Deutsch

FUJITSU Software BS2000

UDS/SQL V2.8

Ready Reference

Ready Reference

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

The **U**niversal **D**atabase **S**ystem 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 13.

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 restructuring	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 program-ming	Creation and re-structuring	Administra-tion 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 reorgan-isation 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 program-ming	Creation and re-structuring	Administra-tion 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

This manual is a practical reference for the experienced user of the UDS/SQL system.

1.3 Summary of contents

What does this manual contain?

This manual contains a summary of all important syntax descriptions, tables and error recovery measures found in the following UDS/SQL manuals:

– Design and Definition
– Application Programming
– Creation and Restructuring
– Database Operation
– Recovery, Information and Reorganization

Using the manual

The table of contents and running headers help you to locate information you may require. The manual contains cross-references to the various UDS/SQL manuals indicating where more detailed information can be found.

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
KEYWORD	Keywords are shown in underlined uppercase letters. You must specify at least the underlined parts of a keyword.	DATABASE-KEY MANUAL
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
variable	Variables are shown in italic lowercase letters. In a format which contains variables, a current value must be entered in place of each variable.	*item-name* *literal-3* *integer*
{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.	{CALC INDEX} {VALUE IS VALUES ARE}
[optional]	The expression in square brackets can be omitted. UDS/SQL then uses the default value The brackets themselves must no be specified.	[IS *integer*] [WITHIN *realm-name*]

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.	*item-name*,... {SEARCH KEY.....}...
..... or . .	Indicates where entries have been omitted for reasons of clarity. When the formats are used, these omissions are not allowed.	SEARCH KEY IS ..... RECORD NAME .. .
.	The period must be specified and must be followed by at least one blank. The underline must not be specified.	SET SECTION. 03 *item-name*..... .
Space	Means that at least one blank has to be specified.	USING CALC

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 *.	OPEN DATABASE COPY-NAME = *NONE
=	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>
Underscoring	Underscoring denotes the default value of an operand.	SCHEMA-NAME = *STD
/	A slash separates alternative operand values.	CMD = *ALL / <dal-cmd>
(...)	Parentheses denote operand values that initiate a structure.	*KSET-FORMAT(...)

Table 4: Metasyntax (part 1 of 2)

Representation	Meaning	Examples
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	USER-GROUP-NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host>
|	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 preceding an operand corresponds to the depth of the structure.	USER-GROUP-NAME = *ALL-EXCEPT(...) *ALL-EXCEPT(...) NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host> ...
,	A comma precedes further operands at the same structure level.	,SPACE = STD
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 2)

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 and whicch 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 recommendable 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: {hh:mm:ss | hh:mm | hh} 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
x..y *unit*	For the “integer” data type: range specification. x Minimum value permitted for “integer”. x is an (optionally signed) integer. y Maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* for “integer” only: additional units. The following units may be specified: *Mbyte, Kbyte, seconds*

Table 6: Data type suffixes

1.5.4 Column conventions for Schema DDL, SSL, Subschema DDL and COBOL-DML

From column 7

identifies special lines for DDl and SSL:

* Comment line; the text in columns 8-72 is recorded by the COBOL compiler as a comment.

/ Page feed

\- Continuation line; for entries from the previous line that would have gone beyond column 72.

From column 8

The following must be entered:

- for Schema DDL and SSL:
  - the first clause of an entry
  - the MEMBER clause
- for Subschema DDL:
  - the first clause of an entry
  - the MEMBER clause
  - the first line of a DIVISION
  - the first line of a SECTION
  - level number 01
- for COBOL DML:
  clauses depending on the COBOL reference format

From column 12

COBOL DML statements can be entered up to and including column 72.

2 Schema DDL

2.1 Structure of the Schema DDL

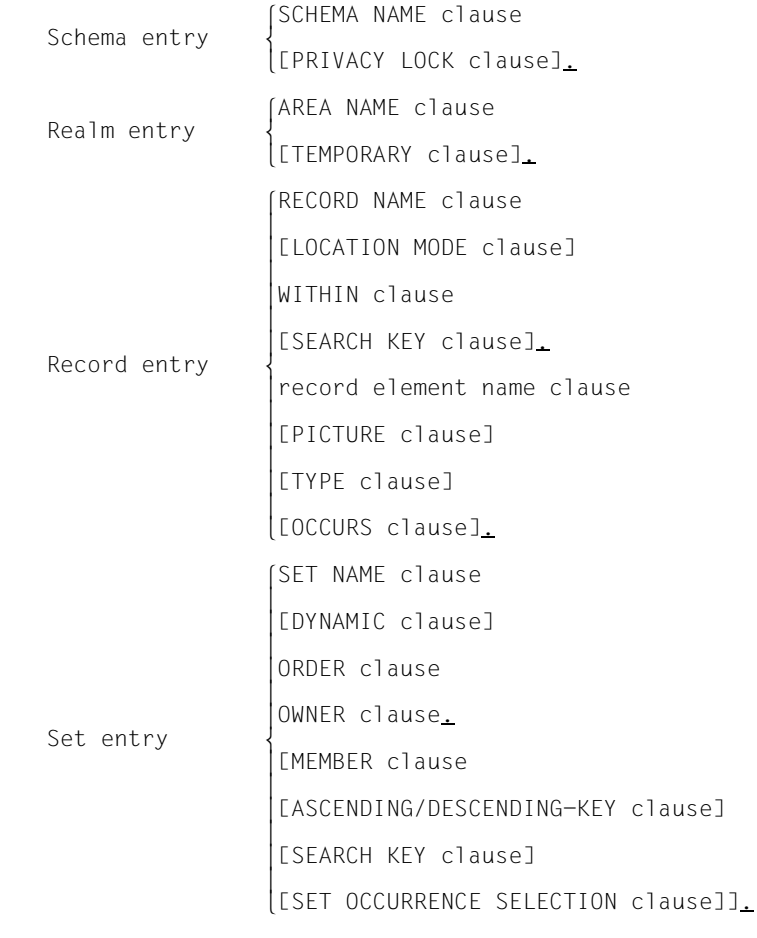


Figure 1: Structure of schema DDL

The description of the logical data structure should always be started with the schema entry and at least one realm entry.

The following applies for the subsequent realm, record and set entries:

– The two associated record types must be defined before a set can be defined.

– All the realms mentioned in the WITHIN clause for the record must be defined before a record type can be defined.

2.2 Schema entry

(cf. the "Design and Definition" manual, section 9.1.1)

```
SCHEMA NAME IS schema-name
     [PRIVACY LOCK FOR COPY IS literal-1[ OR literal-2]].
```

The schema entry is used to assign a name to the schema. Passwords can be specified to prevent unauthorized creation of subschemas from the schema.

2.3 Realm entry

(cf. the "Design and Definition" manual, section 9.1.2)

```
AREA NAME IS realm-name
    [AREA IS TEMPORARY].
```

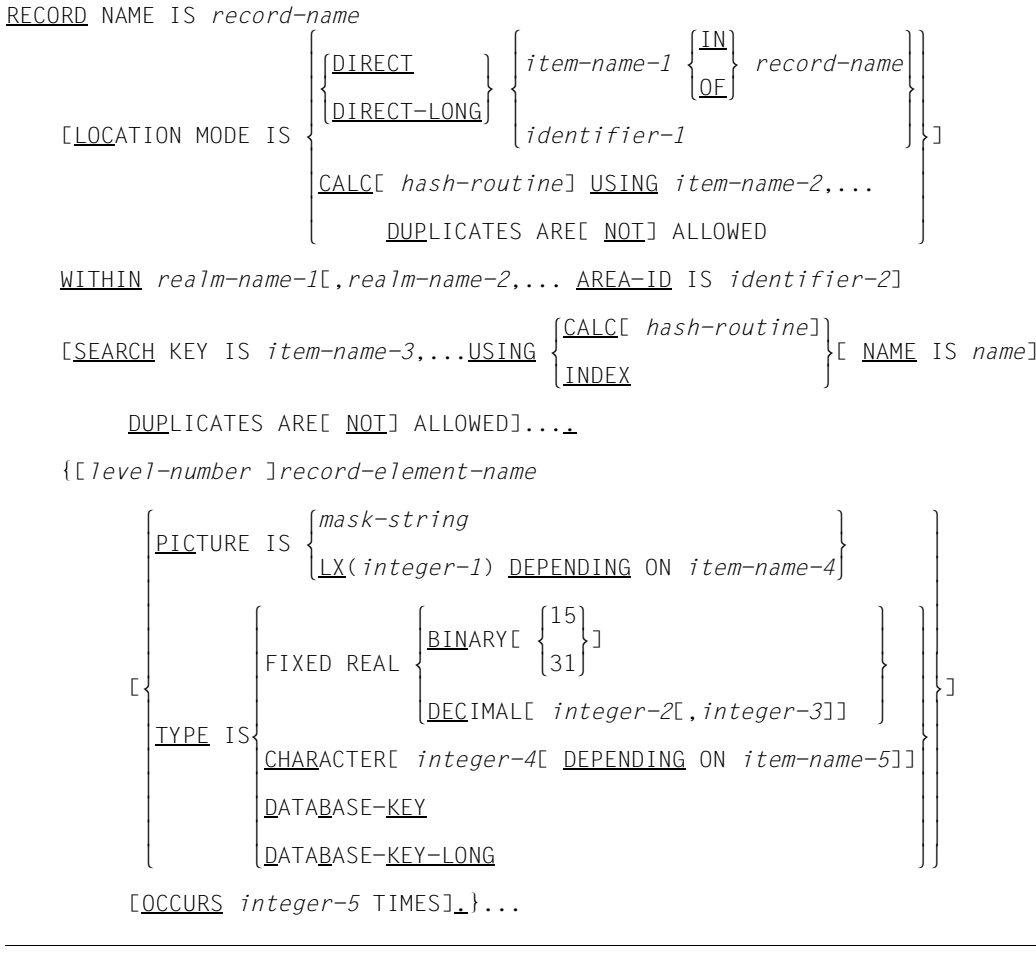
The realm entry is used to assign a name to a realm and, if necessary, to define it as a temporary realm.

A maximum of 123 realms may be defined in a database with a page length of 2048 bytes, and
a maximum of 245 realms may be defined in databases with a page length of 4000 or 8096 bytes.

Only *one* temporary realm may be defined.

2.4 Record entry

(cf. the "Design and Definition" manual, section 9.1.3)



The record entry is used to assign a name to a record type. At the same time, it can be used to define:

– the allocation of records to realms,
– the sequence of records for sequential processing,
– additional access paths for direct access via primary and secondary keys,
– all record elements to be included in the record type.

A maximum of 253 record types may be defined in a database with a page length of 2048 bytes, and
a maximum of 32 766 record types may be defined in databases with a page length of 4000 or 8096 bytes.

The individual clauses of the record entry are explained below.

