<br>DBKSET<br>RLMNAM<br>RLMREC<br>RLMSET<br>RLMDBK |         | DB key<br>DB key<br>DB key<br>DB key<br>realm name<br>realm name<br>realm name<br>DB key,<br>realm name | -<br>record name<br>-<br>-<br>-<br>record name<br>-<br>- | -<br>-<br>set name<br>-<br>-<br>set name<br>- | -<br>-<br>realm name<br>-<br>-<br>-<br>- |         |
| ACCPTL  | DB-KEY<br>DBKREC<br>DBKRLM<br>DBKSET<br>RLMNAM<br>RLMREC<br>RLMSET<br>RLMDBK |         | DB key<br>DB key<br>DB key<br>DB key<br>realm name<br>realm name<br>realm name<br>DB key,<br>realm name | -<br>record name<br>-<br>-<br>-<br>record name<br>-<br>- | -<br>-<br>set name<br>-<br>-<br>set name<br>- | -<br>-<br>realm name<br>-<br>-<br>-<br>- |         |
| CONNEC  | TO-ALL<br>TO-SET                                                             | } [RET] |                                                                                                         | [record name]<br>[record name]                           | -<br>set name...                              |                                          |         |
| DISCON  | FRMALL<br>FRMSET<br>ALLFRM                                                   |         |                                                                                                         | [record name]<br>[record name]<br>-                      | -<br>set name...<br>set name...               |                                          |         |
| ERASEC  | CORUNT<br>PERMAN<br><br>SELTIV<br>ALLMEM                                     |         |                                                                                                         | } record name                                            |                                               |                                          |         |
| FINISC  | ALLRLM<br>ALLCAN                                                             |         |                                                                                                         |                                                          |                                               |                                          |         |
| FREEC   | ALLREC<br>CORUNT                                                             |         |                                                                                                         |                                                          |                                               |                                          |         |

| 9. RECA | 10. SPP1                                                                                                                      | 11. SPP2 | 12. SPP3 | Function                                                                                                                                                                                                                                                                                                              |
|---------|-------------------------------------------------------------------------------------------------------------------------------|----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |                                                                                                                               |          |          | Transfers <ul style="list-style-type: none"> <li>– The database key value of the CRR, CRS, CRA or CRU to the user information area (see <a href="#">page 205</a>).</li> <li>– The name of the realm in which the CRR, CRS, CRU or the record belonging to the specified database key value is stored.</li> </ul>      |
|         |                                                                                                                               |          |          | Transfers <ul style="list-style-type: none"> <li>– The DATABASE-KEY-LONG value of the CRR, CRS, CRA or CRU to the user information area (see <a href="#">page 205</a>).</li> <li>– The name of the realm in which the CRR, CRS, CRU or the record belonging to the specified database key value is stored.</li> </ul> |
|         | <pre>           } for RET:           { SET           { STNset-name.. }           { STEset-name.. }           }         </pre> |          |          | Inserts the CRU in set occurrences                                                                                                                                                                                                                                                                                    |
|         |                                                                                                                               |          |          | <ul style="list-style-type: none"> <li>– Removes the CRU from set occurrences</li> <li>– Removes all records from dynamic sets</li> </ul>                                                                                                                                                                             |
|         |                                                                                                                               |          |          | Erases the CRU with its member records, if any, from the database.                                                                                                                                                                                                                                                    |
|         |                                                                                                                               |          |          | Terminates a transaction and releases locked realms and pages.                                                                                                                                                                                                                                                        |
|         |                                                                                                                               |          |          | Cancels the effect of the KEEP status                                                                                                                                                                                                                                                                                 |

| 1. FCOD | 2. FOPT                                                  | 3. SOPT          | 4. UINF | 5. RECN                                              | 6. SETN                                 | 7. RLMN                                     | 8. ITMN                                |
|---------|----------------------------------------------------------|------------------|---------|------------------------------------------------------|-----------------------------------------|---------------------------------------------|----------------------------------------|
| FIND1   | [DBKPRI]<br>[DBKNXT]                                     | [RET]<br>[NOW]   | DB key  | [record name]                                        |                                         |                                             |                                        |
| FIND1L  | [DBKPRI]<br>[DBKNXT]                                     | [RET]<br>[NOW]   | DB key  | [record name]                                        |                                         |                                             |                                        |
| FIND2   | ANYREC<br>ANYIMP<br>DUPLIC                               | } [RET]<br>[NOW] |         | } record name                                        |                                         |                                             |                                        |
| FIND3   | SETNAM<br>SETITM<br>RECNAM<br>RECITM                     | } [RET]<br>[NOW] |         | -<br>}<br>} record name                              | set name<br>set name<br>-<br>-          |                                             | -<br>item name...<br>-<br>item name... |
| FIND4   | SETNXT<br>SETPRI<br>SETFST<br>SETLST<br>SETSPC           | } [RET]<br>[NOW] |         | } [ { record name } ]<br>[ RECORD ]                  | } set name                              |                                             |                                        |
|         | RLMNXT<br>RLMPRI<br>RLMFST<br>RLMLST<br>RLMSPC           | } [RET]<br>[NOW] |         | } [ { record name } ]<br>[ RECORD ]                  |                                         | } realm name                                |                                        |
|         | RECNXT<br>RECPRI<br>RECFST<br>RECLST<br>RECSPC           | } [RET]<br>[NOW] |         | } record name                                        |                                         |                                             |                                        |
| FIND5   | CORUNT<br>RECNAM<br>RECSET<br>SETNAM<br>RECRLM<br>RLMNAM | } [RET]<br>[NOW] |         | - record name<br>- record name<br>- record name<br>- | -<br>- set name<br>- set name<br>-<br>- | -<br>-<br>-<br>- realm name<br>- realm name |                                        |
| FIND6   |                                                          | [RET]<br>[NOW]   |         |                                                      | set name                                |                                             |                                        |

| 9. RECA                             | 10. SPP1                                                                                                                     | 11. SPP2                          | 12. SPP3 | Function                                                                                                                                                                              |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     | for RET:<br><br>{<br>MULTIPLE<br>[RLM][REC]<br>{<br>SET<br>STNset-<br>name...<br>[<br>SEtset-<br>name...<br>]<br>}<br>}<br>} |                                   |          | Access via a database key value of type DATABASE-KEY                                                                                                                                  |
|                                     |                                                                                                                              |                                   |          | Access via a database key value of type DATABASE-KEY-LONG                                                                                                                             |
| item contents<br>item contents<br>- |                                                                                                                              | -<br>impl. def.<br>data area<br>- |          | Access via CALC key (hashing)                                                                                                                                                         |
|                                     |                                                                                                                              |                                   |          | Access to a record which corresponds to the CRR or CRS in certain item contents, or to a record which satisfies a previously processed search expression (FIND7A/FTCH7A).             |
|                                     |                                                                                                                              | -<br>-<br>-<br>pos. integer       |          | Access to the last or first record, to the record prior or next to the CRR, CRS or CRA to a record whose position corresponds to a specified numeric value                            |
|                                     |                                                                                                                              | -<br>-<br>-<br>pos. integer       |          | (integer) within the collection of records to be searched. The collection of records can be a record type, a set occurrence, a realm or an intersection of a record type and a realm. |
|                                     |                                                                                                                              | -<br>-<br>-<br>pos. integer       |          |                                                                                                                                                                                       |
|                                     |                                                                                                                              |                                   |          | Access to the CRR, CRS, CRA or CRU                                                                                                                                                    |
|                                     |                                                                                                                              |                                   |          | Access to the owner record of a CRS                                                                                                                                                   |

| 1. FCOD | 2. FOPT                                                                      | 3. SOPT                                                     | 4. UINF                           | 5. RECN     | 6. SETN                       | 7. RLMN                                          | 8. ITMN             |
|---------|------------------------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------|-------------|-------------------------------|--------------------------------------------------|---------------------|
| FIND7A  | RECFST<br>SELFST<br>CURFST                                                   | [RET]<br>[NOW]<br>[RES]<br>[LMS]<br>[TAL]<br>{SOA}<br>{SOD} | for TAL:<br><br>number of records | record name | -<br><br>set name<br>set name | item name...<br><br>item name...<br>item name... |                     |
|         | RECSEX<br>CURSEX                                                             | [RET]<br>[NOW]<br>[RES]<br>[LMS]<br>[TAL]<br>{SOA}<br>{SOD} |                                   | record name | -<br><br>set name<br>set name | item name...<br><br>item name...<br>item name... | } search expression |
|         | RECITM<br>RECITN<br>SELITM<br>SELITN<br>SELITP<br>CURITM<br>CURITN<br>CURITP | [RET]<br>[NOW]                                              |                                   | record name | -<br><br>set-name<br>set-name |                                                  | } item-name...      |



| 9. RECA                        | 10. SPP1                                                                       | 11. SPP2                                  | 12. SPP3                                   | Function                                                                                                                 |
|--------------------------------|--------------------------------------------------------------------------------|-------------------------------------------|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
|                                | <pre> MULTIPLE [RLM][REC] } { SET STNset- name... } [ STEset- name... ] </pre> | <pre> for RES: } result set name } </pre> | <pre> for LMS: } limited set name } </pre> | <p>Access to records via any items; counting and storing found records intermediately if required; search using mask</p> |
| <pre> } item contents } </pre> |                                                                                |                                           |                                            |                                                                                                                          |

| 1. FCOD | 2. FOPT                                                       | 3. SOPT          | 4. UINF | 5. RECN                                            | 6. SETN                               | 7. RLMN                              | 8. ITMN                               |
|---------|---------------------------------------------------------------|------------------|---------|----------------------------------------------------|---------------------------------------|--------------------------------------|---------------------------------------|
| FTCH1   | [DBKPRI]<br>[DBKNXT]                                          | [RET]<br>[NOW]   | DB key  | [record name]                                      |                                       |                                      |                                       |
| FTCH1L  | [DBKPRI]<br>[DBKNXT]                                          | [RET]<br>[NOW]   | DB key  | [record name]                                      |                                       |                                      |                                       |
| FTCH2   | ANYREC<br>ANYIMP<br>DUPLIC                                    | } [RET]<br>[NOW] |         | } record name                                      |                                       |                                      |                                       |
| FTCH3   | SETNAM<br>SETITM<br>RECNAM<br>RECITM                          | } [RET]<br>[NOW] |         | } record name                                      | set name<br>set name<br>-<br>-        |                                      | -<br>item name...<br><br>item name... |
| FTCH4   | SETNXT<br>SETPRI<br>SETFST<br>SETLST<br>SETSPC                | } [RET]<br>[NOW] |         | } [ { record name }<br>[ RECORD ] }                | } set name                            |                                      |                                       |
|         | RLMNXT<br>RLMPRI<br>RLMFST<br>RLMLST<br>RLMSPC                | } [RET]<br>[NOW] |         | } [ { record name }<br>[ RECORD ] }                |                                       | } realm name                         |                                       |
|         | REC NXT<br>REC PRI<br>REC FST<br>REC LST<br>REC SPC           | } [RET]<br>[NOW] |         | } record name                                      |                                       |                                      |                                       |
| FTCH5   | CORUNT<br>REC NAM<br>REC SET<br>SET NAM<br>REC RLM<br>RLM NAM | } [RET]<br>[NOW] |         | - record name<br>record name<br>- record name<br>- | -<br>- set name<br>set name<br>-<br>- | -<br>-<br>- realm name<br>realm name |                                       |
| FTCH6   |                                                               | [RET]<br>[NOW]   |         |                                                    | set name                              |                                      |                                       |

| 9. RECA                                                                 | 10. SPP1                                                                                                                                                                                                                                        | 11. SPP2                          | 12. SPP3                    | Function                                                  |                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|-----------------------------|-----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rec. cont's                                                             | $\left\{ \begin{array}{l} \text{MULTIPLE} \\ \text{[RLM][REC]} \end{array} \right\}$ $\left\{ \begin{array}{l} \text{SET} \\ \text{STNset-} \\ \text{name...} \\ \text{[} \\ \text{SEtset-} \\ \text{name...} \\ \text{]} \end{array} \right\}$ |                                   |                             | Access via a database key value of type DATABASE-KEY      |                                                                                                                                                                                                                             |
| rec. cont's                                                             |                                                                                                                                                                                                                                                 |                                   |                             | Access via a database key value of type DATABASE-KEY-LONG |                                                                                                                                                                                                                             |
| item cont's<br>rec. cont's<br>item cont's<br>rec. cont's<br>rec. cont's |                                                                                                                                                                                                                                                 | -<br>impl. def.<br>data area<br>- |                             |                                                           | Access via CALC key (hashing)                                                                                                                                                                                               |
| record contents                                                         |                                                                                                                                                                                                                                                 |                                   |                             |                                                           | Access to a record which corresponds to the CRR or CRS in certain item contents, or to a record which satisfies a previously processed search expression (FIND7A/FTCH7A)                                                    |
|                                                                         |                                                                                                                                                                                                                                                 |                                   | -<br>-<br>-<br>pos. integer |                                                           | Access to the last or first record, to the record next or prior to the CRR, CRS or CRA, or to the record whose position corresponds to a specified numeric value (integer) within the collection of records to be searched. |
|                                                                         |                                                                                                                                                                                                                                                 |                                   | -<br>-<br>-<br>pos. integer |                                                           | The collection of records can be a record type, a set occurrence, a realm or the intersection of a record type and a realm.                                                                                                 |
|                                                                         |                                                                                                                                                                                                                                                 |                                   | -<br>-<br>-<br>pos. integer |                                                           |                                                                                                                                                                                                                             |
|                                                                         |                                                                                                                                                                                                                                                 |                                   |                             | Access to the CRR, CRS, CRA or CRU                        |                                                                                                                                                                                                                             |
|                                                                         |                                                                                                                                                                                                                                                 |                                   |                             | Access to the owner record of a CRS                       |                                                                                                                                                                                                                             |

| 1. FCOD | 2. FOPT                                                                      | 3. SOPT                                                     | 4. UINF                               | 5. RECN                       | 6. SETN                           | 7. RLMN                                              | 8. ITMN                       |
|---------|------------------------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------|-------------------------------|-----------------------------------|------------------------------------------------------|-------------------------------|
| FTCH7A  | RECFST<br><br>SELFST<br><br>CURFST                                           | [RET]<br>[NOW]<br>[RES]<br>[LMS]<br>[TAL]<br>{SOA}<br>{SOD} | for TAL:<br><br><br>number of records | }<br><br>record name<br><br>} | -<br><br>set name<br><br>set name | item name...<br><br>item name...<br><br>item name... |                               |
|         | RECSEX<br><br>CURSEX                                                         | [RET]<br>[NOW]<br>[RES]<br>[LMS]<br>[TAL]<br>{SOA}<br>{SOD} |                                       | }<br><br>record name<br><br>} | -<br><br>set name<br><br>set name | item name...<br><br>item name...<br><br>item name... | }<br><br>search<br>expression |
|         | RECITM<br>RECITN<br>SELITM<br>SELITN<br>SELITP<br>CURITM<br>CURITN<br>CURITP | [RET]<br>[NOW]                                              |                                       | }<br><br>record name<br><br>} | -<br><br>set name<br><br>set name |                                                      | }<br><br>item<br>name-1...    |

| 9. RECA                                              | 10. SPP1                                                                                                                    | 11. SPP2                                             | 12. SPP3                                              | Function                                                                                                                          |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <p>}<br/>record<br/>contents<br/>}</p>               | <p>{<br/>MULTIPLE<br/>[RLM][REC]<br/>}<br/>{<br/>SET<br/>STNset-<br/>name...<br/>[<br/>STEset-<br/>name...<br/>]}<br/>}</p> | <p>} for RES:<br/><br/>result<br/>set name<br/>}</p> | <p>} for LMS:<br/><br/>limited<br/>set name<br/>}</p> | <p>Access to records via any items;<br/>counting and storing records found inter-<br/>mediately if required; search with mask</p> |
| <p>}<br/>contents/<br/>record<br/>contents<br/>}</p> | <p>}</p>                                                                                                                    | <p>}</p>                                             | <p>}</p>                                              | <p>}</p>                                                                                                                          |

| 1. FCOD | 2. FOPT                                                                                                               | 3. SOPT          | 4. UINF | 5. RECN                      | 6. SETN                                                     | 7. RLMN                                     | 8. ITMN             |
|---------|-----------------------------------------------------------------------------------------------------------------------|------------------|---------|------------------------------|-------------------------------------------------------------|---------------------------------------------|---------------------|
| GETC    | CORUNT<br>ITMNAM                                                                                                      | -<br>[VAR]       |         | [record name]<br>record name |                                                             |                                             | -<br>item name-1... |
| IFC     | OWNALL<br>OWNSET<br>MEMALL<br>MEMSET<br>TENALL<br>TENSET<br>EMPTYS                                                    |                  |         |                              | -<br>set name<br>-<br>set name<br>-<br>set name<br>set name |                                             |                     |
| KEEPC   |                                                                                                                       |                  |         |                              |                                                             |                                             |                     |
| MODIF1  | CORUNT<br>INCALL<br>ONLALL<br><br>INCSET<br>ONLSET                                                                    | } [RET]          |         | } record name                | -<br>-<br>-<br>set name...<br>set name...                   |                                             |                     |
| MODIF2  | CORUNT<br>INCALL<br>INCSET                                                                                            | } [RET]<br>[VAR] |         | } record name                | -<br>set name                                               |                                             | } item<br>name...   |
| READYC  | ALLRTR<br>ALLLUPD<br>ALLPRT<br>ALLPUP<br>ALLERT<br>ALLEUP<br>RLMRTR<br>RLMUPD<br>RLMPRT<br>RLMPUP<br>RLMERT<br>RLMEUP | } [NOW]          |         |                              |                                                             | -<br>-<br>-<br>-<br>-<br>} realm<br>name... |                     |

| 9. RECA                              | 10. SPP1         | 11. SPP2              | 12. SPP3 | Function                                                                                                               |
|--------------------------------------|------------------|-----------------------|----------|------------------------------------------------------------------------------------------------------------------------|
| rec. cont's<br>item cont's           |                  |                       |          | Makes the CRU or individual items of the CRU available                                                                 |
|                                      |                  |                       |          | Checks set memberships                                                                                                 |
|                                      |                  |                       |          | Protects the CRU from access by other transactions until a FREE statement is entered or until the end of a transaction |
| rec. cont's<br>_<br>rec. cont's<br>_ | } for RET:<br>}  |                       |          | Modifies record contents or item contents of the CRU and/or transfers them to another set occurrence within the set    |
| } item contents                      |                  |                       |          |                                                                                                                        |
|                                      | } subschema name | } privacy information |          | Opens a transaction or a processing chain                                                                              |

| 1. FCOD | 2. FOPT          | 3. SOPT          | 4. UINF | 5. RECN    | 6. SETN | 7. RLMN | 8. ITMN           | 9. RECA              |
|---------|------------------|------------------|---------|------------|---------|---------|-------------------|----------------------|
| STORE1  | RECNAM<br>IMPDAT | } [RET]          |         | } rec-name |         |         |                   | } record<br>contents |
| STOR1L  | RECNAM<br>IMPDAT | } [RET]          |         | } rec-name |         |         |                   | } record<br>contents |
| STORE2  | ITMNAM<br>IMPDAT | } [RET]<br>[VAR] |         | } rec-name |         |         | } item<br>name... | } item<br>contents   |
| STOR2L  | ITMNAM<br>IMPDAT | } [RET]<br>[VAR] |         | } rec-name |         |         | } item<br>name... | } item<br>contents   |



| 10. SPP1                                                                                                                              | 11. SPP2                              | 12. SPP3 | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> } for RET:   {     MULTIPLE     [RLM][REC]   }   {     SET     STNset-     name...   }   [     STEset-     name...   ] } </pre> | <pre> - impl. def. data area </pre>   |          | <ul style="list-style-type: none"> <li>- Transfers a record or individual items or compressed records from the UWA to the database as a new record</li> <li>- Inserts the new record into all sets for which its record type has been defined as an AUTOMATIC member in the schema</li> <li>- Sets up a new set occurrence for each set for which the record type is defined as owner record type in the schema</li> </ul> <p>If you want to specify a database with a REC-REF &gt; 254 and/or an RSQ &gt; 2<sup>24</sup>-1 with an IMPDAT function option, you must use STOR1L or STOR2L</p> |
|                                                                                                                                       | <pre> - impl. def. data area - </pre> |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

### 8.3.2 Functions of CALL DML

The function of a call is determined by the function code and function option parameters.

The rules for execution of a CALL DML call correspond to those applicable in COBOL DML (see [section “COBOL DML statements” on page 139](#)). They will therefore not be explained again here.

So that the relevant COBOL DML rules can be easily referenced, the keywords of parameter “function-option” are listed alongside the corresponding COBOL DML statements. Where there are variations from COBOL DML, an exact description will be given in each case.

Normally the selected function option governs whether and how a parameter is used. The descriptions of the parameters therefore give the function option keywords to which each description is relevant.

#### *Example*

```
Function code: ACCPTC
 .
 .
 .
Set name: DBKSET/RLMSET Input of a set name
```

The set name parameter is used only for the two function options specified, not for others.

If no function option keyword is given, the description applies to all possible entries.

A detailed description of the formats of CALL DML calls is given in the following text.

## Saving currency information (ACCPTC, ACCPTL)

The ACCPTC and ACCPTL functions determine

- the contents of the specified currency information (database key value),
- the realm name belonging to a specific database key value.

ACCPTC and ACCPTL differ with respect to the return or input of the database key value.

ACCPTL can only be executed in combination with an SSITAB module of UDS/SQL Version 2.0 or higher (see the “[Creation and Restructuring](#)“ manual). Otherwise, UDS/SQL reports status code C98.

Function code:       **ACCPTC**

Function code:       **ACCPTL**

Function option:

| CALL DML | Corresponding COBOL DML statement                          |
|----------|------------------------------------------------------------|
| DB-KEY   | ACCEPT <i>item-name</i> FROM CURRENCY                      |
| DBKREC   | ACCEPT <i>item-name</i> FROM <i>record-name</i> CURRENCY   |
| DBKRLM   | ACCEPT <i>item-name</i> FROM <i>realm-name</i> CURRENCY    |
| DBKSET   | ACCEPT <i>item-name</i> FROM <i>set-name</i> CURRENCY      |
| RLMNAM   | ACCEPT <i>item-name</i> FROM REALM-NAME                    |
| RLMREC   | ACCEPT <i>item-name</i> FROM <i>record-name</i> REALM-NAME |
| RLMSET   | ACCEPT <i>item-name</i> FROM <i>set-name</i> REALM-NAME    |
| RLMDBK   | ACCEPT <i>item-name</i> FROM <i>item-name</i> REALM-NAME   |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User information: | DB-KEY/DBK...<br>Database key value returned.<br>For ACCPTC, UDS/SQL can only return database key values with a REC-REF $\leq 254$ and an RSQ $\leq 2^{24}-1$ . See <a href="#">page 205-207</a> for details.<br><br>RLM...<br>Realm name returned in the realm name item of the System Communication Locations<br><br>RLMDBK<br>Input of a database key.<br>For ACCPTC, you can only specify database key values with a REC-REF $\leq 254$ and an RSQ $\leq 2^{24}-1$ . See <a href="#">page 205-207</a> for details.<br><br>DATABASE-STATUS returned |
| Record name:      | DBKREC/RLMREC<br>Input of a record name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Set name:         | DBKSET/RLMSET<br>Input of a set name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Realm name:       | DBKRLM<br>Input of a realm name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Establishing set connections (CONNEC)

The CONNec function ensures that the Current Record of Run-unit (CRU) becomes a member in one or more sets in which its record type is an optional member (i.e. MANDATORY MANUAL, OPTIONAL AUTOMATIC or OPTIONAL MANUAL).

The RETAINING option may be used to prevent the record from becoming the Current Record of Set (CRS) for all sets or for the specified sets into which it is inserted.

Function code:           **CONNec**

Function option:

| CALL DML | Corresponding COBOL DML statement                       |
|----------|---------------------------------------------------------|
| TO-ALL   | CONNECT[ <i>record-name</i> ] TO ALL                    |
| TO-SET   | CONNECT[ <i>record-name</i> ] TO <i>set-name-1</i> ,... |

Secondary option:       Input of RET (optional)

User information:       DATABASE-STATUS returned

Record name:           Input of record name (optional)

Set name:               TO-SET  
Input of one or more set names

Special parameter 1:   Input of retaining parameters associated with secondary option RET; in the case of CONNec, the retaining parameter is possible only for sets.

## Releasing existing set connections (DISCON)

The DISCON function terminates the membership of the Current Record of Run-unit (CRU) in one or more sets in which it is currently a member, provided that its record type is declared as OPTIONAL member. The membership of all records of the current set occurrence (dynamic sets only) can be terminated with ALLFRM.

Function code:           **DISCON**

Function option:

| CALL DML | Corresponding COBOL DML statement                            |
|----------|--------------------------------------------------------------|
| FRMALL   | DISCONNECT[ <i>record-name</i> ] FROM ALL                    |
| FRMSET   | DISCONNECT[ <i>record-name</i> ] FROM <i>set-name-1</i> ,... |
| ALLFRM   | DISCONNECT ALL FROM <i>set-name-2</i> ,...                   |

User information:       DATABASE-STATUS returned

Record name:           FRMALL/FRMSET  
Input of a record name (optional)

Set name:               FRMSET/ALLFRM  
Input of one or more set names

## Deleting records and their set connections (ERASEC)

The ERASEC function erases the Current Record of Run-unit (CRU) in the database and removes it from all set occurrences in which it was a member. The ERASEC function also:

- erases all MANDATORY member records of the CRU or
- removes all OPTIONAL member records from set occurrences in which the deleted record was the owner, erasing them if required.

Function code:           **ERASEC**

Function option:

| <b>CALL DML</b> | <b>Corresponding COBOL DML statement</b>   |
|-----------------|--------------------------------------------|
| CORUNT          | ERASE <i>record-name</i>                   |
| PERMAN          | ERASE <i>record-name</i> PERMANENT MEMBERS |
| SELTIV          | ERASE <i>record-name</i> SELECTIVE MEMBERS |
| ALLMEM          | ERASE <i>record-name</i> ALL MEMBERS       |

User information:       DATABASE-STATUS returned

Record name:            Input of a record name

## Retrieval of data (FIND/FTCH)

The FIND/FTCH functions make a record found by a record selection expression into the

- Current Record of Run-unit (CRU),
- Current Record of Area (CRA) of the realm in which it is stored,
- Current Record of Record (CRR) of its record type and
- Current Record of Set (CRS) for all sets in which it is the owner or a member.

The FTCH functions additionally transfer the contents of the found record as defined in the subschema into the record area.

You can optionally ensure that no changes are made to the currency table information affected by the FIND/FTCH function by specifying RET in the “secondary option” parameter.

You may also specify NOW in the “secondary option” parameter to ensure that if a page locked by a transaction is accessed, control is immediately returned to the application program (DATABASE-STATUS 04020) instead of waiting for the page to be freed.

The FIND/FETCH functions can be accessed via 7 different function codes. Each function code corresponds to one of the 7 formats of COBOL DML.

Input to parameters 2 to 12 is identical for FIND and FTCH.



The FIND1/FTCH1 or FIND1L/FTCH1L functions gain access to a record via the database key value. FIND1/FTCH1 and FIND1L/FTCH1L differ with respect to the input of the database key value.

FIND1L and FTCH1L can only be executed in combination with an SSITAB module of UDS/SQL Version 2.0 or higher (see the “[Creation and Restructuring](#)” manual). Otherwise, UDS/SQL reports status code C98.

Function code: **FIND1/FTCH1**

Function code: **FIND1L/FTCH1L**

Function option:

| CALL DML | Corresponding COBOL DML statement                                    |
|----------|----------------------------------------------------------------------|
|          | FIND <i>record-name</i> DATABASE-KEY IS <i>item-name</i>             |
| DBKPRI   | FIND <i>record-name</i> DATABASE-KEY IS <i>item-name</i><br>OR PRIOR |
| DBKNXT   | FIND <i>record-name</i> DATABASE-KEY IS <i>item-name</i> OR<br>NEXT  |

Function option: Is omitted; the corresponding COBOL format is as follows:

FIND[ *record-name*] DATABASE-KEY IS *item-name*

Secondary option: Input of RET (optional), NOW (optional)

User information: Input of the database key value of the record to be found.  
For Find1/FTCH1, you can only specify database key values with a REC-REF  $\leq 254$  and an RSQ  $\leq 2^{24}-1$ . See [page 205-207](#) for details.

DATABASE-STATUS returned

Record name: Input of a record name (optional)

Record area: DBKPRI/DBKNXT  
When DBKPRI or DBKNXT is specified, the record with the next lowest or next highest database key is supplied if no record with the specified database key exists. If anything else is specified, the OR PRIOR/NEXT functionality is not employed.

FTCH1: Record searched for returned;

Special parameter 1: Input of retaining parameters for secondary option RET

The FIND2/FTCH2 function provides direct access via the CALC key.

ANYIMP must be used whenever the record type is distributed over several realms.

Function code: **FIND2/FTCH2**

| Function option: | CALL DML | Corresponding COBOL DML statement |
|------------------|----------|-----------------------------------|
|                  | ANYREC   | FIND ANY <i>record-name</i>       |
|                  | ANYIMP   | FIND ANY <i>record-name</i>       |
|                  | DUPLIC   | FIND DUPLICATE <i>record-name</i> |

Secondary option: Input of RET (optional), NOW (optional)

User information: DATABASE-STATUS returned

Record name: Input of a record name; the record type must be defined with LOCATION MODE CALC.

Record area: ANYREC/ANYIMP  
 Input of the CALC key of the LOCATION MODE clause at that point in the record area which corresponds to the subschema format of the record.

DUPLIC

Input of CALC key required; DBH searches for a record in the realm of the CRR which has the same CALC key value as the CRR.

FTCH2

Record returned

Special parameter 1: Input of retaining parameters for secondary option RET

Special parameter 2: ANYIMP

Input of the realm name of the realm to be searched for the record:

Bytes 1-4: not used

Bytes 5-34: full length of realm name, padded with blanks if required.

The contents of bytes 5-34 are only evaluated if the record type *record-name* is distributed over several realms (due to a corresponding WITHIN clause of the schema DDL definition for the record type *record-name*).

The FIND3/FTCH3 function accesses a record which matches the CRR or CRS in specific item contents or which satisfies the conditions of a previously processed search expression.

Function code: **FIND3/FTCH3**

Function option:

| CALL DML | Corresponding COBOL DML statement                                         |
|----------|---------------------------------------------------------------------------|
| SETNAM   | FIND DUPLICATE WITHIN <i>set-name</i>                                     |
| SETITM   | FIND DUPLICATE WITHIN <i>set-name</i><br>USING <i>item-name-1,...</i>     |
| RECNAM   | FIND DUPLICATE WITHIN <i>record-name</i>                                  |
| RECITM   | FIND DUPLICATE WITHIN <i>record-name</i><br>USING <i>item-name-1,....</i> |

Secondary option: Input of RET (optional), NOW (optional)

User information: DATABASE-STATUS returned

Record name: SETITM/RECNAM/RECITM  
Input of a record name; with SETITM this is necessary to identify the items.

Set name: SET...  
Input of a set name

Item name: ...ITM  
Input of one or more item names

Record area: FTCH3  
Record returned

Special parameter 1: Input of retaining parameters for secondary option RET

The FIND4/FTCH4 function accesses the first or last record, the next or prior record to the CRR, CRS or CRA, or a record whose position within the collection of records in which the search is made corresponds to a numeric value to be specified.

Function code: **FIND4/FTCH4**

Function option:

| CALL DML                             | Corresponding COBOL DML statement                                                                                        |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| SET...                               | FIND ..... $\left. \begin{array}{l} \textit{record-name} \\ \text{RECORD} \end{array} \right\}$ WITHIN <i>set-name</i>   |
| RLM...                               | FIND ..... $\left. \begin{array}{l} \textit{record-name} \\ \text{RECORD} \end{array} \right\}$ WITHIN <i>realm-name</i> |
| REC...                               | FIND ..... record-name                                                                                                   |
| ...NXT<br>...PRI<br>...FST<br>...LST | FIND NEXT .....<br>FIND PRIOR .....<br>FIND FIRST .....<br>FIND LAST .....                                               |
| ...SPC                               | FIND $\left. \begin{array}{l} \textit{integer} \\ \textit{item-name} \end{array} \right\}$                               |

All 15 possible combinations are allowed and correspond to the full range of combinations in COBOL DML.

Secondary option: Input of RET (optional), NOW (optional)

User information: DATABASE-STATUS returned

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Record name:         | Input of a record name;<br>SET.../RLM...<br>The input of a record name is optional; input of blank characters has the same effect as input of RECORD_...<br>SET...<br>If a record name is input, the DBH also checks whether the record type specified is the member record type in the named set.<br>RLM...<br>The input of a record name restricts the collection of records in which the search is made to this record type; otherwise the DBH takes into consideration all record types of the subschema which are stored in the realm. |
| Set name:            | SET...<br>Input of a set name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Realm name:          | RLM...<br>Input of a realm name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Record area:         | FTCH4<br>Record returned                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Special parameter 1: | Input of retaining parameters for secondary option RET                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Special parameter 2: | ...SPC<br>Input of an integer. The value must be coded as a 4-byte binary number; negative values are allowed, but not 0.                                                                                                                                                                                                                                                                                                                                                                                                                   |

The FIND5/FTCH5 function accesses the CRR, CRS, CRA or CRU.

Function code: **FIND5/FTCH5**

Function option:

| CALL DML | Corresponding COBOL DML statement                        |
|----------|----------------------------------------------------------|
| CORUNT   | FIND CURRENT                                             |
| RECNAM   | FIND CURRENT <i>record-name</i>                          |
| RECSET   | FIND CURRENT <i>record-name</i> WITHIN <i>set-name</i>   |
| SETNAM   | FIND CURRENT WITHIN <i>set-name</i>                      |
| RECRLM   | FIND CURRENT <i>record-name</i> WITHIN <i>realm-name</i> |
| RLMNAM   | FIND CURRENT WITHIN <i>realm-name</i>                    |

Secondary option: Input of RET (optional), NOW (optional)

User information: DATABASE-STATUS returned

Record name: REC...  
Input of a record name; the collection of records in which the search is made is limited to the specified record type, i.e. the DBH checks whether the CRS or CRA is of the specified record type.

Set name: RECSET/SETNAM  
Input of a set name

Realm name: RECRLM/RLMNAM  
Input of a realm name

Record area: FTCH5  
Record returned

Special parameter 1: Input of retaining parameters for secondary option RET (only for sets concerned)

The FIND6/FTCH6 function accesses the owner record of a CRS.

|                      |                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------|
| Function code:       | <b>FIND6/FTCH6</b>                                                                                   |
| Function option:     | Is omitted; the corresponding COBOL DML format is:<br><code>FIND OWNER WITHIN <i>set-name</i></code> |
| Secondary option:    | Input of RET (optional), NOW (optional)                                                              |
| User information:    | DATABASE-STATUS returned                                                                             |
| Set name:            | Input of a set name                                                                                  |
| Record area:         | FTCH6<br>Owner record returned                                                                       |
| Special parameter 1: | Input of retaining parameters for secondary option RET (only for relevant sets).                     |

The FIND7A/FTCH7A accesses records via user-defined items; if required, selected records are counted and buffered and searched for using a mask.

Function code: **FIND7A/FTCH7A**

Function option:

| <b>CALL DML</b> | <b>Corresponding COBOL DML statement</b>                                                            |
|-----------------|-----------------------------------------------------------------------------------------------------|
| RECFST          | FIND <i>record-name</i>                                                                             |
| CURFST          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT                                              |
| SELFST          | FIND <i>record-name</i> WITHIN <i>set-name</i>                                                      |
| RECSEX          | FIND <i>record-name</i> USING <i>search-expression</i>                                              |
| CURSEX          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT<br>USING <i>search-expression</i>            |
| SELSEX          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT<br>USING <i>search-expression</i>            |
| RECITM          | FIND <i>record-name</i> <i>item-name-1</i>                                                          |
| CURITM          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT<br>USING <i>item-name-1</i> ,...             |
| SELITM          | FIND <i>record-name</i> WITHIN <i>set-name</i><br>USING <i>item-name-1</i> ,...                     |
| RECITN          | FIND <i>record-name</i> USING <i>item-name-1</i> ,...<br>OR NEXT                                    |
| CURITN          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT<br>USING <i>item-name-1</i> ,...<br>OR NEXT  |
| SELITN          | FIND <i>record-name</i> WITHIN <i>set-name</i><br>USING <i>item-name-1</i> ,...<br>OR NEXT          |
| CURITP          | FIND <i>record-name</i> WITHIN <i>set-name</i> CURRENT<br>USING <i>item-name-1</i> ,...<br>OR PRIOR |
| SELITP          | FIND <i>record-name</i> WITHIN <i>set-name</i><br>USING <i>item-name-1</i> ,...<br>OR PRIOR         |



---

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Secondary option: | Optional input of<br>RET<br>for retaining functions<br>RES<br>if the user wishes to specify a dynamic result set<br>(special parameter 2)<br>LMS<br>if the user wishes to specify limited by dynamic set<br>(special parameter 3)<br>TAL<br>if the user wishes to use the tallying function<br>SOA<br>if the user wishes to use the sort function in ascending order<br>SOD<br>if the user wishes to use the sort function in descending order<br>NOW<br>if you don't want to wait when there is a lock<br>The entries RES, LMS, TAL, SOA and SOD are allowed only with<br>...FST and ...SEX.<br>Input of RES or TAL leads to the whole collection of selected<br>records being determined. For the input format, see <a href="#">page 203</a> . |
| User information: | TAL<br>Result of the TALLYING function returned in the counter item as a<br>4-byte binary number<br>DATABASE-STATUS returned                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Record name:      | Input of a record name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

- Set name: CUR.../SEL...  
Input of a set name
- With CUR... the current set occurrence of the specified set is defined as the collection of records in which the search is made.
  - With SEL... the set occurrence in which the search is to be made is defined with the aid of SET OCCURRENCE SELECTION. Sets with the SET OCCURRENCE SELECTION clause THRU LOCATION MODE OF OWNER are handled in the same way as sets with the SET OCCURRENCE SELECTION clause THRU CURRENT OF SET. The effect of SEL... therefore corresponds to that of CUR....
  - CUR... can be used for SYSTEM sets even when a CRS does not exist.
- Realm name: ...SOA/...SOD  
Input of one or more item names; the records satisfying a search expression are sorted by these items. The items must be elements of the record type addressed in the search expression. The records are sorted according to type.
- ...SOA causes sorting in ascending order.
  - ...SOD causes sorting in descending order.
- All items specified must be sorted in the same direction.
- Item name: ...ITM/...ITN/...ITP  
Input of one or more item names; a search is made for a record containing the same values given in the record area.
- If ...ITN is specified and no record with the values specified in the record area are found, the next record of the set occurrence is selected in accordance with the sort sequence. This sort sequence is determined by the specified combination of item names, which in turn must designate the ASCENDING KEY, DESCENDING KEY or a SEARCH KEY USING INDEX of the specified set. Any violation of these conditions will be acknowledged by UDS/SQL with a DATABASE-STATUS of 04101 at runtime.

If there are duplicate records for a key value, UDS/SQL outputs the record with the smallest record sequence number (RSQ). In other words, you can find all duplicates

- with a FIND7A/FTCH7A ...ITN call and a sequence of FIND4/FTCH4 ... NXT calls or
- with a FIND7A/FTCH7A ...ITN call and a sequence of FIND/FTCH3 calls.

If the key value specified in the record area is greater than the largest key value present in the set occurrence, UDS/SQL reports DATABASE-STATUS 04024.

- If ...ITP is specified and no record with the values specified in the record area are found, the immediately preceding record of the set occurrence is selected in accordance with the sort sequence. This sort sequence is determined by the specified combination of item names, which in turn must designate the ASCENDING KEY or DESCENDING KEY of the specified set. Any violation of these conditions will be acknowledged by UDS/SQL with a DATABASE-STATUS of 04101 at runtime. If there are duplicate records for a key value, UDS/SQL outputs the record with the highest record sequence number (RSQ), which means that you can find all duplicates a FIND7A/FTCH7A ...ITP call and a sequence of FIND4/FTCH4 ... PRI calls. If the key value specified in the record area is less than the smallest key value present in the set occurrence, UDS/SQL reports DATABASE-STATUS 04024.

#### ...SEX

Input of a search expression; a search is made for one or more records which satisfy the condition(s) of the search expression. The search expression is described after special-parameter-3 below.

Record area:

Input of the item contents of the specified items for function option ...ITM/...ITN/...ITP; their position in the record area must correspond to that in the subschema format of the record.

#### FTCH7A

Return of (first) record found

- Special parameter 1: Input of the retaining parameters for secondary option RET
- Special parameter 2: RES  
Input of a name for a result set.  
If you specify RES, UDS/SQL will save the selected records in an explicit dynamic set, which you can then address with additional FIND4/FTCH4 calls. This allows you to program the search for records so that a total of several hierarchy levels of the data structure are checked, i.e. to use items of different record types via set relationships for the search (complex search query).  
If the specified set is an unsorted dynamic set, you cannot specify ...SOA or ...SOD in the "realm name" parameter (see above), since the intersection of a selected set (hit list) and a dynamic set cannot be sorted.  
RES cannot be used if ...ITM was specified in the "item name" parameter.
- Special parameter 3: LMS  
Input of the name for the set to be intersected with the selected set (hit list).  
You may specify LMS only if the set specified in the "set name" parameter is not a dynamic set.  
LMS limits the collection of records to records which
- belong to the record type specified in the "record name" parameter or to the selected set occurrence of the set given in the "set name" parameter
  - and which are also contained in the set specified in "special parameter 3".
- The set specified in "special parameter 3" must not be a sorted dynamic set, since it is not possible to create the intersection of a selected set (hit list) and a sorted dynamic set.  
LMS cannot be used if ...ITM was specified in the "item name" parameter.

$$\begin{aligned} \text{Search expression} &::= \left. \begin{array}{l} \text{complex-1[ AN complex-2]} \\ \text{complex-2} \end{array} \right\} \text{END} \\ \text{complex-1} &::= \text{condition-1} \left. \begin{array}{l} \text{[AN]} \\ \text{[OR]} \end{array} \right\} \text{condition-1} \dots \\ \text{complex-2} &::= \text{condition-2[ AN condition-2]} \dots \\ \text{condition-1} &::= n \text{ item-name [MSK mask ]rel value } m \\ \text{condition-2} &::= 0 \text{ item-name} \left. \begin{array}{l} \text{[NXT rel value]} \\ \text{[MAX]} \\ \text{[MIN]} \end{array} \right\} 0 \end{aligned}$$

A search expression may not contain more than 31 conditions in all.