2.4.1 Clauses of record entry

Clauses	Meaning
RECORD NAME IS *record-name*	A name is assigned to a record type.
[LOCATION MODE IS { {DIRECT | DIRECT-LONG} {*item-name-1* {IN | OF} *record-name* | *identifier*} | CALC[*hash-routine*] USING *item-name-2*, ... | DUPLICATES ARE[NOT] ALLOWED }]	You make it possible to assign the database key of a record which is to be stored and to specify the sequence for sequential processing. You specify a primary key to permit direct access to a particular record or to a set of records with the same key values.
WITHIN *realm-name-1*[,*realm-name-2*,... AREA-ID IS *identifier*] [SEARCH KEY IS *item-name*,... USING {CALC[*hash-routine*] | INDEX}[NAME IS *name*] DUPLICATES ARE[NOT] ALLOWED]...	The records of the record type are allocated to certain realms. Additional direct access paths via secondary key are specified and a name is assigned to the SEARCH KEY table or the hash area, which can be referenced in the SSL.
[*level-number*]*record-element-name*	A name is assigned to a record element and optionally a level number can be defined.
PICTURE IS {*mask-string* | LX(*integer-1*) DEPENDING ON *item-name*}	The PICTURE IS clause is used to define unpacked numeric items, alphanumeric items of fixed or variable length or national items.
TYPE IS { FIXED REAL {BINARY[{15 | 31}] | DECIMAL[*integer-1*[,*integer-2*]]} | CHARACTER[*integer-3*[DEPENDING ON *item-name*]] | DATABASE-KEY | DATABASE-KEY-LONG }	This clause is used to define packed numeric items, binary items, alphanumeric items of fixed or variable length, or database key items
[OCCURS *integer* TIMES]	A repetition factor is defined for a vector or a repeating group.

Table 7: Clauses of the record entry of the Schema DDL

2.5 Set entry

(cf. the "Design and Definition" manual, section 9.1.4)

```
SET NAME IS set-name

    [SET IS DYNAMIC]

                ⎧LAST                                                    ⎫
                ⎪FIRST                                                   ⎪
                ⎪NEXT                                                    ⎪
                ⎪PRIOR                                                   ⎪
    ORDER IS    ⎨IMMATERIAL                                              ⎬
                ⎪SORTED[ INDEXED[ NAME IS name]]                         ⎪
                ⎪     ⎧DATABASE-KEY                                      ⎫⎪
                ⎪  BY ⎨                                                  ⎬⎪
                ⎩     ⎩DEFINED KEYS DUPLICATES ARE[ NOT] ALLOWED         ⎭⎭

    OWNER IS    ⎧record-name⎫
                ⎨           ⎬.
                ⎩SYSTEM     ⎭

[MEMBER IS record-name ⎧MANDATORY⎫ ⎧AUTOMATIC⎫
                      ⎩OPTIONAL  ⎭ ⎩MANUAL   ⎭

    [⎧ASCENDING ⎫ KEY IS item-name-1,...]
     ⎩DESCENDING⎭

    [SEARCH KEY IS item-name-2,... USING ⎧CALC[ hash-routine]⎫
                                         ⎩INDEX               ⎭

        [NAME IS name]

        DUPLICATES ARE[ NOT] ALLOWED]...

    [SET OCCURRENCE SELECTION IS

                ⎧CURRENT OF SET                                                    ⎫
                ⎪                                                                  ⎪
                ⎪                                                ⎧item-name-3 ⎫    ⎪
         THRU   ⎨LOCATION MODE OF OWNER[ ALIAS FOR               ⎨            ⎬]]. ⎪
                ⎪                                                ⎩identifier-1⎭    ⎪
                ⎪                                  IS identifier-2]...             ⎪
                ⎩                                                                  ⎭
```

This clause is used to assign a name to a set and to

– declare the set a dynamic set if required,
– define the sequence of the member records within the set occurrences for sequential processing,
– define additional access paths via primary and secondary keys,
– declare a record type to be the owner record type of the set,
– declare a record type to be a member record type of the set if required, and define the type of membership of member records in a set, and
– specify the selection option for the set occurrences.

A maximum of 32 766 sets can be defined per database.

For each record type which is owner of a set you can generate a maximum of 255 tables in these sets. A table is created when the set mode pointer array or list or chain is of the type sorted indexed, also for each secondary key in these sets.

Irrespective of this you may define up to 255 secondary keys per record type on record type level and per singular set on set level; hash routines are not counted here.

The individual clauses of the set entry are explained below.

2.5.1 Clauses of set entry

Clauses	Meaning
SET NAME IS *set-name*	A name is assigned to the set.
[SET IS DYNAMIC]	The set is declared a dynamic set.
ORDER IS { LAST | FIRST | NEXT | PRIOR | IMMATERIAL | SORTED[INDEXED[NAME IS *name*]] | BY { DATABASE-KEY | DEFINED KEYS DUPLICATES ARE[NOT] ALLOWED } }	This clause is used to define – the sequence of the records within the set occurrences for sequential processing. – an additional direct access path via the primary key.

Table 8: Clauses of the set entry of the Schema DDL (part 1 of 2)

Clauses	Meaning
OWNER IS { *record-name* / SYSTEM }.	A record type defined by the user or a symbolic record type SYSTEM is declared owner record type of the set.
MEMBER IS *record-name* { MANDATORY / OPTIONAL } { AUTOMATIC / MANUAL }	A description of the member record type is not required for dynamic sets. In all other cases, the above clause is used to declare a record type member record type and to specify the type of membership of the member records in the set.
[{ ASCENDING / DESCENDING } KEY IS *item-name*,...]	This clause is used to define an item or a combination of items of the member record type as sort key. The member records within the set occurrence are sorted in ascending or descending order, according to the values of this key.
[SEARCH KEY IS *item-name*,... USING { CALC[*hash-routine*] / INDEX }[NAME IS *name*] DUPLICATES ARE[NOT] ALLOWED]...	This clause is used to define additional direct access paths via secondary keys and to assign a name to the SEARCH key table or the hash area, which can be referenced in the SSL.
[SET OCCURRENCE SELECTION IS { CURRENT OF SET / THRU { LOCATION MODE OF OWNER[ALIAS FOR { *item-name* / *identifier-1* } IS *identifier-2*]... } }]	This clause must be specified if the set is not a SYSTEM set. It is used to define the selection option for the set occurrences.

Table 8: Clauses of the set entry of the Schema DDL (part 2 of 2)

3 SSL

3.1 Structure of SSL

```
Schema entry    STORAGE clause.

                ⎧[RECORD NAME clause
                ⎪[DATABASE-KEY-TRANSLATION-TABLE clause]
                ⎪[record POPULATION clause]
Record entry    ⎨[PLACEMENT-OPTIMIZATION clause]
                ⎪[INDEX clause]
                ⎩[COMPRESSION clause]].

                ⎧[SET NAME clause
                ⎪[set POPULATION clause]
Set entry       ⎨[MODE clause]
                ⎪[INDEX clause]
                ⎩[PHYSICALLY LINKED clause]].
```

Figure 2: Structure of SSL

The description of the physical storage structure is optional. If it is omitted, UDS/SQL uses the default values indicated in the explanations for the individual syntax elements.

Otherwise, the description always starts with the STORAGE clause. The sequence of the record and set descriptions is arbitrary. All names referred to in the storage structure description must have been previously defined in the schema DDL.

3.2 Schema entry

(cf. the "Design and Definition" manual, section 9.2.1)

```
STORAGE STRUCTURE OF SCHEMA schema-name.
```

The schema entry is used to specify the name of the schema to which the storage structure description applies.

3.3 Record entry

(cf. the "Design and Definition" manual, section 9.2.2)

```
RECORD NAME IS record-name
    [DATABASE-KEY-TRANSLATION-TABLE[ IS integer-1][ WITHIN realm-name-1]]
    [POPULATION IS {integer-2 WITHIN realm-name-2},...]
    [PLACEMENT OPTIMIZATION FOR SET set-name]
    [INDEX NAME IS name
        [PLACING IS WITHIN realm-name-3]
                 ⎧DATABASE-KEY-LIST                                  ⎫
        [TYPE IS ⎨REPEATED-KEY                                       ⎬]]...
                 ⎩  [DYNAMIC REORGANIZATION SPANS integer-3 PAGES]   ⎭
    [COMPRESSION FOR ALL ITEMS].
```

The record entry is used to specify the name of the record type to which the storage structure description applies and

- to describe the size and physical position of the DBTT and the size of the hash areas for record SEARCH keys,
- to specify the number of records of the record type or the size of the hash area for the primary key within certain realms,
- to describe the physical position of the records within a realm if the record type is a member of a set,

– to describe the physical position, the type and the extent of reorganization employed for record SEARCH key tables or the physical position of hash areas for record SEARCH keys, and
– to indicate compression.

The individual clauses of the record entry are explained below.

3.3.1 Clauses of record entry

Clauses	Meaning
RECORD NAME IS *record-name*	This clause is used to specify the name of the record type to which the record entry applies.
[DATABASE-KEY-TRANSLATION-TABLE[IS *integer*] [WITHIN *realm-name*]]	This clause is used to describe the size and position of the DBTT and at the same time the size of the hash areas for the record SEARCH keys.
[POPULATION IS {*integer* WITHIN *realm-name*},...]	This clause is used to describe size and position of the hash areas for the primary key (LOCATION MODE IS CALC). In addition, UDS/SQL bases its assessment of the realm sizes on this entry.
[PLACEMENT OPTIMIZATION FOR SET *set-name*]	This clause is used to store the member records of the set *set-name* in the vicinity of their owner record.
[INDEX NAME IS *name* [PLACING IS WITHIN *realm-name*] [TYPE IS { DATABASE-KEY-LIST | REPEATED-KEY | [DYNAMIC REORGANIZATION SPANS *integer* PAGES] }]]...	This clause is used to specify the name of the record SEARCH key table or the hash area to which the description applies and to define – for a record SEARCH key table; the physical placement, the type and the extent of reorganization, – for a hash area; the physical placement.
[COMPRESSION FOR ALL ITEMS]	This clause causes UDS/SQL to store the records in compressed form, provided they are made available in this form at the DML interface.

Table 9: Clauses of the record entry (SSL)

3.4 Set entry

(cf. the "Design and Definition" manual, section 9.2.3)

```
SET NAME IS set-name

    [POPULATION IS integer-1[ INCREASE IS integer-2]]

                ┌CHAIN[ LINKED TO PRIOR]                          ┐
                │┌POINTER-ARRAY┐ ┌ATTACHED TO OWNER               ┐│
    [MODE IS {  ├┤             ├ ┤                               ├┤}]
                │└LIST          ┘ └DETACHED[ WITHIN realm-name-1]┘│
                │                          [ WITH PHYSICAL LINK] │
                └                                                 ┘

    [DYNAMIC REORGANIZATION SPANS integer-3 PAGES]

    [INDEX NAME IS name

        [PLACING IS {ATTACHED TO OWNER                 }]
                    {DETACHED[ WITHIN realm-name-2]    }

                ┌DATABASE-KEY-LIST                                  ┐
                │                                                   │
        [TYPE IS {REPEATED-KEY                                      }]]...
                │                                                   │
                └   [DYNAMIC REORGANIZATION SPANS integer-4 PAGES]   ┘

    [MEMBER IS PHYSICALLY LINKED TO OWNER].
```

The set entry is used to specify the set to which the storage structure description applies as well as

- to indicate the average size of the set occurrences
- to give information on the connection of the records within the set occurrences
- to define placement and extent of reorganization for pointer arrays, lists, sort key tables and set SEARCH key tables
- to indicate the type of set SEARCH key tables
- to add a pointer from member to owner.

The individual clauses of the set entry are explained below.

3.4.1 Clauses of set entry

Clauses	Meaning
SET NAME IS *set-name*	This clause is used to specify the name of the set to which the set entry applies.
[POPULATION IS *integer-1*[INCREASE IS *integer-2*]]	This clause is used to indicate the average number of member records which are expected to be in the set occurrences when the database is initially loaded or when the set occurrences are extended at a later date.
[MODE IS { CHAIN[LINKED TO PRIOR] | { POINTER-ARRAY | LIST } { ATTACHED TO OWNER | DETACHED[WITHIN *realm-name*] [WITH PHYSICAL LINK] }}] [DYNAMIC REORGANIZATION SPANS *integer* PAGES]	This clause is used to specify the linking of records within the set occurrences. When setting up a pointer array, list or sort key table, the user can also define the extent of reorganization for such tables.
[INDEX NAME IS *name* [PLACING IS { ATTACHED TO OWNER | DETACHED[WITHIN *realm-name*] }] [TYPE IS { DATABASE-KEY-LIST | REPEATED-KEY [DYNAMIC REORGANIZATION SPANS *integer* PAGES] }]]..	If a SEARCH key or sort key table has been specified for this set in the schema DDL, this table can be referenced here in order to define its placement, type and extent of reorganization. If a SEARCH key has been defined for storage in a hash area, the realm which is to contain this hash area can be specified here. Default values – for PLACING: DETACHED – for TYPE: REPEATED-KEY
MEMBER IS PHYSICALLY LINKED TO OWNER	This clause is used to include an additional pointer to its owner in each member record of the set. The SCD of a member record also contains the probable position pointers (PPP) of the associated owner record.

Table 10: Clauses of the set entry (SSL)

4 Subschema DDL

4.1 Structure of Subschema DDL

```
                    IDENTIFICATION DIVISION.
                    SUB-SCHEMA NAME clause
                    [PRIVACY LOCK clause]
                    [PRIVACY KEY clause].
                    DATA DIVISION.
                    AREA SECTION.
                    COPY clause.
                    RECORD SECTION.
                    [COPY clause.]

                   ⎧[record name clause.
                   ⎪
                   ⎪record element name clause
                   ⎪
                   ⎪[GROUP-USAGE clause]
                   ⎪
                   ⎪[PICTURE clause]
Record entry       ⎨
                   ⎪[USAGE clause]
                   ⎪
                   ⎪[OCCURS clause].
                   ⎪
                   ⎪[condition name clause
                   ⎪
                   ⎩VALUE clause.]]

                    [SET SECTION.
                    COPY clause.]
```

Figure 3: Structure of subschema DDL

The sequence of clauses shown in figure 3 must be observed, with the following exceptions:

– The sequence of PICTURE and USAGE clauses is arbitrary.

– In the RECORD SECTION, a COPY clause may also follow a record entry.

Names used in the definition of the subschema must have been defined in the schema DDL. This does not apply to group item and condition names, which are newly defined in the subschema.

4.2 IDENTIFICATION DIVISION

(cf. the "Design and Definition" manual, section 9.3.1)

```
IDENTIFICATION DIVISION.
    SUB-SCHEMA NAME IS subschema-name OF SCHEMA NAME schema-name
    [PRIVACY LOCK FOR COMPILE IS literal-1[ OR literal-2]]
    [PRIVACY KEY FOR COPY IS literal-3].
```

This entry is used to assign a name to the subschema, and

– to indicate from which schema the subschema is to be copied,

– to define passwords to prevent unauthorized compilation of a DML program with this subschema, and

– to enter, where appropriate, one of the passwords preventing unauthorized copying of a subschema from the schema.

4.3 AREA SECTION

(cf. the "Design and Definition" manual, section 9.3.2)

```
DATA DIVISION.
AREA SECTION.

    ⎧COPY ALL AREAS.          ⎫
    ⎨                         ⎬
    ⎩{COPY realm-name,...}... ⎭
```

This entry is used to copy all realms or a selection of realms from the schema into the subschema.

4.4 RECORD SECTION

(cf. the "Design and Definition" manual, section 9.3.3)

```
RECORD SECTION.

     [{COPY ALL RECORDS.                    }]
      {                                     }
      {COPY record-name-1,...}.... }

[01  record-name-2.
     {level-number record-element-name[ PICTURE IS mask-string]

                                   [ GROUP-USAGE IS NATIONAL]

                                             {DISPLAY           }
                                             {                  }
                                             {COMPUTATIONAL-3   }
                                             {                  }
                                             {COMPUTATIONAL     }
                                   [ USAGE IS {                 }]
                                             {NATIONAL          }
                                             {                  }
                                             {DATABASE-KEY      }
                                             {                  }
                                             {DATABASE-KEY-LONG }

                                   [ OCCURS integer TIMES].}...

     [88 condition-name

          {VALUE IS  }
          {          } {literal-1[ THROUGH literal-2]},... .]...]...
          {VALUES ARE}
```

The user has the choice of either completely copying all record types contained in the schema into the subschema or only a selection of records or items. In the latter case, the user specifies the record types that are to be copied completely or in part. For records to be copied in part, the user must specify all record elements that are to be copied.
It is also possible to define group items and conditions.

4.5 SET SECTION

(cf. the "Design and Definition" manual, section 9.3.4)

```
SET SECTION.

    ⎧COPY ALL SETS.          ⎫
    ⎨                        ⎬
    ⎩{COPY set-name,...,}... ⎭
```

This entry is used to copy all sets or a selection of sets from the schema into the subschema.

5 COBOL DML

5.1 Structure of a COBOL program

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
[PRIVACY.]
ENVIRONMENT DIVISION,
Usual COBOL information
DATA DIVISION.
Usual COBOL section
SUB-SCHEMA-SECTION.
DB-Entry
PROCEDURE DIVISION.
Usual COBOL information
```

The range of ANSCOBOL is extended with the following DML statements:

ACCEPT, CONNENCT, DISCONNECT, ERASE, FETCH, FIND, FINISH, FREE, GET, IF, KEEP, MODIFY, READY, SET, STORE, USE.

IDENTIFICATION DIVISION

(cf. the "Application Programming" manual, section 6.2, 7.2)

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.]
```

DATA DIVISION

(cf. the "Application Programming" manual, section 6.2, 7.3)

The data division of a COBOL program extended by DML statements contains a new section: the SUB-SCHEMA SECTION. It must be included in every program containing DML statements.

This section must be the last section of the Data Division. Its purpose is to:

– name a previously defined subschema and to

– assure that the required areas in the UWA are reserved and named.

```
DATA DIVISION.

Usual COBOL sections.

SUB-SCHEMA SECTION.

    DB subschema-name WITHIN schema-name.
```

PROCEDURE DIVISION

(cf. the "Application Programming" manual, section 6.2, 7.4)

The Procedure Division contains the COBOL and DML statements. In the DECLARATIVES section the extended USE statement can be used to respond to DML database exception conditions.

5.2 Statements

(cf. the "Application Programming" manual, section 7.4.1)

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

Table 11: COBOL DML statements (overview) (part 1 of 6)

Statement	Function
{FIND / FETCH} *record-selection-expression*[RETAINING CURRENCY FOR {MULTIPLE / [REALM][{SETS / *set-name*,...}][RECORD]}]	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 – CRR, – CRS in all sets in which it is owner or member, – CRA in the realm in which it is stored FETCH: additionally transfers the selected record into the UWA of the application program
Formats of the record selection expression Format 1: [*record-name*] DATABASE-KEY IS *item-name* [OR {PRIOR / NEXT}]	Access via the database key
Format 2: {ANY / DUPLICATE} *record-name*	Access via CALC key (hash procedure)
Format 3: DUPLICATE WITHIN {*record-name* / *set-name*} [USING *record-element-name*,...]	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)

Table 11: COBOL DML statements (overview) (part 2 of 6)

Statement	Function
Format 4: `{LAST / FIRST / NEXT / PRIOR / integer / item-name} {record-name / RECORD} [ WITHIN {set-name / realm-name}]`	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.
Format 5: `CURRENT[ record-name][ WITHIN {set-name / realm-name}]`	Access to the CRR, CRS, CRA or CRU
Format 6: `OWNER WITHIN set-name`	Access to the owner record of a CRS

Table 11: COBOL DML statements (overview) (part 3 of 6)

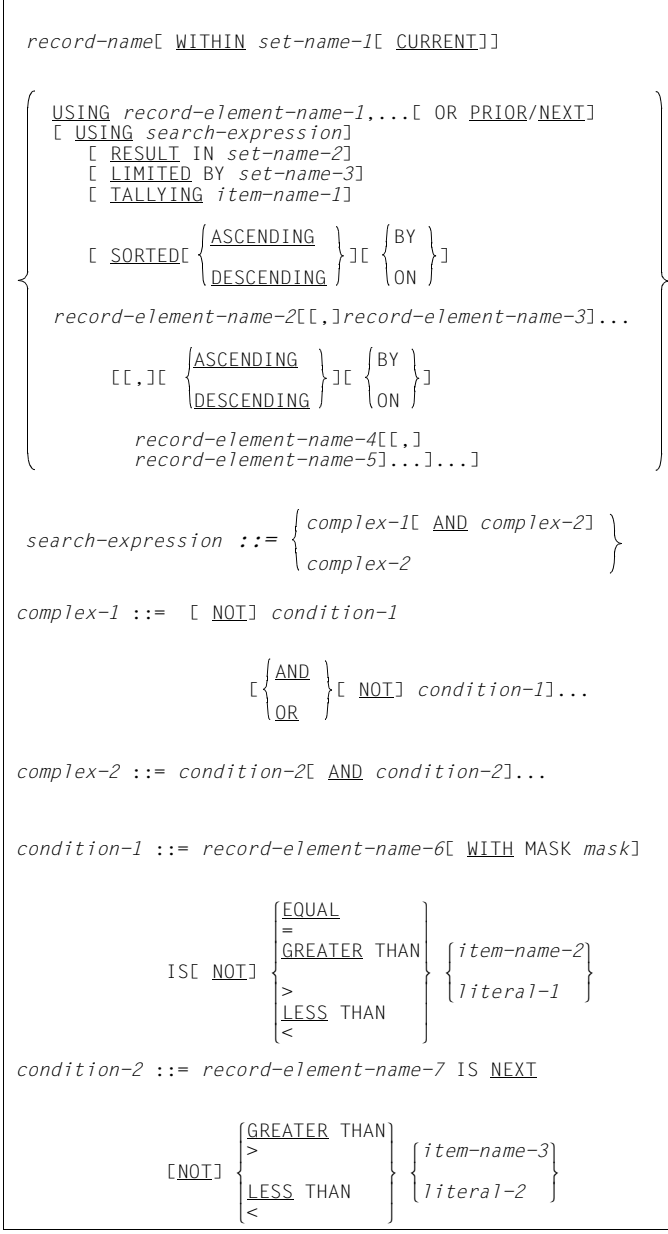
Statement	Function
Format 7:  *record-name*[WITHIN *set-name-1*[CURRENT]] { USING *record-element-name-1*,...[OR PRIOR/NEXT] [USING *search-expression*] [RESULT IN *set-name-2*] [LIMITED BY *set-name-3*] [TALLYING *item-name-1*] [SORTED[{ASCENDING | DESCENDING}][{BY | ON}] *record-element-name-2*[[,]*record-element-name-3*]... [[,][{ASCENDING | DESCENDING}][{BY | 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*] IS[NOT] {EQUAL | = | GREATER THAN | > | LESS THAN | <} {*item-name-2* | *literal-1*} *condition-2* ::= *record-element-name-7* IS NEXT [NOT] {GREATER THAN | > | LESS THAN | <} {*item-name-3* | *literal-2*}	Access to records via freely selected items or counting and buffering of records found and searches using masks searches using masks

Table 11: COBOL DML statements (overview) (part 4 of 6)

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

Table 11: COBOL DML statements (overview) (part 5 of 6)

Statement	Function
SET *item-name-1*,...TO *item-name-2*	Transfers the contents of a database key item into one or more database key items
STORE *record-name*[RETAINING CURRENCY FOR {MULTIPLE / [REALM][{SETS / *set-name*,...}][RECORD]}]	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 for each set for which the record type is defined in the schema as owner record type
USE FOR DATABASE-EXCEPTION[ON {OTHER / *literal*,...}]	Defines sequences of instructions to be executed if a DML statement is terminated with a database exception condition

Table 11: COBOL DML statements (overview) (part 6 of 6)

5.3 Application program generation

Compiling a COBOL program

(cf. the "Application Programming" manual, section 6.2)

- 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.