The rules for the logical processing of the search expression and the evaluation of the multi-level tables of ASC keys and SEARCH keys are the same as in COBOL DML.

*n*

Number of left (opening) parentheses  $n = 0 \dots 9$

The sum of opening parentheses and the sum of closing parentheses in *complex-1* must be the same.

No parentheses are allowed in *complex-2*; <0> must be specified on grounds of format compatibility.

*m*

Number of right (closing) parentheses  $m = 0 \dots 9$

*rel* (Relation):

EQU: equal

GTH: greater than

LTH: lower than

GEQ: greater than or equal

LEQ: lower than or equal

NEQ: not equal

NGR: not greater than

NLW: not lower than

EQU and NEQ are forbidden in a *condition-2*. The boolean operator NOT is not provided, but the same effect can be achieved by specifying opposing relational operators.

*item-name*

Item-name with a length of 8 bytes or 30 bytes corresponding to the (CALL8) and (CALL30) variants.

*mask*

Literal of the same length as item *item-name*

significant bytes: X'F1'

non-significant bytes: X'F0'

*value*

Literal of the same length and same type as item *item-name*; it contains the comparison value.

To avoid alignment problems with the comparison value (depending on the conventions of the programming language used), the literal *value* may begin with up to 7 characters which do not belong to the comparison value. The last (right-justified) bytes of *value*, equal in length to item *item-name*, are then taken for the comparison.

Comparisons can only be made with literals inserted in the search expression, not with the contents of other items.

## MAX/MIN

Are used to find the greatest and smallest value (not present in COBOL DML).

## END

Terminates the search expression.

*Example for (CALL8) variant*

A search is to be made for the farmer with the biggest potatoes; members of parliament with an interest in agriculture are included with farmers in the selection of records in which the search is made.

## Subschema record:

```
01 CITIZEN.
 02 PROFESSION PIC X(12).
 02 SECTOR PIC X(15).
 02 POTATO PIC S999.
```

## COBOL DML search expression:

```
(PROFESSION="FARMER" OR (PROFESSION="MP" AND SECTOR="AGRICULTURE"))
AND POTATO
NEXT NOT GREATER HIGH VALUE
```

## CALL DML search expression:

```
1_PROFESSI EQU FARMER.....0 OR
1_PROFESSI EQU MP.....0 AND
0_SECTOR EQU AGRICULTURE.....2 AND
0_POTATO EQU MAX 0 END
```

The conditions are normally specified contiguously; they have been entered on separate lines here simply for the sake of clarity.

## Concluding processing (FINISC)

The FINISC function terminates a transaction. Updates made by the transaction are confirmed (fixed) or may be optionally cancelled. Locked realms and pages are released.

Function code: **FINISC**

Function option:

| CALL DML | Corresponding COBOL DML statement |
|----------|-----------------------------------|
| ALLRLM   | FINISH                            |
| ALLCAN   | FINISH WITH CANCEL                |

User information: DATABASE-STATUS returned

## Deactivating extended record protection (FREEC)

The FREEC function terminates the KEEP status for one or more records.

Function code: **FREEC**

Function option:

| CALL DML | Corresponding COBOL DML statement |
|----------|-----------------------------------|
| CORUNT   | FREE                              |
| ALLREC   | FREE ALL                          |

User information: DATABASE-STATUS returned

## Transporting a record to the record area (GETC)

The GETC function places the Current Record of Run-unit (CRU) or individual CRU items in the record area of the UWA.

Function code:           **GETC**

| Function option: | CALL DML | Corresponding COBOL DML statement |
|------------------|----------|-----------------------------------|
|                  | CORUNT   | GET[ <i>record-name</i> ]         |
|                  | ITMNAM   | GET <i>item-name-1</i> ,...       |

ITMNAM is not allowed for records with variable-length items.

Secondary option:    ITMNAM  
 If VAR (optional) is specified, the specified items are stored in the record area in the specified order, irrespective of the format of the subschema record and without spaces.

User information:    DATABASE-STATUS returned

Record name:        Input of a record name  
 CORUNT  
 The input of a record name is optional; however, the DBH always transfers the complete CRU into the record area.

If a record name is specified, the DBH first checks whether the Current Record of Run-unit (CRU) is of this record type before transferring it to the record area.

ITMNAM  
 The DBH only transfers the specified items to the record area; a record name must be specified.

Item name:           ITMNAM  
 Input of one or more item names



Record area:

CORUNT

Current Record of Run-unit (CRU) returned

ITMNAM

Individual items of the CRU returned

If the VAR secondary option is not specified, these items are transferred to positions in the record area which correspond to their location in the subschema format of the record.

If VAR is specified, the contents of the specified items are transferred contiguously (abbreviated record format).

## Testing database conditions (IFC)

The IFC function can be used to interrogate the database, i.e.:

- to check whether the Current Record of Run-unit (CRU) is owner or member in one or more specified sets or
- to check whether the current set occurrence contains member records.

Function code:       **IFC**

Function option:

| CALL DML | Corresponding COBOL DML statement |
|----------|-----------------------------------|
| OWNALL   | IF OWNER                          |
| OWNSET   | IF <i>set-name</i> OWNER          |
| MEMALL   | IF MEMBER                         |
| MEMSET   | IF <i>set-name</i> MEMBER         |
| TENALL   | IF TENANT                         |
| TENSET   | IF <i>set-name</i> TENANT         |
| EMPTY    | IF <i>set-name</i> EMPTY          |

No provision is made for COBOL NOT; however, this does not entail any actual limitation of the function.

User information:   Results returned in database status item

000: The condition is met

C11: The condition is not met

Regardless of the result, the program always continues at the specified return address (with the statement following the CALL or register 14).

DATABASE-STATUS returned

Set name:           ...SET/EMPTY

Input of a set name; only the specified set is checked.

## Activating extended record protection (KEEPC)

The KEEP function protects the Current Record of Run-unit (CRU) from access by other transactions until record protection is cancelled with FREEC or the transaction is terminated with FINISC.

Function code:           **KEEPC**

Function option:        Not applicable

User information:       DATABASE-STATUS returned

## Modifying records already stored (MODIF1/2)

The MODIF1/2 functions

- replace the values of all items or specified items of the CRU by values from the record area
- modify the position of the CRU within the set occurrence to match the set order if a sort key is modified
- may be used to disconnect the CRU from its current set occurrence and connect it into a new one determined using set selection mechanisms.

The user has the option of suppressing the updating of all or parts of the relevant currency information.

The MODIF1/2 functions can be accessed via 2 different function codes.

**Modifying the entire record**Function code: **MODIF1**

Function option:

| <b>CALL DML</b> | <b>Corresponding COBOL DML statement</b>                                 |
|-----------------|--------------------------------------------------------------------------|
| CORUNT          | MODIFY <i>record-name</i>                                                |
| INCALL          | MODIFY <i>record-name</i> INCLUDING ALL MEMBERSHIP                       |
| ONLALL          | MODIFY <i>record-name</i> ONLY ALL MEMBERSHIP                            |
| INCSET          | MODIFY <i>record-name</i> INCLUDING <i>set-name-1</i> ,...<br>MEMBERSHIP |
| ONLSET          | MODIFY <i>record-name</i> ONLY <i>set-name-1</i> ,...<br>MEMBERSHIP      |

Secondary option: Input of RET (optional), NOW (optional);

Retaining is only possible for sets

User information: DATABASE-STATUS returned

Record name: Input of the record name of the Current Record of Run-unit (CRU)

Set name: ...SET

Input of one or more set names; the membership of the CRU in the specified sets is checked and, if necessary, modified if this is possible solely via currency information.

If the membership of the CRU is modified in a set defined with the SET OCCURRENCE SELECTION clause THRU LOCATION MODE OF OWNER, MODIF1 reacts in the same way as for the SET OCCURRENCE SELECTION clause THRU CURRENT OF SET.

Record area: CORUNT/INCALL/INCSE

The record must be transferred in its complete subschema format to the record area. All items of the subschema are overwritten with contents from the record area.

ONLALL/ONLSET

The record area is not used; item contents remain unchanged.

Special parameter 1: Input of retaining parameters for secondary option RET; retaining is only possible for sets since the CRSs are changed.

## Modifying individual items

Function code: **MODIF2**

Not allowed for records with variable items.

| Function option: | CALL DML | Corresponding COBOL DML statement                                           |
|------------------|----------|-----------------------------------------------------------------------------|
|                  | CORUNT   | MODIFY <i>item-name</i> ,...                                                |
|                  | INCALL   | MODIFY <i>item-name</i> ,...<br>INCLUDING ALL MEMBERSHIP                    |
|                  | INCSET   | MODIFY <i>item-name</i> ,...<br>INCLUDING <i>set-name-1</i> ,... MEMBERSHIP |

Secondary option: Input of:  
RET (optional); retaining for sets only  
VAR (optional); abbreviated format

User information: DATABASE-STATUS returned

Record name: Input of the record name of the Current Record of Run-unit (CRU)

Set name: INCSET  
Input of one or more set names; the remarks concerning MODIF1 also apply here.

Item name: Input of one or more item names; only the contents of the specified items of the record are modified; the contents of the remaining items are unchanged.

Record area: Input of the item contents to be modified  
  
If the secondary option VAR is not specified, the contents must be placed in positions within the record area which correspond to the positions of the items in the subschema format of the record.  
If VAR is specified, the item contents must be written left-justified (starting at the beginning of parameter item <record-area>) and contiguously in order of item name.

Special parameter 1: See [page 252](#), MODIF1

## Preparing for processing (READYC)

The READYC function opens a transaction/processing chain and readies one or more realms for processing.

If the realm you want to access is locked when opening a transaction, you can return control to the application program (DATABASE-STATUS 12099) rather than waiting.

Function code: **READYC**

Function option:

| CALL DML | Corresponding COBOL DML statement |
|----------|-----------------------------------|
| ALL...   | Open all realms of the subschema  |
| RLM...   | Open specified realms only        |
| ...RTR   | USAGE MODE IS RETRIEVAL           |
| ...UPD   | USAGE MODE IS UPDATE              |
| ...PRT   | USAGE MODE IS PROTECTED RETRIEVAL |
| ...PUP   | USAGE MODE IS PROTECTED UPDATE    |
| ...ERT   | USAGE MODE IS EXCLUSIVE RETRIEVAL |
| ...EUP   | USAGE MODE IS EXCLUSIVE UPDATE    |

All 12 possible combinations are allowed.

Secondary option: Input of: NOW (optional)

User information: DATABASE-STATUS returned  
DB identifier returned

The database identifier must be specified again in all subsequent DML calls of the processing chain.

Realm name: ...RLM  
Input of one or more realm names;

Special parameter 1: Input of the subschema name in its full length (30 bytes); if the name is shorter than this, it must padded with blanks. CALL DML allows different subschemas to be called from the same module; the subschema name must therefore be transferred with the READYC call. The associated SSITAB modules must be available to the CALL DML converter.

Special parameter 2: Is no longer used; existing specifications, if any, are ignored by UDS/SQL.  
PRIVACY data is passed to UDS/SQL outside the program by openUTM or BS2000.

## Storing records (STORE1/2, STOR1L/2L)

The STORE1/2 and STOR1L/2L functions store a record in the database and connect it into all sets in which its record type is defined as AUTOMATIC member. The new record thus becomes

- Current of Run-unit,
- Current of Realm of the realm in which it is stored,
- Current of Record of its record type, and
- Current of Set of all sets in which its record type is defined as owner record type or AUTOMATIC member record type.

The user has the option of retaining all or some of the currency information affected by a STORE1/2 or STOR1L/2L call, except for the currency information for the CRU.

The STORE1/2 and STOR1L/2L functions treat sets defined with the SET OCCURRENCE SELECTION clause THRU LOCATION MODE OF OWNER in the same way as sets defined with the SET OCCURRENCE SELECTION clause THRU CURRENT OF SET.

STORE1/2 and STOR1L/2L differ with respect to the database key value to be specified.

STOR1L and STOR2L can only be executed in combination with an SSITAB module of UDS/SQL Version 2.0 or higher (see the [“Creation and Restructuring”](#) manual). Otherwise, UDS/SQL reports status code C98.



**Storing complete records**Function code: **STORE1**

Function code: STOR1L

Function option:

| CALL DML | Corresponding COBOL DML statement |
|----------|-----------------------------------|
| RECNAM   | STORE <i>record-name</i>          |
| IMPDAT   | STORE <i>record-name</i>          |

IMPDAT must be used if LOCATION MODE DIRECT or LOCATION MODE DIRECT-LONG is defined or several realms are specified in the WITHIN clause, where the record type is not the member record type of a distributable list. This implicitly defined data (database key value or realm name) must be transferred in special parameter 2.

Secondary option: Input of RET (optional)

User information: DATABASE-STATUS returned

Record name: Input of a record name

Record area: Input of the record to be stored in its subschema format. For records with variable-length items, the associated length item must be supplied with a binary value. Only the specified length of the record is stored.

Special parameter 1: Input of retaining parameters for secondary option RET

Special parameter 2: IMPDAT

Input of a database key value (for LOCATION MODE DIRECT or LOCATION MODE DIRECT-LONG) and/or a realm name if more than one realm name is specified in the WITHIN clause (see [page 210](#) for details).

If you want to specify a database key value with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1 in combination with LOCATION MODE DIRECT-LONG, you must use STOR1L.

**Storing individual items or compressed records**Function code: **STORE2**Function code: **STOR2L**

Function option:

| <b>CALL DML</b> | <b>Corresponding COBOL DML statement</b>                               |
|-----------------|------------------------------------------------------------------------|
| ITMNAM          | Storage of the specified items                                         |
| IMPDAT          | Storage of the specified items and transfer of implicitly defined data |

This format extends the functionality offered by COBOL DML. It allows abbreviated record formats to be used. This format is not allowed for records with variable-length items.

Secondary option: Input of:  
RET (optional) for retaining  
VAR (optional) for abbreviated record format

User information: DATABASE-STATUS returned

Record name: Input of a record name

Item name: Input of one or more item names.  
Items for which no names are specified are filled by UDS/SQL with binary zero. If the clause "COMPRESSION FOR ALL ITEMS" is contained in the SSL definition for the record type specified under "record name", items filled with binary zeros will not be stored in the database by UDS/SQL.

Record area: Input of the item contents of the specified items.  
If VAR is not specified, the item contents must be placed at points in the record area which correspond to their positions in the format of the subschema record.  
If VAR is specified, the item contents must be stored left-justified and contiguously in the specified order. This parameter, together with an SSL definition COMPRESSION FOR ALL ITEMS for the record type involved, leads to compressed storage of the record.

Special parameter 1: Input of retaining parameters for secondary option RET.

Special parameter 2: As for STORE1, STOR1L

## 8.4 CALL DML Assembler macros

The following macros are available for the (CALL8) variant to support UDS/SQL users working with CALL DML from within Assembler programs:

| Macro | Function                                                                                  |                                                                                                                                                                                                                                 | Application                                                                                                        |
|-------|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
|       | statically at assembly time                                                               | dynamically at runtime                                                                                                                                                                                                          |                                                                                                                    |
| DSCAL | Generate and initialize implicit parameter list (in instruction code)                     | <ul style="list-style-type: none"> <li>– where necessary complete implicit parameter list</li> <li>– execute CALL DML call with the implicit parameter list</li> </ul>                                                          | CALL DML call with implicit parameter list and convenient parameter mechanism<br><br>Alternative:<br>DSCAP + DSCDF |
| DSCAP | -                                                                                         | <ul style="list-style-type: none"> <li>– accept explicitly defined parameter list</li> <li>– where necessary fill out explicitly defined parameter list</li> <li>– execute CALL DM call with explicit parameter list</li> </ul> | CALL DML call with explicit parameter list<br><br>and<br><br>convenient parameter mechanism (optional)             |
| DSCDF | Generate and enter values for the (explicit) parameter list in the current CSECT or DSECT | -                                                                                                                                                                                                                               | Generate (explicit) CALL DML parameter list                                                                        |
| DSCPA | Generate user information area and/or predefined CALL DML parameter constants             | -                                                                                                                                                                                                                               | Support for ASSEMBLER programming at the CALL DML data interface                                                   |

Table 27: CALL DML Assembler macros

## Parameter mechanism

The CALL DML parameters are referenced in the DSCAL, DSCAP and DSCDF macros by means of the first twelve positional parameters of these macros and given the symbolic designations FCOD, FOPT, SOPT, UINF, RECN, SETN, RLMN, ITMN, RECA, SPP1, SPP2, SPP3 (see [section “Parameter definitions” on page 198](#)).

For parameters which specify the address of a data area, the following three methods of representation are possible:

1. *symbol*

The address of the data area is defined by the symbolic expression <symbol>. It must be possible to represent it as an A-type constant, i.e. the data area may not lie within a DSECT but does not have to be within the current base addresses.

*Example of “symbol”*

```
RECORD + L'ITEM-2
```

2. *\*symbol*

The address of the data area is defined by the symbolic expression <\*symbol>. It must be possible to represent it as an S-type constant, i.e. the data area must lie either within an (addressed) DSECT or within the current base addresses.

*Example of “\*symbol”*

```
*BUFFER + 16
```

3. *(r)*

The address of the data area is stored in register (r).

*Example of “(r)”*

```
(3) or (R12) or (WORKREG) where
R12 is EQU 12
WORKREG is EQU 7
```

Which of these three methods of representation is allowed in individual cases will be indicated in the description of the macro parameters.

## DSCAL

Macro DSCAL generates a CALL DML call with an implicit parameter list, i.e. the user does not have to worry about defining and supplying values for the parameter list.

---

```
[name] DSCAL fcod,fopt,...,spp3
```

---

*name* gives the macro call a symbolic name (optional)

*fcod,fopt,...spp3*

correspond to the 12 CALL DML parameters (see [section “Parameter definitions” on page 198](#)); the *symbol*, *\*symbol* and *(r)* methods of representation are allowed for each of these parameters. The rules for the definition of parameters (see [page 199](#)) must be strictly adhered to. Only parameters which are not used and come at the end of the list may be omitted.

### Example

```
CONN1 DSCAL CONNED,TOALL,RTSHORT,(RUSINF),*LEER8,NIL,NIL,NIL,NIL, -
 RESET
```

## DSCAP

The DSCAP macro generates a CALL DML call with an explicit parameter list, i.e. the user is free to define and supply values for the parameter list.

DSCAP must be used instead of DSCAL in the following cases:

- There should be only one parameter list for each transaction.
- The parameter list should not to be contained in the instruction code (but in temporary storage areas, for example)
- The application program itself should supply all the values for the parameter list.

---

```
[name] DSCAP [fcod,fopt,...,spp3][,PARAM=param]
```

---

*name* Assigns a symbolic name to the macro call (optional)

*fcod,fopt,...spp3*

Specification as for DSCAL (optional)

If you have not entered any *fcod,fopt,...* parameters, you must define the parameter list (see the DSCDF macro on [page 263](#)) and fill in the addresses of the data areas required.

*param*

Allows the address of a parameter list to be specified using the *symbol*, *\*symbol* or (*r*) method of representation. If this parameter is omitted, the address of the parameter list is expected in register 1.

### Examples

- a) DSCAP PARAM=\*DMLP0001
- b) DSCAP PARAM=(R7)
- c) L 1,PARADDR  
DSCAP
- d) DSCAP CONNEC,TOALL,RTSHORT,(RUSINF),\*LEER8,NIL,NIL,NIL,NIL, -  
RESET,PARAM=PARLEIST

## DSCDF

The DSCDF macro generates a parameter list for CALL DML calls with the DSCAP macro and supplies it with initial values (statically).

---

```
[name] DSCDF fcod,fopt,...,spp3[,SUFFIX=x]
```

---

*name* Assigns a symbolic name to the generated parameter list (optional)

*fcod,fopt,...,spp3*

Correspond to the 12 CALL DML parameters (see [page 197](#)). Only the *symbol* method of representation is allowed for each of these parameters. Individual parameters or all parameters can be left out; the corresponding values in the parameter list are then initialized with binary zero.

*x* This is used to avoid conflicts of names between different parameter lists (optional).

A DSCDF call generates the following code at its location in the current CSECT or DSECT:

```

 DS OF
DMLPx DS OCL48
[Name EQU DMLPx]
FCODx DC A(symbol or 0) Function code
FOPTx DC A(") Function option
SOPTx DC A(") Secondary option
UINFx DC A(") User information
RECNx DC A(") Record name
SETNx DC A(") Set name
RLMNx DC A(") Realm name
ITMNx DC A(") Item name
RECAx DC A(") Record area
SPE1x DC A(") Special parameter 1
SPE2x DC A(") Special parameter 2
SPE3x DC A(") Special parameter 3

```

The first four characters of the parameter SUFFIX are entered for *x*.

*Example***Macro call:**

```
CONNPAR DSCDF CONNED,TOALL,RTSHORT,USINFO,BLANK8,NIL,NIL,NIL,NIL, -
 RESET,SUFFIX=CONN
```

**Generated parameter list:**

```
 DS OF
DMLPCONN DS OCL48
CONNPAR EQU DMLPCONN
FCODCONN DC A(CONNED)
FOPTCONN DC A(TOALL)
SOPTCONN DC A(RTSHORT)
UINFCONN DC A(USINFO)
RECNCNN DC A(BLANK8)
SETNCONN DC A(NIL)
RLMNCONN DC A(NIL)
ITMNCONN DC A(NIL)
REACONN DC A(NIL)
SPE1CONN DC A(RETSET)
SPE2CONN DC A(O)
SPE3CONN DC A(O)
```



## DSCPA

The DSCPA macro supports the user at the CALL DML data interface.

---

*[name]* DSCPA *[option]*

---

*name* Assigns a symbolic name to the macro call (optional); this is equated with the start of the generated constants when DMLPAR is specified for OPTION.

*option* The following can be specified here (optional):

### USERINF

The definition of the user-information area is generated in the current CSECT or DSECT. This area definition should be aligned on a word boundary.

### USERINFDS

The definition of the user-information area is generated as DSECT USERINF.

### DMLPAR

The following are generated as symbolically addressable constants in the current DSECT or CSECT:

- all CALL DML function codes
- all function option parameters of all function codes (except LOOKC)
- the most important secondary option parameters

### DMLPARDS

Generates a DSECT DMLPAR with the symbolically addressable constants named in DMLPAR so that symbolic access to these constants is also possible from other modules.

A **missing** OPTION parameter has the same effect as a combination of DSCPA USERINFDS and DSCPA DMLPAR.

The generated symbolic names and values can be taken from a DSCPA macro listing or from relevant test calls.

### Example

```

 DSCPA USERINFDS
CDMLCON DSCPA DMLPAR

```

## 8.5 LOOKC function

The LOOKC function enables structure information to be interrogated at subschema level. The following structural information can be obtained:

| Information specified | Structural information provided on                      |
|-----------------------|---------------------------------------------------------|
| SET                   | Owner and member records of the set                     |
| RECORD                | Sets in which the record is owner or member             |
| RECORD                | Items in the record                                     |
| ITEM STRUCTURE        | Items in the structure                                  |
| RECORD                | Areas in which RECORD KEYS of the record can be located |
| SET                   | Keys of the set                                         |
| KEY                   | Items in the key                                        |
| ITEM                  | Keys in which the item is contained                     |
| SET AND ITEM          | Keys of the sets in which the item is contained         |

Table 28: Structural information

A classification of the LOOKC functions according to search argument and information contained in the answer produces the following two function groups:

1. LOOKC for a name (function option: '...NAM')

The search argument consists of the full name (length: 30 bytes) and the type (realm, record etc.) of element. The answer comprises a short reference which identifies the element internally, the type itself and additional information.

2. LOOKC for a type (function option: '...RLM', '...REC', '...SET', '...ITM', '...KEY')

The type involved can be a realm, record, set, item or key type. The search argument is the internal reference (in some cases two or more) and a special description (see [page 268](#)).

*Example*

The objective is to obtain the key of a set:

Required is the set reference, possibly the record reference and the position within the record of the item associated with the key.

A further distinction between LOOKC functions can be made, depending on whether a LOOKC call will return a single answer (a LOOKC block) or a result vector whose components are individual LOOKC blocks. In the second case, the user specifies a LOOKC number (2 byte in length, binary coded, between 1 and 255, inclusive) in special parameter 1 (10th. SPP1) to indicate the maximum number of successive LOOKC blocks to be located in the record area. The length of a LOOKC block is 56 bytes. Note that the following condition must be observed:

Length of record area  $\geq$  LOOKC number \* 56

### Simple LOOKC functions

- Function option 'SPC...' and 'FST...' (SPECIFIED, FIRST)

The user must initialize the LOOKC block with search arguments according to the rules of the interface table (for example, the search arguments for optional entry 'SPCNAM' are name and type, and for optional entry 'FSTNAM' just type).

- Function option 'NXT...' (NEXT)

The result of a LOOKC call can serve as input for a further call if the optional entry NEXT is used. However, the LOOKC block must be protected from being overwritten between the two calls (where necessary by saving it).

### Compound LOOKC functions

- Function option 'ALL...'

A LOOKC block at the start of the record area must be initialized. The answer consists of the number of LOOKC blocks which can be filled by the system (corresponding to the subschema data contained in the database). The user can limit the number of LOOKC blocks for each call by specifying the LOOKC number.

- Function option 'FRT...' (FROM-TO)

In this case the user has the option of determining the limits of the answer set by initializing two LOOKC blocks. The answer includes all elements of which the value (e.g. name) is not less than the search argument specified in the first LOOKC block and not greater than the search argument specified in the second LOOKC block. The LOOKC blocks which contain the search arguments are overwritten by the answer.

- Function option 'LIS...' (LIST)

This optional entry corresponds to the 'SPC...' option except for the fact that the number of LOOKC blocks specified by the LOOKC number are to be initialized here. The answers are written back into precisely those blocks in which the search arguments were previously entered.

- Function option 'OM-NAM' (OWNER-MEMBER) and secondary option 'SET...'

In the case of this special LOOKC call, a LOOKC block with the desired set reference is initialized as the search argument. The answer consists of one LOOKC block for the owner and one for the member of the specified set, i.e. the LOOKC number must have been initialized with the value 2. In the case of SYSTEM sets or dynamic sets, only the external name 'SYSTEM\_...\_(30)' or 'DYNAMIC\_...\_(30)' is entered into the LOOKC block of the owner.

- Secondary option NXA.../...NXA: NEXT PART OF ANSWER

If the result vector of a compound LOOKC function contains more components than are catered for by the LOOKC number, it is possible to reissue the preceding LOOKC call with the secondary option NXA... or ...NXA (according to function option) in order to call the next part of the result vector. This step can be repeated as often as is necessary to output the whole result vector. Note in this case that no other CALL DML calls may be executed within this sequence of calls.

### 8.5.1 The LOOKC block

The interface for the exchange of input/output data during a LOOKC call is known as the **LOOKC block** and is an overlay on the record area (RECA). The length of a LOOKC block is 56 bytes. The LOOKC block contains:

- a general description for all LOOKC calls (length: 38 bytes)
- a special description for individual LOOKC calls (length: 18 bytes)

The values in the “Contents” column of the tables describing the LOOKC block below are to be interpreted as follows:

- If “(Byte)” is specified, the entire bit pattern (i.e. the value) of the byte in question in the LOOKC block must be tested for the value specified in the “Contents” column.
- If “(Bit)” is specified, only the bit at the indicated position is relevant within the corresponding byte of the LOOKC block and needs to be tested to check whether or not “it is set”. For example, if 01 is given in the “Contents” column, the last bit of the corresponding byte must be evaluated; if 08 is given, the 5th. bit (from the left), and so on.

The input data required for the various LOOKC calls and the output data returned are shown in the overview contained in the [section “LOOKC tables” on page 278](#).

**General description of all LOOKC calls**

| Meaning                                                                                  | Contents                                   | Length          | Displ. | Type      |
|------------------------------------------------------------------------------------------|--------------------------------------------|-----------------|--------|-----------|
| Name of the element                                                                      |                                            | 30 <sup>1</sup> | 0      | character |
| Reference 1                                                                              |                                            | 2               | 30     | binary    |
| Reference 2                                                                              |                                            | 2               | 32     | binary    |
| Data type<br>– no type specified<br>– realm<br>– record type<br>– set<br>– item<br>– key | (Byte)<br>00<br>01<br>02<br>03<br>04<br>05 | 1               | 34     | binary    |
| Name ambiguity<br>– name unique<br>– name not unique                                     | (Bit)<br>00<br>40                          | 1               | 35     | binary    |
| Result<br>– o.k.<br>– not found<br>– no further element found                            | (Bit)<br>00<br>01/02/08<br>04              | 1               | 36     | binary    |
| Spare                                                                                    |                                            | 1               | 37     |           |

Table 29: General description in the LOOKC block

<sup>1</sup> also applies to (CALL8) variant

## Special description

The 18 bytes in the special description are formatted differently, depending on whether information is entered for a realm, a record, a set, an item or a key.

Due to compatibility reasons, a separate short and long variant of the following areas are present in the special description of the LOOKC block:

- The areas “short/long entry for position in record area (displacement within the COBOL-BIB)” in the special description of a record type (see [page 271](#)).

UDS/SQL stores the “displacement within the COBOL-BIB” information as follows:

- If the displacement  $\leq 2^{16}-1$ , UDS/SQL stores the displacement in both the short and the long areas.
- If the displacement  $\geq 2^{16}$ , UDS/SQL stores the displacement in the long area and enters the value X'FFFF' (an invalid value for displacement) in the short area.
- The (short/long) areas for record references and (short/long) areas for set references in the special descriptions of a set (see [page 272](#)), item (see [page 273](#)) and key (see [page 274](#)).

The following is generally applicable in the context of filling areas for record or set references:

- For input:
  - A record reference number or set number  $\leq 254$  may be optionally specified in the relevant short or long area. If you specify the number in the short area, the content of the long area is not significant. If you specify a binary 0 in the short area, UDS/SQL will expect the number in the long area.
  - A record reference number or set number  $> 254$  must be specified in the relevant long area, and the corresponding short area must be assigned the (invalid) value X'00'.
- For output:
  - A record reference number or set number  $\leq 254$  is stored by UDS/SQL in both the short and the long areas.
  - A record reference number or set number  $> 254$  is stored by UDS/SQL in the relevant long area, and the value X'00' (for an invalid record reference/set number) is entered in the corresponding short area.

*Special description of a realm in the LOOKC block*

| Meaning                                            | Contents                              | Length | Displ. | Type   |
|----------------------------------------------------|---------------------------------------|--------|--------|--------|
| Filler                                             |                                       | 4      | 0      | binary |
| Realm status<br>– not temporary<br><br>– temporary | (Bit)<br>not equal<br><b>80</b><br>80 | 1      | 4      | binary |
| Spare                                              |                                       | 13     | 5      |        |

Table 30: Special description of a realm in the LOOKC block

*Special description of a record type in the LOOKC block*

| Meaning                                                                                                               | Contents                       | Length | Displ. | Type   |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------------|--------|--------|--------|
| Long entry for position in record area<br>(displacement within the COBOL BIB)                                         |                                | 4      | 0      | binary |
| Length of a record type                                                                                               |                                | 2      | 4      | binary |
| LOCATION MODE<br>– DIRECT<br>– CALC DUPLICATES<br>– CALC NO DUPLICATES<br>– no LOCATION MODE                          | (Byte)<br>00<br>01<br>02<br>03 | 1      | 6      | binary |
| Details <sup>1</sup><br>– rec. type, compressed<br>– record type with SEARCH key<br>– record type in different realms | (Bit)<br>80<br>20<br>10        | 1      | 7      | binary |
| Short entry for position in record area<br>(displacement within the COBOL BIB)                                        |                                | 2      | 8      | binary |
| Spare                                                                                                                 |                                | 8      | 10     |        |

Table 31: Special description of a record type in the LOOKC block

<sup>1</sup> Combinations are possible

*Special description of a set in the LOOKC block*

| Meaning                                                                             | Contents                                   | Length | Displ. | Type   |
|-------------------------------------------------------------------------------------|--------------------------------------------|--------|--------|--------|
| Owner record reference (long)                                                       |                                            | 2      | 0      | binary |
| Member record reference (long)                                                      |                                            | 2      | 2      | binary |
| Owner record reference (short)                                                      |                                            | 1      | 4      | binary |
| Member record reference (short)                                                     |                                            | 1      | 5      | binary |
| Set order<br>– SORTED<br>– FIRST<br>– LAST<br>– NEXT<br>– PRIOR<br>– SORTED INDEXED | (Byte)<br>00<br>11<br>22<br>44<br>78<br>80 | 1      | 6      | binary |
| CONNECT type<br>– AUTOMATIC<br>– MANUAL                                             | (Byte)<br>00<br>01                         | 1      | 7      | binary |
| DISCONNECT type<br>– MANDATORY<br>– OPTIONAL                                        | (Byte)<br>00<br>01                         | 1      | 8      | binary |
| SET SELECTION<br>– THRU OWNER<br>– THRU CURRENT                                     | (Byte)<br>00<br>01                         | 1      | 9      | binary |
| Special types <sup>1</sup><br>– singular set<br>– DYNAMIC SET<br>– implicit set     | (Bit)<br>80<br>40<br>20                    | 1      | 10     | binary |
| Spare                                                                               |                                            | 7      | 11     |        |

Table 32: Special description of a set in the LOOKC block

<sup>1</sup> Combinations are possible.

If the corresponding bits are not specified: no singular set / DYNAMIC SET / implicit set



*Special description of an item in the LOOKC block*

| Meaning                                                                                                                                                                                                                                                 | Contents                                                     | Length | Displ. | Type   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|--------|--------|--------|
| Filler                                                                                                                                                                                                                                                  |                                                              | 2      | 0      | binary |
| Connection to key item set:<br>set reference (long) to the set belonging to the key<br>referenced with displacement 14                                                                                                                                  |                                                              | 2      | 2      | binary |
| Level number                                                                                                                                                                                                                                            |                                                              | 1      | 4      | binary |
| Next level number                                                                                                                                                                                                                                       |                                                              | 1      | 5      | binary |
| Item type<br>– database key<br>– packed decimal<br>– binary<br>– character<br>– signed unpacked decimal<br>– unsigned unpacked decimal<br>– group item<br>– national<br>– national data group                                                           | (Byte)<br>00<br>01<br>02<br>04<br>05<br>06<br>0F<br>14<br>1F | 1      | 6      | binary |
| Scale modifier<br>Bit 0 = sign<br>Bit 1-7 = value                                                                                                                                                                                                       |                                                              | 1      | 7      | binary |
| Item length                                                                                                                                                                                                                                             |                                                              | 2      | 8      | binary |
| Number of occurrences                                                                                                                                                                                                                                   |                                                              | 2      | 10     | binary |
| LOCATION MODE IS CALC indicator<br>– simple key<br>– compound key<br>– no key                                                                                                                                                                           | (Bit)<br>40<br>20<br>00                                      | 1      | 12     | binary |
| SEARCH KEY...USING CALC /SEARCH KEY...USING<br>INDEX- / sort key indicator <sup>1</sup><br>– no key<br>– simple key<br>– compound key<br>– item used in more than one key<br>– repeating group item<br>– item is a group item that forms a compound key | (Bit)<br>80<br>40<br>20<br>10<br>01<br>08                    | 1      | 13     | binary |

Table 33: Special description of an item in the LOOKC block

(part 1 of 2)

| Meaning                                                                      | Contents | Length | Displ. | Type   |
|------------------------------------------------------------------------------|----------|--------|--------|--------|
| Connection to key item set:                                                  |          |        |        |        |
| – reference to the first key in which the item is a key item                 |          | 2      | 14     | binary |
| – set reference (short) to the set belonging to this key                     |          | 1      | 16     | binary |
| Additional information                                                       | (Bit)    | 1      | 17     | binary |
| – number of digits is even<br>(only relevant for item type = packed decimal) | 80       |        |        |        |

Table 33: Special description of an item in the LOOKC block

(part 2 of 2)

<sup>1</sup> Combinations are possible*Special description of a key in the LOOKC block*

| Meaning                               | Contents | Length | Displ. | Type   |
|---------------------------------------|----------|--------|--------|--------|
| Record reference (long)               |          | 2      | 0      | binary |
| Filler                                |          | 2      | 2      | binary |
| Key length                            |          | 1      | 4      | binary |
| Details of key <sup>1</sup>           | (Bit)    | 1      | 5      | binary |
| – DUPLICATES NOT ALLOWED              | 80       |        |        |        |
| – DUPLICATES ALLOWED                  | 00       |        |        |        |
| – DESCENDING                          | 40       |        |        |        |
| – ASCENDING                           | 00       |        |        |        |
| – Index table for key                 | 20       |        |        |        |
| – implicit set                        | 10       |        |        |        |
| – explicit se                         | 00       |        |        |        |
| Number of items which make up the key |          | 1      | 6      | binary |
| Record reference (short)              |          | 1      | 7      | binary |
| Position of 1st. key item in record   |          | 2      | 8      | binary |
| Item length                           |          | 2      | 10     | binary |
| Item type (cf. item description)      |          | 1      | 12     | binary |
| Spare                                 |          | 5      | 13     | binary |

Table 34: Special description of a key in the LOOKC block

<sup>1</sup> Combinations are possible

## 8.5.2 Description of LOOKC parameters

All LOOKC functions use the following CALL DML parameters:

Function code:           **LOOKC**

as well as the CALL DML parameters

- Function option
- Secondary option
- Record area
- Special parameter 1
- Special parameter 2

The individual parameters are explained in detail in the overview ([LOOKC tables](#)) starting on [page 278](#).

### Function option for LOOKC calls

The function option is always 6 characters in length and consists of a combination of the following symbols:

|     |                                        |
|-----|----------------------------------------|
| ALL | All elements are to be processed       |
| FRT | From-to: sequence of elements          |
| FST | First element                          |
| ITM | Item                                   |
| KEY | Key                                    |
| LIS | List of elements                       |
| MEM | Member records                         |
| NAM | Element name is specified or requested |
| NXT | Next element                           |
| OM- | Owner/Member record type               |
| OWN | Owner record type                      |
| REC | Record type                            |
| RLM | Realm                                  |
| SET | Set                                    |
| SPC | Specified element                      |

## Secondary options for LOOKC calls

- Fixed-format secondary options

The fixed-format representation of the secondary option is always 6 characters in length for the LOOKC function and consists of a combination of the symbols:

|     |                                  |
|-----|----------------------------------|
| AGG | Vector or group item (aggregate) |
| FST | First                            |
| ITM | Item                             |
| KEY | Key                              |
| MEM | Member                           |
| NXA | Next part of answer              |
| OWN | Owner                            |
| REC | Record type                      |
| SET | Set                              |
| SPC | Specified                        |
| ___ | Blanks                           |

- Free-format secondary options

The LOOKC function also allows a free-format representation of the secondary option. For the rules, refer to [page 203](#); the symbols are the same as for the fixed-format secondary option for LOOKC.

*Examples*

() or (\_\_\_) is the same as \_\_\_\_\_

(SET,ITM) is the same as ITMSET

The permitted combinations of function options and secondary options and their meanings can be found in the overview on [page 278](#).

### Special parameter 2 (SPP2) for LOOKC calls

The LOOKC function option ALLRLM shows all realms in which the records of a particular record type can be stored. The record reference number (REC-REF) of the desired record type is passed in special parameter 2 (SPP2), which has the following format for LOOKC calls:

| Byte 1                                    | Byte 2 | Bytes 3...4                                    |
|-------------------------------------------|--------|------------------------------------------------|
| Record reference no. (REC-REF) $\leq 255$ | Filler | Record reference no. (REC-REF) $\leq 2^{15}-1$ |

Table 35: Transfer format for special parameter 2

The record reference number must be entered in SPP2 as follows:

- A record reference number  $> 255$  must always be entered in bytes 3...4 of SPP2, and the invalid value X'00' must be entered in byte 1 of SPP2 in this case.
- A record reference number  $\leq 255$  may be optionally entered in byte 1 or in bytes 3...4 of SPP2.

If you enter a record reference number  $\leq 255$  in byte 1, the contents of bytes 3...4 are not relevant.

If you enter a record reference number  $\leq 255$  in bytes 3...4, you must enter the invalid value X'00' as the record reference number in byte 1.