- 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.

Linking

(cf. the "Application Programming" manual, section 6.4.2)

```
/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=UDSLNKx) ------------------------------------------------------ (1)
//RESOLVE-BY-AUTOLINK LIBRARY=crtesyslnk
//SAVE-LLM MODULE-CONTAINER=*LIB(LIBRARY=library-2,ELEMENT=module)
//END
```

(1) UDSLNKI: independent DBH
UDSLNKL: linked-in DBH
UDSLNKA: free DBH selection

Program execution

The following DBH variants can be used:

For	**independent DBH**	**linked-in DBH**
mono-DB	x	x
multi-DB	x	x
***open*UTM**	x	-

Table 12: DBH variants

- Starting an application program with the independent DBH

  (cf. the "Application Programming" manual, section 6.4.3)

```
[/MODIFY-JOB-SWITCHES ON=28]

 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version
[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]

                                                 ⌈dbname            ⌉
 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=    {                  }
                                                 ⌊configuration-name⌋

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

 [application program parameters]
```

- Starting an application program with the linked-in DBH

  (cf. the "Application Programming" manual, section 6.4.3, and the "Database Operation" manual, section 2.3.1)

```
[/MODIFY-JOB-SWITCHES OFF=28]

 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version

                                                 (dbname             )
 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=(                       )
                                                 (konfiguration-name)

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

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

[/ADD-FILE-LINK LINK-NAME=PPFILE,FILE-NAME=DBH parameter file]

[further DBH-specific commands]

/START-EXECUTABLE-PROGRAM FROM-FILE=(LIBRARY=library-2,ELEMENT=modul)
 ,DBL-PAR=(ERROR-PROC(NAME-COLLISION=*STD))
[application program parameters]
```

6 CALL DML

6.1 Parameter definitions

(cf. the "Application Programming" manual, section 8.2)

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)

6.2 Format table

(cf. the "Application Programming" manual, section 8.2.2)

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, cf. the "Application Programming" manual, section 8.2.3.
	RET, RES, LMS, TAL SOA/SOD, NOW	12/≥2		
4. UINF	DBH communication	126		cf. the "Application Programming" manual, section 8.2.4.
5 .RECN	Record-name	8	30	Single name format
	Record-name or ␣␣␣␣␣␣␣␣	8	30	Single name format or blanks
	RECORD	8□□□□	30	RECORD␣␣/RECORD␣␣␣...
6. SETN	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 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 secondary option SOA or 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	-		cf. the "Application Programming" manual, section 8.3.2.
9. RECA[1)]	Complete record	-		Subschema format
	Selection of items without VAR option	-		Item positions in record area as subschema format
	Selection of items using VAR option	-		Record format corresponds to item name list, length = sum of item lengths

Table 13: CALL DML formats (part 1 of 2)

Parameter	Contents	Length for variant		Format
		(CALL8)	(CALL30)	
10. SPP1	RETAINING without set names	9		cf. the "Application Programming" manual, section 8.2.5.
	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 ≠ 0
	Result set name (FIND7A/FTCH7A)	8	30	Single name format
	Implicitly defined data: DB key of LOCATION MODE clause and/or realm name of WITHIN clause (STORE/STOR1L, STORE2/STOR2L, FIND2/FTCH2)	42		see the "Application Programming" manual, section 8.2.6
12. SPP3	Limited set name (FIND7A/FTCH7A)	8	30	Single name format

Table 13: CALL DML formats (part 2 of 2)

[1] 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 ≠ 000). However, if the database status begins with C, P or S, the record area remains unchanged.

6.3 User information area

(cf. the "Application Programming" manual, section 8.2.4, section 8.2.5)

Contents	Input	Output	Length	Displ.	Type
System Communication Locations - realm name - record name - set name		 X X X	 30 30 30	 0 30 60	 character character character
Result items - statement code - status code		 X X	 2 3	 90 92	 character 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: USINF*/ UINF1*

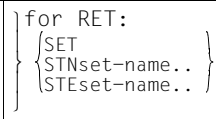
Table 14: User information area

6.4 Overview of the CALL DML functions

(cf. the "Application Programming" manual, section 8.3.1)

Information in response to invoked CALL DML functions can be found in the shaded columns.

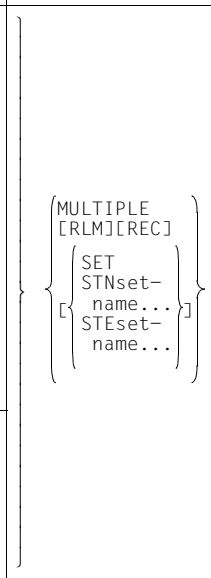
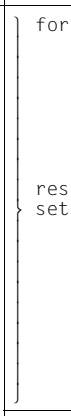
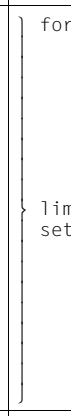
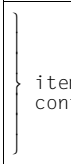
1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN
ACCPTC	DB-KEY DBKREC DBKRLM DBKSET RLMNAM RLMREC RLMSET RLMDBK		DB key DB key DB key DB key realm name realm name realm name DB key, realm name	- record name - - - record name - -	- - - set name - - set name -	- - realm name - - - - -	
ACCPTL	DB-KEY DBKREC DBKRLM DBKSET RLMNAM RLMREC RLMSET RLMDBK		DB key DB key DB key DB key realm name realm name realm name DB key, realm name	- record name - - - record name - -	- - - set name - - set name -	- - realm name - - - - -	
CONNEC	TO-ALL TO-SET	} [RET]		[record name] [record name]	- set name...		
DISCON	FRMALL FRMSET ALLFRM			[record name] [record name] -	- set name... set name...		
ERASEC	CORUNT PERMAN SELTIV ALLMEM			} record name			
FINISC	ALLRLM ALLCAN						
FREEC	ALLREC CORUNT						

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
				Transfers – The database key value of the CRR, CRS, CRA or CRU to the user information area (cf. the "Application Programming" manual, section 8.2.4). – The name of the realm in which the CRR, CRS, CRU or the record belonging to the specified database key value is stored.
				Transfers – The DATABASE-KEY-LONG value of the CRR, CRS, CRA or CRU to the user information area (cf. the "Application Programming" manual, section 8.2.4). – The name of the realm in which the CRR, CRS, CRU or the record belonging to the specified database key value is stored.
				Inserts the CRU in set occurrences
				– Removes the CRU from set occurrences – Removes all records from dynamic sets
				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] [DBKNXT]	[RET] [NOW]	DB key	[record name]			
FIND1L	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB key	[record name]			
FIND2	ANYREC ANYIMP DUPLIC	[RET] [NOW]		record name			
FIND3	SETNAM SETITM RECNAM RECITM	[RET] [NOW]		– record name	set name set name – –		– item name... – item name...
FIND4	SETNXT SETPRI SETFST SETLST SETSPC	[RET] [NOW]		[{record name / RECORD}]	set name		
	RLMNXT RLMPRI RLMFST RLMLST RLMSPC	[RET] [NOW]		[{record name / RECORD}]		realm name	
	RECNXT RECPRI RECFST RECLST RECSPC	[RET] [NOW]		record name			
FIND5	CORUNT RECNAM RECSET SETNAM RECRLM RLMNAM	[RET] [NOW]		– record name record name – record name –	– – set name set name – –	– – – – realm name realm name	
FIND6		[RET] [NOW]			set name		

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
	for RET:			Access via a database key value of type DATABASE-KEY
				Access via a database key value of type DATABASE-KEY-LONG
item contents item contents –		– impl. def. data area –		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).
	MULTIPLE [RLM][REC] [SET STNset-name... STEset-name...]	– – – – pos. integer		Access to the last or first record, to the record prior or next to the CRR, CRS or CRA or to a record whose position corresponds to a specified numeric value (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.
		– – – – pos. integer		
		– – – – 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 SELFST CURFST	[RET] [NOW] [RES] [LMS] [TAL] [SOA / SOD]	for TAL: number of records	record name	– set name set name	item name... item name... item name...	
	RECSEX CURSEX	[RET] [NOW] [RES] [LMS] [TAL] [SOA / SOD]		record name	– set name set name	item name... item name... item name...	search expression
	RECITM RECITN SELITM SELITN SELITP CURITM CURITN CURITP	[RET] [NOW]		record name	– set-name set-name		item-name...

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
				Access to records via any items; counting and storing found records intermediately if required; search using mask
				

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN
FTCH1	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB key	[record name]			
FTCH1L	[DBKPRI] [DBKNXT]	[RET] [NOW]	DB key	[record name]			
FTCH2	ANYREC ANYIMP DUPLIC	[RET] [NOW]		record name			
FTCH3	SETNAM	[RET] [NOW]		record name	set name		–
	SETITM				set name		item name...
	RECNAM				–		item name...
	RECITM				–		
FTCH4	SETNXT SETPRI SETFST SETLST SETSPC	[RET] [NOW]		[{record name | RECORD}]	set name		
	RLMNXT RLMPRI RLMFST RLMLST RLMSPC	[RET] [NOW]		[{record name | RECORD}]		realm name	
	RECNXT RECPRI RECFST RECLST RECSPC	[RET] [NOW]		record name			
FTCH5	CORUNT	[RET] [NOW]		–	–	–	
	RECNAM			record name	–	–	
	RECSET			record name	set name	–	
	SETNAM			–	set name	–	
	RECRLM			record name	–	realm name	
	RLMNAM			–	–	realm name	
FTCH6		[RET] [NOW]			set name		

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
rec. cont´s				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 rec. cont´s item cont´s rec. cont´s rec. cont´s		– impl. def. data area –		Access via CALC key (hashing)
record contents	MULTIPLE [RLM][REC] [SET STNset- name... STEset- name...]			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)
		– – – – 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. The collection of records can be a record type, a set occurrence, a realm or the intersection of a record type and a realm.
		– – – – pos. integer		
		– – – – 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	[RET] [NOW] [RES]	for TAL:		–	item name...	
	SELFST	[LMS] [TAL]		record name	set name	item name...	
	CURFST	[SOA|SOD]			set name	item name...	
	RECSEX	[RET] [NOW] [RES]	number of records		–	item name...	
		[LMS] [TAL]		record name	set name	item name...	search expression
	CURSEX	[SOA|SOD]			set name	item name...	
	RECITM RECITN SELITM SELITN SELITP CURITM CURITN CURITP	[RET] [NOW]		record name	– set name set name		item name-1...