### 8.5.3 LOOKC tables

The following abbreviations are used in the overview:

|     |                                                                                                                 |
|-----|-----------------------------------------------------------------------------------------------------------------|
| I   | Input                                                                                                           |
| O   | Output                                                                                                          |
| I,O | Output different from input                                                                                     |
| AR  | Realm reference (binary; length: 1 byte)                                                                        |
| RR  | Record reference (binary; the length of a short reference is 1 byte, the length of a long reference is 2 bytes) |
| SR  | Set reference (binary; the length of a short reference is 1 byte, the length of a long reference is 2 bytes)    |
| IR  | Position of item within record (binary; length: 2 bytes)                                                        |
| K   | Key reference (binary; length: 2 bytes)                                                                         |

The individual LOOKC tables are illustrated in the following on facing pages.

| Meaning of the function | Function option | Secondary option | Record area (LOOKC block)      |             |             |
|-------------------------|-----------------|------------------|--------------------------------|-------------|-------------|
|                         |                 |                  | General description (38 bytes) |             |             |
|                         |                 |                  | external name                  | reference 1 | reference 2 |
|                         | character       | format-free      | character                      | binary      | binary      |
|                         | 6               |                  | 30                             | 2           | 2           |

**LOOKC for a name**

|                                            |        |       |     |                      |                 |
|--------------------------------------------|--------|-------|-----|----------------------|-----------------|
| specified name (of the specified type)     | SPCNAM | ()    | I   | I,O <sup>1) 3)</sup> | O <sup>2)</sup> |
| first name                                 | FSTNAM | ()    | O   | O <sup>1)</sup>      | O <sup>2)</sup> |
| next name                                  | NXTNAM | ()    | I,O | O <sup>1)</sup>      | O <sup>2)</sup> |
| all names, starting with the one specified | ALLNAM | (SPC) | I,O | I,O <sup>1) 3)</sup> | O <sup>2)</sup> |
| all names, starting with the first one     | ALLNAM | (FST) | O   | O <sup>1)</sup>      | O <sup>2)</sup> |
| all names; next part of answer             | ALLNAM | (NXA) | O   | O <sup>1)</sup>      | O <sup>2)</sup> |
| owner/member record names of a set         | OM-NAM | (SET) | O   | I(SR)O(RR)           | O (Null)        |
| FROM-TO names                              | FRTNAM | ()    | I,O | I,O <sup>1) 3)</sup> | O <sup>2)</sup> |
| FROM-TO names; next part of answer         | FRTNAM | (NXA) | O   | O <sup>1)</sup>      | O <sup>2)</sup> |
| list of specified names                    | LISNAM | ()    | I   | I,O <sup>1) 3)</sup> | O <sup>2)</sup> |

**LOOKC for a realm**

|                                                                        |        |          |   |        |  |
|------------------------------------------------------------------------|--------|----------|---|--------|--|
| list of all realms with the specified record type                      | ALLRLM | (REC)    | O | O (AR) |  |
| list of all realms with the specified record type, next part of answer | ALLRLM | (RECNXA) | O | O (AR) |  |

1) AR in data type = realm      2) zero in data type = realm      3) input only for  
 RR in data type = record type ) zero in data type = record type      data type = item  
 SR in data type = set      zero in data type = set  
 RR in data type = item      IR in data type = item  
 RR in data type = key      IR in data type = key

|           |                 |        |        | Special description<br>(18 bytes) | Special parameter 1    | Special parameter 2 |
|-----------|-----------------|--------|--------|-----------------------------------|------------------------|---------------------|
| data type | duplicate names | result | spare  |                                   | number of LOOKC blocks | record reference    |
| binary    | binary          | binary | binary |                                   | binary                 | binary              |
| 1         | 1               | 1      | 1      |                                   | 2                      | 4                   |

|     |  |   |  |                                                                              |        |  |
|-----|--|---|--|------------------------------------------------------------------------------|--------|--|
| I,O |  | O |  | Only for data type = item or key;<br>special item description is transferred |        |  |
| I,O |  | O |  |                                                                              |        |  |
| I,O |  | O |  |                                                                              |        |  |
| I,O |  | O |  |                                                                              | I      |  |
| I,O |  | O |  |                                                                              | I      |  |
| O   |  | O |  |                                                                              | I      |  |
|     |  |   |  |                                                                              | I (=2) |  |
| I,O |  | O |  |                                                                              | I      |  |
| O   |  | O |  |                                                                              | I      |  |
| I,O |  | O |  |                                                                              | I      |  |

|   |  |   |  |   |   |        |
|---|--|---|--|---|---|--------|
| O |  | O |  | O | I | I (RR) |
| O |  | O |  | O | I |        |



| Meaning of the function | Function option | Second. option | Record area (LOOKC block)      |             |             |
|-------------------------|-----------------|----------------|--------------------------------|-------------|-------------|
|                         |                 |                | General description (38 bytes) |             |             |
|                         |                 |                | external name                  | reference 1 | reference 2 |
|                         | character       | format-free    | character                      | binary      | binary      |
|                         | 6               |                | 30                             | 2           | 2           |

**LOOKC for a record type**

|                                     |        |       |   |          |        |
|-------------------------------------|--------|-------|---|----------|--------|
| specified record types              | SPCREC | ( )   | O | I (RR)   |        |
| first record type                   | FSTREC | ( )   | O | O (RR)   |        |
| next record type                    | NXTREC | ( )   | O | I,O (RR) |        |
| owner record type of specified set  | OWNREC | (SET) | O | O (RR)   | I (SR) |
| member record type of specified set | MEMREC | (SET) | O | O (RR)   | I (SR) |
| list of specified record types      | LISREC | ( )   | O | I (RR)   |        |

**LOOKC for a set**

|                                                                              |        |          |   |          |        |
|------------------------------------------------------------------------------|--------|----------|---|----------|--------|
| specified set                                                                | SPCSET | ( )      | O | I (SR)   |        |
| first set                                                                    | FSTSET | ( )      | O | O (SR)   |        |
| next set                                                                     | NXTSET | ( )      | O | I,O (SR) |        |
| first set of owner record type                                               | FSTSET | (OWNREC) | O | O (SR)   | I (RR) |
| next set of owner record type                                                | NXTSET | (OWNREC) | O | I,O (SR) | I (RR) |
| first set of member record type                                              | FSTSET | (MEMREC) | O | O (SR)   | I (RR) |
| next set of member record type                                               | NXTSET | (MEMREC) | O | I,O (SR) | I (RR) |
| all sets in which the specified record type is an owner                      | ALLOWN | (REC)    | O | O (SR)   | I (RR) |
| all sets in which the specified record type is an owner, next part of answer | ALLOWN | (RECXA)  | O | O (SR)   |        |
| all sets in which the specified record type is a member                      | ALLMEM | (REC)    | O | O (SR)   | I (RR) |
| all sets in which the specified record type is a member, next part of answer | ALLMEM | (RECXA)  | O | O (SR)   |        |
| list of specified sets                                                       | LISSET | ( )      | O | I (SR)   |        |

|           |                 |        |        | Special description<br>(18 bytes) | Special parameter 1    | Special parameter 2 |
|-----------|-----------------|--------|--------|-----------------------------------|------------------------|---------------------|
| data type | duplicate names | result | spare  |                                   | number of LOOKC blocks | record reference    |
| binary    | binary          | binary | binary |                                   | binary                 | binary              |
| 1         | 1               | 1      | 1      |                                   | 2                      | 4                   |

|   |  |   |  |   |   |  |
|---|--|---|--|---|---|--|
| 0 |  | 0 |  | 0 |   |  |
| 0 |  | 0 |  | 0 |   |  |
| 0 |  | 0 |  | 0 |   |  |
| 0 |  | 0 |  | 0 |   |  |
| 0 |  | 0 |  | 0 |   |  |
| 0 |  | 0 |  | 0 | 1 |  |

|  |  |   |  |   |   |  |
|--|--|---|--|---|---|--|
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 |   |  |
|  |  | 0 |  | 0 | 1 |  |
|  |  | 0 |  | 0 | 1 |  |
|  |  | 0 |  | 0 | 1 |  |
|  |  | 0 |  | 0 | 1 |  |
|  |  | 0 |  | 0 | 1 |  |

| Meaning of the function | Function option | Second. option | Record area (LOOKC block)      |             |             |
|-------------------------|-----------------|----------------|--------------------------------|-------------|-------------|
|                         |                 |                | General description (38 bytes) |             |             |
|                         |                 |                | external name                  | reference 1 | reference 2 |
|                         | character       | format-free    | character                      | binary      | binary      |
|                         | 6               |                | 30                             | 2           | 2           |

**LOOKC for an item**

|                                                           |        |          |   |        |          |
|-----------------------------------------------------------|--------|----------|---|--------|----------|
| specified item of specified record type                   | SPCITM | (REC)    | O | I (RR) | I (IR)   |
| first item of specified record type                       | FSTITM | (REC)    | O | I (RR) | O (IR)   |
| next item of specified record type                        | NXTITM | (REC)    | O | I (RR) | I,O (IR) |
| all items of specified record type                        | ALLITM | (REC)    | O | I (RR) | O (IR)   |
| all items of specified record type in next part of answer | ALLITM | (RECNXA) | O | O (RR) | O (IR)   |
| all items of specified combination                        | ALLITM | (AGG)    | O | I (RR) | I,O (IR) |
| all items of specified combination, next part of answer   | ALLITM | (AGGNXA) | O | O (RR) | O (IR)   |
| FROM-TO items                                             | FRTITM | ()       | O | I (RR) | I,O (IR) |
| FROM-TO items, next part of answer                        | FRTITM | (NXA)    | O | O (RR) | O (IR)   |
| list of specified items                                   | LISITM | ()       | O | I (RR) | I (IR)   |

|            |                 |        |        |                                |              |                       | Special parameter 1    | Special parameter 2 |
|------------|-----------------|--------|--------|--------------------------------|--------------|-----------------------|------------------------|---------------------|
|            |                 |        |        | Special description (18 bytes) |              |                       |                        |                     |
| data types | duplicate names | result | spare  | filler                         | level number | remaining description | number of LOOKC blocks | record reference    |
| binary     | binary          | binary | binary | binary                         | binary       |                       | binary                 | binary              |
| 1          | 1               | 1      | 1      | 4                              | 1            | 13                    | 2                      | 4                   |

|   |  |   |   |   |                  |   |  |  |
|---|--|---|---|---|------------------|---|--|--|
| 0 |  | 0 | 0 | 0 | 1,0 <sup>1</sup> | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 0                | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 1,0              | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 0                | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 0                | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 1,0              | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 0                | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 1,0              | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 0                | 0 |  |  |
| 0 |  | 0 | 0 | 0 | 1,0              | 0 |  |  |

<sup>1</sup> If the specified level number (input) is not present, UDS/SQL returns the next level number found as the output.

| Meaning of the function | Function option | Secondary option | Record area (LOOKC block)      |             |             |
|-------------------------|-----------------|------------------|--------------------------------|-------------|-------------|
|                         |                 |                  | General description (38 bytes) |             |             |
|                         |                 |                  | external name                  | reference 1 | reference 2 |
|                         | character       | format-free      | character                      | binary      | binary      |
|                         | 6               |                  | 30                             | 2           | 2           |

**LOOKC for a key**

|                                                                    |        |          |  |          |         |
|--------------------------------------------------------------------|--------|----------|--|----------|---------|
| first key of specified set                                         | FSTKEY | (SET)    |  | I (SR)   | O (K)   |
| next key of specified set                                          | NXTKEY | (SET)    |  | I (SR)   | I,O (K) |
| first key of specified set which contains specified item           | FSTKEY | (ITMSET) |  | I (SR)   | O (K)   |
| next key of item in specified set                                  | NXTKEY | (ITMSET) |  | I (SR)   | I,O (K) |
| first key of item                                                  | FSTKEY | (ITM)    |  | O (SR)   | O (K)   |
| next key of item                                                   | NXTKEY | (ITM)    |  | I,O (SR) | I,O (K) |
| all keys of specified set                                          | ALLKEY | (SET)    |  | I (SR)   | O (K)   |
| all keys of specified set; next part of answer                     | ALLKEY | (SETNXA) |  | O (SR)   | O (K)   |
| all keys of specified item                                         | ALLKEY | (ITM)    |  |          | O (K)   |
| all keys of specified item in specified set                        | ALLKEY | (ITMSET) |  | I (SR)   | O (K)   |
| all keys of specified item (in specified set); next part of answer | ALLKEY | (ITMNXA) |  | O (SR)   | O (K)   |

|           |                 |        |        | Special description (18 bytes) |                     |                          |                |                     | Special parameter 1    | Special parameter 2 |
|-----------|-----------------|--------|--------|--------------------------------|---------------------|--------------------------|----------------|---------------------|------------------------|---------------------|
| data type | duplicate names | result | spare  | record reference (long)        | cf. key description | record reference (short) | item reference | cf. key description | number of LOOKC blocks | record reference    |
| binary    | binary          | binary | binary | binary                         |                     | binary                   | binary         |                     | binary                 | binary              |
| 1         | 1               | 1      | 1      | 2                              | 5                   | 1                        | 2              | 8                   | 2                      | 4                   |

|   |  |   |  |        |   |        |        |   |   |  |
|---|--|---|--|--------|---|--------|--------|---|---|--|
| O |  | O |  | O (RR) | O | O (RR) | O (IR) | O |   |  |
| O |  | O |  | O (RR) | O | O (RR) | O (IR) | O |   |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O |   |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O |   |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O |   |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O |   |  |
| O |  | O |  | O (RR) | O | O (RR) | O (IR) | O | I |  |
| O |  | O |  | O (RR) | O | O (RR) | O (IR) | O | I |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O | I |  |
| O |  | O |  | I (RR) | O | I (RR) | I (IR) | O | I |  |
| O |  | O |  | O (RR) | O | O (RR) | O (IR) | O | I |  |

| Meaning of the function | Function option | Second. option | Record area (LOOKC block)      |             |             |
|-------------------------|-----------------|----------------|--------------------------------|-------------|-------------|
|                         |                 |                | General description (38 bytes) |             |             |
|                         |                 |                | external name                  | reference 1 | reference 2 |
|                         | character       | format-free    | character                      | binary      | binary      |
|                         | 6               |                | 30                             | 2           | 2           |

**LOOKC for items of a key**

|                                                 |        |          |   |                |               |
|-------------------------------------------------|--------|----------|---|----------------|---------------|
| first item of specified key                     | FSTITM | (KEY)    | O | I (SR), O (RR) | I (K), O (IR) |
| next item of key                                | NXTITM | (KEY)    | O | I (SR)         | I (K), O (IR) |
| all items of key                                | ALLITM | (KEY)    | O | I (SR)         | I (K), O (IR) |
| all items of specified key; next part of answer | ALLITM | (KEYNXA) | O | O (SR)         | O (IR)        |

|                                       |                 |        |        |              |                     |               |                |                     |                      | Special parameter 1    | Special parameter 2 |
|---------------------------------------|-----------------|--------|--------|--------------|---------------------|---------------|----------------|---------------------|----------------------|------------------------|---------------------|
| Special description (18 bytes) in the |                 |        |        |              |                     |               |                |                     |                      |                        |                     |
|                                       |                 |        |        | Input        |                     |               |                |                     |                      | Output                 |                     |
| data type                             | duplicate names | result | spare  | long rec-ref | cf. key description | short rec-ref | item reference | cf. key description | cf. item description | number of LOOKC blocks | record reference    |
| binary                                | binary          | binary | binary | binary       |                     | binary        | binary         |                     |                      | binary                 | binary              |
| 1                                     | 1               | 1      | 1      | 2            | 5                   | 1             | 2              | 8                   | 18                   | 2                      | 4                   |

|   |  |   |  |        |  |        |        |  |   |   |  |
|---|--|---|--|--------|--|--------|--------|--|---|---|--|
| O |  | O |  |        |  |        |        |  | O |   |  |
| O |  | O |  | I (RR) |  | I (RR) | I (IR) |  | O |   |  |
| O |  | O |  |        |  |        |        |  | O | I |  |
| O |  | O |  |        |  |        |        |  | O | I |  |



## 8.6 Examples using different programming languages

### DMLTEST:

The following examples have been generated with the DMLTEST program (see [page 301](#)).

```

*** (IN) : * *****LOOKC FOR NAMES *****
*** (IN) : SET LOOKC,ALLNAM,FST
*** (IN) : DEF RECORD,DATTYP,D=34,L=1
*** (IN) : * *****14 LOOKC-BLOCKS*****
*** (IN) : SET SPP1='X'0010'
*** (IN) : * ***** NO ENTRY FOR DATA TYPE *****
*** (IN) : M DATTYP,'X'00'
*** (IN) : SH PARAM
 (OUT) : DB-PARAMS (FIRST 8B)
 (OUT) : FCOD :LOOKC .. FOPT :ALLNAM.. SOPT :FST UINF : RECN :
 (OUT) : SETN : RLMN : ITMN : RECA : SPP1 : ..
 (OUT) : SPP2 :SALES SPP3 : SUBS :ADMIN
*** (IN) : EX
 (OUT) : RECORD - AREA :
 (OUT) : CST-ORD-PLACED CURR-STOCK
*** (IN) : SH RECORD,L=840
 (OUT) : RECORD - AREA :
 (OUT) : CST-ORD-PLACED CURR-STOCK
 (OUT) : STATISTICS+.....
 (OUT) : NOT-AVAILABLE-CODE QUANTITY
 (OUT) : COL-NO
 (OUT) : .. COL-NAME MAT-CODE
 (OUT) : MAT-NAME
 (OUT) : SUPPL-NO
 (OUT) : SUPPL-NAME SUPPL-PCODE
 (OUT) : SUPPL-TOWN
 (OUT) : SUPPL-STREET SUPPL-STREET-NO
 (OUT) :
*** (IN) : SH RECORD,L=840,FORM=X
 (OUT) : RECORD - AREA :
 (OUT) : C1C2C7C5C7C5C2C5D5C560C2C5E2E3404040404040404040404040000F0000030000404040
 (OUT) : 40404040404040404040404040404040C1D2E360C2C5E2E3C1D5C44040404040404040404040
 (OUT) : DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.
*** (IN) : C
 (OUT) : 40404040400009004804000040404040400202010000060000008000000040E2E3C1E3C9E2E3C9
 (OUT) : D2404040404040404040404040404040404040400009004E0200000000000000404040404040
 (OUT) : 40404040404040D2C5D5E960D5C9C3C8E360D3C9C5C6C5D9C2C1D940404040404040400009
 (OUT) : 0056020000000A0C000040
 (OUT) : 40404040404040404040404040000A000002000000000000404040404040404040404040404040
 (OUT) : C6C1D9C260D5D940000B0000020000001120
 (OUT) : 000040
 (OUT) : 40404040404000B000202000001020000040
 (OUT) : 40404040404040404040404040404040404040000C00000200000012210000404040404040
 (OUT) : 4040404040404040D4C1E360C2C5E940000C
 (OUT) : 00010200000013210000404040404040404040404040404040404040D3C9C5C6C5D960D5D9404040404040
 (OUT) : 40404040404040404040404000D000002000000080B000040404040404040404040404040404040
 (OUT) : D3C9C5C6C5D960D5C1D4C5400000050200000080B
 (OUT) : 0000404040404040404040404040404040D3C9C5C6C5D960D7D3E94040404040404040404040404040404040
 (OUT) : 404040404000D0023020000000000040
 (OUT) : E3C1C4E3404040404040404040404040404040404000D00270200000000000000404040404040
 (OUT) : 40404040404040D3C9C5C6C5D960E2E3D9C1E2E2C5404040404040404040404040404040400000
 (OUT) : 0045020000000000000040404040404040404040404040404040404040D3C9C5C6C5D960C8C1E4E2D5D9404040
 (OUT) : 40404040404040404040404000D00630200000000000040

```

```

***(IN) : S RECA=C' '
(IN) : * ** ALL REALMS *****
***(IN) : M DATTYP,'X'01'
***(IN) : SH PARAM
 (OUT) : DB-PARAMS (FIRST 8B)
 (OUT) : FCOD :LOOKC .. FOPT :ALLNAM.. SOPT :FST UINF : RECN :
 (OUT) : DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.
***(IN) : C
 (OUT) : SETN : RLNM : ITMN : RECA : SPP1 : ...
 (OUT) : SPP2 :SALES SPP3 : SUBS :ADMIN
***(IN) : EX
 (OUT) : RECORD - AREA :
 (OUT) : ARTICLE-RLM :
***(IN) : SH RECORD,L=840
 (OUT) : RECORD - AREA :
 (OUT) : ARTICLE-RLM :
 (OUT) :
 (OUT) : HOUSEHOLD GOODS PURCHASE-ORDER-RLM
 (OUT) : FOOD LEISURE
 (OUT) : STATIONERY SPORTS-ARTICLES ..
 (OUT) : SEARCH-RLM
 (OUT) :
 (OUT) :
 (OUT) :
 (OUT) :
***(IN) : SH RECORD,L=840,FORM=X
 (OUT) : RECORD - AREA :
 (OUT) : C1D9E3C9D2C5D3D9D3D4404040404040404040404040404040000B0000010000404040
 (OUT) : 404040404040404040404040404040C1E4C6E3D9C1C7E2D9D3D4404040404040404040
 (OUT) : 40404040400003000001000040
 (OUT) : D3D440000400000100004040404040
 (OUT) : DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.
***(IN) : C
 (OUT) : 4040404040404040C8C1E4E2C8C1D3E340404040404040404040404040404040400006
 (OUT) : 000001000040D2D3C5C9C4E4D5C740404040404040
 (OUT) : 40404040404040404040404040400005000001000040
 (OUT) : D3C5C2C5D5E2D4C9E3E3C5D3404040404040404040404040404040404000080000010000404040
 (OUT) : 40404040404040404040404040404040E2C3C8D9C5C9C2E6C1D9C5D54040404040404040404040
 (OUT) : 4040404040000A00000100004040404040404040404040404040404040E2D7C9C5D3C560C8
 (OUT) : D6C2C2E840404040404040404040404040404040000900000100004040404040404040404040
 (OUT) : 40404040404040E2D7D6D9E3400007
 (OUT) : 000001000040E2E4C3C8D9D3D4404040404040404040
 (OUT) : 4040404040404040404040404040000C000001000040
 (OUT) : 40
 (OUT) : 40
 (OUT) : 40
 (OUT) : 40
 (OUT) : 40
 (OUT) : 40

```

The following programming examples have been generated using COBOL, FORTRAN, and Assembler.

### COBOL:

The COBOL program is based on the CALL-DML variant (CALL30). In a loop, the program executes a FTCH7A statement with search condition. Record name, item name and value of the item are queried on the terminal. The length of the value is set at 10 characters. The first 80 bytes of the record found are displayed on the screen.

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CALLDML.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
 TERMINAL IS T.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FCOD PIC X(6).
01 FOPT PIC X(6).
01 SOPT PIC X(12).
01 UINF.
 02 SYSTEM-COMMUNICATION-LOCATIONS.
 03 DATABASE-REALM-NAME PIC X(30).
 03 DATABASE-RECORD-NAME PIC X(30).
 03 DATABASE-SET-NAME PIC X(30).
 03 DATABASE-STATUS PIC X(5).
*
 02 FILLER PIC X.
 02 UINF-DBKEY PIC X(4).
 02 UINF-TALLY PIC 9(8) COMP.
 02 FILLER PIC X(7).
 02 UINF-DBKZ PIC X.
 02 FILLER PIC X(8).
 02 UINF-END PIC X(6).
*
01 RECN PIC X(30).
01 SETN PIC X(30).
01 RLMN PIC X(30).
01 ITMN PIC X(53).
01 RECA.
 02 RECA1 PIC X(80).
 02 RECA2 PIC X(2000).
01 SPP1 PIC X(30).
*
01 SEARCH EXPRESSION.
 02 SX-BEGIN PIC X(2) VALUE "0 ".

```

```

02 SX-NAME PIC X(30).
02 SX-REL PIC X(5) VALUE " EQU ".
02 SX-VALUE PIC X(10).
02 SX-END PIC X(6) VALUE " 0 END".
*
PROCEDURE DIVISION.
* INITIALIZE UINF.
 MOVE "UINF1*" TO UINF-END.
*
* OPEN TRANSACTION:
 MOVE "READYC" TO FCOD.
 MOVE "ALLRTR" TO FOPT.
 MOVE "SUBSCH" TO SPPI.
 PERFORM DMLCALL.
*
 FETCH-LOOP.
*
* FTCH7A STATEMENT:
 MOVE "FTCH7A" TO FCOD.
 MOVE "RECSEX" TO FOPT.
 MOVE SPACES TO SOPT.
 DISPLAY "PLEASE ENTER RECORD NAME OR STOP" UPON T.
 ACCEPT RECN FROM T.
 IF RECN = "STOP" THEN GO TO FINISH.
*
 DISPLAY "PLEASE ENTER ITEM NAME" UPON T.
 ACCEPT SX-NAME FROM T.
 DISPLAY "PLEASE ENTER ITEM VALUE" UPON T.
 ACCEPT SX-VALUE FROM T.
 MOVE SEARCH EXPRESSION TO ITMN.
 PERFORM DMLCALL.
 DISPLAY RECA1 UPON T.
 GO TO FETCH-LOOP.
*
 FINISH.
* CLOSE TRANSACTION:
 MOVE "FINISC" TO FCOD.
 MOVE "ALLRLM" TO FOPT.
 PERFORM DMLCALL.
 STOP RUN.
*
 DMLCALL.
* EXECUTE CALL DML STATEMENT:
 CALL "DML" USING FCOD FOPT SOPT UINF RECN SETN RLMN
 ITMN RECA SPPI.
 IF DATABASE-STATUS NOT EQUAL "00000" THEN
 DISPLAY "DATABASE-STATUS: " DATABASE-STATUS

```

```

 UPON T.
*
 DSCEXT.
* CALL DML ERROR EXIT DSCEXT:
 ENTRY "DSCEXT".
 DISPLAY "BRANCH TO DSCEXT IN COBOL WAS EFFECTED"
 UPON T.
 STOP RUN.
*
```

The following programming examples are based on the (CALL8) variant. They are identical in function; the 12 CALL DML parameters can be entered interactively and thus it is possible to call most of the DML functions.

Input is unformatted and may be up to 80 characters in length. Each parameter is represented by a two-character symbol, e.g. FC for FCOD, which is followed by a character string enclosed in apostrophes.

### *Example*

```
FC ' FTCH4 ' FO ' RECFST ' RN ' REC1 '
```

Parameters FCOD, FOPT and RECN are either filled with the desired strings or with the appropriate number of blanks.

Input of EX or E\_ causes a CALL DML statement to be called.

When DP (Display Parameters) is specified, the first 8 bytes of each parameter are displayed on the screen; DR (Display Record) causes the first 40 bytes of the record area to be displayed. Programs are terminated by inputting ST or STOP.

In FORTRAN and Assembler the parameter codes are:

- |                   |               |
|-------------------|---------------|
| 1. FC (FCOD)      | 6. SN (SETN)  |
| 2. FO (FOPT)      | 7. RL (RLMN)  |
| 3. SO (SOPT)      | 8. IT (ITMN)  |
| 4. not applicable | 9. RA (RECA)  |
| 5. RN (RECN)      | 10. S1 (SPP1) |

Furthermore the following code exists:

```
SU Subschema
```

The DSCEXT entry is not covered by these examples; DSCEXT requires a special module to be linked into the programs (see the [section "CALL DML error handling routine DSCEXT" on page 124](#)).

**FORTRAN:**

```

PROGRAM FORDML
CHARACTER * 80 LINE
CHARACTER * 80 STRING
CHARACTER * 2 CMD
INTEGER * 2 POS,POSS

C
CHARACTER * 6 FCOD /'READYC' /
CHARACTER * 6 FOPT /'ALLUPD' /
CHARACTER * 12 SOPT /' '/
CHARACTER * 8 RECNAME
CHARACTER * 80 SETN,RLMN
CHARACTER *800 ITMN,RECA
CHARACTER * 30 SPP1 /'SUBTI1' /

C
CHARACTER *126 UINF
CHARACTER * 30 REALMNAME,RECNAME,SETNAME
CHARACTER * 5 DBSTATUS /'00000' /
F FIL1 *1,
F DBKEY *4,
F TALLY *4,
F RUNID *2,
F RUNREF *2,
F FIL2 *3,
F DBREF *1,
F SSITABADR *4,
F FIL3 *4,
F USINF *6 /'USINF*'/
INTEGER * 4 ZAEHLER (ZAEHLER=counter)
COMMON /UINF/ REALMNAME,RECNAME,SETNAME,DBSTATUS,
F FIL1,DBKEY,TALLY,RUNID,RUNREF,FIL2,
F DBREF,SSITABADR,FIL3,USINF
EQUIVALENCE (UINF,REALMNAME),
F (TALLY,ZAEHLER)

20 FORMAT(A80)
22 FORMAT(' ',A80)

C _____ READ INPUT _____
30 READ (1,20) LINE
 POS = 1

C
C _____ GET NEW COMMAND _____
80 DO 100, POS = POS,80,1
 IF (LINE(POS:POS).NE.' ') GOTO 104
100 CONTINUE
 IF (POS.GE.80) GO TO 30

```

```

104 CONTINUE
 CMD = LINE(POS:POS+1)
 DO 108, POS = POS+2,80,1
 IF (LINE(POS:POS).NE.' ') GOTO 112
108 CONTINUE
112 IF (LINE(POS:POS).NE.'''') GOTO 160
C ----- GET NEW STRING -----
 DO 116, POSS = 1,80,1
 STRING(POSS:POSS) = ' '
116 CONTINUE
 POSS = 0
 DO 120, POS=POS+1,80,1
 IF (LINE(POS:POS).EQ.'''') GOTO 124
 POSS = POSS+1
 STRING(POSS:POSS) = LINE(POS:POS)
120 CONTINUE

124 POS = POS+1
C
160 CONTINUE
C ----- CHECK AND EXECUTE COMMAND -----
 IF(CMD.EQ.'FC') GOTO 200
 IF(CMD.EQ.'FO') GOTO 204
 IF(CMD.EQ.'SO') GOTO 208
 IF(CMD.EQ.'RN') GOTO 212
 IF(CMD.EQ.'SN') GOTO 216
 IF(CMD.EQ.'RL') GOTO 220
 IF(CMD.EQ.'IT') GOTO 224
 IF(CMD.EQ.'RA') GOTO 228
 IF(CMD.EQ.'S1') GOTO 232
 IF(CMD.EQ.'SU') GOTO 232
 IF(CMD.EQ.'E ') GOTO 260
 IF(CMD.EQ.'EX') GOTO 260
 IF(CMD.EQ.'DP') GOTO 280
 IF(CMD.EQ.'DR') GOTO 288
 IF(CMD.EQ.'ST') GOTO 304
 IF(CMD.EQ.'FF') GOTO 312

C
180 FORMAT(' STATEMENT UNKNOWN: ',A2)
 WRITE(2,180) CMD
 GOTO 300

C
200 FCOD = STRING(1: 6)
 GOTO 300

204 FOPT = STRING(1: 6)
 GOTO 300

208 SOPT = STRING(1:12)
 GOTO 300

```

```

212 RECN = STRING(1: 8)
 GOTO 300
216 SETN = STRING(1:80)
 GOTO 300
220 RLMN = STRING(1:80)
 GOTO 300
224 ITMN = STRING(1:80)
 GOTO 300
228 RECA = STRING(1:80)
 GOTO 300
232 SPP1 = STRING(1:30)
 GOTO 300
260 CONTINUE
C _____ CALL A DML-STATEMENT _____
 CALL DML (FCOD,FOPT,SOPT,UINF,RECN,SETN,
F RLMN,ITMN,RECA,SPP1)

264 FORMAT(' DBSTATUS: ',A5)
 IF (DBSTATUS.NE.'00000') WRITE(2,264) DBSTATUS
 GOTO 300
278 FORMAT (' FCOD:',A6,' FOPT:',A6,' SOPT:',A8,
F ' RECN:',A8,' SETN:',A8,' RLMN:',A8,
F ' ITMN:',A8,' RECA:',A8,' SPP1:',A8,
280 WRITE(2,278) FCOD,FOPT,SOPT(1:8),RECN,SETN(1:8),RLMN(1:8),
F ITMN(1:8),RECA(1:8),SPP1(1:8),SPP2(1:8),
 GOTO 300

288 WRITE(2,22) RECA
C
300 IF (POS.GE.80) GO TO 30
 GOTO 80
304 CONTINUE
 STOP
C
C _____ PRODUCE A P-ERROR _____
312 POS = POS/(POS-POS)
 GOTO 30
 END

```



**Assembler:**

If users use the Assembler programming language, they must program the CALL DML parameter list themselves, using A-type constants. The DSCAL, DSCAP, DSCDF and DSCPA macros can be used for this purpose (see [section “CALL DML Assembler macros” on page 259](#)).

```

ASSDML START
 BALR 12,0
 USING *,12
 PRINT NOGEN
 SPACE
 MVC UINF+120(6),='USINF*'
 SPACE
READ RDATA LINEL,ERROR,84
 XR 4,4
 LH 4,LINEL
 SH 4,=H'4'
 LA 3,LINE
 LA 4,0(3,4) R4 AT END OF INPUT
 MVI 0(4),' '
NEXT DS 0H
BLANKS DS 0H IGNORE BLANKS
 CR 3,4
 BNL READ
 CLI 0(3),' '
 BNE MOVCMO
 LA 3,1(3)
 B BLANKS
 SPACE
MOVCMO MVC CMD,0(3) GET COMMAND
 LA 3,1(3)
BLANK2 LA 3,1(3) IGNORE BLANKS
 CR 3,4
 BH EXECMO
 CLI 0(3),' '
 BE BLANK2
 CLI 0(3),''''
 BNE EXECMO
MSTRING DS 0H READ A STRING
 MVC STRING,STRING-1
 LA 5,STRING
 LA 3,1(3)
MSTR1 CR 3,4
 BNL EXECMO
 CLI 0(3),''''
 BE MSTREND

```

```

MVC 0(1,5),0(3)
LA 3,1(3)
LA 5,1(5)
B MSTR1
MSTREND LA 3,1(3)
SPACE
EXECMD DS OH CHECK AND EXECUTE COMMAND
CLC CMD,='FC'
BE PAR1
CLC CMD,='FO'
BE PAR2
CLC CMD,='SO'
BE PAR3
CLC CMD,='RN'
BE PAR5
CLC CMD,='SN'
BE PAR6
CLC CMD,='RL'
BE PAR7
CLC CMD,='IT'
BE PAR8
CLC CMD,='RA'
BE PAR9
CLC CMD,='S1'
BE PAR10
CLC CMD,='SU'
BE PAR11
CLC CMD,='E '
BE EXDML
CLC CMD,='EX'
BE EXDML
CLC CMD,='DP'
BE DISPAR
CLC CMD,='DR'
BE DISREC
CLC CMD,='ST'
BE STOP
WROUT UNKNOWN,ERROR MESSAGE IF COMMAND IS WRONG
B NEXT
SPACE
PAR1 MVC FCOD,STRING
B NEXT
PAR2 MVC FOPT,STRING
B NEXT
PAR3 MVC SOPT,STRING
B NEXT
PAR5 MVC REC�,STRING
B NEXT

```

```

PAR6 MVC SETN,STRING
 B NEXT
PAR7 MVC RLMN,STRING
 B NEXT
PAR8 MVC ITMN,STRING
 B NEXT
PAR9 MVC RECA(80),STRING
 B NEXT
PAR10 MVC SPP1,STRING
 B NEXT
EXDML DS OH CALL A DML-STATEMENT
 DSCAP PARAM=PARADR
 SPACE
 CLC DBSTATUS(5),='00000'
 BE NEXT
 WROUT MESSTAL,ERROR
 SPACE
 B NEXT
DISPAR DS OH
 MVC POUT1,FCOD
 MVC POUT2,FOPT
 MVC POUT3,SOPT
 MVC POUT5,RECN
 MVC POUT6,SETN
 MVC POUT7,RLMN
 MVC POUT8,ITMN
 MVC POUT9,RECA
 MVC POUT10,SPP1
 WROUT PAROUT,ERROR
 B NEXT
DISREC WROUT RECAOUT,ERROR
 B NEXT
 SPACE
STOP DS OH
ERROR DS OH
 TERM
 SPACE
MESSTAL DC X'0018404040'
 DC C'DBSTATUS: '
MESSTA DS CL5
 DC C' '
 DS OH
LINEL DC C' '
LINE DS CL80
UNKNOWN DC X'0050404040'
 DC C'ANWEISUNG UNBEKANNT : '
CMD DS CL2

```

```

 DC C' '
STRING DS CL80
 SPACE
PARADR DSCDF FCOD,FOPT,SOPT,UINF,RECN,SETN,RLMN,ITMN,RECA,SPP1, C
 SUFFIX=ADR DEFINITION OF OPERAND LIST
 SPACE
FCOD DC CL6'READYC'
FOPT DC CL6'ALLUPD'
SOPT DC CL20' '
RECN DS CL8
SETN DS CL80
RLMN DS CL80
ITMN DS CL160
RECAOUT DC X'002D404040' WROUT FIRST 40 BYTES OF RECA
RECA DS CL800
SPP1 DS CL30
 SPACE
UINF DS OF UDS USER INFORMATION AREA
 DSCPA USERINF
DBSTATUS EQU USREDMCO DEFINED IN MACRO DSCPA
 SPACE
PAROUT DC Y(PAROUTEN-PAROUT)
 DC X'404040'
 DC C' FCOD:'
POUT1 DC C' '
 DC C' FOPT:'
POUT2 DC C' '
 DC C' SOPT:'
POUT3 DS CL8
 DC C' RECN:'
POUT5 DS CL8
 DC C' SETN:'
POUT6 DS CL8
 DC C' RLMN:'
POUT7 DS CL8
 DC C' ITMN:'
POUT8 DS CL8
 DC C' RECA:'
POUT9 DS CL8
 DC C' SPP1:'
POUT10 DS CL8
PAROUTEN EQU *
 LTORG
 DS OF
 DC C'***** END OF ASSDML *****'
 END

```

---

## 9 Testing DML functions using DMLTEST

The DMLTEST program enables the user to

- test individual DML functions interactively
- run test procedures
- access any database configuration
- interoperate with KDBS.

The following chapter comprises

- an introduction
- the commands of the DMLTEST program
- DML statements under DMLTEST
- DMLTEST execution
- error messages

## 9.1 Introduction

The DMLTEST program offers the following options:

- behavior in compliance with environment
- mostly unformatted input
- for CALL interfaces, the possibility to explicitly address each byte of the CALL parameter list individually
- independence from the processed databases
- independence from UDS variants. All UDS-specific components are dynamically loaded. Link-editing is not required.
- storing of items with symbolic names
- a number of convenient commands for the output of results and for execution control
- processing of predefined command sequences (either defined temporarily or stored in system files)
- any number of user connections
- the SYSTEM command (interrupt program and switch to system mode)
- the TRACE command (TRACE of DMLTEST commands)
- DMLTEST supports the CALL8 as well as the CALL30 interface of CALL DML
- connection to EDT
- DMLTEST is XS-compatible

### Standard functions

Unless specified otherwise by the user, the program runs with the linked-in DBH and COBOL DML with the CALL8 interface.

Internally, DMLTEST uses the CALL DML interface. The scope of DML functions corresponds to that of CALL DML (see the [section “Special features of CALL DML” on page 92](#)).

**SYSDTA assignment**

If SYSDTA has been assigned to a terminal, DML assumes that the user wishes to work interactively. He or she then has the option of responding to interrupts directly.

If SYSDTA has been assigned to a file, DMLTEST assumes that the user does not want to work interactively, i.e. would prefer automatic operation (procedure or batch job). No direct user responses to interrupts are possible in this case. Automatic processing is continued until one of the following events is detected:

- a HALT command
- EOF
- an error occurs
- SYSDTA is assigned to the terminal.

If an error occurs, DMLTEST will issue a message and, if a call to the UDS/SQL DBH has already occurred, will generate the language-specific statement FINISH WITH CANCEL and then terminate.

## Command sequence for starting DMLTEST

The DMLTEST program is a CALL DML application that works together with either the linked-in DBH (default) or the independent DBH (see the DMLTEST command DBH on [page 316](#)). DMLTEST is started with the START-UDS-DMLTEST command (alias name: DMLTEST). Which commands are required before the actual DMLTEST start depends on the DBH variant. Their sequence is then the same as the command sequence required before CALL DML application programs are called.

### 1. Linked-in DBH

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=configuration-name
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-dynamic-loading-library
[/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=filename] _____ (1)
/CREATE-FILE FILE-NAME=confname.DBSTAT,SUPPRESS-ERR=*FILE-EXISTING
/CREATE-FILE FILE-NAME=confname.DBSTAT.SAVE,SUPPRESS-ERR=*FILE-EXISTING
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/START-UDS-DMLTEST
```

The command sequence is the same as when starting linked-in CALL DML application programs (see the "[Database Operation](#)" manual, section "Commands for starting DBH").

- (1) The link name PPFILE is used to assign a file from which the DBH is to read in the load parameters. The load parameters can also be in a PLAM library element or in a procedure file. The assignment commands required for this can be found in the "[Database Operation](#)" manual, section "Commands for starting DBH".