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
record contents	MULTIPLE [RLM][REC] [SET STNset-name... STEset-name...]	for RES: result set name	for LMS: limited set name	Access to records via any items; counting and storing records found intermediately if required; search with mask
contents/ record contents				

1. FCOD	2. FOPT	3. SOPT	4. UINF	5. RECN	6. SETN	7. RLMN	8. ITMN
GETC	CORUNT ITMNAM	– [VAR]		[record name] record name			– item name-1...
IFC	OWNALL OWNSET MEMALL MEMSET TENALL TENSET EMPTYS				– set name – set name – set name set name		
KEEPC							
MODIF1	CORUNT INCALL ONLALL INCSET ONLSET	[RET]		record name	– – – set name... set name...		
MODIF2	CORUNT INCALL INCSET	[RET] [VAR]		record name	– – set name		item name...
READYC	ALLRTR ALLUPD ALLPRT ALLPUP ALLERT ALLEUP RLMRTR RLMUPD RLMPRT RLMPUP RLMERT RLMEUP	[NOW]				– – – – – – realm name...	

9. RECA	10. SPP1	11. SPP2	12. SPP3	Function
rec cont´s 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 rec. cont´s – rec. cont´s –	for RET: SET STNset name.. STEset name..			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 iforma– tion		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 IMPDAT	[RET]		rec-name				record contents
STOR1L	RECNAM IMPDAT	[RET]		rec-name				record contents
STORE2	ITMNAM IMPDAT	[RET] [VAR]		rec-name			item name...	item contents
STOR2L	ITMNAM IMPDAT	[RET] [VAR]		rec-name			item name...	item contents

10. SPP1	11. SPP2	12. SPP3	Function
for RET: MULTIPLE [RLM][REC] { SET / STNset-name... / STEset-name... } [ ]	– impl. def. data area		– Transfers a record or individual items or compressed records from the UWA to the database as a new record – Inserts the new record into all sets for which its record type has been defined as an AUTOMATIC member in the schema – Sets up a new set occurrence for each set for which the record type is defined as owner record type in the schema If you want to specify a database with a REC-REF > 254 and/or an RSQ > 2^{24}-1 with an IMPDAT function option, you must use STOR1L or STOR2L
	– impl. def. data area –		

6.5 LOOKC

(cf. the "Application Programming" manual, section 8.5.1)

General description of all LOOKC calls

Meaning	Contents	Length	Displ.	Type
Name of the element		30[1]	0	character
Reference 1		2	30	binary
Reference 2		2	32	binary
Data type – no type specified – realm – record type – set – item – key	(Byte) 00 01 02 03 04 05	1	34	binary
Name ambiguity – name unique – name not unique	(Bit) 00 40	1	35	binary
Result – o.k. – not found – no further element found	(Bit) 00 01/02/08 04	1	36	binary
Spare		1	37	

Table 15: General description in the LOOKC block

[1] 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.

Special description of a realm in the LOOKC block

Meaning	Contents	Length	Displ.	Type
Filler		4	0	binary
Realm status – not temporary – temporary	(Bit) not **equal 80** 80	1	4	binary
Spare		13	5	

Table 16: 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 (displacement within the COBOL BIB)		4	0	binary
Length of a record type		2	4	binary
LOCATION MODE – DIRECT – CALC DUPLICATES – CALC NO DUPLICATES – no LOCATION MODE	(Byte) 00 01 02 03	1	6	binary
Details[1] – rec. type, compressed – record type with SEARCH key – record type in different realms	(Bit) 80 20 10	1	7	binary
Short entry for position in record area (displacement within the COBOL BIB)		2	8	binary
Spare		8	10	

Table 17: Special description of a record type in the LOOKC block

[1] 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 – SORTED – FIRST – LAST – NEXT – PRIOR – SORTED INDEXED	(Byte) 00 11 22 44 78 80	1	6	binary
CONNECT type – AUTOMATIC – MANUAL	(Byte) 00 01	1	7	binary
DISCONNECT type – MANDATORY – OPTIONAL	(Byte) 00 01	1	8	binary
SET SELECTION – THRU OWNER – THRU CURRENT	(Byte) 00 01	1	9	binary
Special types[1] – singular set – DYNAMIC SET – implicit set	(Bit) 80 40 20	1	10	binary
Spare		7	11	

Table 18: Special description of a set in the LOOKC block

[1] 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: set reference (long) to the set belonging to the key referenced with displacement 14		2	2	binary

Table 19: Special description of an item in the LOOKC block (part 1 of 2)

Meaning	Contents	Length	Displ.	Type
Level number		1	4	binary
Next level number		1	5	binary
Item type – database key – packed decimal – binary – character – signed unpacked decimal – unsigned unpacked decimal – group item – national – national data group	(Byte) 00 01 02 04 05 06 0F 14 1F	1	6	binary
Scale modifier Bit 0 = sign Bit 1-7 = value		1	7	binary
Item length		2	8	binary
Number of occurrences		2	10	binary
LOCATION MODE IS CALC indicator – simple key – compound key – no key	(Bit) 40 20 00	1	12	binary
SEARCH KEY...USING CALC /SEARCH KEY...USING INDEX- / sort key indicator[1] – no key – simple key – compound key – item used in more than one key – repeating group item – item is a group item that forms a compound key	(Bit) 80 40 20 10 01 08	1	13	binary
Connection to key item set: – reference to the first key in which the item is a key item – set reference (short) to the set belonging to this key		 2 1	 14 16	 binary binary
Additional information – number of digits is even (only relevant for item type = packed decimal)	(Bit) 80	1	17	binary

Table 19: Special description of an item in the LOOKC block (part 2 of 2)

[1] 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[1] – DUPLICATES NOT ALLOWED – DUPLICATES ALLOWED – DESCENDING – ASCENDING – Index table for key – implicit set – explicit se	(Bit) 80 00 40 00 20 10 00	1	5	binary
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 20: Special description of a key in the LOOKC block

[1] Combinations are possible

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 1) 3)	O 2)
first name	FSTNAM	()	O	O 1)	O 2)
next name	NXTNAM	()	I,O	O 1)	O 2)
all names, starting with the one specified	ALLNAM	(SPC)	I,O	I,O 1) 3)	O 2)
all names, starting with the first one	ALLNAM	(FST)	O	O 1)	O 2)
all names; next part of answer	ALLNAM	(NXA)	O	O 1)	O 2)
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 1) 3)	O 2)
FROM-TO names; next part of answer	FRTNAM	(NXA)	O	O 1)	O 2)
list of specified names	LISNAM	()	I	I,O 1) 3)	O 2)

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 parameter 1	Special parameter 2
				Special description (18 bytes)		
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; 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	(RECNXA)	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	(RECNXA)	O	O (SR)	
list of specified sets	LISSET	( )	O	I (SR)	

					Special parameter 1	Special parameter 2
				Special description (18 bytes)		
data type	duplicate names	result	spare		number of LOOKC blocks	record reference
binary	binary	binary	binary		binary	binary
1	1	1	1		2	4

O		O		O		
O		O		O		
O		O		O		
O		O		O		
O		O		O		
O		O		O	I	

		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O		
		O		O	I	
		O		O	I	
		O		O	I	
		O		O	I	
		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 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

O		O	O	O	I,O[1]	O		
O		O	O	O	O	O		
O		O	O	O	I,O	O		
O		O	O	O	O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	
O		O	O	O	O	O	I	
O		O	O	O	I,O	O	I	

[1] 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 parameter 1	Special parameter 2
				Special description (18 bytes)						
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	

6.6 CALL DML assembler macros

(cf. the "Application Programming", section 8.4)

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)	– where necessary complete implicit parameter list – execute CALL DML call with the implicit parameter list	CALL DML call with implicit parameter list and convenient parameter mechanism Alternative: DSCAP + DSCDF
DSCAP	-	– accept explicitly defined parameter list – where necessary fill out explicitly defined parameter list – execute CALL DM call with explicit parameter list	CALL DML call with explicit parameter list and 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 21: CALL DML Assembler macros

```
[name] DSCAL fcod,fopt,...,spp3

[name] DSCAP [fcod,fopt,...,spp3][,PARAM=param]

[name] DSCDF fcod,fopt,...,spp3[,SUFFIX=x]

[name] DSCPA [option]
```

6.7 Application program generation

Compiling the subschema with BCALLSI

(cf. the "Creation and Restructuring" manual, section 3.5)

Compiling an application program

CALL DML application in COBOL:

(cf. section “Application program generation” on page 53)

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

CALL DML application in other programming languages:

For information on compiling the application please refer to the examples in the relevant compiler manual.

Linking

CALL DML application in COBOL:

(cf.section “Application program generation” on page 53)

CALL DML application in other programming languages:

As with linking COBOL applications. For information on linking the language-specific runtime systems please refer to the examples in the relevant compiler manual.

Program execution

- Starting an application program with the independent DBH

  (cf. the "Application Programming" manual, section 6.4.3)

```
[/MODIFY-JOB-SWITCHES ON=28]

 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version
[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D,VERSION=version]

                                                 ⌈dbname            ⌉
 /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=    {                  }
                                                 ⌊configuration-name⌋

 /ADD-FILE-LINK LINK-NAME=$UDSSSI,FILE-NAME=SSITAB-library-1
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=SSITAB-library-nn]

[/ADD-FILE-LINK LINK-NAME=$UDSPLEX,FILE-NAME=PLITAB-library-1]
[/ADD-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=PLITAB-library-nn]

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

 [application program parameters]
```

7 COBOL and CALL DML statement codes and status codes

7.1 Statement codes

(cf. the "Application Programming" manual, section 6.6.2)

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.

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

Table 22: Statement codes and associated functions

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

7.2 Meaning of the status codes

(cf. the "Application Programming" manual, section 6.6.2)

The following table 23 shows the correlation between UDS/SQL behavior and status codes:

	Status codes					
	000	**001**	**018, 113, 122, 218**	**≠ 000 READY**	**200**	**any other**
DML statement successfully completed	yes	yes	no	no	so far	no
Transaction aborted by UDS/SQL	no[1)]	no[1)]	yes	yes	no	no[1)]
Contents of record area UWA ...	corresponding to statement	the next record	undefined because transaction has been aborted		un-changed	un-changed

Table 23: Meaning of the status codes

[1] Unless FINISH WITH CANCEL is executed.

7.3 Combinations of statement codes and status codes

(cf. the "Application Programming" manual, section 6.6.2)

	Statement code															
Status code	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
001 010													X				
011 012 013 018 020		X	X	X	see table 25 on page 100		X	X	X	X	X	X	X X X X X	X	X		X
021 022 023 024 027 028 029		X						X			X X	X	X	X X	X		
031 032 033		X X X	X X	X X				X X	X X		X X X			X			
042 043 044		X X	X	X					X		X X	X	X	X X	X		
051		X									X			X			
071 072				X													
081 082 083		X	X X								X X						
091 092 093 099		X X	X X	X X					X		X X	X X X	X X X	X X	X		
101 102 103		X	X	X		X		X	X		X	X	X	X	X	X	
113	X	X	X	X		X		X	X		X	X	X	X	X		X

Table 24: Combinations of statement codes and status codes (part 1 of 3)

	Statement code															
Status code	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
122	X	X	X	X	see table 25 on page 100	X	X	X	X	X	X	X	X	X	X		X
123												X	X				
124												X	X				
131	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				
141												X	X				
142												X	X				
144	X	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 24: Combinations of statement codes and status codes (part 2 of 3)

	Statement code															
Status code	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**12**	**13**	**14**	**15**	**16**	**25**
781					see table 25 on page 100								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				
954												X	X				

Table 24: 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						
04...	**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 25: Combinations of statement code 04 and status codes (part 1 of 2)

Status code	Formats of the FIND/FETCH record selection expression						
04...	**1**	**2**	**3**	**4**	**5**	**6**	**7**
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 25: Combinations of statement code 04 and status codes (part 2 of 2)

7.4 DML status codes

(cf. the "Application Programming" manual, section 10.1)

Status code giving information

001 FIND/FETCHformat 1 or 7 with OR PRIOR/OR NEXT specification:
No record has been found which matches the values given. The next record following in the sort sequence has been 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 O18 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 DRS can be found.
FIND/FETCH format 3 (without USING):
The end of the hits 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 deliver 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 sequence number or one 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 schema modification, the application program was not recompiled (COBOL DML) or the BCALLSI run was omitted (CALL DML), or

an error occurred at 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, 4 and 5:
The entry point of the DML statement (CRR, CRA or CRS) has been deleted or removed 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 specified with an ERASE PERMANENT/SELECTIVE/ALL was not opened with EXCLUSIVE UPDATE, or
in conjunction with the P parameter PP TA-ACCESS=SHARED an attempt has been made to open a processing chain in 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)
While opening a transaction, a realm is locked by a different 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 for a search in a CHAIN that is not chained backward

FIND/FETCH format 6:
setname 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 *setname-1* USING *itemname-1*,..." was specified.
  Specification of a dynamic set in *setname-1* is not permitted.
- LIMITED BY *dynamic-set* ... SORTED BY ... was specified.
  It is not possible to sort the intersection of a hit list and a dynamic set.
- LIMITED BY *sorted-dynamic-set* ... was specified.
  It is not possible to obtain the intersection of a hit list 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^{24}-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 at 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 When accessing a database page a serious error was discovered in the Database Handler or in the database.

Status codes relating to UDS/SQL resources:

122 The transaction was 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 in a deadlock which has been resolved 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 (by means of a UDS/SQL message).
- For UDS-D:
  Transaction rollback can also be due to errors or administrator intervention in a remote configuration or due to errors (e.g. ABORT, CLOSE CALLS, CLOSE RUN-UNITS, %TERM) 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 shadow 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 prematurely terminated with CANCEL by the DBH.

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 (subschema reference).

145 The subschema specified in the READY statement cannot be processed because it does not match the current status of the schema (subschema 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 received (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 SET specified is not a distributable list.

166 The realm specified is not permitted for the record type.

167 Conflicting 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 on 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 has been 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 one 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 The specified user group is not known (see the manual "Creation and Restructuring", BPRIVACY).

954 No access authorization has been defined for the specified user group.

7.5 CALL DML status codes

(cf. the "Application Programming" manual, section 10.1)

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.

8 DMLTEST

8.1 Testing DML functions

(cf. the "Application Programming" manual, chapter 9)

The DMLTEST program enables the user to

– test individual DML functions interactively
– run test procedures
– access any database configuration
– interoperate with KDBS.

8.2 Keyword parameters

The following keyword parameters are used in the formats of the DMLTEST commands:

Keyword parameter	Default value
value: [VAL=]{ *literal* | [*n1*]{C | X | P}[L*n2*]'*literal*' | '*literal*' }	-
distance: [DIST=]{ *n* | X'*n*' }]	D=0
length: [LNG=]{ *n* | X'*n*' }]	L=8
name: [NAME=]*literal* maximum length : 20 allowed characters: 1st position A-Z from 2nd position A-Z,-,0-9	-
filename: [{ OML= | FILE= }]*literal*	OML= DMLTEST.MODLIB for PERFORM
repetition: [{ REP= | STEP= }]{ *n* | X'*n*' }	4 for TRACE 1 in all other cases
form: [FORM=] {C | X | D}	F=C
condition: [{ CASE= | COND= }]{ RECORD{ (*n*) | (X'*n*') } | TIME | RCODE | *definition* }{ EQ | NE | LT | GT | LE | GE }*value*	-

Table 26: 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

8.3 Keywords

The *parameter* variable in the DMLTEST commands can be replaced by the following values (see also page 58):

Position in the CALL parameter list	Parameter keyword	
	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[2]
12	SPP3	–
	SEX[1]	–
	SUBS	–

Table 27: Values for CDML and KDBS

[1] Redefine ITMN: This parameter must be used if the user requires a search expression in DMLTEST syntax to be edited into CALL DML syntax

[2] Redefine SI: This parameter must be used if the user requires a search expression in DMLTEST syntax to be edited into KDBS syntax

Any number of commands may be written in a sequence. The separator is ";".

8.4 Overview of the DMLTEST commands

Command	Function
ADD *name*,*value*[,*condition*]	Add
CONTINUE	Resume processing after interrupt
DBH {INDEPENDENT | INLINKED}	Select DBH variant Default value: INLINKED
{DECLARE | DCL} *name*[,*length*]	Define an item in the work area
DEFINE {RCODE | RECORD | *parameter*},*name*[,*distance*][,*length*]	Define an item in the specified area
DELETE *name*[,*condition*]	Delete a procedure or an item
DISPLAY {RECA | RECORD | RCODE | TIME}[,*distance*][,*length*][,*form*][,*condition*]	Output the specified area or value to SYSOUT after each DML statement
{DISPOFF | DOFF}[{RECA | RECORD | RCODE | TIME}][,*condition*]	Deactivate DISPLAY function
DO *name*[,*repetition*][,*condition*]	Start a procedure
EDT	Call EDT
END	Terminate procedure definition
ESCAPE[*condition*]	Terminate interrupt, or abort procedure
EXECUTE[*repetition*][,*condition*]	Execute DML statement
{HALT | STOP}[*condition*]	Terminate the DMLTEST program
HELP[*condition*]	Request information on preceding command, or request preceding DML statement

Table 28: Overview of DMLTEST commands (part 1 of 3)

Command	Function
LANGUAGE { CDML | CDML30 | COBOL | COBOL30 | KDBS | KKDS | KLDS }	Select data manipulation language Default value: COBOL
LEAVE[*condition*]	Abort a procedure call
{ LIST | LS } { { { CMD | DCL | } | DEF | PROC } [,*name*] | *command-name* | *declaration* | *definition* | *procname* }	Output specified information to SYSOUT
MOVE { RECORD | RCODE | *parameter* | *definition* | *declaration* },*value*[,*distance*][,*condition*]	Enter values in the specified areas
NEXT	Respond to interrupts, or abort current command
PERFORM *name*[,*filename*][,*condition*]	Call a module
PRINT { RECORD | RCODE | TALLY | TIME } [,*distance*][,*length*][,*form*][,*condition*]	Output specified area or value to SYSLST after each DML statement
PROC *procname*	Open a procedure definition
{ PROFF | POFF } [{ RECORD | RCODE | TIME }][,*condition*]	Deactivate PRINT function
PROT[{ ON | OFF | OUT }][,*condition*]	Activate or deactivate logging function Default value: PROT ON
{ REMARK | * } *literal*	Insert comment lines
RUN *filename*[,*repetition*][,*condition*]	Start a sequence of commands or statements stored in an ISAM file.

Table 28: Overview of DMLTEST commands (part 2 of 3)

Command	Function
SET[*parameter*[(n | X'n')]=]*value*,...	Enter values in the CALL DML parameter list
SHOW { RCODE | TALLY | { RECORD | PARAM[,*name*] } | { DCL | DEF | PROC },*name* [,*distance*] [,*length*] [,*form*] | *parameter* | *declaration* | *definition* | *procname* } [,*condition*]	Output the specified area to SYSOUT in the specified format
SUBSCHEMA IS *subschema*	Select subschema
SYSTEM[*condition*]	Go to system mode
TRACE[{ ON | OFF }][,*repetition*][,*condition*]	Log commands and statements on the screen during processing Default value: TRACE ON
WAIT[*condition*]	Effect an interrupt

Table 28: Overview of DMLTEST commands (part 3 of 3)

8.5 Overview of the differences between DMLTEST DML and COBOL DML statements

(cf. the "Application Programming" manual, section 9.3.1)

Statement		COBOL DML	DMLTEST DML
ACCEPT	format 1:	*item-name-1*	DB-KEY, DB-KEY-LONG
	format 2:	*item-name-2* *item-name-3*	AREA-ID DB-KEY, DB-KEY-LONG
FIND/FETCH	format 1:	*item-name* OR PRIOR/NEXT	DB-KEY, DB-KEY-LONG 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: ...*record-name*[IMP]
	format 3:		USING can be extended by ...[{IN | OF} *record-name*]
	format 4:	*item-name*	–
	format 7:	*item-name-1* OR PRIOR/NEXT	TALLY 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 ...[{IN | OF} *record-name*]
IF	format 1: format 2:		NEXT SENTENCE, ELSE and NOT must not be used.
MODIFY			This statement is extended by ... {IN | OF} *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: ...*record-name*[IMP]...or ...*record-name*[IMP-LONG]...

Table 29: Differences between COBOL DML and DMLTEST DML statements

9 Compiling the Schema DDL, SSL and Subschema DDL

9.1 Command sequence for compiling the Schema DDL

(cf. the "Creation and Restructuring" manual, section 3.2.2)

```
01  /CREATE-FILE FILE-NAME=dbname.COSSD ...
02  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04  /CREATE-FILE FILE-NAME=dbname.DBSTAT,SUPPRESS-ERRORS=*FILE-EXISTING
    /CREATE-FILE FILE-NAME=dbname.DBSTAT.SAVE,SUPPRESS-ERRORS=*FILE-EXISTING
05  /START-UDS-DDL
06   ddl-compiler-statements
07   END
```

06 The individual statements can be entered in one line if they are separated by commas or blanks.

9.2 Command sequence for compiling the SSL

(cf. the "Creation and Restructuring" manual, section 3.2.3)

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-SSL
04   ssl-compiler-statements
05   END
```

04 The individual statements can be entered in one line if they are separated by commas or blanks.

9.3 Command sequence for compiling the Subschema DDL

(cf. the "Creation and Restructuring" manual, section 3.4.1)

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-DDL
04  sddl-compiler-statements
05  END
```

04 The individual statements can be entered in one line if they are separated by commas or blanks.

9.4 DDL compiler statements/SSL compiler statements

(cf. the "Creation and Restructuring" manual, section 3.2.2)

Statement	Compiler	Default value	Meaning
PARLIST IS {YES / NO}	DDL SDDL SSL	NO	optional; YES all statements are listed on SYSLST NO statements are not listed
SORCLIST IS {YES / NO}	DDL SDDL SSL	YES	optional; YES a listing is printed out on SYSLST, possibly containing error messages NO no listing is printed
SOURCE IS {'*file-name*' / '*lib*(*element*)'}	DDL SDDL SSL	-	not required for inputs in interactive mode or if SYSDTA is assigned as the input file - in this case, it should be noted that all the statements, (at least END) must be entered first followed by the actual source. assigns the compiler the file containing the Schema DDL/Subschema DDL/SSL. Instead of '*file-name*' it is also possible to specify an element of a program library (see "Program libraries" in the "LMS (BS2000)" manual). *lib*: name of program library *element*: name of element SYSDTA is switched to the input file. It is reset to SYSCMD upon completion of the compiler run. The statements “SOURCE IS“ and “DELETE SCHEMA“ or “DELETE SUBSCHEMA“ may not be used within the same DDL compiler run.