### 2. Independent DBH

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=configuration-name
/ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-dynamic-loading-library
/START-UDS-DMLTEST
```

The command sequence is the same as when starting CALL DML application programs which use the independent DBH (see [section "Starting a COBOL program" on page 103ff](#)).



## 9.2 DMLTEST commands

### Overview of the DMLTEST commands and general rules

| Command                                                                                                                                                          | Function                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <u>A</u> DD <i>name,value[,condition]</i>                                                                                                                        | Add                                                                   |
| <u>C</u> ONTINUE                                                                                                                                                 | Resume processing after interrupt                                     |
| <u>DBH</u> { <u>I</u> NDEPENDENT<br><u>I</u> NLINKED }                                                                                                           | Select DBH variant<br>Default value: INLINKED                         |
| <u>D</u> ECLARE }<br><u>D</u> CL } <i>name[,length]</i>                                                                                                          | Define an item in the work area                                       |
| <u>D</u> EFINE { <u>R</u> CODE<br><u>R</u> ECORD } , <i>name[,distance][,length]</i><br><i>parameter</i>                                                         | Define an item in the specified area                                  |
| <u>D</u> ELETE <i>name[,condition]</i>                                                                                                                           | Delete a procedure or an item                                         |
| <u>D</u> ISPLAY { <u>R</u> ECA<br><u>R</u> ECORD } [, <i>distance</i> ][, <i>length</i> ][, <i>form</i> ]<br><u>R</u> CODE<br><u>T</u> IME [, <i>condition</i> ] | Output the specified area or value to SYSOUT after each DML statement |
| <u>D</u> ISPOFF } [ { <u>R</u> ECA<br><u>R</u> ECORD } ] [, <i>condition</i> ]<br><u>D</u> OFF } [ { <u>R</u> CODE<br><u>T</u> IME } ] [, <i>condition</i> ]     | Deactivate DISPLAY function                                           |
| <u>D</u> O <i>name[,repetition][,condition]</i>                                                                                                                  | Start a procedure                                                     |
| <u>E</u> DT                                                                                                                                                      | Call EDT                                                              |
| <u>E</u> ND                                                                                                                                                      | Terminate procedure definition                                        |
| <u>E</u> SCAPE[ <i>condition</i> ]                                                                                                                               | Terminate interrupt, or abort procedure                               |
| <u>E</u> XECUTE[ <i>repetition</i> ][, <i>condition</i> ]                                                                                                        | Execute DML statement                                                 |
| { <u>H</u> ALT }<br>{ <u>S</u> TOP } [ <i>condition</i> ]                                                                                                        | Terminate the DMLTEST program                                         |

Table 36: Overview of DMLTEST commands

(part 1 of 3)

| Command                                                                                                                                                                             | Function                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| <u>H</u> ELP[ <i>condition</i> ]                                                                                                                                                    | Request information on preceding command, or request preceding DML statement |
| <u>L</u> ANGUAGE {<br>{<br><u>C</u> DML<br><u>C</u> DML30<br><u>C</u> OBOL<br><u>C</u> OBOL30<br><u>K</u> DBS<br><u>K</u> KDS<br><u>K</u> LDS<br>}<br>}                             | Select data manipulation language<br><br>Default value: COBOL                |
| <u>L</u> EAVE[ <i>condition</i> ]                                                                                                                                                   | Abort a procedure call                                                       |
| <u>L</u> IST } {<br><u>D</u> CL } [ , <i>name</i> ]<br><u>D</u> EF }<br><u>P</u> ROC }<br>} <i>command-name</i><br>} <i>declaration</i><br>} <i>definition</i><br>} <i>procname</i> | Output specified information to SYSOUT                                       |
| <u>M</u> OVE {<br><u>R</u> ECORD<br><u>R</u> CODE } , <i>value</i> [ , <i>distance</i> ][ , <i>condition</i> ]<br>} <i>parameter</i><br>} <i>definition</i><br>} <i>declaration</i> | Enter values in the specified areas                                          |
| <u>N</u> EXT                                                                                                                                                                        | Respond to interrupts, or abort current command                              |
| <u>P</u> ERFORM <i>name</i> [ , <i>filename</i> ][ , <i>condition</i> ]                                                                                                             | Call a module                                                                |
| <u>P</u> RINT {<br><u>R</u> ECORD<br><u>R</u> CODE } [ , <i>distance</i> ][ , <i>length</i> ][ , <i>form</i> ][ , <i>condition</i> ]<br><u>T</u> ALLY<br><u>T</u> IME }             | Output specified area or value to SYSLST after each DML statement            |
| <u>P</u> ROC <i>procname</i>                                                                                                                                                        | Open a procedure definition                                                  |
| <u>P</u> ROFF } [ {<br><u>R</u> ECORD } ] [ , <i>condition</i> ]<br><u>P</u> OFF } {<br><u>R</u> CODE }<br><u>T</u> IME }                                                           | Deactivate PRINT function                                                    |
| <u>P</u> ROT {<br><u>O</u> N } ] [ , <i>condition</i> ]<br><u>O</u> FF }<br><u>O</u> UT }                                                                                           | Activate or deactivate logging function<br>Default value: PROT ON            |
| {<br><u>R</u> EMARK } <i>literal</i><br>* }                                                                                                                                         | Insert comment lines                                                         |

Table 36: Overview of DMLTEST commands

(part 2 of 3)

| Command                                                                                                                                                                                                                                                                                                                | Function                                                                               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <u>R</u> N <i>filename</i> [, <i>repetition</i> ][, <i>condition</i> ]                                                                                                                                                                                                                                                 | Start a sequence of commands or statements stored in an ISAM file.                     |
| <u>S</u> ET[ <i>parameter</i> ( { $\begin{matrix} n \\ X' n' \end{matrix}$ } )]= <i>value</i> , ...                                                                                                                                                                                                                    | Enter values in the CALL DML parameter list                                            |
| <u>S</u> HOW { $\begin{matrix} \text{RCODE} \\ \text{TALLY} \\ \text{RECORD} \\ \text{PARAM}[ ,name] \\ \text{DCL} \\ \text{DEF} \\ \text{PROC} \end{matrix}$ } , <i>name</i> [ , <i>distance</i> ] [ , <i>length</i> ] [ , <i>form</i> ] } [ , <i>condition</i> ]<br><i>parameter declaration definition procname</i> | Output the specified area to SYSOUT in the specified format                            |
| <u>S</u> UBSCHEMA IS <i>subschema</i>                                                                                                                                                                                                                                                                                  | Select subschema                                                                       |
| <u>S</u> YSTEM[ <i>condition</i> ]                                                                                                                                                                                                                                                                                     | Go to system mode                                                                      |
| <u>T</u> RACE { $\begin{matrix} \text{ON} \\ \text{OFF} \end{matrix}$ } [ , <i>repetition</i> ][, <i>condition</i> ]                                                                                                                                                                                                   | Log commands and statements on the screen during processing<br>Default value: TRACE ON |
| <u>W</u> AIT[ <i>condition</i> ]                                                                                                                                                                                                                                                                                       | Effect an interrupt                                                                    |

Table 36: Overview of DMLTEST commands

(part 3 of 3)

The DMLTEST commands can be

- entered interactively
- stored in temporary DMLTEST procedures
- stored in BS2000 procedure files
- stored in BS2000 Enter files
- stored in BS2000 ISAM system files

The user may enter any number of commands at a time, provided the following input length is not exceeded:

The contents of 1 screen and 256 bytes in the case of an interrupt

*Example*

```

1) { /SET-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=SHIPPING
 { /ADD-FILE-LINK LINK-NAME=$UDSSSI, FILE-NAME=LMS.SSITAB
 { /ADD-FILE-LINK LINK-NAME=PPFILE, FILE-NAME=UDSDBB.PP.FILE
 { /CREATE-FILE FILE-NAME=SHIPPING.DBSTAT, SUPPRESS-ERRORS=*FILE-EXISTING
 { /CREATE-FILE FILE-NAME=SHIPPING.DBSTAT.SAVE, SUPPRESS-ERRORS=*FILE-EXISTING
 { /ASSIGN-SYSDTA TO-FILE=*SYSCM7
 { /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.8A00
 { /START-UDS-DMLTEST

2) LANG COB
3) DISPLAY RCODE, COND=RCODE NE C'00000'
4) PROT ON
5) DISPLAY RECA, L=80
6) SUBSCHEMA IS ADMIN
7) READY
8) E

% UDS0215 UDS STARTING UDS/SQL V2.8 (LINKED-IN), DATE=2015-06-28
(ILL2038,11:41:01/OYBG)
% UDS0746 UDS PUBSET DECLARATION (CURRENT) FOLLOWS
(ILL1746,11:41:01/OYBG)
OYBG: UDS-PUBSET-JV: :SQL2:$XXXXXXXXX.PUBSDECL.ALL
OYBG: PUBSETS: *
OYBG: DEFAULT PUBSET: SQL2
OYBG: -----
% UDS0722 UDS ORDER ADD RLOG 150628094100 IN EXECUTION
(ILL1283,11:41:01/OYBG)
% UDS0354 UDS ALOG CHECKPOINT FOR SHIPPING
(ILL1307,11:41:01/OYBG)
OYBG: ALOG-CKPT OMITTED: DB WITHOUT ALOG-LOGGING.
OYBG: MAXDB = 1
OYBG: TRANSACTION = 1
OYBG: SUBSCHEMA = 1
OYBG: 2KB-BUFFER-SIZE= 1
OYBG: 4KB-BUFFER-SIZE= 1
OYBG: 8KB-BUFFER-SIZE= 0
OYBG: LOG = PUBLIC
OYBG: LOG-2 = NO
OYBG: LOG-SIZE = (192, 192)
OYBG: RESERVE = NONE
OYBG: WARMSTART = STD
OYBG: CONSOLE = NO
OYBG: STDCKPT = YES
OYBG: LOCK = STD
OYBG: PRIVACY-CHECK = OFF
OYBG: CONFNAME = $XXXXXXXXX.SHIPPING
OYBG: DATABASES OF CONFIGURATION:

```

```

OYBG: $XXXXXXXX.SHIPPING ,EXCLUSIVE-UPD
,*SYSTEM
% UDS0356 UDS EXECUTION OF ORDERS FOR SHIPPING TERMINATED
(ILL1309,11:41:01/OYBG)
RECORD - AREA :
.....

9) SYS
% IDA0199 PROGRAM BREAK AT ADDRESS X'02B052', AMODE=24
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=*TIME(TIMEOUT=*STD)
10) /RESUME-PROGRAM
11) EDT
12) @RET
 EDT ENDED. DMLTEST CONTINUES
13) RUN DMLTEST.EXAMPLE1
 .
 .
 .
DB-PARAMS (FIRST 8B)
FCOD :ACCPCT.. FOPT :DB-KEY.. SOPT : UINF : RECN :ARTICLE
SETN : RLMN : ITMN : RECA :..... SPP1 :ADMIN
SPP2 :..... SPP3 : SUBS :ADMIN
.
.
.
DB-PARAMS (FIRST 8B)
FCOD :CONNEC.. FOPT :TO-SET.. SOPT : UINF :CLOTHING RECN :ARTICLE
DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.

14) ESC
15) FINISH;E

16) HALT
 DMLTEST NORMAL TERMINATION
 % UDS0354 UDS ALOG CHECKPOINT FOR SHIPPING (ILL1307,11:40:57/OYBG)
 OYBG: ALOG-CKPT 20150628094106: FIXED (ALOG-NR:000000005, START-CKPT:
 20150628094105).
 % UDS0758 NUMBER OF DML-STATEMENTS AND I/O COUNTERS PER DATABASE
 (ILL1758,11:40:57/OYBG)
 OYBG: DATABASE NAME DMLS LOG READ PHYS READ LOG WRITE
 PHYS WRITE
 OYBG: -----

 OYBG: SHIPPING 76 338 103 91
 63
 % UDS0213 UDS NORMAL SYSTEM TERMINATION WITH *****76 DML-
 STATEMENTS
 2015-06-28 (ILLY033,11:40:57/OYBG)

```

- 1) See "Command sequence for starting DMLTEST" ([page 304](#))
- 2) Define application language
- 3) Define in which case the DATABASE-STATUS is to be output
- 4) Activate logging
- 5) Define that RECA is to be output after each DML statement
- 6) Assign subschema
- 7) Set up READY statement
- 8) Execute READY statement
- 9) Transfer control to operating system
- 10) Transfer control to DMLTEST
- 11) Call EDT as subroutine
- 12) Transfer control to DMLTEST
- 13) Call DMLTEST procedure
- 14) Abort DMLTEST procedure
- 15) Terminate transaction
- 16) Terminate DMLTEST program

### General rules

*variable* must be replaced by a currently valid value when a format is used. The following categories of variables can be distinguished:

|                                                                                   |                                                                                                                                                          |
|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>definition</i><br><i>declaration</i><br><i>command-name</i><br><i>procname</i> | These names must be defined in the corresponding DMLTEST commands.                                                                                       |
| <i>literal</i>                                                                    | May be composed of any characters. If the literal is not alphanumeric or is alphanumeric, but contains blanks, it must be enclosed within single quotes. |

Table 37: DMLTEST variables

The following keyword parameters are used in the formats of the DMLTEST commands:

| Keyword parameter                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Default value                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| $value: [\underline{VAL}]= \left\{ \begin{array}{l} literal \\ [n1] \left\{ \begin{array}{l} C \\ X \\ P \end{array} \right\} [Ln2] \cdot literal' \\ 'literal' \end{array} \right\}$                                                                                                                                                                                                                                                                                 | -                                   |
| $distance: [\underline{DIST}]= \left\{ \begin{array}{l} n \\ X \cdot n' \end{array} \right\}$                                                                                                                                                                                                                                                                                                                                                                         | D=0                                 |
| $length: [\underline{LNG}]= \left\{ \begin{array}{l} n \\ X \cdot n' \end{array} \right\}$                                                                                                                                                                                                                                                                                                                                                                            | L=8                                 |
| name: $[\underline{NAME}]= literal$<br>maximum length: 20<br>allowed characters: 1st position A-Z<br>from 2nd position A-Z,-,0-9                                                                                                                                                                                                                                                                                                                                      | -                                   |
| $filename: \left[ \left\{ \begin{array}{l} \underline{OML} \\ \underline{FILE} \end{array} \right\} \right] literal$                                                                                                                                                                                                                                                                                                                                                  | OML=<br>DMLTEST.MODLIB for PERFORM  |
| $repetition: \left[ \left\{ \begin{array}{l} \underline{REP} \\ \underline{STEP} \end{array} \right\} \right] \left\{ \begin{array}{l} n \\ X \cdot n' \end{array} \right\}$                                                                                                                                                                                                                                                                                          | 4 for TRACE<br>1 in all other cases |
| $form: [\underline{FORM}]= \left\{ \begin{array}{l} C \\ X \\ D \end{array} \right\}$                                                                                                                                                                                                                                                                                                                                                                                 | F=C                                 |
| $condition: \left[ \left\{ \begin{array}{l} \underline{CASE} \\ \underline{COND} \end{array} \right\} \right] \left\{ \begin{array}{l} \underline{RECORD} \left\{ \begin{array}{l} (n) \\ (X \cdot n') \end{array} \right\} \\ \underline{TIME} \\ \underline{RCODE} \\ definition \end{array} \right\} \left\{ \begin{array}{l} \underline{EQ} \\ \underline{NE} \\ \underline{LT} \\ \underline{GT} \\ \underline{LE} \\ \underline{GE} \end{array} \right\} value$ | -                                   |

Table 38: Keyword parameters with default values

*definition* must be enclosed in single quotes.

- C alphanumeric representation
- X hexadecimal representation
- D dump format
- P packed representation
- n* integer
- n1* multiplication factor
- n2* length of literal

The *parameter* variable in the DMLTEST commands can be replaced by the following values (see also [table 23 on page 201](#)).

|    | for CDML         | for KDBS         |
|----|------------------|------------------|
| 1  | FCOD             | OP               |
| 2  | FOPT             | RE               |
| 3  | SOPT             | DB               |
| 4  | UINF             | AR               |
| 5  | RECN             | FS               |
| 6  | SETN             | SI               |
| 7  | RLMN             | KB               |
| 8  | ITMN             | KE               |
| 9  | RECA             | RT               |
| 10 | SPP1             | ST               |
| 11 | SPP2             | FSI <sup>2</sup> |
| 12 | SPP3             | –                |
|    | SEX <sup>1</sup> | –                |
|    | SUBS             | –                |

Table 39: Values for CDML and KDBS

- <sup>1</sup> Redefine ITMN: This parameter must be used if the user requires a search expression in DMLTEST syntax to be edited into CALL DML syntax (see the SET command on [page 334](#)).
- <sup>2</sup> Redefine SI: This parameter must be used if the user requires a search expression in DMLTEST syntax to be edited into KDBS syntax (see the SET command on [page 334](#)).

Any number of commands may be written in a sequence. The separator is “;”.



## Predefined items

The items described below are predefined and can be addressed with the DMLTEST commands LIST, SHOW, MOVE and ADD.

The following items are always defined:

|        |                                                                  |
|--------|------------------------------------------------------------------|
| RCODE  | contains statement code and status code                          |
| TALLY  | contains the counter for TALLYING                                |
| RECORD | contains the RECORD AREA                                         |
| TIME   | contains the processing time of the statement (in milli-seconds) |

The following items are defined when COBOL DML is used:

|                 |                                                                                                                                                                                                                                      |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AREA-ID         | contains the realm name for ACCEPT-2.<br>The AREA-ID item is a redefinition of the area for the realm name in the “System Communication Locations” section of the user information area (see <a href="#">table 24 on page 205</a> ). |
| DB-KEY          | contains the DATABASE-KEY value for ACCEPT-1, ACCEPT-2, FIND/FETCH-1.<br>The DB-KEY item is a redefinition of the DATABASE-KEY item of the user information area (see <a href="#">table 24 on page 205</a> ).                        |
| DB-KEY-LONG     | contains the DATABASE-KEY-LONG value for ACCEPT-1, ACCEPT-2, FIND/FETCH-1.<br>The DB-KEY-LONG item is a redefinition of the DATABASE-KEY-LONG item of the user information area (see <a href="#">table 24 on page 205</a> ).         |
| IMP-AREA-ID     | contains the realm name for FIND/FETCH-2 and STORE...IMP or STORE...IMP-LONG.<br>The IMP-AREA-ID item is a redefinition of the area comprising bytes 5...34 of special parameter 2 (SPP2; see <a href="#">page 210</a> ).            |
| IMP-DB-KEY      | contains the DATABASE-KEY value for STORE...IMP.<br>The DB-KEY item is a redefinition of the area comprising bytes 1...4 of special parameter 2 (SPP2; see <a href="#">page 210</a> ).                                               |
| IMP-DB-KEY-LONG | contains the DATABASE-KEY-LONG value for STORE...IMP-LONG.<br>The DB-KEY-LONG item is a redefinition of the area comprising bytes 35...42 of special parameter 2 (SPP2; see <a href="#">page 210</a> ).                              |

When KDBS is used, the following items are defined:

SC     contains the protection code

VA     contains the processing type

The SC and VA items are redefinitions of the RE communication area.

## ADD

The ADD command is used to add in the specified area.

---

```
ADD name,value[,condition]
```

---

*name* Must have been defined using the DEFINE or DECLARE command.

Using DECLARE:

The type is defined by a previous MOVE command.

Using DEFINE:

The type is defined by the description of the redefined area.

*name* contains the result of the addition after execution.

The kind of addition performed is determined by the type of *value* specified, which must match the type specified for *name*:

- P: packed
- C: unpacked
- X: hexadecimal

The maximum length for *name* is:

- for unpacked representation: 16
- for packed representation: 10
- for hexadecimal representation: 4

*value* Must be of the same type as the content of *name*.

### Examples

```
*** (IN) : REMARK ***ADD***
*** (IN) : DEF RECORD, ITEM2, L=4
*** (IN) : SH ITEM2, F=X
 (OUT): ITEM2 :00000000
*** (IN) : ADD ITEM2, X'F5'
*** (IN) : SH ITEM2, F=X
 (OUT): ITEM2 :000000F5

*** (IN) : DEC ITEM3, L=8
*** (IN) : MOVE ITEM3, 8
*** (IN) : SH ITEM3
 (OUT): ITEM3 :8
*** (IN) : ADD ITEM3, 9
*** (IN) : SH ITEM3
 (OUT): ITEM3 :17
```

## CONTINUE

The CONTINUE command is used to resume processing after an interrupt. If no interrupt has occurred, the command has no effect.

---

CONTINUE

---

## DBH

The DBH command is used to select the DBH variant.

---

DBH { INDEPENDENT  
          INLINKED }

---

This command must be entered prior to the first SET or DML command. The default value is INLINKED.

## DECLARE

The DECLARE command is used to define an item in the work area.

---

DECLARE }  
DCL        } *name[, length]*

---

*name*    Must be unique and may only occur once in each program run.

The item type is not qualified. It is defined by a MOVE command. Each further MOVE command can change it again.

### Example

```

(IN): REMARK ***DECLARE
***(IN): DEC ITEM3,L=8
***(IN): MOVE ITEM3,8
***(IN): SH ITEM3
 (OUT): ITEM3 :8

```

## DEFINE

The DEFINE command is used to define an item in the specified area.

---

```
DEFINE { RCODE
 RECORD
 parameter } ,name[,distance][,length]
```

---

*name* Must be unique and may only occur once in each program run.

DEFINE can be used to redefine areas that are frequently referenced in order to avoid having to respecify the displacement and length repeatedly.

An area can be redefined more than once.

### Example

```
(IN): REMARK ***DEFINE
***(IN): DEFINE RECORD, FIRSTNAME, L=10
***(IN): DEFINE RECORD, LASTNAME, D=10, L=15
***(IN): DEFINE RECORD, DATEOFBIRTH, D=25, L=10
***(IN): DEFINE RECORD, NAMES, L=25
***(IN): MOVE FIRSTNAME, ANTON
***(IN): MOVE LASTNAME, MAIER
***(IN): MOVE DATEOFBIRTH, '12.12.50'
***(IN): SH RECORD, L=35
 (OUT): RECORD - AREA :ANTON....MAIER.....12.12.50.....
***(IN): MOVE NAMES, ALBERT
***(IN): SH RECORD, L=35
 (OUT): RECORD - AREA :ALBERT....MAIER.....12.12.50.....
***(IN): SH FIRSTNAME
 (OUT): FIRSTNAME :ALBERT..
***(IN): SH LASTNAME
 (OUT): LASTNAME :MAIER...
***(IN): SH DATEOFBIRTH
 (OUT): DATEOFBIRTH :12.12.50
***(IN): SH NAMES, L=25
 (OUT): NAMES :ALBERT....MAIER.....
```

## DELETE

The DELETE command is used to delete a procedure, definition or declaration.

---

```
DELETE name[,condition]
```

---

## DISPLAY

The DISPLAY command is used to output the specified area or value on the screen following each DML statement if the specified condition is satisfied.

---


$$\text{DISPLAY } \left\{ \begin{array}{l} \text{RECA} \\ \text{RECORD} \\ \text{RCODE} \\ \text{TIME} \end{array} \right\} [, \textit{distance}] [, \textit{length}] [, \textit{form}] [, \textit{condition}]$$


---

The following formats are available for output:

FORM=C alphanumeric characters (default)  
 FORM=X hexadecimal  
 FORM=D both C and X (dump format)

If RCODE and TIME are specified, *distance*, *length* and *form* specifications are ignored.

The contents of non-alphanumeric items can only be interpreted in hexadecimal format. This also applies for the contents of national items (Unicode: UTF-16, PICTURE N, USAGE NATIONAL).

## DISPOFF

The DISPOFF command deactivates the specified or all previously valid DISPLAY functions. If a condition is specified, this condition must be met.

---


$$\left. \begin{array}{l} \text{DISPOFF} \\ \text{DOFF} \end{array} \right\} \left[ \begin{array}{l} \text{RECA} \\ \text{RECORD} \\ \text{RCODE} \\ \text{TIME} \end{array} \right] [, \textit{condition}]$$


---

## DO

The DO command is used to start a DMLTEST procedure.

---

`DO name[,repetition][,condition]`

---

*name* Must have been defined in the same program run.

*repetition*

Specifies how many times the procedure is to be run (repetition factor).

*condition*

Specifies the condition under which the procedure is to be run. This condition is only evaluated once before the first procedure run even if a repetition factor was specified.

The DO command can also be entered in an interrupt state, but will not be executed so long as the interrupt state prevails.

Up to 20 procedures can be processed within the same DMLTEST run. The maximum nesting level is 20 procedures.

It is possible to program DO-WHILE loops.

```
01) PROC proc1;
02) LEAVE condition;
 .
03) .
 .
04) END
```

01) Open definition of *proc1*

02) The first command stored in *proc1* is a LEAVE command with a condition.

03) Enter further procedure commands and statements

04) Terminate definition of *proc1*

The command

```
DO proc1,W=50000
```

initiates processing of *proc1*.

The procedure is run until one of the following events occur:

- The *condition* in the stored LEAVE command is satisfied.
- The procedure has been run 50 000 times.

*Example*

```
DO STORE-A,R=20,COND=RCODE EQ C'00000' ;
```

DMLTEST first checks if the DB status is equal to C'00000'. If this is the case, procedure STORE-A is run 20 times (provided it was defined).

Sequence:

|                               |   |
|-------------------------------|---|
| I = 20 (repetition factor)    |   |
| DB status equal to C'00000' ? |   |
| Y                             | N |
| As long as I > 0              |   |
| Run procedure STORE-A         |   |
| I = I - 1                     |   |
| Execute next command          |   |



## EDT

The EDT command is used to call the EDT file editor as subroutine.



The call takes place in L mode, which is not Unicode-capable.

---

EDT

---

@RETURN is used to return to the DMLTEST dialog.

## END

The END command is used to conclude procedure definition.

---

END

---

## ESCAPE

The ESCAPE command is used to terminate all active procedures, command sequences and interactive entries; it also terminates an interrupt. After an ESCAPE command, the next command is expected from SYSDTA.

---

ESCAPE[ *condition*]

---

### *condition*

Specifies the condition which has to be satisfied for the procedure(s) and command sequence(s) to be terminated.

*Example*

```

(IN): REMARK ***ESCAPE
***(IN): RUN DMLTEST.EXAMPLES1
 .
 .
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :CONNEC.. FOPT :TO-SET.. SOPT : UINF :CLOTHING RECN :ARTICLE
(OUT): DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.
***(IN): ESCAPE
***(IN): HELP
(OUT): LAST INPUT :ESCAPE
(OUT): DURING BREAK-STATUS STATUS IS NORMAL

```

**EXECUTE**

The EXECUTE command is used to initiate execution of a DML statement. The same DML statement can be repeated any number of times.

---

```
EXECUTE[repetition][, condition]
```

---

*repetition*

Specifies the number of times the command is to be executed (repetition factor).

*condition*

Specifies the condition under which the DML statement is to be executed.

**Sequence:**

|                             |       |
|-----------------------------|-------|
| I = Repetition factor       |       |
| As long as I > 0            |       |
| <i>condition</i> satisfied? |       |
| Y                           | N     |
| Issue DB call               | I = 0 |
| I = I - 1                   |       |
| Execute next command        |       |

If an EXECUTE command is entered during an interrupt, *repetition* and *condition* are ignored.

## HALT

The HALT command is used to terminate the DMLTEST run. Currently open transactions are “rolled back”.

---

```
{ HALT }
 { STOP } [condition]
```

---

### *condition*

Specifies the condition under which the DMLTEST run is to be terminated.

## HELP

The HELP command is used to have the following information output on the screen:

- the command preceding the HELP command
- the name of the input medium from which the command was entered
- the status (NORMAL or BREAK)
- an error description if applicable

---

```
HELP [condition]
```

---

### *condition*

Specifies the condition under which information to be output.

### *Example*

```
(IN): REMARK ***HELP
***(IN): DEF RECORD, LASTNAME, D=10, L=15
 (OUT): DMLTEST: USED NAME. STATUS IS NORMAL.
***(IN): HELP
 (OUT): LAST INPUT :DEF RECORD, LASTNAME, D=10, L=15
 (OUT): FROM SYSDTA STATUS IS NORMAL
 (OUT): ERROR DESCRIPTION :NAME IST SCHON FUER EINE DEFINITION VERGEBEN!
 (= name already assigned to definition)
```

## LANGUAGE

The LANGUAGE command is used to select the data manipulation language.

---

```

LANGUAGE {
 CDML
 CDML30
 COBOL
 COBOL30
 KDBS
 KKDS
 KLDS
}

```

---

CDML CALL DML; the CALL8 interface is used.

CDML30 CALL DML; the CALL30 interface is used.

KDBS KDBS (compatible database interface)

KKDS KDBS (compatible interface for the processing of complex data structures).

KLDS KDBS (compatible interface for the processing of linear data structures).

COBOL COBOL DML; the CALL8 interface is used internally.

COBOL30 COBOL DML; the CALL30 interface is used internally.

The default value is COBOL.

The LANGUAGE command causes parameter areas to be created and initialized as required. It can only be entered before the first SET command or the first DML statement. It is therefore not possible to switch between the CALL8 (CDML, COBOL) and CALL30 (CDML30, COBOL30) interfaces within the same DMLTEST run.

If LANGUAGE COBOL or LANGUAGE COBOL30 has been specified, DML statements may be entered in COBOL DML format, subject to certain restrictions. These COBOL DML statements are then translated into CALL DML by DMLTEST. Consequently, only the CALL DML functionality is available to the user. If LANGUAGE COBOL30 is set, full-length names (30 bytes) are passed.

The SET command can also be used with LANGUAGE COBOL or LANGUAGE COBOL30.

### Example

```

LANGUAGE
***(IN): LAN CDML
***(IN): DISP RCODE,COND=RCODE NE C'0000'
***(IN): DISP RECA,L=80
***(IN): SET FCOD=C'READYC',FOPT=C'ALLEUP',SUBS=C'ADMIN'
...

```

## LEAVE

The LEAVE command is used to terminate processing of a procedure or command sequence. Subsequently it passes control to the next higher level of procedure, command sequence etc., or to the terminal. It does not terminate an interrupt.

---

`LEAVE[ condition]`

---

### *condition*

Specifies the condition which has to be satisfied for a procedure or command sequence to be terminated.

If processing of a DO or RUN command is aborted by means of LEAVE, the associated procedure or command sequence is considered completed, regardless of any repetition factor.

### *Example*

```

(IN): REMARK ***LEAVE
***(IN): PROC TEST
***(IN): SH RECORD,L=40
***(IN): LEAVE
***(IN): SH RECORD,L=40
***(IN): END
***(IN): DO TEST
***(IN): SH RECORD,L=40
 (OUT): RECORD - AREA :ANTON....MAIER.....12.12.50.....
***(IN): LEAVE
***(IN): HELP
 (OUT): LAST INPUT :LEAVE
 (OUT): FROM : TEST :STATUS IS NORMAL
***(IN): CONTINUE
***(IN): HELP
***(OUT): LAST INPUT :CONTINUE
***(OUT): FROM SYSDTA :STATUS IS NORMAL

```

## LIST

The LIST command is used to output the specified information on the screen.

---

|      |   |   |              |            |   |            |
|------|---|---|--------------|------------|---|------------|
| LIST | } | { | {            | CMD        | } | [ , name ] |
|      |   |   | DCL          |            |   |            |
| LS   | } | { | {            | DEF        | } |            |
|      |   |   | PROC         |            |   |            |
|      |   |   | command-name |            |   |            |
|      |   |   | declaration  |            |   |            |
|      |   |   |              | definition |   |            |
|      |   |   |              | procname   |   |            |

---

### CMD, DEF, DCL, PROC

Can be combined with the desired *name*.

**CMD** If no name is specified, all command names with their abbreviations are output.

**DEF** If no name is specified, all definition names are output.

**DCL** If no name is specified, all declaration names are output.

### PROC

If no name is specified, all procedure names are output.

### Examples

```

(IN): REMARK ***LIST
***(IN): LIST DCL
(OUT): LIST OF DECLARATIONS
(OUT): NAME : ITEM1
(OUT): NAME : ITEM3
***(IN): LIST DEF
(OUT): LIST OF DEFINITIONS
(OUT): NAME : USERGROUP
(OUT): NAME : USERNAME
(OUT): NAME : PASSWORD
(OUT): NAME : DB-KEY
(OUT): NAME : AREA-ID
(OUT): NAME : IMP-DB-KEY
(OUT): NAME : IMP-AREA-ID
(OUT): NAME : ITEM2
(OUT): NAME : FIRSTNAME
(OUT): NAME : DATEOFBIRTH
(OUT): NAME : LASTNAME
***(IN): LIST DEC
(OUT): NAME : DECLARE (DEC)
(OUT): OTHER-NAME: DCL (DC)
(OUT): 1. OPERAND: <NAME>
(OUT): 2. OPERAND: <LENGTH>

```

## MOVE

The MOVE command is used to enter values in the record area, the CALL parameter list or items.

---

```

MOVE { RECORD
 RCODE
 parameter
 definition
 declaration
 } , value [, distance] [, condition]

```

---

### *parameter*

Specifies the desired CALL parameter.

### *declaration*

*distance* must not be specified.

The type assumed for *declaration* is dependent on the type of *value*. It is unaffected by the type which was valid in any previous *declaration*.

The type of *value* determines the type of transfer:

| Type:            | Transfer:                                                  |
|------------------|------------------------------------------------------------|
| alphanumeric     | left-justified, padded with blanks, truncated to the right |
| packed numeric   | right-justified, truncated to the left, padded with X' 00' |
| unpacked numeric | right-justified, truncated to the left, padded with 0      |
| hexadecimal      | left-justified, truncated to the right                     |

### *definition*

Is transferred left-justified; if *distance* is specified, starting at the corresponding displacement. If *value* is shorter than *definition*, it is not padded.

*value* The following types are available:

```

C'-123' alphanumeric
ABC1 alphanumeric
P'23' packed numeric
-12.3 unpacked numeric
X'12' hexadecimal

```

The MOVE command does not support national literals (Unicode: UTF-16).

RECORD, RCODE, *parameter*

Is transferred in the same way as *definition*.

*Examples*

```
(IN): REMARK ***MOVE
***(IN): MOVE ITEM3,8
***(IN): SH ITEM3
 (OUT): ITEM3 :8

***(IN): MOVE LASTNAME,MAIER
***(IN): SH LASTNAME
 (OUT): LASTNAME :MAIER...

***(IN): MOVE DATEOFBIRTH,'12.12.50'
***(IN): SH DATEOFBIRTH
 (OUT): DATEOFBIRTH :12.12.50
```



## NEXT

The NEXT command is used to abort processing of a command and start the next one. An interrupt is terminated.

---

### NEXT

---

In the case of an interrupt due to screen overflow, no further output is generated.

If a command with a repetition factor is interrupted, it is considered completed.

### *Example*

```

(IN): REMARK ***NEXT
***(IN): PROC TEST
***(IN): SH RECORD,L=45
***(IN): LIST CMD
***(IN): SH RECORD,L=45,F=X
***(IN): END
***(IN): DO TEST
***(IN): SH RECORD,L=45
 (OUT): RECORD - AREA :ANTON.....MAIER.....12.12.50.....
***(IN): LIST CMD
 (OUT): LIST OF COMMANDS
 (OUT): NAME : ADD (A)
 (OUT): NAME : CONTINUE (C)
 (OUT): NAME : DBH (DB)
 (OUT): NAME : DECLARE (DEC)
 (OUT): OTHER-NAME: DCL (DC)
 (OUT): NAME : DEFINE (DEF)
 (OUT): NAME : DELETE (DEL)
 (OUT): NAME : DISPLAY (DIS)
 (OUT): NAME : DISPOFF (DISPO)
 (OUT): OTHER-NAME: DOFF (DOF)
 (OUT): NAME : DO (DO)
 (OUT): NAME : EDT (ED)
 (OUT): NAME : END (EN)
 (OUT): NAME : ESCAPE (ES)
 (OUT): NAME : EXECUTE (E)
 (OUT): NAME : HALT (HALT)
 (OUT): OTHER-NAME: STOP (STOP)
 (OUT): NAME : HELP (H)
 (OUT): NAME : LANGUAGE (LA)
 (OUT): DMLTEST: SCREEN OVERFLOW. STATUS IS BREAK.
***(IN): NEXT
***(IN): SH RECORD,L=45,F=X
 (OUT): RECORD - AREA :
 (OUT): C1D5E3D6D50000000000D4C1C9C5D900000000000000000000F1F24BF1F24BF1F2000000000000
 (OUT): 0000000000

```

## PERFORM

The PERFORM command is used to call a user-specific module.

---

`PERFORM name [, filename] [, condition]`

---

*name* Must be the name of a module.

The module must observe the following conventions:

- R1 contains the address of the CALL parameter list.  
The 21st word of the address list contains, in binary format, the time required for the DML call for which the time was last taken (in 1/10000 secs.).
- R13 contains the address of the save area (18 words).
- R14 contains the return address.

*file-name*

If *file-name* is not specified, the module must exist in DMLTEST.MODLIB or in your own TASKLIB.

*condition*

Specifies the condition under which the module is to be called.

## PRINT

The PRINT command is used to output the specified area or value to SYSLST after each DML statement.

---

```
PRINT { RECORD
 RCODE
 TIME
 TALLY } [, distance] [, length] [, form] [, condition]
```

---

The following output formats are available:

- FORM=C: alphanumeric characters (standard)
- FORM=X: hexadecimal
- FORM=D: both C and X (dump format)

If RCODE and TIME are specified, *distance*, *length* and *form* specifications are ignored.

The contents of non-alphanumeric items can only be interpreted in hexadecimal format. This also applies for the contents of national items (Unicode: UTF-16, PICTURE N, USAGE NATIONAL).

## PROC

The PROC command is used to open a procedure definition.

---

```
PROC procname
```

---

*procname*

May be of any length. The first 8 characters only are used as the procedure identification.

The input following the PROC command is not submitted to a syntax check.

If the user enters a second PROC command without having entered an END command first, the first procedure is deleted.

A maximum of 20 procedures can be processed in each program run.

## PROFF

The PROFF command is used to deactivate the specified or all PRINT functions.

---


$$\left. \begin{array}{c} \{ \text{PROFF} \} \\ \{ \text{POFF} \} \end{array} \right\} [ \left. \begin{array}{c} \{ \text{RECORD} \\ \text{RCODE} \} \\ \{ \text{TALLY} \\ \text{TIME} \} \end{array} \right\} ] [ , \textit{condition} ]$$


---

## PROT

The PROT command is used to control the DMLTEST logging functions on SYSLST.

---


$$\text{PROT} [ \left. \begin{array}{c} \{ \text{ON} \} \\ \{ \text{OFF} \} \\ \{ \text{OUT} \} \end{array} \right\} ]$$


---

ON    Logs all input and output

OFF   Deactivates the log function

OUT   Logs SHOW output only

The default value is PROT ON.

## REMARK

The REMARK command is used to insert comment lines.

---


$$\left. \begin{array}{c} \{ \text{REMARK} \} \\ \{ * \} \end{array} \right\} \textit{literal}$$


---

*literal* Any characters may be specified. The command separator “;” may only be used within single quotes. The number of single quotes must be even.

## RUN

The RUN command is used to start a command sequence stored in an ISAM file.

---

```
RUN filename[,repetition][,condition]
```

---

### *file-name*

The file *filename* must be created according to EDTISAM format with the following characteristics:

RECFORM = V

KEYLEN = 8

KEYPOS = 5

The text may be up to 256 bytes in length and may consist of several commands.

### *repetition*

Specifies the number of times the command sequence is to be run. (repetition factor).

### *condition*

Specifies the condition which has to be satisfied for the command sequence to be run.

Processing of a command sequence takes place in the same way as with a DO procedure.

Up to 20 different RUN commands can be processed in one DMLTEST run (nesting level = 20).

## SET

The SET command is used to enter values in the CALL DML or KDBS parameters.

---


$$\underline{\text{SET}}[\textit{parameter}[(\left\{ \begin{matrix} n \\ \textit{x}'n' \end{matrix} \right\})]=]\textit{value}, \dots$$


---

### *parameter*

The individual parameters can be identified both by their position and their name.

*n* Is the displacement from the beginning of the item. If *n* is specified, only the specified *value* is transferred, and it is not padded with blanks.

If *n* is not specified, *value* is transferred left-justified and is padded with blanks.

The last keyword parameter always determines the position of the subsequent positional parameter. If no keyword parameter is specified, the first value specified is entered in the FCOD.

In positional parameters no displacement can be specified. *value* is transferred left-justified, padded with blanks.

If *value* including *n* exceeds the item in the parameter list, it is truncated without warning.

The SET command does not support national literals (Unicode: UTF-16).

The SET command processes different parameter records for CDML or CDML30 and KDBS.

## Search expressions with the SET command

DMLTEST can be used to edit search expressions which are formulated in CALL DML syntax (parameter SEX) or in the KDBS syntax (parameter FSI).

The following rules apply:

- The search expression must be entered in Assembler syntax as alphanumeric item.
- Syntactical units must be separated from one another by at least one blank.
- Item names are lengthened to 8 bytes.
- Comparison values must be entered in Assembler syntax; they are edited as required.
- Single quotes can be specified as part of the comparison value. Their number must be even.
- Item names are extended to 8 bytes for CDML (CALL8 interface) and to 30 bytes for CDML30 (CALL30 interface).

If you are using the CALL DML syntax, you must specify the *parameter* ITMN.

If a search expression is to be edited by DMLTEST, you must specify the *parameter* SEX.

### Example

```

***(IN): SET ITMN=C'0 SIZE GTH 40 0 END'
***(IN): SH ITMN,L=30
 (OUT): ITMN CONTAINS :0 SIZE GTH 40 0 END

***(IN): SET SEX=C'SIZE GTH CL2''40'' END'
***(IN): SH SEX,L=30
 (OUT): SEX CONTAINS :0 SIZE GTH 40 0 END

```

If you are using the KDBS syntax, you must specify the *parameter* SI.

If a search expression is to be edited by DMLTEST, you must specify the *parameter* FSI.

### Example

```

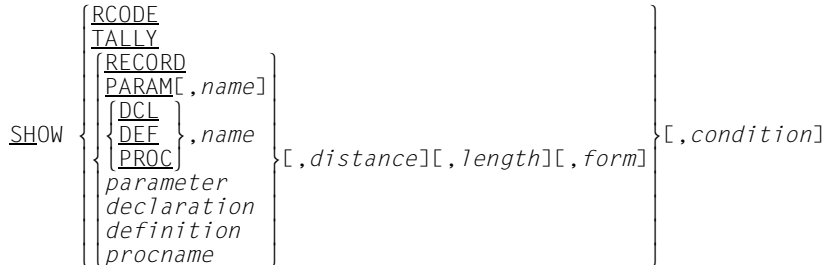
***(IN): SET SI=C'(ITEM1 EQ VALUE1)'
***(IN): SH SI,L=40
 (OUT): SI CONTAINS :(ITEM1 EQ VALUE1)

***(IN): SET FSI=C'(ITEM1 EQ CL10'' VALUE1'')'
***(IN): SH FSI,L=40
 (OUT): FSI CONTAINS :(ITEM1 EQ VALUE1)

```

## SHOW

The SHOW command is used to output the specified area on the screen.



*parameter, declaration, definition, procname*

If a name is specified, DMLTEST checks in the following order if the specified name is

- a CALL parameter
- a definition
- a procedure name
- a declaration

Then the corresponding contents are output on the screen.

### PARAM

The contents of all CALL DML parameters are output..

If the specified *length* is greater than the actual length, it is truncated to the actual length. Depending on the parameter involved, the actual length at the CALL30 interface is greater than at the CALL8 interface.

PROC The specified procedure is output in alphanumeric representation; *form* is ignored.

DCL *distance* and *length* are ignored.

The contents of non-alphanumeric items can only be interpreted in hexadecimal format. This also applies for the content of national items (Unicode).



*Example*

```

(IN): REMARK ***SHOW
***(IN): SH RECORD,L=40
 (OUT): RECORD - AREA :ALBERT...MAIER.....12.12.50.....
***(IN): SH RECORD,D=10,L=15,FORM=X
 (OUT): RECORD - AREA :D4C1C9C5D900000000000000000000
***(IN): SH RECORD,D=10,L=15,FORM=C
 (OUT): RECORD - AREA :MAIER.....
***(IN): SH RECORD,D=10,L=15,FORM=D
 (OUT): RECORD - AREA :
 (OUT): D4C1C9C5 D9000000 00000000 00000000 MAIER.....

```

**SUBSCHEMA**

The SUBSCHEMA command selects the subschema that you want to edit with DMLTEST.

---

SUBSCHEMA IS *subschema*

---

*subschema*

Name of the subschema to be edited with DMLTEST

More information on SUBSCHEMA can be found on pages [341](#) and [362](#), respectively.

**SYSTEM**

The SYSTEM command is used to interrupt the program run and pass control to the system. It enables the user to enter system commands.

---

SYSTEM[ *condition*]

---

*condition*

Specifies the condition under which control is to be passed to system.

The /RESUME command is used to return control to DMLTEST.

## TRACE

The TRACE command is used to log all processed commands and DML statements on the screen, or to deactivate this function.

---

```
TRACE[{ ON }][,repetition][,condition]
```

---

### ON, OFF

If no entry is made, the default value is ON.

### *repetition*

Specifies the number of commands to be logged before processing is interrupted. *repetition* is ignored if OFF is specified.

The default value is 4

### *condition*

Specifies the condition which has to be satisfied for the TRACE function to be executed.

### *Example*

```
*** (IN) : REMARK ***TRACE***
*** (IN) : TRACE ON,R=6
*** (IN) : RUN DMLTEST.EXAMPLES1
 (OUT): **CURRENT COMMAND**:RUN DMLTEST.EXAMPLES1
*** (IN) : * ***** DMLTEST.EXAMPLES1 *****
 (OUT): **CURRENT COMMAND**:
 (OUT): * ***** DMLTEST.EXAMPLES1 *****
*** (IN) : *
 (OUT): **CURRENT COMMAND**:
 (OUT): *
*** (IN) : DISPOFF RECORD
 (OUT): **CURRENT COMMAND**:DISPOFF RECORD
*** (IN) : REMARK *** EXAMPLE ACCEPT FORMAT 1 ***
 (OUT): **CURRENT COMMAND**:REMARK *** EXAMPLE ACCEPT FORMAT 1 ***
*** (IN) : FIND 4 ARTICLE
 (OUT): **CURRENT COMMAND**:FIND 4 ARTICLE
 (OUT): DMLTEST: TRACE-LIMIT. STATUS IS BREAK.
*** (IN) : C
*** (IN) : EX
 (OUT): **CURRENT COMMAND**:EX
*** (IN) : ACCEPT DB-KEY FROM CURRENCY
 (OUT): **CURRENT COMMAND**:ACCEPT DB-KEY FROM CURRENCY
*** (IN) : EX
 (OUT): **CURRENT COMMAND**:EX
*** (IN) : SH PARAM
 (OUT): **CURRENT COMMAND**:SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :ACCP TC.. FOPT :DB-KEY.. SOPT : UINF : RECN :ARTICLE
 (OUT): SETN :(MIN-STOCK-LEVEL RLMN : ITMN : RECA :ANTON... SPP1 :ADMIN
 (OUT): SPP2 :...4 SPP3 : SUBS :ADMIN
```