Table 30: Compiler statements for the Schema DDL/Subschema DDL/SSL (part 1 of 4)

Statement	Com-piler	De-fault value	Meaning
SUBSCHEMA FORM IS OLD	SDDL	-	optional; this statement is only required for subschemas which are used by KDBS applications; it is permissible only in conjunction with the "SOURCE IS *filename*" statement and is ignored when compiling schemas. The "SUBSCHEMA FORM IS OLD" statement causes the transformed subschema and the check table (CHECK-TABLE) to be entered in the COSSD in an internal format which was the standard format up to and including UDS/SQL V1.2 ("old" form; all reference numbers are 1 byte long). A subschema can be compiled to a format compatible with UDS/SQL V1.2 only if the following conditions are satisfied: – No item of the subschema is of type DATABASE-KEY-LONG. – No item of the subschema is of the type NATIONAL. – No record type of the subschema is longer than 2020 bytes. – All record references and set numbers of the schema are ≤ 254. Otherwise, the DDL compiler aborts with syntax errors, and the subschema is not entered in the DBCOM and COSSD.
DELETE SCHEMA '*schema-name*'	DDL	-	optional; deletes the specified schema; useful after restructuring with BALTER if the DDL executes correctly and the SSL compilation reports errors actually attributable to the DDL *schema-name*: name of schema The "SOURCE IS" and "DELETE SCHEMA" statements must not be used within the same DDL compiler run.

Table 30: Compiler statements for the Schema DDL/Subschema DDL/SSL (part 2 of 4)

Statement	Compiler	Default value	Meaning
DELETE [ONLY]SUBSCHEMA '*subschema-name*' { OF / : } SCHEMA '*schema-name*'	SDDL	-	optional; deletes the specified subschema. The subschema being compiled may have the same name as the subschema named in the DELETE statement, as it is deleted before the compiler run. ONLY if the parameter is omitted, a SOURCE statement must follow the DELETE statement. If the parameter is specified, any SOURCE statement is ignored. *subschema-name*: name of subschema *schema-name*: name of schema Both names have to be given in single quotes
DISPLAY IS { YES / NO }	DDL SDDL SSL	NO	optional; YES information held in DBCOM relating to record types, sets, etc. is output in unencoded form. NO values in DBCOM are not output
CREATE COSSD '*schema-name*'	DDL SDDL	-	Retroactive creation of COSSD. If this was forgotten during schema compilation or the DDL compiler was terminated abnormally owing to an error when the COSSD was being configured, this can be carried out in a separate run before the first subschema is compiled; *schema-name*: has to be given in single quotes. The COSSD has to be created with a CREATE-FILE command prior to the compiler run. N.B.: If the SOURCE IS ... parameter is specified at the same time, compilation will be suppressed.

Table 30: Compiler statements for the Schema DDL/Subschema DDL/SSL (part 3 of 4)

Statement	Compiler	Default value	Meaning
COMPARE SUBSCHEMAS	SDDL	-	admissible only after restructuring with BALTER. The subschemas of the old schema are checked for compatibility with the new schema; for this purpose the DDL compiler reads the subschemas from the old COSSD after the BALTER run. If an old subschema is compatible with the new schema, it is entered in the new DBCOM and in the new COSSD.
DIAGNOSTIC IS { YES / NO }	SDDL	NO	only meaningful in conjunction with COMPARE YES diagnoses incompatibilities between old subschemas and the new schema and lists them in the form of error messages NO no error messages are output
QUOTE IS { SINGLE / DOUBLE }	DDL SDDL	DOUBLE	either; SINGLE literals in the Schema DDL/ Subschema DDL are given in single quotes DOUBLE literals in the Schema DDL/ Subschema DDL are given in double quotes
END	DDL SDDL SSL	-	mandatory; terminates statement input

Table 30: Compiler statements for the Schema DDL/Subschema DDL/SSL (part 4 of 4)

10 Database operation

10.1 Database operation using the independent DBH

(cf. the "Database Operation" manual, chapter 3)

Load parameters

Parameter	Default value	Meaning
PP 2KB-BUFFER-SIZE=*n*	1	Defines size of the 2-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 4KB-BUFFER-SIZE=*n*	1	Defines size of the 4-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 8KB-BUFFER-SIZE=*n*	0	Defines size of the 8-Kbyte system buffer pool in Mbytes; *n*=0 or *n*=3..2047
PP ADM={REMOTE / LOCAL}	REMOTE	REMOTE: Administration from any terminal via DCAM LOCAL: The master task reserves the terminal permanently. Administration is possible only via the database administrator´s terminal or the console.
PP ADMPASS=*admpassword*	-	Defines password for administration via DCAM ENTRY: not permitted
PP BCAM-PREFIX=*prefix*	SUD$	Defines a prefix for names of user tasks which execute on virtual hosts

Table 31: Load parameters of the independent DBH (part 1 of 6)

Parameter	Default value	Meaning
PP CATPASS={STANDARD / *password*}	STD	Defines password for files created by the DBH, such as ALOG files and temporary realms
PP CHCKTIME=*n*	60	Specifies period in seconds for connection and transaction checking by the UDS-D task *n*=60..900
PP CP-SIZE=*n*	1024	Specifies minimum size in Kbytes of common pool; *n*=1..16384
PP CPU={MONO-PROCESSOR / MULTI-PROCESSOR}	MONO-PROCESSOR	Specifies processor type used
PP CUP-SIZE=*n*	1024[1] 128 [2]	Specifies minimum size in Kbytes of communication pool; *n*=1..16384
PP DBNAME=[$*userid.*]*dbname* [.*copy-name*] [,SHARED-RETRIEVAL] [,{n / [*n*],*bufferid*}]	-	Names databases in DB configuration; max. 222 databases *n*: defines size of user buffer pools in Mbytes; *n*=0..2047 *bufferid:* buffer pool identifier
PP DEACT={YES / NO}	YES	Enables deactivation of UDS/SQL tasks due to computer workload
PP DEADTIME=*n*	60	Time in seconds to resolve interconfiguration deadlocks or deadlocks involving *open*UTM *n*=5..900
PP DIP-SIZE=n	1024[1] 64[2]	Specifies minimum size in Kbytes of distribution pool; n=1..16384
PP DISDB=*n*	1	Maximum number of remote databases that can be accessed per transaction; *n*=1..32
PP DISTABLE=[:*catid*:][$*userid.*] *file-name*	-	Specifies input file for creating distribution table

Table 31: Load parameters of the independent DBH (part 2 of 6)

Parameter	Default value	Meaning
PP DISTRIBUTION={NO | STANDBY | START}	NO	Controls participation in UDS-D operation NO UDS-D operation not possible STANDBY UDS-D operation is being prepared and can be started later with &START DISTRIBUTION START UDS-D operation is started
PP DUMP={STD | ALL}	ALL	Determines the scope of a dump
PP END	-	Terminates input of load parameter
PP IO={ASYNC | SYNC}	ASYNC	Executes I/Os in server tasks asynchronously or synchronously
PP LOCK={STD | SHARED | EXCLUSIVE}	STD	Defines locking protocol
PP LOG={NO | :*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2*] [,*priv-vsn-3/device-3*]]) | (*vsn-1*[,*vsn-2* [,*vsn-3*]])}	-	Maintains the RLOG file
PP LOG-2={:*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* | [,*priv-vsn-3/device-3*]]) | (*vsn-1*[,*vsn-2* [,*vsn-3*]])}	-	Maintains the duplicate RLOG file
PP LOG-SIZE=([*primary*] [,[*secondary*]])	192,192	Specifies amount of storage (number of PAM pages) in RLOG files
PP MAXDB=*n*	Sum of PP DBNAME	Specifies maximum number of databases in DB configuration; *n*=1..122

Table 31: Load parameters of the independent DBH (part 3 of 6)

Parameter	Default value	Meaning
PP MPSEG={STD / 64K}	STD	Specifies segment size
PP ORDER-DBSTATUS={STD / SPECIAL}	STD	Controls system behavior if new update transactions or processing chains collide with the execution of pending requests
PP PARLIST={YES / NO}	NO	Lists parameters used
PP PASSWORD={NONE / STANDARD / *password* / (*password*, *password*,...)}	STD	Defines password that the DBH must use along with PP CATPASS in order to open files
PP PRIVACY-CHECK={STD / NO-KSET / OFF}	STD	Controls handling of privacy checks
PP PTCSYNCH= ([{WAIT / ABORT / COMMIT}][,[{WAIT / ABORT / COMMIT}]])	WAIT,WAIT	Controls handling of transactions in PTC state; the first value applies to warm starts, the second to the current session if the state of the primary subtransaction cannot be ascertained
PP RESERVE={NONE / :*catid*: / PUBLIC / (*priv-vsn-1*/*device-1* [,*priv-vsn-2*/*device-2* [,*priv-vsn-3*/*device-3*]]) / (*vsn-1*[,*vsn-2* [,*vsn-3*]])}	NONE	Specifies reserve volumes for RLOG files
PP RESULT-DELAY=*n*	0	Groups together request results for user tasks; *n*=1..*m* *m*=PP TRANSACTION
PP SCHEDULING={SYMMETRIC / ASYMMETRIC}	SYMMETRIC	Controls optimization of communications between the user task and server task and the processing of pending DML jobs in the server task

Table 31: Load parameters of the independent DBH (part 4 of 6)

Parameter	Default value	Meaning
PP SERVERTASK=*n*	1	Defines number of server tasks under independent DBH *n*=1..30
PP SIP-SIZE=*n*	1024[1] 128[2]	Specifies size of the SSITAB pool in Kbytes ; *n*=1..16384
PP SQL=*n*	4	Defines maximum number of simultaneously active SQL conversations; *n*=0...9999
PP SQL-LIMIT=*n*	10	Sets minimum time for which UDS/SQL maintains the conversation-specific data for inactive conversations; *n*=5..999 minutes
PP STDCKPT={YES / NO}	NO	Writes standard checkpoints in AFIM logging at a DBH start, DBH end, and a session restart
PP SUBSCHEMA=*n*	1	Defines maximum number of subschemas that can be used at one time in a database; *n*=1..100
PP SUBTRANSACTION=*n*	0	Defines maximum number of logical file names open at one time per database (KDBS only); *n*=1..254
PP TA-ACCESS={STD / SHARED}	STD	Defines usage modes for transactions
PP TRANSACTION={*n* / ([*n*][,*m*])}	(4,1)	*n*: maximum number of simultaneously active transactions and user tasks *n* = 1..225; *m*: maximum number of secondary subtransactions that this DBH can process simultaneously *m* = 1..*n* and *m* ≤ *n*
PP UCON=C'{(*mn*) / <*x* / *nnnn*}'[,{MSG / UDS}]	C' <U'	Defines operator terminal (UCON) on which DCAM administration is to be logged

Table 31: Load parameters of the independent DBH (part 5 of 6)

Parameter	Default value	Meaning
PP WAIT={EVENT / BUSY}=*n*	EVENT	Sets wait mode
PP WARMSTART={STD / FAST / VERY-FAST}	STD	Determines duration of a warm start

Table 31: Load parameters of the independent DBH (part 6 of 6)

1 if PP MPSEG=STD is specified

2 if PP MPSEG=64K is specified

Commands for starting the session

(cf. the "Database Operation" manual, section 2.3.1)