```
*** (IN) : SH DB=KEY,FORM=X
 (OUT) : **CURRENT COMMAND**:SH DB=KEY,FORM=X
 (OUT) : DB=KEY :09000004
*** (IN) : REMARK *** EXAMPLE ACCEPT FORMAT 2 ***
 (OUT) : **CURRENT COMMAND**:REMARK *** EXAMPLE ACCEPT FORMAT 2 ***
 (OUT) : DMLTEST: TRACE-LIMIT. STATUS IS BREAK.
*** (IN) : TRACE OFF
```

## WAIT

The WAIT command is used to generate an interrupt.

---

WAIT[ *condition*]

---

### *condition*

Specifies the condition under which the interrupt to be generated.

If SYSDTA is assigned to a file, the interrupt is terminated with CONTINUE. This cannot be influenced by the user.

## 9.3 The DML statements of DMLTEST

You may enter DML statements in COBOL DML format (see [section “COBOL DML statements” on page 139](#)); DMLTEST will convert them to the CALL DML format. Depending on whether you have specified the keyword COBOL or COBOL30 in the LANGUAGE command (see [page 324](#)), DMLTEST will supply the CALL8 or CALL30 interface.

### 9.3.1 Overview of differences between DMLTEST DML and COBOL DML statements

|            |                        |                                          |                                                                                                                                                                                                                                                                   |
|------------|------------------------|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCEPT     | format 1:              | <i>item-name-1</i>                       | DB-KEY, DB-KEY-LONG                                                                                                                                                                                                                                               |
|            | format 2:              | <i>item-name-2</i><br><i>item-name-3</i> | AREA-ID<br>DB-KEY, DB-KEY-LONG                                                                                                                                                                                                                                    |
| FIND/FETCH | format 1:              | <i>item-name</i><br>OR PRIOR/NEXT        | DB-KEY, DB-KEY-LONG<br>These clauses may be used in DMLTEST in the form SET FOPT,DBKPRI/DBKNXT after entering the FIND1/FTCH1 or FIND1L/FTCH1L statement in COBOL-DML syntax (without EXECUTE).                                                                   |
|            | format 2:              |                                          | If the record type can be stored in several realms, IMP must be used:<br>... <i>record-name</i> [ <u>IMP</u> ]                                                                                                                                                    |
|            | format 3:              |                                          | USING can be extended by<br><br>...[ $\left. \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right\} \textit{record-name}$ ]                                                                                                                                 |
|            | format 4:              | <i>item-name</i>                         | -                                                                                                                                                                                                                                                                 |
|            | format 7:              | <i>item-name-1</i><br>OR PRIOR/NEXT      | TALLY<br>These clauses may be used in DMLTEST in the form SET FOPT,...ITP/...ITN after entering the FIND7A/FTCH7A statement in COBOL-DML syntax (without EXECUTE).                                                                                                |
| GET        |                        |                                          | This statement can be extended by<br><br>...[ $\left. \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right\} \textit{record-name}$ ]                                                                                                                        |
| IF         | format 1:<br>format 2: |                                          | } NEXT SENTENCE, ELSE and NOT must not be used.                                                                                                                                                                                                                   |
| MODIFY     |                        |                                          | This statement is extended by<br><br>...[ $\left. \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right\} \textit{record-name}$ ]                                                                                                                            |
| STORE      |                        |                                          | If the record type can be stored in multiple realms or if the database key is assigned by the user (DDL clause LOC MODE), you must use the IMP or IMP-LONG option:<br>... <i>record-name</i> [ <u>IMP</u> ]...or<br>... <i>record-name</i> [ <u>IMP-LONG</u> ]... |

Table 40: Differences between COBOL DML and DMLTEST DML statements

If LANGUAGE COBOL or LANGUAGE COBOL30 has been specified, the following format can be used for subschema assignment:

---

SUBSCHEMA[ IS] *subschema-name*

---

This statement must be entered prior to the READY statement.

The SUBSCHEMA statement can also be used to assign DML statements to transaction chains or to individual databases if multi-DB transactions are to be opened.

### 9.3.2 The DML statements

#### ACCEPT

*ACCEPT (format1):*

$$\text{ACCEPT } \left\{ \begin{array}{l} \text{DB-KEY} \\ \text{DB-KEY-LONG} \end{array} \right\} \text{ FROM} \left[ \left\{ \begin{array}{l} \text{record-name} \\ \text{set-name} \\ \text{realm-name} \end{array} \right\} \right] \text{ CURRENCY}$$

DMLTEST stores the database key value in the predefined item DB-KEY or DB-KEY-LONG.

*Example*

```

***(IN): REMARK *** EXAMPLE ACCEPT FORMAT 1 ***
***(IN): FIND 4 ARTICLE
***(IN): EX
***(IN): ACCEPT DB-KEY FROM CURRENCY
***(IN): EX
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :ACPTC.. FOPT :DB-KEY.. SOPT : UINF : RECN :ARTICLE
(OUT): SETN : RLMN : ITMN : RECA :..... SPP1 :ADMIN
(OUT): SPP2 :..... SPP3 : SUBS :ADMIN
***(IN): SH DB-KEY,FORM=X
(OUT): DB-KEY :09000004

```

*ACCEPT (format 2):*

$$\text{ACCEPT } \text{AREA-ID} \text{ FROM} \left[ \left\{ \begin{array}{l} \text{record-name} \\ \text{set-name} \\ \text{DB-KEY} \\ \text{DB-KEY-LONG} \end{array} \right\} \right] \text{ REALM-NAME}$$

The database key value must be entered in the predefined DB-KEY or DB-KEY-LONG item in binary format. DMLTEST stores the result (realm name) in the predefined item AREA-ID.

*Example*

```

***(IN): REMARK *** EXAMPLE ACCEPT FORMAT 2 ***
***(IN): ACCEPT AREA-ID FROM DB-KEY REALM-NAME
***(IN): EX
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :ACPTC.. FOPT :RLMDBK.. SOPT : UINF :CLOTHING RECN :ARTICLE
(OUT): SETN : RLMN : ITMN : RECA :..... SPP1 :ADMIN
(OUT): SPP2 :..... SPP3 : SUBS :ADMIN
***(IN): SH AREA-ID
(OUT): AREA-ID :CLOTHING

```

**CONNECT**

```

CONNECT [record-name] TO { set-name-1, ... }
 ALL
 }
 [RETAINING CURRENCY FOR { set-name-2, ... }]
 SETS
 }

```

*Example*

```

***(IN): REMARK ***EXAMPLE CONNECT ***
***(IN): FIND 1 ARTICLE
***(IN): EX
***(IN): CONNECT ARTICLE TO MIN-STOCK-LEVEL
***(IN): EX
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :CONNEC.. FOPT :TO-SET.. SOPT : UINF : RECN :ARTICLE
(OUT): SETN :(MIN-STOCK-LEVEL RLMN : ITMN : RECA :..... SPP1 :ADMIN
(OUT): SPP2 :.... SPP3 : SUBS :ADMIN
***(IN): FETCH 1 ARTICLE WITHIN MIN-STOCK-LEVEL
***(IN): EX
***(IN): SH RECORD,LNG=48
(OUT): RECORD - AREA :00000110SUMMER DRESS WITH JACKET

```

**DISCONNECT***DISCONNECT (format 1):*

```
DISCONNECT[record-name] FROM { set-name,... }
 { ALL }
```

*Example*

```
***(IN): REMARK *** EXAMPLE DISCONNECT FORMAT 1 ***
***(IN): FETCH 1 ARTICLE WITHIN MIN-STOCK-LEVEL
***(IN): EX
***(IN): DISCONNECT ARTICLE FROM MIN-STOCK-LEVEL
***(IN): EX
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :DISCON.. FOPT :FRMSET.. SOPT : UINF : RECN :ARTICLE
 (OUT): SETN :(MIN-STOCK-LEVEL RLMN : ITMN : RECA :00000110 SPP1 :ADMIN
 (OUT): SPP2 :.... SPP3 : SUBS :ADMIN
***(IN): FINISH
***(IN): EX
```

*DISCONNECT (format 2):*

```
DISCONNECT ALL FROM set-name,...
```

**ERASE**

```
ERASE record-name[{ PERMANENT }
 { SELECTIVE } MEMBERS]
 { ALL }
```

*Example*

```
***(IN): REMARK *** EXAMPLE ERASE ***
***(IN): FETCH LAST ARTICLE
***(IN): E
***(IN): ERASE ARTICLE ALL
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :ERASEC.. FOPT :ALLMEM.. SOPT : UINF : RECN :ARTICLE
 (OUT): SETN :(MIN-STOCK-LEVEL RLMN : ITMN :(SUPPL- RECA :901145 0 SPP1 :
 (OUT): SPP2 :...r SPP3 : SUBS :ADMIN
```



**FIND/FETCH**

$$\left. \begin{array}{l} \text{[FIND]} \\ \text{[FETCH]} \end{array} \right\} \text{record-selection-expression [RETAINING CURRENCY FOR}$$

$$\left. \begin{array}{l} \text{[MULTIPLE]} \\ \left. \begin{array}{l} \text{[REALM]} \left\{ \begin{array}{l} \text{[SETS]} \\ \text{set-name, ...} \end{array} \right\} \text{[RECORD]} \end{array} \right\} \end{array} \right\}$$

*Formats of the record selection expression:*

*FIND/FETCH (format 1):*

$$[\text{record-name}] \text{ DATABASE-KEY IS } \left. \begin{array}{l} \text{DB-KEY} \\ \text{DB-KEY-LONG} \end{array} \right\}$$

The database key value must have been specified in the predefined item DB-KEY or DB-KEY-LONG in binary format.

*Example*

```

***(IN): REMARK *** EXAMPLE FETCH 1 ***
***(IN): M DB-KEY,X'09000004'
***(IN): FETCH DATABASE-KEY IS DB-KEY
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :FTCH1 .. FOPT :DB-KEY.. SOPT : UINF :CLOTHING RECN :
(OUT): SETN :(MIN-STOCK-LEVEL RLMN : ITMN : RECA :00000110 SPP1 :ADMIN
(OUT): SPP2 :.... SPP3 : SUBS :ADMIN
***(IN): SH RECORD,L=80
(OUT): RECORD - AREA :
(OUT): 00000110SUMMER DRESS WITH JACKET 23010742.....m...

```

You can implement access to the previous or next record for the database key (COBOL parameter OR PRIOR/NEXT) for which no record exists in the database by first entering the FIND/FETCH statement and then supplying the FOPT parameter with DBKPRI or DBKNXT before executing the statement (e.g. FETCH DATABASE-KEY IS DB-KEY; MOVE FOPT,DBKNXT;E).

*FIND/FETCH (format 2):*

```
{ ANY
 DUPLICATE } record-name[IMP]
```

When the record type *record-name* can be stored in more than one realm, the IMP option must be specified and the IMP-AREA-ID item must be supplied with the realm name of the realm in which the record is to be searched for if the record type is not the member record type of a distributable list.

*Example*

```
***(IN): REMARK *** EXAMPLE FETCH 2 ***
***(IN): FETCH 9 ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :000005SO
***(IN): S SPP2=C' '
***(IN): M IMP-AREA-ID, 'CLOTHING'
***(IN): FETCH ANY ARTICLE-DESCR IMP
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH2 .. FOPT :ANYIMP.. SOPT : UINF :CLOTHING RECN :ARTICLE
 (OUT): SETN : (MIN-STOCK-LEVEL RLMN : ITMN : RECA :000001SO SPP1 :ADMIN
 (OUT): SPP2 : CLOT SPP3 : SUBS :ADMIN
***(IN): SH RECORD,L=6
 (OUT): RECORD - AREA :000001
***(IN): PROC ULI
***(IN): FETCH DUPLICATE ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD,LNG=6
***(IN): END
***(IN): DO ULI,R=4
***(IN): FETCH DUPLICATE ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD,LNG=6
 (OUT): RECORD - AREA :000002
***(IN): FETCH DUPLICATE ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD,LNG=6
 (OUT): RECORD - AREA :000003
***(IN): FETCH DUPLICATE ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD,LNG=6
 (OUT): RECORD - AREA :000004
***(IN): FETCH DUPLICATE ARTICLE-DESCR
***(IN): E
***(IN): SH RECORD,LNG=6
 (OUT): RECORD - AREA :000005
***(IN): DELETE ULI
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH2 .. FOPT :DUPLIC.. SOPT : UINF :CLOTHING RECN :ARTICLE
 (OUT): SETN : (MIN-STOCK-LEVEL RLMN : ITMN : RECA :000005SO SPP1 :ADMIN
 (OUT): SPP2 : CLOT SPP3 : SUBS :ADMIN
```

*FIND/FETCH (format3):*

DUPLICATE WITHIN  $\left\{ \begin{array}{l} \text{rec-name} \\ \text{set-name} \end{array} \right\}$  [ USING record-element-name,... ] [  $\left\{ \begin{array}{l} \text{IN} \\ \text{OF} \end{array} \right\}$  rec-name ]

In order to enable CALL DML to correctly identify the items, you must specify “IN rec-name” with “DUPLICATE WITHIN set-name USING record-element-name,...”.

*Example*

```

***(IN): REMARK *** EXAMPLE FETCH 3 ***
***(IN): FETCH 2 PURCHASE-ORDER
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :799E8201
***(IN): FETCH DUPLICATE WITHIN PURCH-ORD-PLACED USING P-ORDER-MTH
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH3 .. FOPT :SETITM.. SOPT : UINF :PURCHASE RECN :PURCHASE
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-O-ORD RECA :854.8201 SPP1 :ADMIN
 (OUT): SPP2 :...CLOT SPP3 : SUBS :ADMIN
***(IN): SH RECORD
 (OUT): RECORD - AREA :854.8201

```

*FIND/FETCH (format 4):*

$$\left. \begin{array}{l} \text{FIRST} \\ \text{NEXT} \\ \text{PRIOR} \\ \text{LAST} \\ \text{integer} \end{array} \right\} \left\{ \begin{array}{l} \text{record-name} \\ \text{RECORD} \end{array} \right\} \left[ \text{WITHIN} \left\{ \begin{array}{l} \text{set-name} \\ \text{realm-name} \end{array} \right\} \right]$$

*Example*

```

***(IN): REMARK *** EXAMPLE FETCH 4 ***
***(IN): FETCH FIRST SUPPLIER
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :00001MON
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH4 .. FOPT :REFST.. SOPT : UINF :PURCHASE RECN :SUPPLIER
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-ORD RECA :00001MON SPP1 :ADMIN
 (OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
***(IN): FETCH 2 PURCHASE-ORDER WITHIN PURCH-ORD-PLACED
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :854.8201
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH4 .. FOPT :SETSPC.. SOPT : UINF :PURCHASR RECN :PURCHASE
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-ORD RECA :854.8201 SPP1 :ADMIN
 (OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN

```

*FIND/FETCH (format 5):*

```
CURRENT[record-name][WITHIN { set-name }
 { realm-name }]
```

*Example*

```
*** (IN): REMARK *** EXAMPLE FETCH 5 ***
*** (IN): FIND FIRST ARTICLE
*** (IN): E
*** (IN): FIND NEXT ARTICLE RETAINING CURRENCY FOR ARTICLES-AVAILABLE
*** (IN): E
*** (IN): FETCH CURRENT ARTICLE RETAINING CURRENCY FOR ARTICLES-AVAILABLE
*** (IN): E
*** (IN): SH RECORD,L=80
(OUT): RECORD - AREA :
(OUT): 00000110SUMMER DRESS WITH JACKET 23010738.....rn...
*** (IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :FTCH5 .. FOPT :RECNAM.. SOPT :RET UINF :CLOTHING RECN :ARTICLE
(OUT): SETN :P-ORD-PL RLMN : ITMN :(P-ORD-M RECA :00000110 SPP1 : ST
(OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
*** (IN): FETCH CURRENT ARTICLE WITHIN ARTICLES-AVAILABLE
*** (IN): E
*** (IN): SH RECORD,L=80
(OUT): RECORD - AREA :
(OUT): 00000110SUMMER DRESS WITH JACKET 23010736.....rn...
```

*FIND/FETCH (format 6):*

OWNER WITHIN *set-name*

*Example*

```

***(IN): REMARK *** EXAMPLE FETCH 6 ***
***(IN): FIND 1 PURCHASE-ORDER
***(IN): E
***(IN): FETCH CURRENT PURCHASE-ORDER WITHIN P-ORD-PLACED
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :000C8201
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH5 .. FOPT :RECSET.. SOPT : UINF :PURCHASE RECN :PURCHASE
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-ORD RECA :000C8201 SPP1 :ST
 (OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
***(IN): FETCH OWNER WITHIN PURCH-ORD-PLACED
***(IN): E
***(IN): SH RECORD
 (OUT): RECORD - AREA :00001MON
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH6 .. FOPT :RECSET.. SOPT : UINF :PURCHASE RECN :PURCHASE
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-ORD RECA :00001MON SPP1 :ST
 (OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN

```

*FIND/FETCH (format 7):*

```

record-name[WITHIN set-name-1[CURRENT]]
{
 [USING record-element-name-1,...
 [USING search-expression]
 [RESULT IN set-name-2]
 [LIMITED BY set-name-3]
 [TALLYING TALLY]
 [SORTED[{ASCENDING}]][{BY}]
 [{DESCENDING}]][{ON}]
 record-element-name-2[[,]record-element-name-3]...
 [[,][{ASCENDING}]][{BY}]
 [{DESCENDING}]][{ON}]
 record-element-name-4[[,]record-element-name-5]...]]
}

search-expression { complex-1[AND complex-2]
 { complex-2
}

complex-1 [NOT] condition-1
 { {AND}
 [NOT] condition-1]...
 { {OR}
}

complex-2 condition-2[AND condition-2]...

condition-1 record-element-name-6[WITH MASK mask]
 { {EQUAL
 =
 GREATER THAN } { declaration
 >
 LESS THAN } { literal
 <
}

condition-2 record-element-name-7 IS NEXT
 { {GREATER THAN
 >
 LESS THAN } { declaration
 <
 }
 [NOT]
}

```

*record-element-name-2...5*

corresponds to a record element in the database

*declaration*

corresponds to a DECLARE item

When using DECLARE items as compare items, you should note how the contents of the items have been entered by the MOVE command (see [page 327](#)).

When DECLARE items are used as mask items, they must be of the same length as the corresponding items in the database.  
Each byte must be identified as either significant (X'F1') or non-significant (X'F0').

### Examples

```

***(IN): REMARK *** EXAMPLE FETCH 7 ***
***(IN): REMARK **FETCH-7 RECORD ELEMENT NAMES**

***(IN): S RECA=C' '
***(IN): M RECA,C' 8201'
***(IN): FETCH PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH7A.. FOPT :RECITM.. SOPT : UINF :PURCHASE REC N :PURCHASE
 (OUT): SETN :PURCH-OR RLMN : ITMN :(P-ORD RECA :000C8201 SPP1 :ST
 (OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
***(IN): SH RECORD,L=10
 (OUT): RECORD - AREA :000C8201E
***(IN): PROC ULI
***(IN): FETCH DUPLICATE WITHIN PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR
***(IN): E
***(IN): SH RECORD,L=10
***(IN): END
***(IN): DO ULI,R=3
***(IN): FETCH DUPLICATE WITHIN PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR
***(IN): E
***(IN): SH RECORD,L=10
 (OUT): RECORD - AREA :799E8201.
***(IN): FETCH DUPLICATE WITHIN PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR
***(IN): E
***(IN): SH RECORD,L=10
 (OUT): RECORD - AREA :854.82010C
***(IN): FETCH DUPLICATE WITHIN PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR
***(IN): E
***(IN): SH RECORD,L=10
 (OUT): RECORD - AREA :785D82010C
***(IN): DEL ULI

***(IN): REMARK **FETCH-7 SEARCH EXPRESSION**

***(IN): RUN PO
***(IN): DCL PARAMS,L=41
***(IN): M PARAMS,*FOPT*SOPT*REC N*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): SH PARAMS,L=41
 (OUT): PARAMS :*FOPT*SOPT*REC N*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): DEL PARAMS
***(IN): SET FOPT=C' '
***(IN): SET SOPT=C' '
***(IN): SET REC N=C' '
***(IN): SET ITMN=C' '
***(IN): SET RECA=C' '
***(IN): S SPP1=C' '
***(IN): S SPP2=C' '
***(IN): S RLMN=C' '
***(IN): FIND 1 ARTICLE-DESCR
***(IN): E
***(IN): FETCH ARTICLE WITHIN P-ORD-SPEC USING SIZE IS > 40
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :FTCH7A.. FOPT :SELSEX.. SOPT : UINF :CLOTHING REC N :ARTICLE

```



```

 (OUT): SETN :PURCHASE-ORDER RLMN : ITMN :0 SIZE RECA :00000110 SPP1 :
 (OUT): SPP2 :.... SPP3 : SUBS :ADMIN
*** (IN) : SH RECA,L=80
 (OUT): RECA CONTAINS :
 (OUT): 00000110SUMMER DRESS WITH JACKET 23010742.....rn...
*** (IN) : PROC ULI
*** (IN) : FETCH DUPLICATE WITHIN P-ORD-SPEC
*** (IN) : E
*** (IN) : SH RECA,L=80
*** (IN) : END
*** (IN) : DO ULI,R=3,COND=RCODE EQ C'00000'
*** (IN) : FETCH DUPLICATE WITHIN P-ORD-SPEC
*** (IN) : E
*** (IN) : SH RECA,L=80
 (OUT): RECA CONTAINS :
 (OUT): 00000110SUMMER DRESS WITH JACKET 23010744.....rn...
*** (IN) : FETCH DUPLICATE WITHIN P-ORD-SPEC
*** (IN) : E
*** (IN) : SH RECA,L=80
 (OUT): RECA CONTAINS :
 (OUT): 00000110SUMMER DRESS WITH JACKET 23010746.....q...
*** (IN) : FETCH DUPLICATE WITHIN P-ORD-SPEC
*** (IN) : E
*** (IN) : SH RECA,L=80
 (OUT): RECA CONTAINS :
 (OUT): 00000110SUMMER DRESS WITH JACKET 23010748.....q...

```

If no record exists in the database for the specified values, you can implement access to the previous or next record (COBOL parameter OR PRIOR/NEXT) by first entering the FIND/FETCH statement and then supplying the FOPT parameter with RECITP/SECITN or ...ITN before executing the statement (e.g. FETCH PURCHASE-ORDER USING P-ORD-MTH,P-ORD-YEAR; MOVE FOPT,RECITN;E).

**FINISH**

FINISH [ WITH CANCEL ]

**FREE**

FREE [ ALL ]

**GET**

GET [ { *record-name* } ] [ { IN } *record-name* ]  
 [ { *item-name,...* } ] [ { OF } ]

**IF**

*IF (format 1):*

IF [ *set-name-1* ] { OWNER }  
 { MEMBER }  
 { TENANT }

*IF (format 2):*

IF *set-name-2* IS EMPTY

**KEEP**

KEEP

**MODIFY**

MODIFY { *record-name* }  
 { *record-element-name,...* { IN } *record-name* }  
 { OF }  
 [ { INCLUDING } { ALL }  
 { ONLY } { *set-name-1,...* } MEMBERSHIP ]  
 [ RETAINING CURRENCY FOR { SETS }  
 { *set-name-1,...* } ]

In order to enable CALL DML to correctly identify the items, “IN *record-name*” must be specified.

*Examples*

```

(IN): * **MODIFY RECORD ELEMENT*****
***(IN): FETCH 1 SUPPLIER
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :FTCH4 .. FOPT :RECSPC.. SOPT : UINF :PURCHASE RECN :SUPPLIER
(OUT): SETN :P-ORD-PL RLMN : ITMN :(P-ORD-M RECA :00001MON SPP1 :ADMIN
(OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
***(IN): SH RECA,L=104
(OUT): RECA CONTAINS :
(OUT): 00001MONA FASHIONS 7500KARLSRUHE 1 AUGUSTENSTR
(OUT): ASSE 1 00
***(IN): M RECA,999,D=99
***(IN): MODIFY SUPP-STREET-NO
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :MODIF2.. FOPT :CORUNT.. SOPT : UINF : RECN :SUPPLIER
(OUT): SETN :P-ORD-PL RLMN : ITMN :(SUPP-ST RECA :00001MON SPP1 :ADMIN
(OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN
***(IN): SH RECA,L=104
(OUT): RECA CONTAINS :
(OUT): 00001MONA FASHIONS 7500KARLSRUHE 1 AUGUSTENSTR
(OUT): ASSE 99900
***(IN): M RECA,001,D=99
***(IN): MODIFY SUPP-STREET-NO
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :MODIF2.. FOPT :CORUNT.. SOPT : UINF : RECN :SUPPLIER
(OUT): SETN :P-ORD-PL RLMN : ITMN :(SUPP-ST RECA :00001MON SPP1 :ADMIN
(OUT): SPP2 :....CLOT SPP3 : SUBS :ADMIN

***(IN): REMARK *** EXAMPLE MODIFY OWNER-ASSIGNMENT ***

***(IN): RUN PO
***(IN): DCL PARAMS,L=41
***(IN): M PARAMS,*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): SH PARAMS,L=41
(OUT): PARAMS :*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): DEL PARAMS
***(IN): SET FOPT=C' '
***(IN): SET SOPT=C' '
***(IN): SET RECN=C' '
***(IN): SET ITMN=C' '
***(IN): SET RECA=C' '
***(IN): S SPP1=C' '
***(IN): S SPP2=C' '
***(IN): S RLMN=C' '
***(IN): FIND 1 PURCHASE-ORDER
***(IN): E
***(IN): FETCH OWNER WITHIN PURCH-ORD-PLACED
***(IN): E
***(IN): SH RECA,L=130
(OUT): RECA CONTAINS :
(OUT): 00001MONA FASHIONS 7500KARLSRUHE 1 AUGUSTENSTR
(OUT): ASSE 001000721609422.....
(OUT):
***(IN): M RECA,00002BRAKSPEARS BREWERY

```

```

***(IN): E
***(IN): FIND 1 PURCHASE-ORDER RETAINING CURRENCY FOR MULTIPLE
***(IN): E
***(IN): MODIFY PURCHASE-ORDER ONLY PURCH-ORD-PLACED MEMBERSHIP
***(IN): E
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :MODIF1.. FOPT :ONLSET.. SOPT : UINF : RECN :PURCHASE
 (OUT): SETN :(PURCH-O RLMN : ITMN : RECA :0000283 SPP1 :MULTIPLE
 (OUT): SPP2 :... SPP3 : SUBS :ADMIN
***(IN): FETCH OWNER WITHIN PURCH-ORD-PLACED
***(IN): E
***(IN): SH RECA,L=130
 (OUT): RECA CONTAINS :
 (OUT): 00002BRAKSPEARS BREWERY 8000MUENCHEN 2 MARSSTRASSE
 (OUT): 77 000899128741.....
***(IN): FIND 1 SUPPLIER
***(IN): E
***(IN): FIND 1 PURCHASE-ORDER RETAINING MULTIPLE
***(IN): E
***(IN): E

```

**READY**

READY[ *realm-name*,...]

[USAGE-MODE IS { EXCLUSIVE } ] { RETRIEVAL }  
                                   { PROTECTED } { UPDATE }

*Example*

```

***(IN): SUBS IS ADMIN
***(IN): M USERGROUP,CL8'SALES'
***(IN): M USERNAME,CL24'MAYER'
***(IN): M PASSWORD,CL48'HUGO'
***(IN): READY USAGE-MODE IS EXCLUSIVE UPDATE
***(IN): EX
***(IN): SH PARAM
 (OUT): DB-PARAMS (FIRST 8B)
 (OUT): FCOD :READYC.. FOPT :ALLEUP.. SOPT : UINF : RECN :ARTICLE
 (OUT): SETN : RLMN : ITMN : RECA :..... SPP1 :ADMIN
 (OUT): SPP2 :SALES SPP3 : SUBS :ADMIN

```

**STORE**

```

STORE rec-name [{ IMP
 IMP-LONG }] [RETAINING CURRENCY FOR
 { MULTIPLE
 [REALM] [{ SETS
 { set-name, ... } }] [RECORD] }]

```

The IMP or IMP-LONG option must be specified in the following cases:

- The record type *rec-name* can be stored in several realms.

If the record type *rec-name* was **not** defined with the DDL clause LOCATION MODE (see the “[Design and Definition](#)” manual), you may optionally specify IMP or IMP-LONG. In addition, the predefined item IMP-AREA-ID must be supplied with the name of the realm to be searched for the record.

Distributable lists are an exception to the aforementioned rule:

When a record which is a member of a distributable list is stored, a free page is searched for in the DBH in the preferred realm if needed. In this case any realm specification in the IMP-AREA-ID item is ignored.

- The record type *record-name* was defined with LOCATION DIRECT or LOCATION DIRECT-LONG (see the “[Design and Definition](#)” manual).

IMP must be specified for LOCATION DIRECT, and IMP-LONG must be specified for LOCATION DIRECT-LONG.

In addition, the appropriate database key or binary zero must be entered in the predefined item IMP-DB-KEY or IMP-DB-KEY-LONG prior to storing.

*Examples*

```

***(IN): REMARK *** EXAMPLE STORE DB-KEY ***

***(IN): RUN PO
***(IN): DCL PARAMS,L=41
***(IN): M PARAMS,*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): SH PARAMS,L=41
 (OUT): PARAMS :*FOPT*SOPT*RECN*ITMN*RECA*SPP1*SPP2*RLMN*
***(IN): DEL PARAMS
***(IN): SET FOPT=C' '
***(IN): SET SOPT=C' '
***(IN): SET RECN=C' '
***(IN): SET ITMN=C' '
***(IN): SET RECA=C' '
***(IN): S SPP1=C' '
***(IN): S SPP2=C' '
***(IN): S RLMN=C' '
***(IN): FETCH LAST ARTICLE
***(IN): E
***(IN): S RECA=C' '
***(IN): M UINF,FOOD

```

```

***(IN): M IMP-DB-KEY,X'09000099'
***(IN): M RECA,C'001144 ODR.MAYERORANGEJUICE'
***(IN): STORE ARTICLE IMP
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :STORE1.. FOPT :IMPDAT.. SOPT : UINF :FOOD RECN :ARTICLE
(OUT): SETN :(P-ORD-P RLMN : ITMN :(SUPP-ST RECA :001144 0 SPP1 :MULTIPLE
(OUT): SPP2 :...rCLOT SPP3 : SUBS :ADMIN
***(IN): FETCH LAST ARTICLE
***(IN): E
***(IN): SH RECA,L=80
(OUT): RECA CONTAINS :
(OUT): 001144 ODR.MAYERORANGEJUICE
***(IN): ERASE ARTICLE
***(IN): E

***(IN): REMARK *** EXAMPLE STORE SET OCCURRENCE ***

***(IN): S RECA=C' '
***(IN): M RECA,000003
***(IN): FETCH ART-DESCR USING ART-NO
***(IN): E
***(IN): S RECA=C' '
***(IN): M RECA,00002BRAKSPEAR
***(IN): FETCH SUPPLIER USING SUPPL-NO
***(IN): E
***(IN): S RECA=C' '
***(IN): M RECA,C'901145 OMILLER DIET MEALS'
***(IN): STORE ARTICLE
***(IN): E
***(IN): SH PARAM
(OUT): DB-PARAMS (FIRST 8B)
(OUT): FCOD :STORE1.. FOPT :RECNAM.. SOPT : UINF :FOOD RECN :ARTICLE
(OUT): SETN :(P-ORD-P RLMN : ITMN :(SUPPL-N RECA :901145 0 SPP1 :MULTIPLE
(OUT): SPP2 :...rCLOT SPP3 : SUBS :ADMIN
***(IN): FETCH LAST ARTICLE
***(IN): E
***(IN): SH RECA,L=80
(OUT): RECA CONTAINS :
(OUT): 901145 OMILLER DIET MEALS

```

## 9.4 DMLTEST program execution

This section contains information on the following topics:

- interrupts that can occur during a program run
- communication between DMLTEST and one or more databases.

Before executing DMLTEST, you should first prepare the subschema that you want to work with by means of the BCALLSI utility routine (see the [“Creation and Restructuring”](#) manual).

### 9.4.1 Interrupts

An interrupt occurs in the following cases:

- Logical and syntactical errors in commands
- Reaching the TRACE limit
- Entering the WAIT command
- Screen overflow

If interactive mode is to be used, input is expected from the terminal.

An interrupt status is indicated by the message:

```
STATUS IS BREAK
```

The following DMLTEST commands can be used to respond to an interrupt:

- NEXT to abort processing of a command. The next command in the current command sequence is started.
- CONTINUE to simply continue processing.

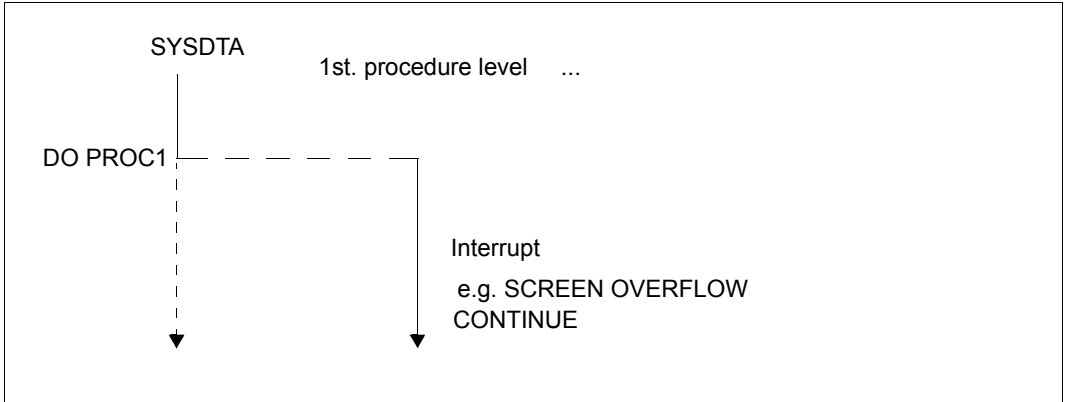


Figure 11: Response to interrupts using CONTINUE

ESCAPE to abort processing of all started procedures or command sequences. The next command is expected from SYSDTA.

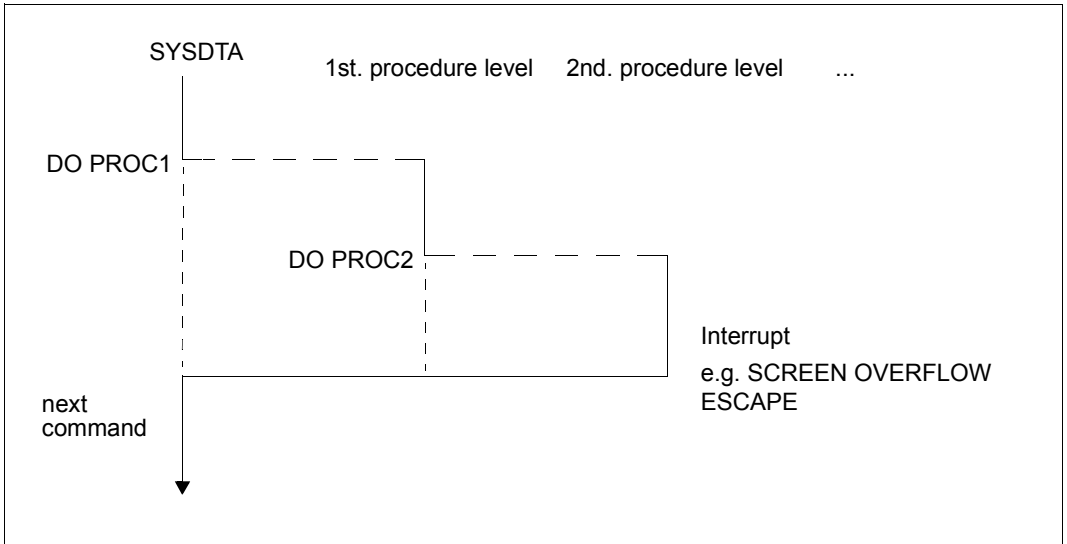


Figure 12: Response to interrupts using ESCAPE



**LEAVE** to abort processing of the current procedure or command sequence and pass control to the next-higher level of procedure or command sequence or to the terminal.

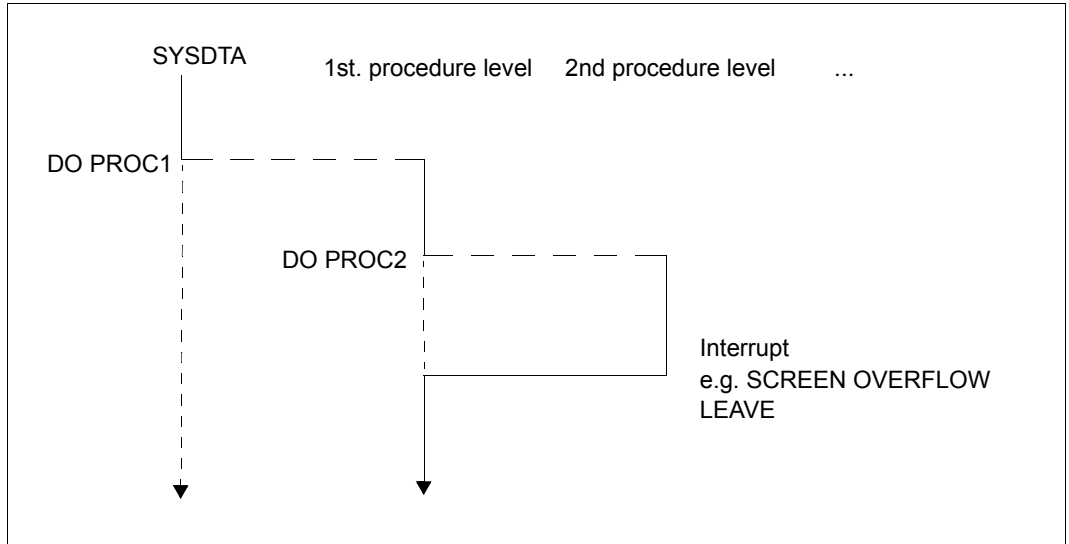


Figure 13: Response to interrupts using LEAVE

When a CONTINUE, ESCAPE or NEXT command is entered, the interrupt is automatically terminated. A LEAVE command, by contrast, must be followed by one of the other three commands if you want the interrupt to be terminated.

## 9.4.2 Communication with one or more databases

In order to work with one database, you must first enter the following specifications:

- the desired DBH variant (DBH command)
- the desired language (LANGUAGE command)
- the desired subschema  
(SET SUBS=*subschemaname* or SUBSCHEMA IS *subschemaname*)

You can then execute the READY statement.

The following applies for each DML statement:

If you have supplied values in the parameter list, the items of the parameter list will remain unchanged until you overwrite them with new values. The record area and the user information area can also be overwritten by DML statements.

The EXECUTE command can be repeated with the same parameter values any number of times.

The record area and user information area refer to the task last executed.

The SHOW, PRINT or DISPLAY command can be used to view the results.

### Multi DB

If you are running DMLTEST on several databases, you must ensure that each database remains symbolically addressable.

A database is identified via the subschema name. The corresponding information can be passed to DMLTEST by specifying SET SUBS=*subschemaname* or SUBS IS *subschemaname*. The first 6 characters of the specified name are used as the symbolic name of the database; however, you must specify its full length (30 characters). Different subschema names must be unique in the first 6 characters.

When KDBS is used, the converter ensures the selection of the appropriate database.

## 9.5 Error messages

If an error is the cause of an interrupt, an appropriate message is output. To obtain further explanations, the user can enter the HELP command.

### Example

```

***(IN): DEF RECORD, LASTNAME, D=10, L=15
 (OUT): DMLTEST: USED NAME. STATUS IS NORMAL.
***(IN): HELP
 (OUT): LAST INPUT :DEF RECORD, LASTNAME, D=10, L=15
 (OUT): FROM SYSDBA STATUS IS NORMAL
 (OUT): ERROR DESCRIPTION :NAME IST SCHON FUER EINE DEFINITION VERGEBEN!
 (= name is already assigned to definition)

```

The designations used in the general error texts have the following significance:

|            |                       |
|------------|-----------------------|
| PARAM-NAME | Parameter or item     |
| DB-NAME    | Item in subschema     |
| ELEMENT    | Non-identifiable item |

### Overview of general error texts

```

COMMAND NOT FOUND
MISSING NECESSARY OPERAND
SYNTAX ERROR
OPERAND ERROR
OPERAND-NAME NOT FOUND
TOO MANY OPERANDS
INTERNAL BUFFER OVERFLOW
TOO LARGE DISTANCE
PARAM-NAME NOT FOUND
COMMAND ILLEGAL IN CONTEXT
EDT NOT AVAILABLE
DSCEXT WAS CALLED BY UDS1
ERROR ON RUN-FILE
NOT READY EXECUTED
DB-NAME UNKNOWN
DB-ERROR STATUS
ELEMENT NOT ALLOWED
NUMERICAL ERROR
USED NAME
WARNING

```

<sup>1</sup> If the user information area has been destroyed, it must be reconstructed by the user (see [page 124](#)).



---

# 10 Appendix

## 10.1 Status codes

### DML status codes

#### Status code giving information

001 FIND/FETCH format 1 or 7 with OR PRIOR/OR NEXT specification: No record has been found which matches the values given. The next record in the sort sequence was made available.

#### Status codes with progress information of the online utility

010 RELOCATE DML: Source and target levels are identical. Relocation has been completed.

REORGPPP DML: End of realms reached. Reorganization has been completed.

011 RELOCATE DML: Source and target levels are 0 when INITIALIZE=\*NO.

REORGPPP DML: Current page number is 0 when INITIALIZE=\*NO.

When an attempt is made to continue the relocations with INITIALIZE=\*NO, it is determined that no further information is available, e.g. because the database has been detached in the meantime or a new session section has been started.

012 RELOCATE DML: A locking conflict occurred with a parallel transaction when a source page was read.

REORGPPP DML: A locking conflict occurred with a parallel transaction when a page was read.

013 A locking conflict occurred with a parallel transaction when a target page was read.

**Status codes relating to data consistency:**

- 018 Deadlock status (mutual locking of several transactions involving UDS/SQL resources);  
FINISH WITH CANCEL is executed. It is advisable to repeat the transaction (a limited number of times). For UDS-D:  
In UDS/SQL applications without openUTM, global deadlock recognition is carried out by means of timeout monitoring (PP DEADTIME) of wait situations. Once the time limit has been exceeded, the status code 018 is indicated, even if there is no actual deadlock.
- 020 FIND/FETCH (only CALL DML)  
A page to be accessed is locked by another transaction.

**Status codes relating to retrieval of records:**

- 021 The end of a record type, set or realm has been reached.  
FIND/FETCH formats 2 (DUPLICATE) and 3 (USING):  
No record with the same values as the corresponding CRR or CRS can be found.  
FIND/FETCH format 3 (without USING):  
The end of the selected records has been reached.  
FIND/FETCH format 4:  
No NEXT or PRIOR record can be found or  
*integer* or *name* contains a value that addresses no record within the realm/record type/set occurrence.
- 022 The transaction attempts to open a database or realm which is locked for UPDATE and RETRIEVAL. Possible reasons for the lock are:  
Database level:  
  - The database administrator has locked the database with the DAL command ACCESS.
  - The DBDIR of the database is locked (see “realm level”).Realm level:  
  - The realm has been excluded from the database during restructuring.
  - The realm has been disconnected by the database administrator or by UDS/SQL error handling.
  - The realm has been locked by the database administrator, using the DAL command ACCESS.

- 023 In SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER only: no set occurrence which satisfies the set selection criteria can be found.
- 024 No record satisfying the record selection expression can be found.  
FIND/FETCH format 1:  
The database key does not return any records for one of the following reasons:  
– Its record type number does not match the record type specified.  
– Its value lies within the limits of its DBTT, but no associated record exists in the database.  
FIND/FETCH formats 2 (ANY) and 7:  
No record matching the initialized data elements or search expression can be found.  
FIND/FETCH format 4:  
No record can be found within the specified record type, realm or set occurrence.
- 027 The subscript of the specified item name does not lie within the range defined by the OCCURS clause in the subschema.
- 028 The specified database key contains an invalid record reference number or a record sequence number that lies outside the range of its DBTT.
- 029 FIND/FETCH formats 4 and 5:  
The Current of Realm or the Current of Set does not have the record type specified in the statement.

**Status codes relating to currency indicators:**

- 031 The Current of Realm, the Current of Set or the Current of Record type is not known.  
FIND/FETCH format 3:  
The Current of Set is owner and not a member of the specified set or the specified set name differs from the set name specified in the preceding FIND7.  
FIND/FETCH format 6 and format 7:  
The owner has been erased.  
IF format 2:  
The CRS has been erased or disconnected from the specified set.
- 032 The Current of Run Unit is not known or has been erased.
- 033 The Current of Run Unit does not have the record type specified in the statement.

**Status codes relating to naming conventions:**

- 042 Record type, set or realm is not defined in the called subschema; or  
an item which is part of an ASC, DESC or CALC key is not defined in the subschema; or  
following a subschema modification, the application program was not recompiled (COBOL DML) or the BCALLSI run was omitted (CALL DML); or  
error on the BIB interface (see status code 103); or  
in the case of an online utility a realm was specified in which no activities are permitted.
- 043 STORE and FIND/FETCH format 2:  
The AREA-ID data element contains the name of a realm which is not specified in the DDL WITHIN clause or does not belong to the called subschema or  
in SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER and owner record type = LOCATION MODE IS CALC: the AREA-ID item of the owner record contains the name of a realm which is not specified in the DDL WITHIN clause or does not belong to the called subschema.
- 044 IF:  
Specification of a dynamic set is not permitted.

**Status codes relating to data elements:**

- 051 Duplicate occurrences of key values in the database. This means that the execution of a DML statement would contradict a DUPLICATES ARE NOT ALLOWED specification in an ORDER IS SORTED BY DEFINED KEYS clause or SEARCH KEY clause of a set in which the record involved is a member, or the LOCATION MODE IS CALC or SEARCH KEY clause of the record involved.

**Status codes relating to records:**

- 071 FIND/FETCH format 2 (DUPLICATE), 3 and 5:  
The entry point of the DML statement (CRR, CRA or CRS) has been deleted or disconnected from the current set occurrence. If the records found are processed (FIND3 without USING), update operations of the user's own transaction do not result in the entry point being lost; only updates performed by foreign transactions cause this to happen.
- 072 ERASE:  
The record involved is owner of a non-empty set occurrence and therefore cannot be erased by the selected ERASE variant.



**Status codes relating to set membership:**

- 081   CONNECT (set-name):  
The CRU is already a member in one of the specified sets or one of the specified sets is not a member of the CRU.  
CONNECT (ALL):  
The CRU is already a member of all its member sets.  
MODIFY (set-name):  
One of the specified sets is not a member set of the CRU.  
MODIFY (ALL):  
The CRU is a member of none of its member sets.
- 082   DISCONNECT (set-name):  
The CRU is a MANDATORY member of one of the specified sets or one of the specified sets is not a member of the CRU  
DISCONNECT (FROM ALL):  
No member set of the CRU is OPTIONAL.
- 083   DISCONNECT (set-name) and MODIFY (set-name):  
The CRU is not a member in one of the specified sets  
DISCONNECT (FROM ALL):  
At least one member set of the CRU is OPTIONAL but the CRU is not a member of any of these OPTIONAL sets

**Status codes relating to a READY status:**

- 091   A realm is not in the READY status (i.e. a realm was not explicitly specified at READY or is not part of the current subschema), or  
realm names were explicitly specified at READY in an ERASE PERMANENT/SELECTIVE/ALL statement, or  
the DBTT of a record type to be relocated by the online utility is in an unopened realm.
- 092   No DML statement containing an update function is permitted in a RETRIEVAL processing chain,  
or the processing chain for an ERASE/PERMANENT/SELECTIVE/ALL was not opened with EXCLUSIVE UPDATE,  
or, if the P parameter is set to PP TA-ACCESS=SHARED, an attempt is made to open a processing chain in the usage mode PROTECTED or EXCLUSIVE.
- 093   The Database Handler (DBH) does not allow the processing chain, since the database involved is already open within the transaction ("second READY" within a processing chain).

- 099 (only CALL DML or online utility)  
When opening a transaction, a realm is locked by another transaction.

**Status codes relating to erroneous DML statements:**

- 101 FIND/FETCH format 4:  
The value zero has been specified for *integer* or *item-name* or a negative value was used in a search in a CHAIN that is not chained backward
- FIND/FETCH format 6:  
*set-name* must not identify a singular set.
- FIND/FETCH format 7:
- OR PRIOR or OR NEXT could not be executed, since no sorted and indexed key was found.
  - “WITHIN *set-name-1* USING *record-element-name-1*,...” was specified. The specification of a dynamic set in *set-name-1* is not permitted.
  - LIMITED BY *dynamic-set* ... SORTED BY ... was specified.  
It is not possible to sort the intersection of a selected set (hit list) and a dynamic set.
  - LIMITED BY *sorted-dynamic-set* ... was specified.  
It is not possible to create the intersection of a selected set and a sorted dynamic set.
- FINISH  
Type of FINISH (with or without CANCEL) cannot be identified.
- 102 SET, ACCEPT (format 1):  
A large database key value (database key value with a REC-REF > 254 and/or an RSQ > 2<sup>24</sup>-1) cannot be copied to an item of type USAGE IS DATABASE-KEY.  
A subschema must be used in which SUBSCHEMA FORM IS OLD is not specified and which was created in UDS/SQL V2.0 or higher. The field specified must also be of the type USAGE IS DATABASE-KEY-LONG.
- 103 Error in the BIB interface.  
Possible cause: incorrect COBOL compiler or incorrect COBOL runtime system, error in the CALL-DML converter, in IQS, in the online utility, or in a utility routine which generates BIBs, or error in the Database Handler.

**Status codes relating to system errors:**

- 113 A serious error was detected in the Database Handler or in the database on accessing a database page.

**Status codes relating to UDS/SQL resources:**

- 122 The transaction has been prematurely terminated with CANCEL by the DBH; possible causes for this are:
- RLOG file too small or split too often
  - UDS/SQL buffer too small, increase PP BUFFERSIZE=n.
  - Transaction rolled back due to a deadlock resolution that has been executed in the meantime.
  - Intervention by the database administrator with DAL (commands ABORT, PERFORM, CLOSE).
  - New update transactions during writing of a checkpoint.
  - Occurrence of a file or programming error which can be bypassed (temporarily) by executing CANCEL for the transaction.
  - An error in a DML statement which cannot be rolled back on its own, and thus requires a CANCEL of the complete transaction. The database administrator was notified (via a UDS/SQL message).
  - For UDS-D:  
Transaction rollback can also be due to errors or administrator intervention in a remote configuration (e.g. ABORT, CLOSE CALLS, CLOSE RUN-UNITS, %TERM) or to errors in the link to a remote configuration.
- 123 The transaction attempts to open a realm, which is locked against updates, with READY USAGE-MODE UPDATE. The lock is due to one of the following reasons:
- Configuration level:  
The current session of the independent DBH was started without RLOG logging (PP LOG=NO).
- Opening of the RLOG file was unsuccessful, which means that RLOG logging is currently impossible.
- Database level:
- The database is being activated as a SHARED RETRIEVAL database.
  - The database is not an original database, but a backup database.
  - Opening of a new ALOG file was unsuccessful, which means that AFIM logging is currently impossible.
  - The database administrator has locked the database against updates by means of the DAL command ACCESS.
  - The DBDIR of the database is locked for updates (see “realm level”).

Realm level:

- The database administrator has locked the realm against updates by means of the DAL command ACCESS.
- The transaction attempts to open a realm of a remote database although the current session runs without RLOG logging (due to PP LOG=NO or unsuccessful open of RLOG file); there is thus no basis for the two-phase commit protocol of distributed transactions.

124 The transaction was rolled back prematurely by the DBH with CANCEL.

Cause:

New update transaction or update processing chain when writing a checkpoint or switching the RLOG file.

This status code is set if the load parameter PP ORDER-DBSTATUS=SPECIAL was specified for the current session. Otherwise, status code 122 is set under the conditions indicated above.

131 The Database Handler does not allow the transaction since the maximum permitted number of parallel transactions or user tasks, which was specified by the load parameter TRANSACTION when the Database Handler was loaded, has been reached.

132 The Database Handler does not allow the transaction since the maximum permitted number of subschemas, which was specified by the load parameter SUBSCHEMA when the Database Handler was loaded, has been reached.

#### **Status codes relating to the sequence of DML statements:**

134 The Database Handler does not allow the DML statement since no transaction is open.

136 A DML statement, though belonging to an existing transaction, is rejected because it refers to a database (supplies a DB reference) for which no processing chain of the transaction currently exists.

137 It is not possible to mix SQL and non-SQL statements in a transaction (exception: accessing different UDS/SQL configurations via openUTM). A mixture of COBOL DML and CALL DML statements in a processing chain is not permitted.

**Status codes relating to subschemas:**

141 The transaction has specified an invalid or unknown subschema name, or the first 6 characters of the subschema name are not unique in the current DB configuration, or the database involved is not active.

For UDS-D:

The specified subschema is

- not in the local configuration and not in the distribution table.
- in the distribution table, but not in the corresponding UDS/SQL configuration.
- in the distribution table, but the corresponding UDS/SQL configuration is not accessible, because
  - a) the computer is not accessible,
  - b) the configuration is not running or is running without distribution active.
- in the distribution table but locked, or the related database or configuration is locked.
- not in the local configuration and UDS-D has not been started in the local configuration.

The number of remote databases addressed by this transaction exceeds the value PP DISDB.

142 The subschema description in DBDIR (SSIA) has been destroyed. Repeat BGSSIA run.

144 The DML statement specifies a different subschema from the one specified in the current READY statement (subschemas reference).

145 The subschema specified in the READY statement cannot be processed because it does not match the current status of the schema (subschemas DDL compilation and/or BGSSIA run missing after database restructuring), or the READY statement is rejected because the UDS/SQL version does not match the database:

- The database was set for the year-2000-compatible processing of two-digit year fields or this setting was not correctly removed. Therefore the statement can only be processed with a UDS/SQL version as of V2.0B30.
- A subschema contains national Daten (Unicode: UTF-16, PICTURE N, USAGE NATIONAL). Therefore it may only be processed with a version as of UDS/SQL V2.5.

- 146 COBOL DML: The subschema with which the module of the current DML statement was compiled does not correspond with the current status of the database.  
CALL DML: The SSITAB module used does not correspond with the current status of the database.

**Status codes relating to DBH availability:**

- 151 The Database Handler is not yet available or is being terminated normally (termination in progress).
- 152 The Database Handler has been terminated abnormally.
- 154 An irrecoverable error has been detected in UDS/SQL; the program should be terminated (STOP RUN for COBOL programs). The transaction was not completed.
- 155 While UDS/SQL was processing a DML statement, a further DML statement for the same transaction was receive (deserialization).  
Possible cause of error:  
Asynchronous activities performed by the user program (e.g. DML statement in STXIT routine) or UDS/SQL system error.

**Further status codes of the UDS online utility**

- 161 A transaction of an online utility is already active in the same realm.
- 162 A user transaction which is running in parallel has activated an online realm extension and thus temporarily hindered the online utility.
- 163 The online utility is not permitted in a temporary realm.
- 164 USAGE-MODE EXCLUSIVE UPDATE is required for this RELOCATE type
- 165 The specified SET is not a distributable list
- 166 The specified realm is not permitted for this record type
- 167 Contending change of a parallel user TA. The utility TA is reset.

**Status codes relating to FIND/FETCH:**

- 183 The search expression exceeds the maximum length.
- 184 The temporary realm is not available.
- 191 Both the object set and the LIMITED set are dynamic sets.
- 192 The LIMITED set is empty.
- 193 FIND/FETCH format 7: The LIMITED set contains a different record type from the object set  
FIND/FETCH formats 4 and 7:  
The object set is dynamic and contains a record type different from the one specified.  
FIND/FETCH format 3:  
The specified record name differs from the record name specified in the preceding FIND/FETCH format 7.
- 194 The comparison value or sort item has the length 0 or a length that is not permitted for the item type.
- 195 The comparison value or sort item has an unknown item type or the comparison value contains incompatible data.
- 197 No preceding FIND/FETCH format 7.
- 198 The CRS of the result set has been disconnected from the object set or connected to another occurrence by a different transaction.

**Status codes relating to interoperation with *openUTM*:**

- 200 FINISH:  
The FINISH statement has been accepted, but the execution of FINISH will be delayed until the openUTM end-of-transaction call to the DC controller (PEND). No further DML statements are accepted.
- 201 A further DML statement was issued after the pended FINISH. The DML statement is ignored.
- 218 Deadlock involving more than one system that can only be resolved by releasing the openUTM application task (e.g. with PEND RS).

*Examples:*

- local UDS/SQL-openUTM operation:  
Deadlock between UDS/SQL resources (data) and openUTM resources (tasks).
- Distributed processing via UDS-D or openUTM-D:  
Deadlock between UDS/SQL resources (data) and/or openUTM resources (tasks).

This type of deadlock is recognized by means of timeout monitoring of wait situations (PP DEADTIME). When this time limit is exceeded, status code 218 is indicated, even if no actual deadlock has occurred.

**Status codes relating to LOOK:**

- 781 Element not found or realm name unknown to the online utility.
- 782 No next element available.
- 783 One element in the list not found.
- 784 The item reference entered does not exist. The description with the next smallest item reference was output.
- 785 The result vector of a compound LOOKC function must be retrieved by a contiguous sequence of corresponding LOOKC statements.
- 786 The record type cannot be processed with this subschema because it contains data of a type which is not known to the application program.
- 789 The specified subschema does not exist.



**Status codes relating to allocation of memory space or database key:**

- 802 The memory space in the realm is exhausted or an activated online realm extension has failed. The record involved cannot be stored or inserted in a set occurrence.
- 804 No further database key is available for storage of a new record or an activated online realm extension has failed.
- 805 The system address space of the DBH is exhausted. The DBH tables can no longer be extended dynamically. The database administrator was notified.

**Status codes relating to variable-length items and compression:**

- 888 The length of the variable item is greater than that defined in the subschema or is negative.
- 898 STORE/MODIFY format 2 is not allowed for variable-length items.
- 899 STORE:  
The number of items to be stored is so great that the size of the compressed record is greater than a page.
- GET:  
One of the desired items is not present in the compressed record in the database.
- MODIFY format 1:  
This format is not allowed if the record accessed is present in compressed form.
- MODIFY format 2:  
One of the items to be modified is not present in the compressed record.

**Status codes relating to access rights:**

- 901 Access to a realm, record or set is not permitted within the user group or the utility routine ONLINE-PRIVACY or ONLINE-UTILITY is trying to access a database which is not contained in the utility routine's execution USER-ID. It is not possible to bypass this behavior of the utility routines by setting the P parameter PRIVACY-CHECK to OFF.
- 950 User group unknown (see the "[Creation and Restructuring](#)" manual, BPRIVACY).
- 954 No access authorization has been defined for the specified user group.

## CALL DML status codes

### DML optional entry error:

- C00 The specified function code is not correct.
- C01 The specified function option is not allowed with the specified function code.
- C02 The specified secondary option is not allowed with the specified combination of function code and function option, or it contains syntax errors.

### Record name error:

- C03 The specified record name is not present in the relevant subschema or is not unique.
- C04 A mandatory record name has not been specified.

### Set name error:

- C05 The specified set name is not present in the current subschema or is not unique.
- C06 Syntax error in the set name list  
(too many set names; incorrect separators or terminators for set names; set name occurs more than once)

### Realm name error:

- C07 The specified realm name is not present in the current subschema or is not unique.
- C08 Syntax error in the realm name list  
(too many realm names; incorrect separators or terminators for realm names; realm name occurs more than once)

### Item name error:

- C09 The specified item name is not present in the relevant record of the current subschema or is not unique.
- C10 Syntax error in the item name list  
(too many item names; incorrect separators or terminators for item names)

**Result of IF statement:**

- C11 The IF condition is not satisfied.  
C11 should not be regarded as an error code but rather as the result of the DML statement IF; the code is 000 if the condition is satisfied.

**Search expression error:**

- C20 The search expression contains too many search conditions.
- C21 A NXT search condition after an OR operator is not allowed.
- C22 The separator before and after an item name or relational operator in a search condition must always be a space.
- C23 The number of parentheses in a NXT search condition must be equal to zero.
- C24 The mask for a search condition may only consist of the characters 0 and 1 and must be terminated with a space.
- C25 A NXT search condition may not be enclosed in parentheses.
- C26 The length of the mask for a search condition must be the same as the length of the item.
- C27 NXT search conditions may only be located at the end of a search expression.
- C28 A search condition is not terminated with `_OR_`, `_AN_` or `_END_`.
- C29 The length of a value in a search condition is incorrect.
- C30 The number of right-hand parentheses in a search condition is not numeric.
- C32 There are more left-hand than right-hand parentheses in a search expression.
- C33 The NEQ relation is not allowed in a NXT search expression.
- C34 The relational operator in a search condition is not correct.
- C35 The number of left-hand parentheses in a search condition is not numeric.
- C37 Too many right-hand parentheses have been specified in a search condition.
- C38 The relational operator in a search condition is not followed by a space.
- C39 The item name of a search condition is not present in the current subschema or is not unique.
- C40 The item type of a search condition is printable numeric but the associated comparison value is not.
- C41 The item type of a search condition is packed decimal but the associated comparison value is not.

C42 Search conditions are not allowed for this item type.

**Retaining entry error:**

C61 The specified retaining option (special parameter 1) is not correct.

C62 A specified retaining set name (special parameter 1) is not present in the current subschema or is not unique.

C63 Syntax error in the retaining set name list  
(too many set names; incorrect separators or terminators for set names; set name occurs more than once).

**Other errors:**

C66 The SSITAB module of the subschema cannot be identified, or the specified subschema name matches the one in the SSITAB module only in the first 6 characters, but not in the full length.  
Execute BCALLSI run.

C72 The integer indicating the record position in a FIND4/FTCH4 call must not be zero.

**Specific FIND7A/FTCH7A errors:**

C74 The specified name of the limited set is not present in the current subschema or is not unique.

C75 The specified name of the result set is not present in the current subschema or is not unique.

**Specific LOOKC errors:**

C80 The number of LOOKC blocks must be between 1 and 255 (inclusive).

**User communication errors:**

C90 A work buffer of the size needed by the UDSCDML converter module cannot be made available. If necessary, the communication pool must be enlarged (see the [“Database Operation”](#) manual).

C91 The error exit DSCEXT was not defined.

C94 The converter module UDSCDML is not present.

C95 The SSITAB module generated by BCALLSI is not present or could not be loaded in the memory (e.g. due to a lack of memory space).

- C98 An attempt is made to execute ACCPTL, FIND1L, FTCH1L, STORE1L or STORE2L with an SSITAB module which was generated before UDS/SQL V2.0 or with a "FORM IS OLD" subschema. An SSITAB module with UDS/SQL V2.0 or higher is required to execute the specified functions.
- C99 The SSITAB module is invalid or incompatible with the version of the CALL DML translating routine.

**Validity check on DML statements based on the subschema structure:**

- P01 A FIND2/FTCH2 statement with optional parameter ANY... is only allowed if LOCATION MODE IS CALC is specified and all keys of the record type are present in the subschema.
- P02 A FIND2/FTCH2 statement with optional parameter DUPLIC is only allowed if LOCATION MODE IS CALC and DUPLICATES ARE ALLOWED are specified and all keys of the record type are present in the subschema.
- P03 Duplicates are not allowed for the current FIND3/FTCH3 statement.
- P04 A FIND7A/FTCH7A statement is only allowed if the referenced record type is a member of the specified set.
- P05 A FIND7A/FTCH7A statement for SET OCCURRENCE IS THRU LOCATION MODE OF OWNER is:
- only allowed in connection with LOCATION MODE IS DIRECT if the item involved is present in the subschema.
  - only allowed in connection with LOCATION MODE IS CALC if all keys of the record type are present in the subschema.
- P06 A FIND4/FTCH4 or FIND5/FTCH5 statement is only allowed if the specified record type is a member of the specified set.
- P07 A FIND4/FTCH4 or FIND5/FTCH5 statement is only allowed if the specified record type is permissible in the specified realm.
- P08 A FIND6/FTCH6 statement is only allowed if the set involved is not a SYSTEM set.
- P09 The form of storage specified for the set does not allow CONNEC or DISCON statements, or, in the case of DISCON ALLFRM, the set specified is not a dynamic set.
- P10 In the set name list of a CONNEC or DISCON statement, only sets which have the same record type as member are allowed.
- P11 For a CONNEC or DISCON statement, the Current of Run Unit must belong to the member record type of the specified set.

- P12 For a CONNEC TO-ALL statement, the subschema must contain at least one set with the referenced record type which is not MANDATORY AUTOMATIC. For a DISCON FRMALL statement, the referenced record type must be OPTIONAL member in at least one set of the subschema.
- P13 The specified MODIF1/2 statement is not allowed.
- P14 The specified STORE1/2 statement is not allowed.
- P15 The specified ERASEC statement is not allowed.
- P16 The set specified in the RESULT and/or LIMITED clause is not a dynamic set.

## 10.2 Description of the "MAIL-ORDERS" schema for the sample database SHIPPING

```

SCHEMA NAME IS MAIL-ORDERS
PRIVACY LOCK FOR COPY IS "SHIP-KEY".
*
*
*
AREA NAME IS CUSTOMER-ORDER-RLM.
AREA NAME IS PURCHASE-ORDER-RLM.
AREA NAME IS CLOTHING.
AREA NAME IS HOUSEHOLD-GOODS.
AREA NAME IS SPORTS-ARTICLES.
AREA NAME IS FOOD.
AREA NAME IS LEISURE.
AREA NAME IS STATIONERY.
AREA NAME IS ARTICLE-RLM.
AREA NAME IS SEARCH-RLM
AREA IS TEMPORARY.
*
*
*
RECORD NAME IS CUSTOMER
LOCATION MODE IS DIRECT-LONG CUST-NO OF CUSTOMER
WITHIN CUSTOMER-ORDER-RLM.
*
01 CUST-NAME TYPE IS CHARACTER 30.
01 CUST-F-NAME TYPE IS CHARACTER 30.
01 CUST-NO TYPE IS DATABASE-KEY-LONG.
*
*
RECORD NAME IS CST-ORDERS
WITHIN CUSTOMER-ORDER-RLM.
*
01 ORD-NO PICTURE IS 9(4).
01 ORD-YEAR PICTURE IS 99.
01 ORD-MONTH PICTURE IS 99.
01 ORD-DAY PICTURE IS 99.
01 ORD-STATUS PICTURE IS X.
*
*
RECORD NAME IS ORD-ITEM
WITHIN CUSTOMER-ORDER-RLM.
*
01 ORD-NO-ITEM PICTURE IS 99.
01 ORD-QTY TYPE IS DECIMAL 6.
01 PAY-INSTAL-CODE PICTURE IS X.
01 ORD-STATUS-ITEM PICTURE IS X.
*
*
RECORD NAME IS INSTALMENT
WITHIN CUSTOMER-ORDER-RLM
SEARCH KEY IS YEAR-NEXT-INSTAL, MONTH-NEXT-INSTAL,
DAY-NEXT-INSTAL
USING INDEX NAME IS SEARCH-TAB-INSTALMENT
DUPLICATES ARE ALLOWED.

```

```

*
01 ORD-NO PICTURE IS 9(4).
01 ORD-NO-ITEM PICTURE IS 99.
01 TOT-PRICE-INSTAL TYPE IS DECIMAL 9,2.
01 SINGLE-INSTAL TYPE IS DECIMAL 7,2.
01 BALANCE TYPE IS DECIMAL 9,2.
01 YEAR-NEXT-INSTAL PICTURE IS 99.
01 MONTH-NEXT-INSTAL PICTURE IS 99.
01 DAY-NEXT-INSTAL PICTURE IS 99.
*
*
RECORD NAME IS ART-TYPE
 WITHIN CLOTHING, HOUSEHOLD-GOODS, SPORTS-ARTICLES, FOOD,
 LEISURE, STATIONERY AREA-ID IS RLM-SELECTION-1
SEARCH KEY IS ART-NAME USING CALC
 NAME IS SEARCH-TAB-ART-TYPE DUPLICATES ARE ALLOWED.
*
01 ART-NAME TYPE IS CHARACTER 25.
*
*
RECORD NAME IS ART-SELECTION
 WITHIN CLOTHING, HOUSEHOLD-GOODS, SPORTS-ARTICLES, FOOD,
 LEISURE, STATIONERY AREA-ID IS RLM-SELECTION-2
SEARCH KEY IS SEL-CRIT USING INDEX
 NAME IS SEARCH-TAB-ARTICLE-SELECTION
 DUPLICATES ARE ALLOWED.
*
01 SEL-CRIT TYPE IS CHARACTER 25.
*
*
RECORD NAME IS ART-DESCR
 LOCATION MODE IS CALC USING ARTICLE-NAME
 DUPLICATES ARE ALLOWED
 WITHIN CLOTHING, HOUSEHOLD-GOODS, SPORTS-ARTICLES, FOOD,
 LEISURE, STATIONERY AREA-ID IS RLM-SELECTION-3.
*
01 ART-NO PICTURE IS 9(6).
01 ARTICLE-NAME TYPE IS CHARACTER 40.
01 MATERIAL OCCURS 4 TIMES.
 02 PERZENT PICTURE IS 99.
 02 MAT-CODE PICTURE IS X.
01 LENGTH-FIELD TYPE IS BINARY 15.
01 ART-INFO PICTURE IS LX(500)
 DEPENDING ON LENGTH-FIELD.
*
*
RECORD NAME IS ARTICLE
 LOCATION MODE IS CALC USING ART-NO, COL-NO, ART-SIZE
 DUPLICATES ARE NOT ALLOWED
 WITHIN CLOTHING, HOUSEHOLD-GOODS, SPORTS-ARTICLES, FOOD,
 LEISURE, STATIONERY AREA-ID IS RLM-SELECTION-4
SEARCH KEY IS ART-NO-AVAIL, COL-NO-AVAIL, ART-SIZE
 USING CALC NAME IS SEARCH-TAB-ARTICLE-1
 DUPLICATES ARE NOT ALLOWED
SEARCH KEY IS ARTICLE-NAME USING CALC
 NAME IS SEARCH-TAB-ARTICLE-2 DUPLICATES ARE ALLOWED.
*
01 ART-NO PICTURE IS 9(6).

```



```

01 COL-NO PICTURE IS 99.
01 ARTICLE-NAME TYPE IS CHARACTER 40.
01 ART-NO-AVAIL PICTURE IS 9(4).
01 COL-NO-AVAIL PICTURE IS 99.
01 ART-SIZE PICTURE IS 99.
01 PRICE TYPE IS DECIMAL 7,2.
01 INSTALMENT-PRICE TYPE IS DECIMAL 7,2.
01 MAX-STOCK TYPE IS DECIMAL 10.
01 MIN-STOCK TYPE IS DECIMAL 3.
01 CURR-STOCK TYPE IS DECIMAL 10.
01 STATISTICS TYPE IS DECIMAL 15.
01 NOT-AVAIL-CODE PICTURE IS X.
*
*
RECORD NAME IS SUBSET
 WITHIN HOUSEHOLD-GOODS, SPORTS-ARTICLES
 AREA-ID IS RLM-SELECTION-5.
*
01 QUANTITY PICTURE IS 99.
*
*
RECORD NAME IS COLORS
 WITHIN ARTICLE-RLM
 SEARCH KEY IS COL-NAME USING CALC DUPLICATES ARE NOT ALLOWED
 SEARCH KEY IS COL-NO USING CALC DUPLICATES ARE NOT ALLOWED.
*
01 COL-NO PICTURE IS 99.
01 COL-NAME TYPE IS CHARACTER 20.
*
*
RECORD NAME IS MATERIALS
 WITHIN ARTICLE-RLM
 SEARCH KEY IS MAT-CODE USING INDEX
 NAME IS SEARCH-TAB-MATERIAL-1 DUPLICATES ARE NOT ALLOWED
 SEARCH KEY IS MAT-NAME USING INDEX
 NAME IS SEARCH-TAB-MATERIAL-2 DUPLICATES ARE NOT ALLOWED.
*
01 MAT-CODE TYPE IS CHARACTER 1.
01 MAT-NAME TYPE IS CHARACTER 20.
*
*
RECORD NAME IS SUPPLIER
 LOCATION MODE IS CALC USING SUPPL-NO, SUPPL-NAME
 DUPLICATES ARE NOT ALLOWED
 WITHIN PURCHASE-ORDER-RLM.
*
01 SUPPL-NO PICTURE IS 9(5).
01 SUPPL-NAME TYPE IS CHARACTER 30.
01 SUPPL-PCODE TYPE IS CHARACTER 4.
01 SUPPL-TOWN TYPE IS CHARACTER 30.
01 SUPPL-STREET TYPE IS CHARACTER 30.
01 SUPP-STREET-NO TYPE IS CHARACTER 3.
01 SUPPL-TEL PICTURE IS 9(12).
01 SUPPL-POBOX PIC 9(4).
01 SUPP-TELEX PIC 9(12).
*
*
RECORD NAME IS PURCHASE-ORDER

```

```

 WITHIN PURCHASE-ORDER-RLM.
*
01 P-ORD-NO PICTURE IS 9(4).
01 P-ORD-YEAR PICTURE IS 99.
01 P-ORD-MONTH PICTURE IS 99.
01 P-ORD-DAY PICTURE IS 99.
*
*
RECORD NAME IS P-ORD-ITEM
 WITHIN PURCHASE-ORDER-RLM.
*
01 P-ORD-NO-ITEM PICTURE IS 99.
01 P-ORD-QTY TYPE IS DECIMAL 10.
*
*
*
SET NAME IS CST-ORD-PLACED
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE NOT ALLOWED
OWNER IS CUSTOMER.
MEMBER IS CST-ORDERS OPTIONAL AUTOMATIC
ASCENDING KEY IS ORD-NO
SEARCH KEY IS ORD-YEAR, ORD-MONTH, ORD-DAY USING INDEX
 NAME IS SEARCH-TAB-C-O-PLCD DUPLICATES ARE ALLOWED
 SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS CST-ORD-CONTENTS
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE NOT ALLOWED
OWNER IS CST-ORDERS.
MEMBER IS ORD-ITEM MANDATORY AUTOMATIC
ASCENDING KEY IS ORD-NO-ITEM
 SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS OUTSTANDING
ORDER IS LAST
OWNER IS CUSTOMER.
MEMBER IS ORD-ITEM OPTIONAL AUTOMATIC
 SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS HIRE-PURCHASE
ORDER IS LAST
OWNER IS CUSTOMER.
MEMBER IS INSTALMENT MANDATORY AUTOMATIC
 SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS OFFER
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE ALLOWED
OWNER IS ART-TYPE.
MEMBER IS ART-DESCR MANDATORY AUTOMATIC
ASCENDING KEY IS ARTICLE-NAME
 SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*

```

```

*
SET NAME IS SHORT-LIST
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE ALLOWED
OWNER IS ART-SELECTION.
MEMBER IS ART-DESCR MANDATORY AUTOMATIC
ASCENDING KEY IS ARTICLE-NAME
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS P-ORD-SPEC
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE NOT ALLOWED
OWNER IS ART-DESCR.
MEMBER IS ARTICLE MANDATORY AUTOMATIC
ASCENDING KEY IS COL-NO, ART-SIZE
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS MIN-STOCK-LEVEL
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE NOT ALLOWED
OWNER IS SYSTEM.
MEMBER IS ARTICLE OPTIONAL MANUAL
ASCENDING KEY IS ART-NO, COL-NO, ART-SIZE.
*
*
SET NAME IS CONTAINING
ORDER IS NEXT
OWNER IS ARTICLE.
MEMBER IS SUBSET MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS CONTAINED-IN
ORDER IS NEXT
OWNER IS ARTICLE.
MEMBER IS SUBSET MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
 ALIAS FOR ART-NO IS SUBST-ART-NO
 ALIAS FOR COL-NO IS SUBST-COL-NO
 ALIAS FOR ART-SIZE IS SUBST-SIZE.
*
*
SET NAME IS SUPPLIERS
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE NOT ALLOWED
OWNER IS SYSTEM.
MEMBER IS SUPPLIER MANDATORY AUTOMATIC
ASCENDING KEY IS SUPPL-NAME, SUPPL-NO.
*
*
SET NAME IS ARTICLES-AVAILABLE
ORDER IS SORTED INDEXED BY DEFINED KEYS
 DUPLICATES ARE ALLOWED
OWNER IS SUPPLIER.
MEMBER IS ARTICLE MANDATORY AUTOMATIC
ASCENDING KEY IS ARTICLE-NAME

```

```

SEARCH KEY IS NOT-AVAIL-CODE USING INDEX
NAME IS SEARCH-TAB-ART-AVAIL DUPLICATES ARE ALLOWED
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS ORDERED-ARTICLES
ORDER IS LAST
OWNER IS ARTICLE.
MEMBER IS ORD-ITEM MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS REORDERED-ARTICLES
ORDER IS LAST
OWNER IS ARTICLE.
MEMBER IS P-ORD-ITEM MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.
*
*
SET NAME IS P-ORD-PLACED
ORDER IS LAST
OWNER IS SUPPLIER.
MEMBER IS PURCHASE-ORDER MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS P-ORD-RECEIVED
ORDER IS FIRST
OWNER IS SUPPLIER.
MEMBER IS PURCHASE-ORDER MANDATORY MANUAL
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
SET NAME IS P-ORD-CONTENTS
ORDER IS NEXT
OWNER IS PURCHASE-ORDER.
MEMBER IS P-ORD-ITEM MANDATORY AUTOMATIC
SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
*
*
*
SET NAME IS RESULT-SET
SET IS DYNAMIC
ORDER IS IMMATERIAL
OWNER IS SYSTEM.
*
SET NAME IS LIMITED-SET
SET IS DYNAMIC
ORDER IS IMMATERIAL
OWNER IS SYSTEM.
*
*
SET NAME IS IQL-DYN1
SET IS DYNAMIC
ORDER IS IMMATERIAL
OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN2

```

```
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN3
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN4
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN5
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN6
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN7
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
SET NAME IS IQL-DYN8
 SET IS DYNAMIC
 ORDER IS IMMATERIAL
 OWNER IS SYSTEM.
*
*
*
***** VERSION 14.02.02 *****
```

## 10.3 UDS/SQL-openUTM return codes

Four bytes of the DB trace information contain task-specific DB return codes (see the openUTM manual “[Messages, Debugging and Diagnostics in BS2000](#)”, DB-DIAGAREA). These codes are version-dependent and only applicable to the specific version involved.

The return codes have the following structure: **abcd**

### **a contains:**

- X'04' Error in eventing (ENABLE)
- X'08' Serialization error
- X'0C' Error in eventing
- X'10' Error in LINK
- X'14' Communication pool no longer contains enough free space
- X'18' Internal error in UDSCON
- X'1C' Memory error
- X'20' Unknown keyword
- X'24' START parameters not valid for UDS /SQL
- X'28' Length specification for START parameters is too small
- X'2C' DMS error returned to status query
- X'30' No application ENTRY available
- X'34' Addressed application ENTRY missing
- X'38' Communication pool closed normally
- X'3C' Communication pool closed abnormally
- X'40' Irrecoverable error in memory administration
- X'44' Error in status message
- X'48' Status of transaction ambiguous
- X'4C' Error in FORWARD eventing
- X'50' Violation of recovery requirements
- X'54' Error in program management
- X'58' Error in result transfer
- X'5C' Task deadlock in TIAM application

X'60' Error in transfer data

X'64' Lifetime exceeded for job sent to master task

**b contains:**

Additional information for diagnostic purposes (for some errors)

**c contains:**

Additional information for diagnostic purposes (for some errors)

In some error situations which are already isolated sufficiently by d, additional information is provided in abc for diagnostic purposes.

**d contains:**

X'00' Function executed without error

X'04' Invalid operation code

X'08' Internal error (see meaning under a)

X'0C' No connection to UDS subtasks

X'10' UDS terminated

X'14' Transaction does not exist (any more)

X'18' Transaction terminated abnormally

X'1C' Transaction could not be opened due to lack of tables

X'20' Transaction termination fails

X'24' Error in START parameter

X'28' Errored termination of a PTC transaction

X'2C' Processing error

X'30' 2nd PTC from user

X'34' User error in CDML

X'38' Processing error during FINISH

X'3C' Processing error during FINISH WITH CANCEL

X'40' Error returned to status query

X'44' Irrecoverable CDML error

X'48' READY error in KDBS

---

|       |                                                                         |
|-------|-------------------------------------------------------------------------|
| X'4C' | No connection possible - all channels occupied                          |
| X'50' | Invalid communication name                                              |
| X'54' | Addressed communication pool does not exist                             |
| X'58' | Communication pool not fully created yet                                |
| X'5C' | Task does not exist for which an asynchronous CANCEL has been attempted |
| X'60' | Channel active                                                          |
| X'64' | Start BIB: User requests connection                                     |
| X'68' | Stop BIB: User requests connection cleardown                            |
| X'6C' | User already connected with UDS                                         |
| X'70' | Versions of the UDS modules mixed                                       |
| X'74' | Error in distributed processing                                         |
| X'78' | Illegal BS2000 version                                                  |
| X'7C' | Subschema change during a transaction                                   |
| X'80' | Serialization error                                                     |
| X'84' | User's own task cannot be aborted asynchronously                        |
| X'88' | No BREAK TA precedes for CONTINUE TA                                    |
| X'8C' | User entered invalid parameters                                         |
| X'90' | Deadlock                                                                |
| X'94' | READY for transaction missing                                           |
| X'98' | Irrecoverable error in open PST                                         |
| X'9C' | Error in master task message                                            |
| X'A0' | Error in update recognition                                             |
| X'A4' | PETA only possible for update transaction                               |
| X'A8' | SQL conversation could not be opened due to lack of tables              |
| X'AC' | No SQL OUTPUT could be created                                          |
| X'B0' | SQL not available in loaded DBH                                         |
| X'B4' | Length in the conversation memory changed incorrectly                   |
| X'B8' | UDS not addressable in AMODE3                                           |
| X'BC' | Application and AMODE not compatible                                    |



- X'C0' Call not permitted in secured UDS/SQL configuration
- X'C4' Illegal module library
- X'C8' No connection to UDS/SQL configuration possible
- X'CC' Inconsistent DML request

## 10.4 Additional diagnostic information in openUTM

openUTM documents events that occur in task-specific trace areas which are written cyclically. Requests sent to the database system are also documented. The 'Secondary DB Trace Information' field in particular is relevant for UDS/SQL. UDS/SQL stores data about the individual request there which you can use to analyze runtime information and to diagnose errors. However, sometimes this information can only be used in conjunction with other diagnostic documents (e.g. a dump) since the data items from which the data was obtained are only used within UDS/SQL. In this respect the fields can only be interpreted in connection with the UDS/SQL version from which they originated.

The 'Secondary DB Trace Information' field is 32 bytes long. It is a component part of a trace record (DB record) of the DB-DIAGAREA. In openUTM v5.3, the DB record is contained in the UTM-DIAGAREA (see the openUTM manual "[Messages, Debugging and Diagnostics in BS2000](#)", DB-DIAGAREA and UTM-DIAGAREA).

The secondary DB trace information from UDS/SQL is structured as follows:

| Byte(s) | Meaning                                                                                                                                                                                                |                                           |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| 1-4     | Version<br>The version string ('U01_', 'U02_', or 'U03_') serves to identify the trace information and to differentiate between pieces of information with the same contents but in different formats. |                                           |
| 5-6     | Type of request<br>Two characters are used to record the type of request sent by openUTM to UDS/SQL.                                                                                                   |                                           |
|         | CB                                                                                                                                                                                                     | COBOL-DML                                 |
|         | CD                                                                                                                                                                                                     | CALL-DML (including KDBS request)         |
|         | CN                                                                                                                                                                                                     | Connection                                |
|         | DC                                                                                                                                                                                                     | Disconnection                             |
|         | FN                                                                                                                                                                                                     | End of transaction                        |
|         | PA                                                                                                                                                                                                     | Passing of start parameters               |
|         | PB                                                                                                                                                                                                     | Special request from COBOL runtime system |
|         | RB                                                                                                                                                                                                     | Continuation of a task of an open TA      |
|         | SB                                                                                                                                                                                                     | Interruption of a task of an open TA      |
|         | SQ                                                                                                                                                                                                     | SQL request                               |
| ST      | Status request from openUTM                                                                                                                                                                            |                                           |
| 7       | openUTM opcode 1                                                                                                                                                                                       |                                           |
| 8       | openUTM opcode 2                                                                                                                                                                                       |                                           |
| 9-32    | Different meanings depending on the version and type of request defined in bytes 1-6 (see the following table).                                                                                        |                                           |

Table 41: Structure of the secondary DB trace information field

The following table contains the meanings of bytes 9-32 of the secondary DB trace information field for the various versions and types of requests.

**Bytes 1-6: U01 CB**

| <b>Byte(s)</b> | <b>Meaning</b>                                                                                                                                      |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 9-12           | Dynamically assigned internal UDS/SQL transaction ID                                                                                                |
| 13             | DML request code 1 in the BIB                                                                                                                       |
| 14             | DML request code 2 in the BIB                                                                                                                       |
| 15             | Dynamically assigned internal UDS/SQL number of the set type accessed in the DML for old style BIBs                                                 |
| 16             | Dynamically assigned internal UDS/SQL number of the set or realm used in the DML for old style BIBs                                                 |
| 17-19          | Status code of the DML used                                                                                                                         |
| 20             | Code stating if the status code recorded in bytes 17-19 matches the status code passed to the user in the BIB ('O' when matched, 'B' when no match) |
| 21             | Dynamically assigned internal UDS/SQL database ID                                                                                                   |
| 22             | Dynamically assigned internal UDS/SQL database ID in the remote configuration                                                                       |
| 23-26          | Dynamically assigned internal UDS/SQL subschema reference                                                                                           |
| 27-32          | Subschema name                                                                                                                                      |

**Bytes 1-6: U02 CB**

| <b>Byte(s)</b> | <b>Meaning</b>                                                                                                                                      |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 9-12           | Dynamically assigned internal UDS/SQL transaction ID                                                                                                |
| 13             | DML request code 1 in the BIB                                                                                                                       |
| 14             | DML request code 2 in the BIB                                                                                                                       |
| 15-16          | Dynamically assigned internal UDS/SQL number of the set type accessed in the DML for new style BIBs                                                 |
| 17-19          | Status code of the DML used                                                                                                                         |
| 20             | Code stating if the status code recorded in bytes 17-19 matches the status code passed to the user in the BIB ('O' when matched, 'B' when no match) |
| 21             | Dynamically assigned internal UDS/SQL database ID                                                                                                   |
| 22             | Dynamically assigned internal UDS/SQL database ID in the remote configuration                                                                       |
| 23-26          | Dynamically assigned internal UDS/SQL subschema reference                                                                                           |
| 27-32          | Subschema name                                                                                                                                      |

**Bytes 1-6: U01 CD**

| Byte(s) | Meaning                                                                                                                                             |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 9-12    | Dynamically assigned internal UDS/SQL transaction ID                                                                                                |
| 13      | DML request code 1 in the BIB created internally                                                                                                    |
| 14      | DML request code 2 in the BIB created internally                                                                                                    |
| 15      | Dynamically assigned internal UDS/SQL number of the set type accessed in the DML for new style BIBs                                                 |
| 16      | Dynamically assigned internal UDS/SQL number of the set type or realm used in the DML for new style BIBs                                            |
| 17-19   | Status code of the DML used                                                                                                                         |
| 20      | Code stating if the status code recorded in bytes 17-19 matches the status code passed to the user in the BIB ('O' when matched, 'B' when no match) |
| 21      | Code stating if there is a KDBS request available                                                                                                   |
| 22      | Dynamically assigned internal UDS/SQL database ID                                                                                                   |
| 23-28   | Subschema name                                                                                                                                      |

**Bytes 1-6: U02 CD**

| Byte(s) | Meaning                                                                                                                                             |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 9-12    | Dynamically assigned internal UDS/SQL transaction ID                                                                                                |
| 13      | DML request code 1 in the BIB created internally                                                                                                    |
| 14      | DML request code 2 in the BIB created internally                                                                                                    |
| 15-16   | Dynamically assigned internal UDS/SQL number of the set type accessed in the DML for new style BIBs                                                 |
| 17-19   | Status code of the DML used                                                                                                                         |
| 20      | Code stating if the status code recorded in bytes 17-19 matches the status code passed to the user in the BIB ('O' when matched, 'B' when no match) |
| 21      | Code stating if there is a KDBS request available                                                                                                   |
| 22      | Dynamically assigned internal UDS/SQL database ID                                                                                                   |
| 23-28   | Subschema name                                                                                                                                      |

**Bytes 1-6: U01 CN**

| Bytes | Meaning                                            |
|-------|----------------------------------------------------|
| 13-20 | Name of the UDS/SQL configuration                  |
| 21-24 | Return code of the ENAMP-SVC to connect to the CUP |

**Bytes 1-6: U01 DC**

| Bytes | Meaning                           |
|-------|-----------------------------------|
| 13-20 | Name of the UDS/SQL configuration |

**Bytes 1-6: U01 FN**

| Bytes | Meaning                                              |
|-------|------------------------------------------------------|
| 9-12  | Dynamically assigned internal UDS/SQL transaction ID |

**Bytes 1-6: U01 PA**

| Bytes | Meaning                                            |
|-------|----------------------------------------------------|
| 9-32  | 24 bytes of the start parameters passed by openUTM |

**Bytes 1-6: U01 PB**

| Bytes | Meaning                   |
|-------|---------------------------|
| 9-32  | No additional information |

**Bytes 1-6: U01 RB**

| Bytes | Meaning                                                                         |
|-------|---------------------------------------------------------------------------------|
| 9-12  | Dynamically assigned internal UDS/SQL transaction ID                            |
| 19-20 | Number of open processing chains and processing chains which are to be restored |

**Bytes 1-6: U03 RB**

| Bytes | Meaning                                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------------------------------|
| 9-12  | Dynamically assigned internal UDS/SQL transaction ID                                                                                   |
| 13-15 | Internal UDS/SQL status displays for the session, the last request in the transaction and the processing chain which is to be restored |
| 17-18 | Internal UDS/SQL number of the processing chain which is to be restored                                                                |
| 19-20 | Number of open processing chains                                                                                                       |
| 21-24 | Location of the BIB which is to be restored in the communication pool                                                                  |
| 27-32 | Subschema name                                                                                                                         |

**Bytes 1-6: U01 SB**

| Bytes | Meaning                                                                      |
|-------|------------------------------------------------------------------------------|
| 9-12  | Dynamically assigned internal UDS/SQL transaction ID                         |
| 13-16 | Number of open processing chains and therefore the number of BIBs to back up |

**Bytes 1-6: U01 SQ**

| Bytes | Meaning                                                       |
|-------|---------------------------------------------------------------|
| 9-12  | Dynamically assigned internal UDS/SQL transaction ID          |
| 13-16 | Dynamically assigned internal UDS/SQL ID of the SQL procedure |
| 17-20 | SQL return code of the SQL request                            |
| 21-24 | SQL return code of the 2 level of the SQL request (operation) |
| 25    | Internal SQL request code                                     |
| 27-28 | Internal SQL error code of the connection                     |
| 29-30 | Internal module code of the connection when an error occurs   |
| 31-32 | Error number internal to the module                           |

**Bytes 1-6: U01 ST**

| <b>Bytes</b> | <b>Meaning</b>                                           |
|--------------|----------------------------------------------------------|
| 13-16        | Internal UDS/SQL RLOG ID referring to the status request |
| 17-20        | Session section number referring to the status request   |
| 21-22        | Error code internal to the module                        |



---

# Glossary

This Glossary contains the definitions of some of the important terms and concepts used in the UDS/SQL manuals. Terms that appear in *italics* within a particular definition have also been defined in this Glossary. In cases where two or more terms are used synonymously, a “See” reference points to the more commonly used term in these manuals.

## A

### **access, contending**

See *contending access*.

### **access, direct**

See *direct access*.

### **access, sequential**

See *sequential access*.

### **access authorization**

The rights of a specified user group with regard to access to the *database*. Access rights are defined during live database operation using ONLINE-PRIVACY utility routine or, in offline mode, using the BPRIVACY utility routine.

### **access path**

Means of finding a certain subset of all *records* qualified by a search query, without having to carry out a sequential search of the whole *database*.

### **access rights**

Right of access to a *database* as defined in the BPRIVACY utility routine.

### **access type**

Type of access, e.g. read, update etc.

**act-key**

(actual key) Actual address of a *page*, consisting of *realm number* and *page number*.

**act-key-0 page**

First *page* of a *realm*; contains general information on the realm such as

- when the realm was created,
- when the realm was last updated,
- *internal version number* of the realm,
- *system break information*
- if applicable, *warm start* information.

**act-key-N page**

Characteristic page of a *realm*, with the highest *page number*.

Copy of the *act-key-0 page*.

**address, physical**

See *act-key* or *probable position pointer (PPP)*.

**administrator task**

Task of the *independent DBH*; The *database administrator* can control execution of the *independent DBH* via this task.

**AFIM**

See *after-image*.

**after-image**

Modified portion of a *page* **after** its content has been updated.

The *DBH* writes after-images to the *RLOG file* as well as the *ALOG file*.

**after-image, ALOG file**

The after-images are written to the ALOG file when the ALOG buffer is full. The purpose of the after-images in the ALOG file is to secure the data contained in the database and thus they must be maintained for a long period of time. They are used to reconstruct an original database or update a *shadow database*.

**after-image, RLOG file**

After-images are logged in the RLOG file **before** the updates are applied to the *database*. The after-images held in the RLOG file are required for *warm start* only. They are thus periodically overwritten.

**ALOG file**

File for securing the data contained in the database in the long term; see *after-image*.

**ALOG sequence number**

See *sequence number*.

**anchor record**

*Record* automatically created by UDS/SQL as *owner record* for *SYSTEM sets*. It cannot contain any *items* defined with the *schema DDL* and cannot be accessed.

**application**

Realization of a job in one or several *user programs* working with UDS/SQL *databases*.

**application program (AP)**

E.g. *COBOL DML* program or *IQS*.

**area**

See *realm*.

**ascending key (ASC key)**

*Primary key* of a *set*. Defines the sequence of *member records* in the *set occurrences* by ascending key values.

**authorization**

Identification used for user groups.

**authorized users**

Specified user groups who are authorized to access the *database*.

**automatic DBTT extension**

Some utility routines automatically extend the number of records possible for a record type if too few are available; no separate administration is required to do this.

See also *online DBTT extension*.

**automatic realm extension**

Some utility routines automatically extend realms when insufficient free space is available; no separate administration is required to do this.

See also *online realm extension*.

## B

**backup database**

See *shadow database*.

**base interface block (BIB)**

(Base Interface Block) Standard interface between UDS/SQL and each individual user; it contains, among other things, the *RECORD AREA* (user records as defined in the *subschema*).

**before-image**

Copy of a *page* taken before its contents are updated.

The *DBH* writes before-images to the *RLOG files* during database operation before the updates are applied to the *database*. A prerequisite is that the RLOG files exist.

**BFIM**

See *before-image*.

**BIB**

See *base interface block*.

**buffer pool**

See *system buffer pools* and *exclusive buffer pool*.

## C

**CALC key**

*Key* whose value is converted into a relative *page number* by means of a *hash routine*.

**CALC page**

*Page* of a *hash area*.

**CALC SEARCH key**

*Secondary key*. Used as *access path* for *direct access* via *hash routine*.

**CALC table**

Table in the direct/indirect *CALC page* whose entries point to the stored records.

Each line contains:

- the *CALC key*,
- the *record sequence number*
- the displacement to the related *page index entry* (direct *CALC page*) or the *probable position pointer* (indirect *CALC page*).

**CALL DML**

*DML* that is called by various programming languages (Assembler, COBOL, FORTRAN, PASCAL, PL/1) via the *CALL* interface.

**catalog identifier**

Name of the public volume set (PVS) under which the BS2000 UDS/SQL files are stored. The catalog identifier is part of the database or file name and must be enclosed in colons: “:catid:”.

**chain**

Storage mode for a *set occurrence* in which every *record* contains a pointer to the subsequent record.

**Character Separated Values (CSV)**

Output format in which the values are separated by a predefined character.

**checkpoint**

*Consistency point*, at which the *ALOG* file was changed and to which it is possible to return at any time using *BMEND* utility routine

**check records**

Elements which provide information for checking the database. They vary in length from 20 to 271 bytes.

**CHECK-TABLE**

Check table produced by the *DDL* compiler during *Subschema DDL* compilation, and used by the COBOL compiler and *CALL DML* to check whether the *DML* statements specified in the *application program* are permitted. It is part of the *COSSD* or *SSITAB* module.

**clone pair, clone pubset, clone session, clone unit**

A clone unit is the copy of an (original) unit (logical disk in BS2000) at a particular time ("Point-in-Time copy"). The TimeFinder/Clone component creates this copy optionally as a complete copy or as a "snapshot".

After they have been activated, the unit and clone unit are split; applications can access both.

The unit and clone unit together form a clone pair. TimeFinder/Clone manages this pair in what is known as a clone session.

If clone units exist for all units of a pubset, these clone units together form the clone pubset.

Details of this are provided in the manual "[Introduction to System Administration](#)".

**COBOL DML**

*DML* integrated in the COBOL language.

**COBOL runtime system**

Runtime system; sharable routines selected by the COBOL compiler (COBOL2000 or COBOL85) for the execution of complex statements.

**COBOL Subschema Directory (COSSD)**

Provides the COBOL compiler with subschema information for compilation of the DB *application programs*.

**common memory**

Shareable memory area used by several different tasks. In UDS/SQL, it always consists of the *common pool* and the *communication pool* and, depending on the application, the *SSITAB pool* (see *SSITAB module*) if *CALL DML* is used.

If UDS-D is used, it also consists of the *distribution pool* and the *transfer pool*.

**common pool**

Communication area of the *independent DBH*. Enables *DBH* modules to communicate with each other. Contains, among other things, an input/output buffer for *pages (buffer pools)*.

**communication partners**

Tasks or data display terminals.

**communication pool**

Communication area of the *independent DBH* for *application programs*. One of its functions is to store base interface blocks (*BIB*).

**compatible database interface (KDBS)**

see *KDBS*

**compiler database**

The *realms* and files of the *database* which are required by the UDS/SQL compiler. They are

- *DBDIR* (*Database Directory*)
- *DBCOM* (*Database Compiler Realm*)
- *COSSD* (*COBOL Subschema Directory*).

**COMPILER-SCHEMA**

UDS/SQL-internal *schema* of the *compiler database*.

**COMPILER-SUBSCHEMA**

UDS/SQL-internal *subschema* of the *compiler database*.

**compound key**

Key consisting of several *key items*.

**compression**

Only the filled *items* of a *record* are stored (see *SSL* clause *COMPRESSION*).

**configuration**

See *DB configuration*.

**configuration user ID**

User ID in which the *database administrator* starts the *DBH*.

**configuration name**

Freely selectable name of the *database configuration* for a particular *session*. The *DBH* uses it to form:

- the name of the *Session Log File*,
- the names of the *DB status file* and its backup copy,
- the names of the *RLOG files*,
- the names of the temporary *realms*,
- the names of session job variables,
- the *event names* of *PI eventing*,
- the *DCAM application* name for the administration,
- the names of the *common pools*
- the names of the dump files.

**connection module**

Module that must be linked into every UDS/SQL *application program* and which establishes the connection with the *DBH*.

**consistency**

State of the database without conflicts in the data stored in it.

**consistency, logical**

State of the database in which the stored data has no internal conflicts and reflects the real-world situation.

**consistency, physical**

State of the database in which the stored data is consistent with regard to correct physical storage, *access paths* and description information.

**consistency, storage**

See *physical consistency*.

**consistency error**

A violation of the *physical consistency* of the stored data.

**consistency point**

Point (in time) at which the *database* is consistent, i.e. all modifying transaction have been terminated and their modifications have been executed in the database.

**consistency record**

Administration record with consistency time and date stamps in the *DBDIR*. For an update in a *realm* the *DBH* enters the date and time in the consistency record and in the updated realm. When realms or *databases* are attached for a *session*, the *DBH* uses this time stamp to check the consistency of the realms within each database.

**contending access**

Different *transactions* attempting to access a *page* simultaneously.

**conversation**

*SQL*-specific administration data is retained across transaction boundaries in an *SQL* application. This kind of data administration unit is called a conversation. In openUTM such an administrative unit is also called a service.

**copy**

See *database copy*.

**COSSD**

See *COBOL Subschema Directory*.

**CRA**

(Current Record of Area) *Record* which is marked in the *currency table* as the current record of a particular *realm* (area).



**CRR**

(Current Record of Record) *Record* which is marked in the *currency table* as the current record of a particular *record type* (Record).

**CRS**

(Current Record of Set) *Record* which is marked in the *currency table* as the current record of a particular *set*.

**CRU**

(Current Record of Rununit) *Record* which is marked in the *currency table* as the current record of the *processing chain*.

**CSV**

see *Character Separated Values*

**currency table**

The currency table contains:

- CURRENT OF AREA table (table of CRAs),
- CURRENT OF RECORD table (table of CRRs) and
- CURRENT OF SET table (table of CRSs).

**CURRENT OF AREA table**

See *currency table*.

**CURRENT OF RECORD table**

See *currency table*.

**CURRENT OF SET table**

See *currency table*.

**D****DAL**

(Database Administrator Language) Comprises the commands which monitor and control a *session*.

**data backup**

Protection against loss of data as a result of hardware or software failure.

**data deadlock**

See *deadlock*.

**data protection (privacy)**

Protection against unauthorized access to data. Implemented in UDS/SQL by means of the schema/subschema concept and access authorization. *Access rights* are granted by means of the BPRIVACY utility routine.

**database (DB)**

Related data resources that are evaluated, processed and administered with the help of a *database system*.

A database is identified by the database name.

An UDS/SQL database consists of the *user database* and the *compiler database*. To prevent the loss of data, a *shadow database* may be operated together with (i.e. parallel to) the original database.

**database administrator**

Person who manages and controls *database* operation. The DB administrator is responsible for the utility routines and the Database Administrator Language (*DAL*).

**database copy**

Copy of a consistent *database*; may be taken at a freely selectable point in time.

**database compiler realm (DBCOM)**

Stores information on the *realms*, *records* and *sets* defined by the user in the *Schema DDL* and *Subschema DDL*.

**database copy update**

Updating of a *database copy* to the status of a *checkpoint* by applying the appropriate *after-images*.

**database directory (DBDIR)**

Contains, among other things, the *SIA*, all the *SSIAs* and information on *access rights*.

**database job variable**

Job variable in which UDS/SQL stores information on the status of a *database*.

**database key (DB key)**

*Key* whose value represents a unique identifier of a *record* in the *database*. It consists of the *record reference number* and the *record sequence number*. The database key values are either defined by the database programmer or automatically assigned by UDS/SQL.

**database key item**

Item of type DATABASE-KEY or DATABASE-KEY-LONG that is used to accommodate *database key* values.

Items of type DATABASE-KEY and DATABASE-KEY-LONG differ in terms of the item length (4 bytes / 8 bytes) and value range.

**DATABASE-KEY item**

See *database key item*.

**DATABASE-KEY-LONG item**

See *database key item*.

**database page**

See *page*.

**DATABASE-STATUS**

Five-byte item indicating the database status and consisting of the *statement code* and the *status code*.

**database system**

Software system that supports all tasks in connection with managing and controlling large data resources. The database system provides mechanisms for stable and expandable data organization without redundancies. They allow many users to access *databases* concurrently and guarantee a consistent data repository.

**DB status file**

(database status file) Contains information on the most recently reset *transactions*.

openUTM-S or, in the case of distributed processing, UDS-D/openUTM-D needs this information for a *session restart*.

**DB configuration**

(database configuration) The *databases* attached to a *DBH* at any one point during *session* runtime. As the result of *DAL* commands or *DBH* error handling, the database configuration can change in the course of a session.

At the *session start*, the DB configuration may be empty. Databases can be attached with *DAL* commands after the start of the session. They can also be detached during the session with *DAL* commands.

**DBC.COM**

See *database compiler realm*.

**DBDIR**

See *database directory*.

**DBH**

Database Handler: program (or group of programs) which controls access to the *database(s)* of a *session* and assumes all the attendant administrative functions.

**DBH end**

End of the *DBH* program run. DBH end can be either a *session end* or a *session abort*.

**DBH, independent**

See *independent DBH*.

**DB key**

See *database key*.

**DBH, linked-in**

See *linked-in DBH*.

**DBH load parameters**

See *load parameters (DBH)*.

**DBH start**

Start of the *DBH* program run. DBH start can be either a *session start* or a *session restart*.

**DBTT**

(Database Key Translation Table) Table from which UDS/SQL can obtain the *page address (act-key)* of a *record* and associated tables by means of the database key value.

The DBTT for the SSIA-RECORD consists only of the DBTT base. For all other record types, the DBTT consists of a base table (DBTT base) and possibly of one or more extension tables (DBTT extents) resulting from an online DBTT extension or created by BREORG.

**DBTT anchor page**

Page lying within the realm of the associated DBTT in which the DBTT base and DBTT extents are administered. Depending on the number of DBTT extents multiple chained DBTT anchor pages may be required for their administration.

**DBTT base**

see *DBTT*

**DBTT extent**

see *DBTT*

**DBTT page**

Page containing the *DBTT* or part of the *DBTT* for a particular *record type*.

**DCAM**

Component of the TRANSDATA data communication program.

**DCAM application**

Communication application using the *DCAM* communication method. A DCAM application enables communication between

- a DCAM application and terminals.
- different DCAM applications within the same or different hosts, and with *remote configurations*.
- a DCAM and a openUTM application.

**DDL**

(Data Description Language) Formalized language for defining the logical data structure.

**deadlock**

Mutual blocking of *transactions*.

A deadlock can occur in the following situations:

- Data deadlock: This occurs when *transactions* block each other with *contending access*.
- Task deadlock: This occurs when a *transaction* that is holding a lock cannot release it, since no openUTM task is free. This deadlock situation can only occur with UDS/SQL-openUTM interoperation.

**descending key (DESC key)**

*Primary key* of a set. Determines the sequence of *member records* in the *set occurrences* to reflect descending key values.

**direct access**

Access to a *record* via an item content. UDS/SQL supports direct access via the *database key*, *hash routines* and *multi-level tables*.

**direct hash area**

See *hash area*.

**distributed database**

A logically connected set of data resources that is distributed over more than one UDS/SQL configuration.

**distributed transaction**

*Transaction* that addresses at least one *remote configuration*. A transaction can be distributed over:

- UDS-D,
- openUTM-D,
- UDS-D and openUTM-D.

**distribution pool**

Area in the *independent DBH* used for communication between *UDSCT*, *server tasks*, *user tasks* and the *master task* with regard to UDS-D-specific data. The distribution pool contains the *distribution table* and the UDS-D-specific system tables.

**distribution table**

Table created by UDS-D using the input file assigned in the *distribution pool*. With the aid of the distribution table, the distribution component in the *user task* decides whether a *processing chain* should be processed locally or remotely. Assigned in the distribution table are:  
*subschema - database*  
*database - configuration*  
*configuration - host computer*.

**DML**

Data Manipulation Language: language for accessing a UDS/SQL *database*.

**dummy subtransaction**

A primary *subtransaction* is created by UDS-D when the first *READY* statement in a *transaction* addresses a *remote database*.

A dummy subtransaction is used to inform the *local configuration* of the transaction so that the *database* can be recovered following an error.

**duplicates header**

Contains general information on a *duplicates table* or a *page* of a *duplicates table*, i.e.

- chaining reference to the next and previous *overflow page*
- the number of free bytes in the page of the *duplicates table*.

**duplicates table**

Special *SEARCH-KEY table* in which a key value which occurs more than once is stored only once.

For each key value, the duplicates table contains:

- a table index entry with the key value and a pointer to the associated table entry
- a table entry (DB key list), which can extend over several pages, containing the *record sequence numbers* of the *records* which contain this key value.

**duplicates table, main level**

Main level, Level 0. Contains a table index entry and the beginning of the associated table entry (DB key list).

**dynamic set**

*Set* which exists only for the life of a *transaction* and which stores *member records* retrieved as result of search queries.

## E

**ESTIMATE-REPORT**

Report produced after BGSIA run. Used to estimate the size of the *user realms*.

**event name**

Identification used in eventing.

**exclusive buffer pool**

Buffer which, in addition to the *system buffer pools*, is used exclusively for buffering *pages* of the specified *database*.

## F

**foreign key**

*Record element* whose value matches the primary key values of another table (UDS/SQL *record type*). Foreign keys in the sense of UDS/SQL are qualified as "REFERENCES owner record type" in the member record type of a set relationship in the BPSQLSIA protocol.

**FPA**

See *free place administration*.

**FPA base**

See *free place administration*.

**FPA extent**

See *free place administration*.

**FPA page**

*Free place administration page*.

**free place administration (FPA)**

Free space is managed both at realm level (*FPA pages*) and at page and table level. Free place administration of the pages is carried out in a base table (FPA base) and possibly in one or more extension tables (FPA extents) created by means of an online realm extension or BREORG.

**function code**

Coding of a *DML* statement; included in information output by means of the *DAL* command DISPLAY or by UDSMON.

**G****group item**

Nameable grouping of *record elements*.

**H****hash area**

Storage area in which UDS/SQL stores data and from which it retrieves data on the basis of key values which are converted into relative *page numbers*. A hash area may contain the *record* addresses as well as the records themselves. A *direct hash area* contains the records themselves; an *indirect hash area*, by contrast, contains the addresses of records stored at some other location.

**hash routine**

Module which performs *hashing*.

**hashing**

Method of converting a key value into a *page address*.



**HASHLIB**

Module library for the storage of *hash routines* for one *database*.

## I

**identifier**

Name allocated by the database designer to an *item* that UDS/SQL creates automatically. UDS/SQL adapts item type and length to the specified item usage.

**implicit set**

*SYSTEM set* created by UDS/SQL when a *SEARCH key* is defined at record type level.

**inconsistency**

State of the database in which the data values contained in it are inconsistent.

**independent DBH**

Independent program system enabling more than one user to access a single *database (mono-DB operation)* or several databases (*multi-DB operation*) simultaneously. The independent DBH is designed as a task family, consisting of

- a *master task (UDSSQL)*
- one or more *server tasks (UDSSUB)*
- an *administrator task (UDSADM)*

**index level**

Hierarchy level of an *index page*.

**index page**

*Page* in which the highest (lowest) key values of the next-lower level of an indexed table are stored.

**INDEX search key**

*Secondary key*. Used as *access path* for *direct access* via a *multi-level table*.

**indirect hash area**

See *hash area*.

**integrity**

State of the database in which the data contained in it is complete and free of errors.

- entity integrity
- *referential integrity*
- user integrity

**interconfiguration**

Concerning at least one *remote configuration*.

**interconfiguration consistency**

A *distributed transaction* that has caused updates in at least one *remote configuration* is terminated in such a way that the updates are either executed on the *databases* in each participating *DB configuration* or on none at all.

Interconfiguration consistency is assured by the *two-phase commit protocol*.

**interconfiguration deadlock**

Situation where *distributed transactions* are mutually locked due to *contending accesses*.

**interface**

In software: memory area used by several different programs for the transfer of data.

**internal version number**

Each *realm* of the *database*, including *DBDIR* and *DBCOM*, has an internal version number which the utility routines (e.g. BREORG, BALTER) increment by one whenever a realm is updated. This internal version number is kept in the *act-key-0 page* of the realm itself and also in the PHYS VERSION RECORD in the DBDIR.

**item**

Smallest nameable unit of data within a *record type*. It is defined by item type and item length.

**K****KDBS**

Compatible database interface. Enables programs to be applied to applications of *DB systems* by different manufacturers.

**key**

*Item* used by the database programmer for *direct access* to records; an optimized *access path* is provided for the key by UDS/SQL in accordance with the *schema* definition.

**key, compound**

Key consisting of several *key items*.

**key item**

*Item* defined as a *key* in the *schema*.

**key reference number**

*Keys* are numbered consecutively in ascending order, beginning at 1.

## L

**linked-in control system**

UDS/SQL component for *linked-in DBH*, responsible for control functions (corresponds to the *subcontrol system* of the *independent DBH*).

**linked-in DBH**

Module linked in to or dynamically loaded for the current DB *application program* and controlling access to a single *database (mono-DB operation)* or several databases simultaneously (*multi-DB operation*).

**list**

Table containing the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

In a distributable list the data pages which contain the member records (level 0 pages) can be distributed over more than one realm. The pages containing the higher-ranking table levels all reside in one realm (table realm of a distributable list).

**load parameters (DBH)**

Parameters requested by the *DBH* at the beginning of the *session*. They define the basic characteristics of a session.

**local application program**

An *application program* is local with regard to a *configuration* if it was linked to the configuration using `/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name`

**local configuration**

The *configuration* assigned to an *application program* before it is called using /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name.

The application program communicates with the local configuration via the *communication pool*. The local configuration is in the same host as the application program.

**local database**

*Database* in a *local configuration*.

**local distribution table**

A *distribution table* is considered local to a *DBH* if it is held in the *DBH's distribution pool*.

**local host**

Host computer containing the *application program*.

**local transaction**

*Transaction* that only addresses the *local configuration*.

**logging**

Recording of all updates in the *database*.

**logical connection**

Assignment of two *communication partners* that enables them to exchange data. *DCAM applications* communicate via logical connections.

---

**M****main reference**

In the *DBH* the main reference is used to manage the resources required for processing a transaction's requests, including those for transferring the requests from the application program to the *DBH* and back.

**mainref number**

Number assigned to the *transaction* at *READY*. This number is unique only at a given time; at the end of the transaction, it is assigned to another transaction.

**master task**

Task of the *independent DBH* in which the *UDSQL* module executes. Controls the start and end of a *session* and communicates with the *database administrator* directly or via the *administrator task*.

**member**

See *member record* or *member record type*.

**member, AUTOMATIC**

*Record* is inserted at storage time.

**member, MANDATORY**

*Record* cannot be removed.

**member, MANUAL**

*Record* is not inserted automatically at storage time.

**member, OPTIONAL**

*Record* can be removed.

**member record**

Lower-ranking *record* in a *set occurrence*.

**member record type**

Lower-ranking *record type* in a *set*.

**mono-DB configuration**

Type of configuration where only one *database* takes part in a *session*.

**mono-DB operation**

Mode of *database* operation where the *DBH* uses only one *database* of a *configuration*.

**multi-DB configuration**

Type of configuration where several *databases* take part in a *session*.

**multi-DB operation**

Mode of *database* operation where the *DBH* uses several *databases* of a *configuration*.

**multi-DB program**

*Application program* that addresses more than one *database*. The *databases* may be part of one or more *mono-DB* or *multi-DB configurations*.

**multi-level table**

*SEARCH KEY table* which contains a line for each *record* of the associated *record type* or each *member record* of the *set occurrence*, as appropriate. Each line comprises the key value of the record and the record pointer. It is also referred to as an indexed table.

**multithreading**

A mechanism that enables the *DBH* to fully exploit the CPU.

Multithreading means that the *DBH* processes several jobs concurrently by using so-called threads. Each thread has information on the current status of a particular job stored in it. When a job needs to wait for the completion of an I/O operation, *DBH* uses the CPU to process some other job.

## N

**network**

All computers linked via *TRANSDATA*.

## O

**OLTP**

(Online Transaction Processing) In an OLTP application, a very large number of users access the same programs and data. This usually occurs under the control of a transaction monitor (TP monitor).

**online backup**

If AFIM logging is active, the *database* can be saved during a session. The ability to save a database online is determined with the *BMEND* utility routine.

**online DBTT extension**

Extension during ongoing database operation of the number of possible records of a record type. The *DAL* commands *ACT DBTT-INCR*, *DEACT DBTT-INCR*, *DISPLAY DBTT-INCR* and *EXTEND DBTT* can be used to administer the online extension of *DBTTs*.

See also *automatic DBTT extension*.

**online realm extension**

Extension of *user realms* and *DBDIR* in ongoing database operation. The *DAL* commands *ACT INCR*, *DEACT INCR*, *DISPLAY INCR*, *EXTEND REALM* and *REACT INCR* are provided for administering the online extensibility of realms.

See also *automatic realm extension*.

**open transaction**

*Transaction* which has not been closed with *FINISH* or *FINISH WITH CANCEL*, or with *COMMIT* or *ROLLBACK*.

**openUTM**

(universal transaction monitor) Facilitates the creation and operation of transaction-oriented applications.

**operator task (OT)**

See *master task*

**original database**

The term “original database” refers solely to the naming of the database files (*dbname.dbfile*), not to the status of the database content (see also *shadow database*).

**overflow page**

*Page in hash areas and duplicates tables* for storing data that does not fit in the primary page. Their structure is the same as that of the pages of the hash area or duplicates table in question.

**owner**

See *owner record* or *owner record type*.

**owner record**

Higher-ranking *record* in a *set occurrence*.

**owner record type**

Higher-ranking *record type* in a *set*.

**P****page**

Physical subunit of a *realm*. UDS/SQL identifies pages by means of unique keys (*act-key*).

The length of a page may be optionally 2048, 4000 or 8096 bytes. All pages within a database must have the same length. Pages with a length of 4000 or 8096 bytes are embedded in a *page container*.

**page address**

In a page address, a distinction is made between the current address of a *page*, i.e. the *act-key*, and the probable address of a page, the *probable position pointer (PPP)*.

**page container**

Pages with a length of 4000 or 8096 bytes are embedded in a so-called page container, which consists of a 64-byte header that precedes the page and a 32-byte trailer at the end of the page.

**page header (page info)**

The first 20 bytes of a database *page* (except for the *FPA* and *DBTT* pages with a length of 2048 bytes). They contain:

- the *act-key* of the *page* itself,
- the number of *page index entries*
- the length and displacement of the bytes which are still vacant in this page.
- the page type (*ACT-Key-0 page*, *FPA page*, *DBTT page*, *DBTT anchor page*, normal data page or *CALC page*)

**page index entry**

Indicates the position of a *record* within a *page*.

**page number**

In each *realm* the *pages* are numbered consecutively in ascending order starting starting from 0. The page number is part of the *page address*.

Page number = PAM page number -1 for databases with a page length of 2048 bytes

Page number = (PAM page number-1) / 2 for databases with a page length of 4000 bytes

Page number = (PAM page number-1) / 4 for databases with a page length of 8096 bytes.

**password for UDS/SQL files**

Password serving to protect the files created by UDS/SQL (default: C'UDS\_'). The *DB administrator* can define other passwords with PP CATPASS or MODIFY-FILE-ATTRIBUTES.

**pattern**

Symbolic representation of all possible *item* contents, used at item definition.

**pattern string**

String defining a *pattern*.



**PETA**

Preliminary end of transaction: UDS-D or openUTM-D statement that causes a preliminary transaction end.

The PETA statement belongs to the first phase of the *two-phase commit protocol* which terminates a *distributed transaction*.

The PETA statement stores the following information failproof in the *RLOG file* of the local *DBH*:

- each updated *page*
- rollback and locking information
- the names of all participating *configurations*.

This information is required for any future *warm start*.

**pointer array**

Table of pointers to the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

**PPP**

See *probable position pointer (PPP)*.

**prepared to commit (PTC)**

Part of the *two-phase commit protocol*:

State of a *subtransaction* after execution of a *PETA* statement and before receipt of the message that the complete *transaction* is to be terminated with FINISH or FINISH WITH CANCEL.

**primary key**

Distinguished from *secondary keys* for reasons of efficiency. Usually a unique identifier for a *record*.

**primary key (DDL)**

The *key* of a *record type* which is defined by means of "LOCATION MODE IS CALC" or the *key* of an order-determining *key* of a set occurrence which is defined by means of "ORDER IS SORTED [INDEXED]". Also used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

**primary key (SQL)**

In the broader sense (SQL), a *record element* uniquely identifying a record.

In UDS-SQL, the database key of an owner record output as the "PRIMARY KEY" in the BPSQLSIA log (see also *foreign key*).

A *record element* which uniquely identifies a record is flagged as "UNIQUE" in the BPSQLSIA log unless it is the aforementioned "PRIMARY KEY".

**primary subtransaction**

*Subtransaction that runs in the local configuration.*

The primary subtransaction is opened by the first *READY* statement in a *transaction* on a *local database*.

If the first *READY* statement addresses a *remote database*, UDS-D generates a *dummy subtransaction* as the primary subtransaction.

**PRIVACY-AND-IQF SCHEMA**

UDS/SQL-internal *schema* for protection against unauthorized access.

**PRIVACY-AND-IQF SUBSCHEMA**

UDS/SQL-internal *subschema* for protection against unauthorized access.

**probable position pointer (PPP)**

Probable address of a *page*, comprising *realm number* and *page number*.

UDS/SQL does not always update probable position pointers (PPP) when the storage location of data is changed.

**processing chain**

Sequence of DML statements applied to a *database* within a *transaction*.

**PTC state**

See *prepared to commit*.

**pubset declaration**

See *UDS/SQL pubset declaration*

**pubset declaration job variable**

Job variable in which a *UDS/SQL pubset declaration* is specified.

**P1 eventing**

Manner in which tasks communicate with each other.

## R

**READY**

Start of a *transaction* or a *processing chain* in *COBOL DML* programs.

**READYC**

Start of a *transaction* or a *processing chain* in *CALL DML* programs.

**realm**

Nameable physical subunit of the *database*. Equivalent to a file. Apart from the *user realms* for user data there are also the realms *DBDIR* and *DBCUM*, which are required by UDS/SQL.

**realm configuration**

Comprises all the database *realms* taking part in a *session*.

**realm copy**

See *database copy*.

**realm reference number**

*Realms* are numbered consecutively in ascending order, starting with 1. The realm reference number (area reference) is part of the *page address*.

**reconfiguration**

Regrouping of databases in a *DB configuration* after a *session abort*. A prerequisite for reconfiguration is that the *SLF* has been deleted or that its contents have been marked as invalid.

**record**

Single occurrence of a *record type*; consists of one item content for each of the *items* defined for the record type and is the smallest unit of data managed by UDS/SQL via a unique identifier, the *database key*.  
The reserved word **RECORD** is used in DDL and SSL syntax to declare a record type.

**record address**

Address of the page containing the *record*. See *page address*.

**RECORD AREA**

Area in the *USER WORK AREA (UWA)* which can be referenced by the user. The record area contains the *record types* and the implicitly defined items (IMPLICITLY-DEFINED-DATA-NAMES) of the database such as the AREA-ID items of the WITHIN clauses of the schema. The length of the record area is essentially defined by the record types contained in it.

**record element**

*Item, vector or group item.*

**record hierarchy**

Owner/member relationship between *record types*: the *owner record type* is the higher-ranking part of the relationship; the *member record type* is the lower-ranking part.

**REC-REF**

See *record reference number*.

**record reference number**

*Record types* are numbered consecutively in ascending order, starting at 1. The record reference number is part of the *database key*.

**record SEARCH KEY table**

*SEARCH KEY* table for selection of a *record* from a *record type*.

**record sequence number (RSQ)**

The record sequence number can be assigned by the database programmer; if not, UDS/SQL numbers the *records* of a *record type* contiguously in ascending order, in the sequence in which they are stored; numbering starts at 1. The record sequence number is part of the *database key*.

**record type**

Nameable grouping of *record elements*.

**record type, linear**

*Record type* that is neither the *owner* nor the *member* of a set (corresponds to record types of a conventional file).

**referential integrity**

*Integrity* of the relationships between tables (UDS/SQL *record types*).

**remote application program**

*Application program* that is not local with regard to a particular *configuration*.

**remote configuration**

*DB-configurations* that are not assigned to the *application program* via `/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name` but via the *distribution table* once the application program is running. The *connection module* of the application program communicates with the remote configurations via *DCAM applications*.

Remote configurations can be situated on *local* or *remote* hosts.

**remote database**

*Database* in a *remote configuration*.

**remote host**

Host computer that is not local.

**repeating group**

*Group item* with repetition factor. The repetition factor, which must be greater than 1, specifies the number of duplicates of the group item to be incorporated in the repeating group.

**request**

The functions of the *DAL* commands ADD DB, ADD RN, DROP DB, DROP RN, NEW RLOG and CHECKPOINT are held in the *DBH* as "requests" and are not executed until the DAL command PERFORM is entered.

**restart of BMEND**

Resumption of an aborted BMEND run.

**restart of a session**

See *session restart*.

**restructuring**

Modification of the *Schema DDL* or *SSL* for *databases* already containing data.

**return code**

Internal code which the called program sends to the calling program; Return code  $\neq 0$  means an error has occurred.

**RLOG file**

Backup file used by the *DBH* during a session to store *before-images* (BFIMs) and *after-images* (AFIMs) of data which is updated. With the aid of the *RLOG file*, the *DBH* can cancel updates effected by incomplete *transactions*. There is one RLOG file per *configuration*. An RLOG file consists of two physical files.

**rollback**

Canceling of all updates effected within a *transaction*.

**RSQ**

See *record sequence number*.

**RUNUNIT-ID**

See *transaction identification*.

**S****schema**

Formalized description of all data structures permitted in the *database*. A UDS/SQL schema is defined by means of the *Schema DDL*.

**Schema DDL**

Formalized language for defining a *schema*.

**Schema Information Area (SIA)**

The SIA contains the complete database definition. The *DBH* loads the SIA into main memory at the start of DB processing.

**SEARCH KEY**

*Secondary key*; *access paths* using secondary keys are created by UDS/SQL by means of *hash routines* and *multi-level tables*.

**SEARCH KEY table**

*Multi-level table* used by UDS/SQL as an *access path* via a *secondary key*.

**secondary key**

Any *key* which is not a *primary key*. Used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

**secondary subtransactions**

*Subtransactions* that address *remote configurations*.

**sequence number**

Identifier in the name of the *ALOG files* (000000001 - 999999999). The first ALOG file of a *database* is always numbered 000000001.

**sequential access**

Accessing a *record* on the basis of its position within a predefined record sequence.

**server task**

Task of the *independent DBH* in which the *UDSSUB* module executes; processes the requests of the *DB application programs*.

**session**

Period between starting and normal termination of the *DBH* (*independent/linked-in*) in which it is possible to work with the *databases* of the *configuration*. Normally, a session consists of a sequence of *session sections* and *session interrupts*.

**session abort**

Occurs when the *DBH* is terminated abnormally after a successful *session start*. A session abort can be caused by: power failure, computer failure, BS2000 problems, *DBH* problems, %TERM.

**session end**

Is the result of:

- *DAL* when using *independent DBH*,
- TERM in the *DML application program* when using *linked-in DBH*,
- *DBH* error handling.

During a *session interrupt*, the user can also effect session end by invalidating the *SLF* contents. Inconsistent *databases* can be made consistent again by a *warm start*, even without an *SLF*.

**session interrupt**

The period between a *session abort* and the related *session restart*.

**session job variable**

Job variable in which *UDS/SQL* stores information about a session.

**Session Log File (SLF)**

File which is permanently assigned to a *session* and which is required by the *DBH* in the event of a *session restart*. It contains information on the current *DB configuration*, the number of current file identifiers and the current values of the *DBH load parameters*.

**session restart**

Starting of the *DBH*, under the same *configuration name* and *configuration user ID*, after a *session abort*. With the aid of the *SLF*, the *DBH load parameters* and the current file identifiers which existed when the session aborted are re-established, and the *databases* of the previous *configuration* are reattached, if necessary by means of a *warm start*.

**session section**

Period from the start of the *DBH*, either at the *session start* or a *restart*, to the normal *session end* or to a *session abort*.

**session section number**

Number which identifies a session section unambiguously.

**session start**

State of a session in which the *DBH* is started under a configuration name for which there is no *Session Log File (SLF)* with valid contents.

**set**

Nameable relationship between two *record types*.

**set, dynamic**

See *dynamic set*.

**set, implicit**

See *implicit set*.

**set, singular**

See *SYSTEM set*.

**set, standard**

See *standard set*.

**Set Connection Data (SCD)**

Linkage information for the *records* of a *set occurrence*.

**set occurrence**

Single instance of a *set*. Comprises exactly one *owner record* and any number of subordinate *member records*.

**set reference number**

*Sets* are numbered contiguously in ascending order, beginning at 1.



**set SEARCH KEY table**

*SEARCH KEY table* for selecting a *member record* from a *set occurrence*.

**SF pubset**

See *single feature pubset*

**shadow database**

Backup of all the files of a database, each saved under the name "*dbname.dbfile.copyname*".

A shadow database can be created at any time and processed parallel to the original database in RETRIEVAL mode.

In addition BMEND can be used to apply *ALOG files* that have already been closed to the database parallel to the UDS/SQL *session*.

**Shared user buffer pool**

Shared buffer of several databases which is used in addition to the *System Buffer Pool*, solely for buffering *pages* of the *databases* that have been assigned to it.

**SIA**

See *Schema Information Area*.

**SIB**

See *SQL Interface Block*.

**single feature pubset**

A single feature pubset (SF pubset) consists of one or more homogeneous disks which must have the same major properties (disk format, allocation unit).

**SLF**

See *session log file*.

**SM pubset**

See *system managed pubset*

**snap pair, snap pubset, snap session, snap unit**

A snap unit is the copy of an (original) unit (logical disk in BS2000) at a particular time (“Point-in-Time copy”). The TimeFinder/Snap component creates this copy as a “snapshot” in accordance with the “Copy-On-First-Write strategy”: Only if data is modified is the original data concerned written beforehand into a central save pool of the Symmetrix system. The snap unit contains the references (track pointers) to the original data. In the case of unmodified data the references point to the unit, in the case of modified data to the save pool.

After they have been activated, the unit and snap unit are split; applications can access both.

The unit and snap unit together form a snap pair. TimeFinder/Snap manages this pair in what is known as a snap session.

If snap units exist for all units of a pubset, these snap units together form the snap pubset.

Details of this are provided in the manual "[Introduction to System Administration](#)".

**sort key table**

Table pointing to the *member records* of a *set occurrence*.

**source program**

Program written in a programming language and not yet translated into machine language.

**spanned record**

Record exceeding the length of a *page*. **Only UDS/SQL-internal records** can be spanned records;

User record types must not exceed

- 2020 bytes for a page length of 2048 bytes
- 3968 bytes for a page length of 4000 bytes
- 8064 bytes for a page length of 8096 bytes.

**SQL**

SQL is a relational database language which has been standardized by ISO (International Organization for Standardization).

**SQL conversation**

See *conversation*.

**SQL DML**

*SQL* Data Manipulation Language for querying and updating data.

**SQL Interface Block (SIB)**

Interface between UDS/SQL and SQL application program(s); contains the SQL statement, any existing parameters and the statement results.

**SQL transaction**

Related sequence of *SQL* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.

**SSIA**

See *Subschema Information Area*.

**SSIA-RECORD**

UDS/SQL-internal *record type*, located in the *DBDIR*. *Records* belonging to this type are, for example, the Schema Information Area (*SIA*) and the Subschema Information Areas (*SSIAs*).

**SSITAB module**

Module generated by the BCALLSI utility routine; makes available the subschema information required by *CALL DML* programs.

**SSL**

See *Storage Structure Language*.

**standard set**

A *set* other than a *dynamic*, *implicit* or *SYSTEM set*.

**statement code**

Number stored in the first part of the *DATABASE-STATUS* item. Its function is to indicate which *DML* statement resulted in an exception condition.

**status code**

Number stored in the second part of the *DATABASE-STATUS* item. It indicates which exception condition has occurred.

**Storage Structure Language (SSL)**

Formalized language for describing the storage structure.

**string**

A series of consecutive alphanumeric characters.

**subcontrol system**

Component for the *independent DBH*. Responsible for control functions.

**subschema**

Section of a *schema* required for a particular *application*; it can be restructured, within limits, for the intended application; a subschema is defined by means of the *Subschema DDL*.

**Subschema DDL**

Formalized language for defining a *subschema*.

**Subschema Information Area (SSIA)**

The SSIA contains all subschema information required by the *DBH* to carry out, on behalf of the user, the *database* accesses permitted within the specified *subschema*. The *DBH* loads the SSIA into main memory when it is referenced in a *READY* command.

**subschema module**

Module resulting from *subschema* compilation when a *COBOL DML* program is compiled. It must be linked in to the *application program* and includes the *USER WORK AREA (UWA)* as well as the *RECORD AREA*, which is also part of the *base interface block (BIB)*. The name of the subschema module is the first 8 bytes of the subschema name.

**subschema record**

*Record* defined in the *Subschema DDL*.

**SUB-SCHEMA SECTION**

In COBOL programs with *DML* statements: section of the DATA DIVISION used for specifying the schema name and the subschema name.

**subtransaction**

In a distributed *transaction*, all the *processing chains* that address the databases in **one** *configuration* form a subtransaction.

**system area**

*Realm* required only by UDS/SQL. The system areas of a database include:

- the *Database Directory (DBDIR)*,
- the *Database Compiler Realm (DBCOM)*,
- the *COBOL Subschema Directory (COSSD)*

**system break information**

Indicates whether the *database* is consistent or inconsistent.

**system buffer pools**

Input/output buffer for database pages (see *page*). The buffer is part of the *common pool (independent DBH)* or the *DBH work area (linked-in DBH)*. Its size is determined by the *DBH load parameters* 2KB-BUFFER-SIZE, 4KB-BUFFER-SIZE or 8KB-BUFFER-SIZE.

**system managed pubset**

A system managed pubset consists of one or more volume sets which, as with an *SF pubset*, comprise a collection of multiple homogeneous disks; here, too, homogeneity relates to particular physical properties such as disk format and allocation unit.

**SYSTEM record**

See *anchor record*.

**SYSTEM set**

*Set* whose *owner record type* is the *symbolic record type* SYSTEM.

## T

**table, multi-level**

See *multi-level table*.

**table (SQL)**

A table in the context of *SQL* corresponds to a *UDS/SQL record type*.

**table header**

Contains general information on a table or *table page*:

- the table type and the level number of the table page,
- the number of reserved and current entries in this table page,
- the chaining reference to other table pages on the same level,
- the pointer to the associated table page on the next higher level,
- the pointer to the page containing the last table on the main level (for the highest-level table only).

**table page**

*Page* containing a table or part of a table. If a *table* which does not extend over several pages or the highest level of a multi-level *table* is concerned, "table page" only refers to the object involved, not the entire *page*.

**TANGRAM**

(Task and Group Affinity Management) Subsystem of BS2000 that plans the allocation of processors for task groups which access large quantities of shared data in multi-task applications.

**task attribute TP**

There are 4 task attributes in BS2000: SYS, TP, DIALOG and BATCH. Special runtime parameters that are significant for task scheduling are assigned to each of these task attributes.

In contrast to the other task attributes, the TP attribute is characterized by optimized main memory management that is specially tailored to transaction processing requirements.

**task communication**

Communication between the *DBH modules*. See also *common pool*.

**task deadlock**

See *deadlock*.

**task priority**

In BS2000, it is possible to define a priority for a task. This priority is taken into account when initiating and activating the task.

Priorities may be fixed or variable. Variable priorities are adapted dynamically; fixed priorities do not change.

Note that UDS/SQL server tasks should be started with a fixed priority in order to ensure consistent performance.

**TCUA**

See *Transaction Currency Area*.

**time acknowledgment**

Message sent by the *UDS-D task* to the remote *application program* to indicate that there is still a *DML* statement being processed.

**transaction (TA)**

Related sequence of *DML* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.

For UDS-D:

The total set of *subtransactions* active at a given time.

**transaction, committing a**

Terminating a *transaction* with FINISH, i.e. all updates performed within the transaction are committed to the *database*.

**transaction, rolling back a**

Terminating a *transaction* with FINISH WITH CANCEL, i.e. all updates performed on the *database* within the transaction are rolled back.

**Transaction Currency Area (TCUA)**

Contains currency information.

**transaction identification (TA-ID)**

Assigned by the *DBH* to identify a particular *transaction*. Can be requested with the *DAL* command DISPLAY.

**transfer pool**

UDS-D-specific storage area in which the *UDSCT* receives the *BIBs* from *remote application programs*.

**two-phase commit protocol**

Procedure by which a *distributed transaction* that has made changes in at least one *remote configuration* is terminated in such a way as to safeguard *inter-configuration consistency* or UDS/SQL openUTM-D consistency. The two-phase commit is controlled

- by the distribution component in the *user task* if the *transaction* is distributed via UDS-D.
- by openUTM-D if the transaction is distributed via openUTM-D or via openUTM-D and UDS-D.

## U

**UDSADM**

Module of the *independent DBH*; executes in the *administrator task*.

**UDSHASH**

Module generated by the BGSIA utility routine. It contains the names of all the *hash routines* defined in the *Schema DDL*.

**UDSNET**

Distribution component in the *user task*.

**UDSSQL**

Start module of the *independent DBH*; executes in the *master task*.

**UDSSUB**

Start module of the *independent DBH*; executes in the *server task*.

**UDS-D task UDSCCT**

Task started for each *configuration* by UDS/SQL so that it can participate in distributed processing with UDS-D.

**UDS/SQL / openUTM-D consistency**

A *transaction* that has updated both openUTM data and UDS/SQL *databases* is terminated in such a way that the openUTM data and the UDS/SQL databases are either updated together or not at all.

**UDS/SQL pubset declaration**

Declaration in a *pubset declaration job variable* for restricting the UDS/SQL pubset environment. This reduces or prevents the risk of file names being ambiguous.

**unique throughout the network**

Unique in all the computers that are included in the *network*.

**user database**

The *realms* and files of the *database* required by the user in order to be able to store data in, and to retrieve data from a database are:

- the *Database Directory (DBDIR)*,
- the *user realms*
- the module library for *hash routines (HASHLIB)*.

**user realm**

A *realm* defined in the realm entry of the *Schema DDL*. It contains, among other things, the user records.

**user task**

Execution of an *application* program or openUTM program, including the parts linked by the system.

**USER-WORK-AREA (UWA)**

Transfer area for communication between the *application program* and the *DBH*.

**UTM**

See openUTM.

**UWA**

See *USER-WORK-AREA (UWA)*.



## V

**vector**

*Item* with repetition factor. The repetition factor must be greater than 1. It specifies how many duplicates of the item are combined in the vector.

**version number, internal**

See *internal version number*.

## W

**warm start**

A warm start is performed by UDS/SQL if an inconsistent *database* is attached to a *session*. For UDS/SQL this involves applying all updates of completed *transactions* to the database which have not yet been applied, *rolling back* all database transactions that are open, and making the database consistent. The related *RLOG file* and the *DB status file* are required for a warm start.



---

## Abbreviations

|         |                                     |
|---------|-------------------------------------|
| ACS     | Alias Catalog Service               |
| Act-Key | ACTual KEY                          |
| AFIM    | AFter-IMage                         |
| AP      | Application Program                 |
| ASC     | ASCending                           |
| BIB     | Base Interface Block                |
| BFIM    | BeFore-IMage                        |
| COBOL   | COmmon Business Oriented Language   |
| CODASYL | COnference on DAta SYstem Languages |
| CRA     | CuRrent of Area                     |
| CRR     | CuRrent of Record                   |
| CRS     | CuRrent of Set                      |
| CRU     | Current of RunUnit                  |
| COSSD   | COBOL SubSchema Directory           |
| DAL     | Database Administration Language    |
| DB      | DataBase                            |
| DBCOR   | DataBase COmpiler Realm             |
| DBDIR   | DataBase DIRectory                  |
| DBH     | DataBase Handler                    |
| DB-Key  | DataBase Key                        |
| DBTT    | DataBase key Translation Table      |
| DDL     | Data Description Language           |
| DESC    | DESCending                          |
| DML     | Data Manipulation Language          |
| DRV     | Dual Recording by Volume            |
| DSA     | Database System Access              |
| DSSM    | Dynamic SubSystem Management        |

## Abbreviations

---

|         |                                                                                             |
|---------|---------------------------------------------------------------------------------------------|
| FC      | Function Code                                                                               |
| FPA     | Free Place Administration                                                                   |
| GS      | Global Storage                                                                              |
| HSMS    | Hierarchic Storage Management System                                                        |
| ID      | IDentification                                                                              |
| IQL     | Interactive Query Language                                                                  |
| IQS     | Interactive Query System                                                                    |
| KDBS    | Kompatible Datenbank-Schnittstelle (= compatible database interface)                        |
| KDCS    | Kompatible Datenkommunikationsschnittstelle<br>(= compatible data communications interface) |
| LM      | Lock Manager                                                                                |
| LMS     | Library Maintenance System                                                                  |
| MPVS    | Multiple Public Volume Set                                                                  |
| MR-NR   | MainRef NumberR                                                                             |
| MT      | Master task                                                                                 |
| OLTP    | OnLine transaction processing                                                               |
| openUTM | Universal Transaction Monitor                                                               |
| OT      | Operator Task                                                                               |
| PETA    | Preliminary End of TrAnsaction                                                              |
| PPP     | Probable Position Pointer                                                                   |
| PTC     | Prepared To Commit                                                                          |
| PTT     | Primäre Teiltransaktion (= primary subtransaction)                                          |
| PVS     | Public Volume Set                                                                           |
| REC-REF | RECOrd REFErence number                                                                     |
| RSQ     | Record Sequence Number                                                                      |
| SC      | SubControl                                                                                  |
| SCD     | Set Connection Data                                                                         |
| SCI     | Software Configuration Inventory                                                            |
| SECOLTP | SECure OnLine Transaction Processing                                                        |
| SECOS   | SECurity COntrol System                                                                     |
| SET-REF | SET-REFerence                                                                               |
| SIA     | Schema Information Area                                                                     |
| SIB     | SQL Interface Block                                                                         |

|         |                                                        |
|---------|--------------------------------------------------------|
| SLF     | Session Log File                                       |
| SQL     | Structured Query Language                              |
| SSD     | Solid State Disk                                       |
| SSIA    | SubSchema Information Area                             |
| SSITAB  | SubSchema Information TABLE                            |
| SSL     | Storage Structure Language                             |
| ST      | ServerTask                                             |
| STT     | Sekundäre Teiltransaktion (= secondary subtransaction) |
| TA      | TrAnsaction                                            |
| TA-ID   | TrAnsaction IDentification                             |
| TANGRAM | TAsk aNd GRoup Affinity Management                     |
| TCUA    | Transaction CUurrency Area                             |
| UDS/SQL | Universal Database System/Structured Query Language    |
| UWA     | User Work Area                                         |



---

## Related publications

You will find the manuals on the internet at <http://manuals.ts.fujitsu.com>. You can order printed copies of those manuals which are displayed with an order number.

**UDS/SQL (BS2000)**  
**Creation and Restructuring**  
User Guide

**UDS/SQL (BS2000)**  
**Database Operation**  
User Guide

**UDS/SQL (BS2000)**  
**Design and Definition**  
User Guide

**UDS/SQL (BS2000)**  
**Messages**  
User Guide

**UDS/SQL (BS2000)**  
**Recovery, Information and Reorganization**  
User Guide

**UDS/SQL (BS2000)**  
**Ready Reference**

**UDS (BS2000)**  
**Interactive Query System IQS**  
User's Guide

**UDS-KDBS (BS2000)**  
Compatible Database Interface  
User Guide

**SQL for UDS/SQL**

Language Reference Manual

**BS2000 OSD/BC**

**Commands**

User Guide

**BS2000 OSD/BC**

**Introduction to System Administration**

User Guide

**BS2000 OSD/BC**

**Executive Macros**

User Guide

**BS2000 OSD/BC**

**Introductory Guide to DMS**

User Guide

**SDF (BS2000)**

**SDF Dialog Interface**

User Guide

**SORT (BS2000)**

User Guide

**SPACEOPT (BS2000)**

**Disk Optimization and Reorganization**

User Guide

**LMS (BS2000)**

**SDF Format**

User Guide

**DSSM/SSCM**

**Subsystem Management in BS2000**

User Guide

**ARCHIVE (BS2000)**

User Guide

**DRV (BS2000)**

**Dual Recording by Volume**

User Guide



**HSMS / HSMS-SV** (BS2000)  
**Hierarchical Storage Management System**  
**Volume 1: Functions, Management and Installation**  
User Guide

**SECOS** (BS2000)  
**Security Control System**  
User Guide

**openNet Server** (BS2000)  
**BCAM**  
Reference Manual

**DCAM** (BS2000)  
**Program Interfaces**  
Reference Manual

**DCAM** (BS2000)  
**Macros**  
User Guide

**OMNIS/OMNIS-MENU** (BS2000)  
**Functions and Commands**  
User Guide

**OMNIS/OMNIS-MENU** (BS2000)  
**Administration and Programming**  
User Guide

**openUTM**  
**Concepts and Functions**  
User Guide

**openUTM**  
**Programming Applications with KDCS for COBOL, C and C++**  
User Guide

**openUTM**  
**Generating Applications**  
User Guide

**openUTM**  
**Administering Applications**  
User Guide

**openUTM**  
**Using openUTM Applications under BS2000**  
User Guide

**openUTM**  
**Messages, Debugging and Diagnostics in BS2000**  
User Guide

**COBOL2000 (BS2000)**  
**COBOL Compiler**  
Reference Manual

**COBOL2000 (BS2000)**  
**COBOL Compiler**  
User's Guide

**COBOL85 (BS2000)**  
**COBOL Compiler**  
Reference Manual

**COBOL85 (BS2000)**  
**COBOL Compiler**  
User's Guide

**CRTE (BS2000)**  
**Common Runtime Environment**  
User Guide

**DRIVE/WINDOWS (BS2000)**  
Programming System  
User Guide

**DRIVE/WINDOWS (BS2000)**  
Programming Language  
Reference Guide

**DRIVE/WINDOWS (BS2000)**  
System Directory of DRIVE Statements  
Reference Manual

**DRIVE/WINDOWS (BS2000/SINIX)**  
Directory of DRIVE SQL Statements for UDS  
Reference Manual

**DAB** (BS2000)  
**Disk Access Buffer**  
User Guide

**Unicode in BS2000**  
Introduction

**XHCS** (BS2000)  
8-Bit Code and Unicode Processing in BS2000  
User Guide

**BS2000 OSD/BC**  
**Softbooks English**  
DVD

**openSM2** (BS2000)  
**Software Monitor**  
User Guide

**SNMP Management** (BS2000)  
User Guide



---

# Index

\$UDSDLIB, link name 97  
\$UDSKLIB, link name 97  
\$UDSKONF, link name 99  
\$UDSLIB, link name 97  
\$UDSPLEX, link name 98  
\$UDSSSI, link name 98  
\$UL, dynamic-loading procedure  
(abbreviation) 97, 98

## A

ACCEPT 46, 63, 139, 227

access

- contending 401
- direct 401
- sequential 401
- to the CRR 55

access authorization 401

access CRA 61

access CRS 60

access owner of CRS 60

access path 401

access rights 401

access type 401

access, direct

- at record level 48
- at set level 56
- via CALC key 49
- via database key 48

access, sequential

- at realm level 61
- at record type level 55
- at set level 59

act-key 402

act-key-0 page 402

act-key-N page 402

ADD 305, 315

address

- physical 402

administrator task 402

AFIM 402

after-image 402

- ALOG file 402

- RLOG file 402

ALOG file 402

ALOG sequence number 403

alog-seq-no 23

anchor record 403

AND groups

- in FIND/FETCH 174

appl 23

application 403

application program (AP) 403

area 403

ascending key (ASC key) 403

Assembler example 297

Assembler macros 259

authorization 403

authorized users 403

automatic DBTT extension 403

AUTOMATIC member 65

automatic realm extension 403

## B

backup database 404

Base Interface Block (BIB) 404

BCALLSI 83

before-image 404

- BFIM 404
- BIB (Base Interface Block) 404
- buffer pools
  - see exclusive buffer pool
  - see system buffer pools
- C**
- CALC key 153, 169, 404
  - in FIND/FETCH 175
- CALC key value 49
- CALC page 404
- CALC SEARCH key 404
- CALC table 405
- CALL DML 83, 92, 197, 335, 405
  - converter 83
  - parameter mechanism 260
- CALL DML functions 211
- CALL DML parameters
  - define 198
  - format table 201
  - formatting rules 199
- CALL DML program 76
  - generate 83
  - link 100
  - structure 84
- CALL DML status codes 378
- CALL interface 29, 197
- catalog identifier 405
- catid 23
- chain 405
- Character Separated Values (CSV) 405
- check records 405
- checkpoint 405
- CHECK-TABLE 405
- clone 406
- closing a transaction 45
- COBOL DML 77, 86, 129, 406
- COBOL DML program 76
  - link 100
  - preparation 77
  - structure 77
- COBOL example 291
- COBOL runtime system 406
- COBOL Subschema Directory (COSSD) 406
  - collection of records 56, 164
  - common memory 406
  - common pool 406
  - communication area 29, 86
  - communication partners 406
  - communication pool 406
  - comparison condition 53
  - compatible database interface 406, 418
  - compiler database 407
  - COMPILER-SCHEMA 407
  - COMPILER-SUBSCHEMA 407
  - compound CALC key
    - in FIND/FETCH 175
  - compound key 407
    - use in FIND/FETCH 171
  - compression 407
  - configuration 407
  - configuration identification 407
  - configuration name 407
  - CONNECT 66, 143, 229
  - connect
    - into a set occurrence 66
    - into set occurrence 189, 256
  - connection module 407
    - version-dependent 100
    - version-independent 100
  - consistency 31, 407
    - logical 408
    - physical 408
    - storage 408
  - consistency error 408
  - consistency point 408
  - consistency record 408
  - contending access 408
  - CONTINUE 305, 316
  - conversation 408
  - copy 408
  - copyname 23
  - COSSD 408
  - COSSD file 90
  - CRA 37, 61, 63, 158, 160, 408
  - CRR 37, 49, 50, 55, 63, 64, 153, 155, 158, 160, 409

- CRS [37](#), [56](#), [63](#), [64](#), [67](#), [74](#), [155](#), [158](#), [160](#), [162](#), [169](#), [181](#), [183](#)
- CRU [37](#), [62](#), [63](#), [64](#), [71](#), [73](#), [143](#), [144](#), [180](#), [181](#)
- c-string [24](#)
- CSV [409](#)
- csv-dateiname [24](#)
- currency information [37](#)
- currency table [37](#), [38](#), [45](#), [409](#)
  - columns [37](#), [38](#), [46](#)
- currency table updating
  - explicit control of [47](#)
  - suppress [46](#)
- CURRENT
  - OF AREA table [409](#)
  - OF RECORD table [409](#)
  - OF SET table [409](#)
- Current
  - Record of Rununit [409](#)
  - Record of Set (CRS) [409](#)
- D**
- dal-cmd [24](#)
- data backup [409](#)
- data deadlock [409](#)
- data division [131](#)
- Data Manipulation Language (DML) [414](#)
- data modification [65](#)
- data protection (privacy) [410](#)
- data retrieval [46](#)
- data type
  - structured-name [25](#)
- data types [23](#)
- database (DB) [410](#)
- database administrator [410](#)
- Database Administrator Language (DAL) [409](#)
- database compiler realm (DBCOM) [410](#)
- database copy [410](#)
- database copy update [410](#)
- database directory (DBDIR) [410](#)
- database exception condition [73](#), [74](#), [89](#), [117](#), [194](#)
- database job variable [410](#)
- database key [410](#)
  - item [411](#)
- database key value [37](#), [46](#), [55](#), [63](#), [68](#), [88](#), [150](#), [190](#)
- database page [411](#)
- database system [411](#)
- DATABASE-KEY item [411](#)
- DATABASE-KEY-LONG item [411](#)
- DATABASE-STATUS [411](#)
- date [24](#)
- DB configuration [411](#)
- DB entry [86](#)
- DB key [412](#)
- DB record [113](#), [394](#)
- DB status file [411](#)
- DB trace information, openUTM [390](#), [394](#)
- DBCOM [411](#)
- DB-DIAGAREA, openUTM [390](#), [394](#)
- DBDIR [412](#)
- DBH [305](#), [316](#), [412](#)
  - end [412](#)
  - independent [95](#), [412](#)
  - linked-in [95](#), [412](#)
  - start [412](#)
- DBH load parameters [412](#)
- DBH variants [103](#)
- dbname [24](#)
- DBTT [412](#)
  - page [413](#)
- DBTT anchor page [412](#)
- DBTT base [412](#)
- DBTT extension
  - automatic [403](#)
  - online [422](#)
- DBTT extent [413](#)
- DCAM [413](#)
- DCAM application [413](#)
- DCL [305](#)
- DDL [413](#)
- deadlock [33](#), [44](#), [413](#)
- DECLARE [305](#), [316](#)
- DEFINE [305](#), [317](#)
- DELETE [305](#), [317](#)
- delete records [69](#), [231](#)
- descending key (DESC key) [413](#)
- device [24](#)

- direct access 413
  - direct access path 170
  - direct hash area 413
  - DISCONNECT 67, 144, 230
  - DISPLAY 305, 318
  - DISPOFF 305, 318
  - distributable list 49, 65, 153, 190, 210, 257, 346, 357, 374, 419
    - preferred realm 66, 190, 357
    - table realm 419
  - distributed database 413
  - distributed transaction 414
  - distribution pool 414
  - distribution table 414
  - DML functions 41
  - DML program
    - execute 103
    - link 100
    - start 103
  - DML status codes 365
  - DMLTEST 301
    - commands 305
    - communication with databases 362
    - error messages 363
    - interrupts 359
  - DO 305, 319
  - DOFF 305
  - DSCAL 259, 261
  - DSCAP 259, 262
  - DSCDF 259, 263
  - DSCPA 259, 265
  - dummy subtransaction 414
  - duplicate 49, 50, 54, 56, 59, 153
  - duplicates header 414
  - duplicates table 415
    - main level 415
  - dynamic loading 95, 100
  - dynamic set 50, 53, 56, 58, 67, 143, 144, 164, 165, 415
  - dynamic-loading strategy 95
- E**
- EDT 305, 321
  - EHPROT 332
  - END 305, 321
  - ERASE 69, 146, 231
  - error handling 114, 124
  - error handling routine 89
  - ESCAPE 305, 321
  - ESTIMATE-REPORT 415
  - event name 415
  - EXCLUSIVE 43
  - exclusive buffer pool 415
  - exclusive locking 34
  - EXECUTE 305, 322
  - execution, see program execution
- F**
- FCOD 198, 201, 211
  - FETCH 46, 46, 122, 148, 232
  - FETCH-1 48, 150, 233
  - FETCH-2 49, 153, 234
  - FETCH-3 50, 56, 155, 235
  - FETCH-4 54, 59, 61, 158, 236
  - FETCH-5 47, 55, 60, 61, 160, 238
  - FETCH-6 60, 162, 239
  - FETCH-7 50, 56, 163, 240
  - FIND 46, 46, 117, 122, 148, 232
  - FIND-1 48, 150, 233
  - FIND-2 49, 153, 234
  - FIND-3 50, 56, 155, 235
  - FIND-4 54, 59, 61, 158, 236
  - FIND-5 47, 55, 60, 61, 160, 238
  - FIND-6 60, 162, 239
  - FIND-7 50, 56, 163, 240
  - FINISH 31, 45, 179, 247
  - FOPT 198, 201, 211
  - foreign key 415
  - FORTRAN example 294
  - FPA 415
  - FPA base 416
  - FPA extent 416
  - FPA page 416
  - FREE 72, 179, 247
  - free place administration 416
  - function code 198, 416
  - function option 198
  - functions of DML 30, 340



**G**

GET 46, 62, 180, 248  
group item 416

**H**

HALT 305, 323  
hash area 416  
hash procedure 49, 153  
hash routine 416  
hashing 416  
HASHLIB 417  
HELP 306, 323  
host 24

**I**

identification division 131  
identifier 417  
IF 73, 73, 181, 250  
IMON 95  
implicit set 417  
inconsistency 417  
independent DBH 103, 417  
index level 417  
index page 417  
INDEX search key 417  
indirect hash area 417  
insert into set occurrence 143, 229  
integer 24  
integrity 418  
interconfiguration 418  
    consistency 418  
    deadlock 418  
interface 418  
internal version number 418  
interrupt handling 114  
    STXIT concurrency 116  
item 418  
    modify 183  
item name 198  
items, national 126, 318, 331  
ITMN 198, 201, 211

**K**

KDBS 314, 335, 362, 406, 418

KEEP 71, 182, 251  
key 86, 419  
    compound 419  
key item 419  
key reference number 419  
keyword 20, 21, 131, 312  
keyword parameters 311  
kset 24

**L**

LANGUAGE 306, 324  
LCCONCT, connection module 100  
LEAVE 306, 325  
levels of locking 34  
link  
    UDS/SQL openUTM application 109  
    UDS/SQL-TIAM application 100  
linked-in control system 419  
linked-in DBH 31, 105, 419  
LIST 306, 326  
list 419  
    distributable, see distributable list  
literals, national 327, 334  
load parameters DBH 419  
local application program 419  
local configuration 420  
local database 420  
local distribution table 420  
local host 420  
local transaction 420  
lock  
    at page level 34  
    at realm level 34, 43, 179, 186, 254  
    at record level 71, 182, 251  
logging 420  
logical connection 420  
LOOKC 94, 266  
    block 268  
    block, general description 269  
    block, special description 270  
    parameter description 275  
    tables 278  
LS 306

### M

- main reference [420](#)
- mainref number [420](#)
- MANDATORY member [66](#)
- MANUAL member [66](#)
- mask [50](#), [53](#), [56](#), [165](#), [166](#), [176](#), [246](#)
- mask condition
  - in FIND/FETCH [171](#)
- masking symbols [102](#)
- master task [420](#)
- MEMBER [73](#)
- member [421](#)
  - AUTOMATIC [65](#), [70](#), [189](#), [421](#)
  - MANDATORY [66](#), [70](#), [146](#), [421](#)
  - MANUAL [66](#), [70](#), [421](#)
  - OPTIONAL [66](#), [70](#), [144](#), [146](#), [421](#)
- member record [58](#), [69](#), [73](#), [158](#), [421](#)
- member record type [421](#)
- metavariable [130](#)
- MODIFY [68](#), [183](#), [251](#)
- modifying
  - data [70](#)
  - item contents [68](#)
- mono-DB configuration [421](#)
- mono-DB operation [33](#), [44](#), [78](#), [104](#), [421](#)
- MOVE [306](#), [327](#)
- multi-DB configuration [421](#)
- multi-DB operation [32](#), [80](#), [84](#), [421](#)
- multi-DB program [421](#)
- multi-level table [421](#)
- multithreading [422](#)

### N

- name [24](#)
- name conflicts during dynamic loading [102](#)
- NATIONAL [126](#), [318](#), [327](#), [331](#), [373](#)
- national items [126](#), [318](#), [331](#)
- national literals [327](#), [334](#)
- nesting [319](#), [333](#)
- network [422](#)
- NEXT [306](#), [329](#)
- notational conventions [20](#), [21](#)
  - SDF statements [22](#)

### O

- OLTP [422](#)
- online backup [422](#)
- online DBTT extension [422](#)
- online realm extension [422](#)
- open transaction [186](#), [422](#)
- opening of transactions [43](#)
- openUTM [423](#)
  - DB record [394](#)
  - DB trace information [390](#), [394](#)
  - DB-DIAGAREA [390](#), [394](#)
- operator task (OT) [423](#)
- OPTIONAL member [66](#)
- optional word [20](#), [21](#)
- OR groups
  - in FIND/FETCH [176](#)
- original database [423](#)
- overflow pages [423](#)
- OWNER [73](#)
- owner [423](#)
- owner record [60](#), [66](#), [69](#), [73](#), [159](#), [162](#), [190](#), [423](#)
- owner record type [423](#)

### P

- P1 eventing [426](#)
- page [423](#)
- page address [423](#)
- page container [424](#)
- page header (page info) [424](#)
- page index entry [424](#)
- page length [129](#)
- page number [424](#)
  - relative [49](#)
- page protection [34](#)
- parameters, CALL DML [198](#)
- password for UDS/SQL files [424](#)
- pattern [424](#)
- pattern string [424](#)
- PERFORM [306](#), [330](#)
- PETA [425](#)
- PLITAB module [98](#)
- POFF [306](#)
- pointer array [425](#)
- PPP (probable position pointer) [425](#), [426](#)

- preferred realm
    - distributable list [66](#), [190](#), [357](#)
  - prepared to commit (PTC) [425](#)
  - primary key [170](#), [425](#)
  - primary key (DDL) [425](#)
  - primary key (SQL) [425](#)
  - primary subtransaction [426](#)
  - PRINT [306](#), [331](#)
  - PRIVACY [86](#), [131](#)
  - PRIVACY-AND-IQF SCHEMA [426](#)
  - PRIVACY-AND-IQF SUBSCHEMA [426](#)
  - probable position pointer (PPP) [425](#), [426](#)
  - PROC [306](#), [331](#)
  - procedure division [133](#), [137](#)
  - processing chain [32](#), [426](#)
  - product version
    - specify [95](#), [112](#)
  - PROFF [306](#), [332](#)
  - program execution
    - DML program [103](#)
  - programming languages [29](#), [197](#)
  - PROT [306](#)
  - PROTECTED [43](#)
  - protection functions [31](#)
  - PTC state [426](#)
  - pubset declaration [426](#)
  - pubset declaration job variable [426](#)
- R**
- Readme file [17](#)
  - READY [31](#), [32](#), [43](#), [186](#), [254](#), [427](#)
  - READYC [427](#)
  - realm [427](#)
  - realm configuration [427](#)
  - realm copy [427](#)
  - realm extension
    - automatic [403](#)
    - online [422](#)
  - realm level [61](#)
  - realm name [64](#), [142](#), [198](#)
  - realm reference number [427](#)
  - realm-name [25](#)
  - realmref [25](#)
  - RECA [93](#), [198](#), [201](#), [211](#)
  - RECN [198](#), [201](#), [211](#)
  - reconfiguration [427](#)
  - record [427](#)
    - delete [69](#)
    - find [46](#)
    - modify [65](#)
    - remove [146](#)
    - store [65](#)
  - record address [427](#)
  - RECORD AREA [428](#)
  - record area [86](#), [93](#), [198](#)
  - RECORD AREA, see record area
  - record element [50](#), [57](#), [62](#), [68](#), [156](#), [166](#), [170](#), [180](#), [183](#), [428](#)
  - record hierarchy [428](#)
  - record level [71](#)
  - record name [198](#)
  - record protection
    - extended [45](#), [71](#), [179](#), [182](#), [247](#), [251](#)
  - record reference number [428](#)
  - record SEARCH KEY table [428](#)
  - record selection expression [46](#), [150](#), [153](#)
  - record sequence number [428](#)
  - record type [428](#)
    - linear [428](#)
  - record type level [48](#), [55](#)
  - record-name [25](#)
  - recordref [25](#)
  - recovery concept [30](#), [33](#)
  - REC-REF [428](#)
  - referential integrity [428](#)
  - relational operators [50](#), [56](#)
  - relationship
    - between records [29](#)
    - of sets [181](#)
  - REMARK [306](#), [332](#)
  - remote application program [428](#)
  - remote configuration [429](#)
  - remote database [429](#)
  - remote host [429](#)
  - repeating group [429](#)
  - request [429](#)
  - reserved words [129](#)

- restart
  - of a session 429
  - of BMEND 429
- restructuring 429
- result set 145
- RETAINING 39, 47, 67, 143, 148, 189
- RETRIEVAL 43
- retrieval of data 232
- retrieve
  - database key values 63
  - realm 64
- return code 429
- RLMN 198, 201, 211
- RLOG file 429
- rollback 33, 430
- RSQ 430
- RUN 307, 333
- RUNUNIT-ID 430
  
- S**
- schema 430
- schema DDL 430
- Schema Information Area (SIA) 430
- schema-name 25
- SCI Software Configuration Inventory 95
- SCSXUSER 114
- SDF statements notational conventions 22
- search expression 50, 58, 155, 165, 170, 245, 335
  - in FIND/FETCH 171
- SEARCH KEY 430
- SEARCH KEY table 430
- secondary key 170, 430
- secondary option 198, 203
  - fixed-format 204
  - free-format 204
- secondary subtransaction 430
- SELECT-PRODUCT-VERSION 95, 104, 105
- sequence number 430
- sequence of processing
  - in FIND/FETCH 174
- sequential access 431
- server task 431
  
- session 431
  - abort 431
  - end 431
  - interrupt 431
  - start 432
- session job variable 431
- Session Log File (SLF) 431
- session restart 432
- session section 432
- session section number 432
- SET 88, 187, 307
- set 432
  - dynamic 50, 53, 56, 58, 67, 143, 144, 164, 165, 432
  - implicit 432
  - singular 432
  - standard 432
- SET command 334
- Set Connection Data (SCD) 432
- set connections, releasing 230
- set level 56, 59
- set membership 66, 70, 73
- set name 198
- set occurrence 56, 65, 68, 69, 73, 74, 143, 144, 155, 159, 164, 169, 183, 189, 191, 432
- set reference number 432
- set SEARCH KEY table 433
- SETN 198, 201, 211
- set-name 25
- SF pubset 433
- shadow database 433
- Shared User buffer pool 433
- SHOW 307, 336
- SIA 433
- SIB 433
- single feature pubset 433
- SLF 433
- SM pubset 433
- snap 434
- SOPT 198, 201, 203, 211
- sort dynamic set 50, 56
- sort key table 434
- sort sequence 51, 57
- source program 434

- spanned record 434
  - special parameter 1 198, 209
  - special parameter 2 198, 210
  - special parameter 3 198
  - special registers 88, 117
  - SPP1 198, 202, 209, 211
  - SPP2 198, 202, 210, 211
  - SPP3 198, 202, 211
  - SQL 434
  - SQL conversation 434
  - SQL DML 434
  - SQL Interface Block (SIB) 435
  - SQL transaction 435
  - SSIA 435
  - SSIA-RECORD 435
  - SSITAB module 76, 83, 435
    - load dynamically 98
  - SSL 435
  - standard set 435
  - start
    - UDS/SQL-TIAM application 103
  - statement code 117, 118, 435
  - status code 117, 119, 435
  - status codes
    - UDS online utility 365, 374
  - STOP 305, 323
  - Storage Structure Language (SSL) 435
  - STORE 65, 117, 189, 256
  - store record 189, 256
  - string 435
  - structure information 266
  - structured-name (data type) 25
  - STXIT 114
    - SCSXUSER 114
  - STXIT concurrency 116
  - subcontrol system 435
  - SUBSCHEMA 307
  - subschemata 436
  - Subschema DDL 436
  - Subschema Information Area (SSIA) 436
  - subschemata module 436
  - subschemata record 436
  - SUB-SCHEMA SECTION 131, 436
  - subschemata-name 25
  - subtransaction 436
  - syntax description 22
  - SYSTEM 307, 337, 337
  - system area 436
  - system break information 436
  - system buffer pools 437
  - system managed pubset 437
  - SYSTEM record 437
  - SYSTEM set 437
- ## T
- table
    - multi-level 437
  - table (SQL) 437
  - table header 437
  - table pages 437
  - table realm
    - distributable list 419
  - TANGRAM 438
  - task attribute TP 438
  - task communication 438
  - task deadlock 438
  - task priority 438
  - TCUA 438
  - TENANT 73
  - terminate transaction 179
  - testing DML functions 301
  - time 26
  - time acknowledgment 438
  - TRACE 307, 338
  - transaction 31, 71, 438
    - close 45
    - committing a 438
    - open 43, 186, 254
    - roll back 439
    - terminate 179, 247
  - Transaction Currency Area 439
  - transaction identification (TA-ID) 439
  - transfer pool 439
  - translation table
    - UDSTRTAB 126
  - TSK, dynamic-loading procedure (abbreviation) 97, 98
  - two-phase commit protocol 439

- U**
- UDS [439](#)
- UDS online utility
  - status codes [365, 374](#)
- UDS/SQL [440](#)
- UDS/SQL / openUTM-D consistency [440](#)
- UDS/SQL openUTM application
  - linking [109](#)
- UDS/SQL pubset declaration [440](#)
- UDS/SQL-TIAM application
  - link [100](#)
  - start [103](#)
- UDSADM [439](#)
- UDSBCCON, connection module [100](#)
- UDS-D task UDSCT [440](#)
- UDSHASH [439](#)
- UDSLNKA, connection module [100](#)
- UDSLNKI, connection module [100](#)
- UDSLNKL, connection module [100](#)
- UDSNET [439](#)
- UDSSUB [439](#)
- UDSTRTAB [126](#)
- UINF [198, 201, 205, 211](#)
- Unicode [126, 318, 327, 331, 373](#)
- unique throughout the network [440](#)
- UPDATE [43](#)
- USAGE-MODE [31, 43, 186](#)
- USE [89, 194](#)
- user database [440](#)
- user information [198, 205](#)
  - area [117](#)
- user information area [205](#)
- user realm [440](#)
- user task [440](#)
- userid [26](#)
- USER-WORK-AREA (UWA) [440](#)
- UTF-16 [126, 318, 327, 331, 373](#)
- UTM-DB-DIAGAREA [113, 394](#)
- UWA [29, 440](#)
  
- V**
- variable [20, 21](#)
- vector [441](#)
  
- version number
  - internal [441](#)
- volume [26](#)
  
- W**
- WAIT [307, 339](#)
- warm start [441](#)
  
- X**
- x-string [26](#)