The independent DBH is started by the following commands:

Commands	Meaning
/SET-FILE-LINK LINK-NAME=DATABASE ,FILE-NAME=*configuration-name*	Name of the DB configuration in the session
[/CREATE-FILE FILE-NAME=*konfname*.TEMPO.*nnn* ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=device[,SPACE=...])]	Temporary user realms on private disk
[/CREATE-FILE FILE-NAME=*konfname*.SLF ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=device[,SPACE=...])]	Session log file on private disk
/CREATE-FILE FILE-NAME=*konfname*.DBSTAT ,SUPPRESS-ERRORS=*FILE-EXISTING /CREATE-FILE FILE-NAME=*confname*.DBSTAT.SAVE ,SUPPRESS-ERRORS=*FILE-EXISTING	DB status files
[/ASSIGN-SYSDTA TO={*LIBRARY-ELEMENT(...) *SYSCMD}]	Input source for the DBH parameters
[/ADD-FILE-LINK LINK-NAME=PPFILE ,FILE-NAME=*filename*]	
[/SET-FILE-LINK LINK-NAME=UDSDBHJV ,FILE-NAME=*jvname*]	Variable part of the name of the session job variable
[/SET-JV-LINK LINK-NAME=UDSPS01 ,JV-NAME=*jvname*]	UDS/SQL pubset declaration
[/ADD-FILE-LINK LINK-NAME=$UDSKONF ,FILE-NAME=*UDSTRTAB-modlib*]	User-specific UDSTRTAB Module
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL ,VERSION=*nn.nann*,SCOPE=*TASK	Selection of the UDS/SQL or UDS-D product libraries and subsystems
[/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-D ,VERSION=*nn.nann*,SCOPE=*TASK]	
Private installation (additional commands)	
[/ADD-FILE-LINK LINKNAME=$UDSLIB ,FILE-NAME=*UDS/SQL-modlib*]	
[/ADD-FILE-LINK LINKNAME=$UDSDLIB ,FILE-NAME=*UDS-D-modlib*]	

Table 32: DBH start commands (independent) DBH (part 1 of 2)

Commands	Meaning
/START-UDS-DBH [RESIDENT-PAGES=*PAR(MIN=...,MAX=...)]	Starting the independent DBH
Private installation (alternative command)	
/START-EXECUTABLE-PROGRAM FROM-FILE=*MODULE (LIB=*UDS/SQL-modlib* ,ELEM=UDSSQL)	

Table 32: DBH start commands (independent) DBH (part 2 of 2)

Syntax of the START-UDS-DBH command:

START-UDS-DBH

VERSION = ***STD** / <product-version>

,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>

,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767 *seconds*>

,**RESID**ENT-**PAGE**S = [***PAR**AMETERS](...)

[***PAR**AMETERS](...)

MINIMUM = ***STD** / <integer 0..32767 *4Kbyte*>

,**MAX**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

DAL commands of the independent DBH

(cf. the "Database Operation" manual, section 4.4)

DAL command	Meaning
ABORT { *transaction-id*[,OPTION=PTC] / ALL }	Rolls back the specified open transactions OPTION=PTC: No allowance made for interconfiguration consistency or UDS/SQL-openUTM consistency
ACCESS { LOCK / RETRIEVAL / UPDATE }, { DB=*dbname* / RN=*realm-name*[,DB=*dbname*] }	Handles access locks at database and realm level
ACT { DBTT-INCR,DB=*dbname*[,RECR=*recordref*] [,SCAN={YES / NO}] / INCR,DB=*dbname*[,RR=*realmref*] [,EXT=*(nr-pages,min-pages)*] }	Activates online extensibility of DBTTs or realms
ADD { DB=[$*userid.*]*dbname*[.*copy-name*] [,OPTION=[SHARED-RETRIEVAL] [,OWN-BUFFER-SIZE=*n*][,ID=*bufferid*] / RN=*realm-name*[,DB=*dbname*] / PW=*password* / ADM=*admpassword* }	Adds databases, realms and passwords
&ADD=DISTRIBUTION, { NODE=*processor-name*, CONF=*confname*[,DB=*dbname*] / DB=*dbname*,SS=*subschema-name* / FILE=*file-name* }	Adds new entries to the distribution table
%BIB	Displays the number of messages to the UDS/SQL DBH forwarded by UDSCON. Counts not only the DML statements coming from the COBOL DML, CALL DML, SQL or KDBS, but also the messages from the runtime systems and from UDSCON.
&CHANGE=DISTRIBUTION, NODE=*processor-name*, CONF=*confname*	Assigns a configuration to a different host

Table 33: DAL commands for the independent DBH (part 1 of 6)

DAL command	Meaning
CHECKPOINT[DB=*dbname*[,OPTION=EVEN-WITHOUT-ALOG]]	Writes checkpoints in AFIM logging for individual databases of the configuration or for the entire configuration
CLOSE {RUN-UNITS / CALLS / ADMINISTRATION}	Terminates the current session normally or terminates DCAM administration
&CLOSE DISTRIBUTION	Terminates UDS-D operation
COMMIT *transaction-id*	Terminates transaction in PTC state and commits updates (FINISH) regardless of inter-configuration consistency or UDS/SQL-openUTM consistency
CONTINUE	Scrolls output of DISPLAY SQL DAL command
DEACT{DBIT-INCR,DB=*dbname*[,RECR=*recordref*] / INCR,DB=*dbname*[,RR=*realmref*]}	Deactivates online extensibility of DBTTs or realms

Table 33: DAL commands for the independent DBH (part 2 of 6)

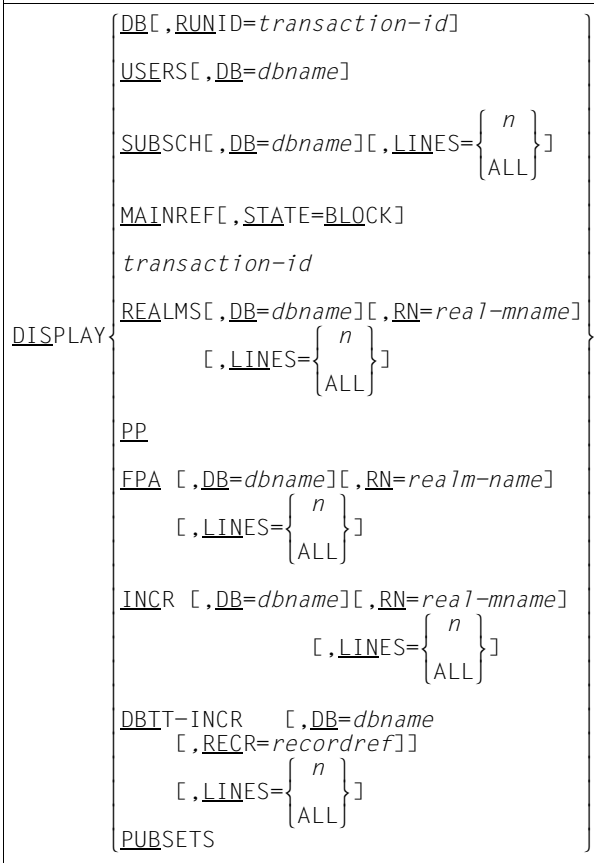
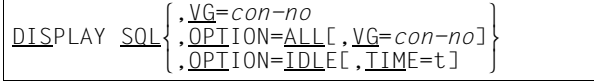
DAL command	Meaning
	Lists databases, transactions, subschemas, mainrefs, available free space or program parameters of the configuration or displays information on the online extensibility of realms or DBTTs or displays UDS/SQL pubset declaration
&DISPLAY DISTRIBUTION [,NODE=*processor-name*] [,CONF=*confname*] [,DB=*dbname*] [,SS=*subschema-name*]	Displays the distribution table
	Displays detailed information on one or more SQL conversations

Table 33: DAL commands for the independent DBH (part 3 of 6)

DAL command	Meaning
%DML	Displays the number of messages to the UDS/SQL DBH forwarded by UDSCON. Counts not only the DML statements coming from the COBOL DML, CALL DML, SQL or KDBS, but also the messages from the runtime systems and from UDSCON.
DROP { DB=*dbname* | RN=*realm-name*[,DB=*dbname*] | PW=*password* | ADM=*admpassword* }	Detaches databases and realms and excludes passwords
&DROP DISTRIBUTION, { ALL | NODE=*processor-name*, CONF=*confname*[,DB=*dbname*][,ALL] | DB=*dbname*,SS=*subschema-name* }	Deletes entries from the distribution table
DUMP [{ ALL | *transaction-id* }]	Generates a DBH memory dump
EXTEND DBIT,DB=*dbname*,RECR=*recordref* [,EXT=*extnmbr*]	Executes online extensibility
EXTEND REALM,DB=*dbname*,RR=*realmref*, EXT=*nr-pages*	Executes online realm extension
FORGET SQL,VG=*con-no*	Releases the resources belonging to the SQL conversation numbered *con-no*
GO { *transaction-id* | ALL | OLD }	Continues execution of the specified transaction(s)
&LOCK DISTRIBUTION, { NODE=*processor-name* | CONF=*confname* | DB=*dbname* | SS=*subschema-name* }	Locks entries in the distribution table
MODIFY { LOG | LOG-2 },VALUE={ :*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2* [,*priv-vsn-3/ device-3*]]) | (vsn-1[,vsn-2 [,vsn-3]]) }	Changes the volume allocation for an original and duplicate RLOG file to be newly created

Table 33: DAL commands for the independent DBH (part 4 of 6)

DAL command	Meaning
MODIFY LOG-SIZE,VALUE=([*primary*] [,[*secondary*]])	Changes the amount of storage in RLOG files
MODIFY PTCSYNCH, VALUE=([{WAIT / ABORT / COMMIT}][,[{WAIT / ABORT / COMMIT}]])	Modifies the value of the DBH load parameter PTCSYNCH
MODIFY RESERVE,VALUE={NONE / :*catid*: / PUBLIC / (*priv-vsn-1/device-1*[,*priv-vsn-2/device-2*][,*priv-vsn-3/device-3*]) / (vsn-1[,vsn-2[,vsn-3]])}	Changes the reserve volumes for the RLOG files
NEW PUBSETS	Checks and notes new UDS/SQL pubset declaration
NEW RLOG	Selects new RLOG files
PERFORM {NOCANCEL / CANCEL}	Initiates execution of the DROP, ADD, NEW or CHECKPOINT commands Default value: NOCANCEL
&PWD DISTRIBUTION,CONF=confname, {PWN=*new-password* / PWO=*old-password* / PWO=*old-password*,PWN=*new-password*}	Assigns and changes password
REACT INCR,DB=*dbname*[,RR=*realmref*]	Reactivate online realm extensibility
RESET ORDERS	Cancels pending requests
&SAVE DISTRIBUTION,FILE=*file-name*	Saves the distribution table
&START DISTRIBUTION	Starts UDS-D operation
STOP {*transaction-id* / NEW / ALL}	Stops execution of the specified transaction(s)
&SYNCHRONIZE DISTRIBUTION	Terminates secondary subtransactions (STTs) in the PTC state

Table 33: DAL commands for the independent DBH (part 5 of 6)

DAL command	Meaning
%TERM	Aborts the session in the quickest way (emergency halt) and optionally generates a complete DBH dump **CAUTION!** %TERM should be used only if the master task stops accepting DAL commands due to an error.
&UNLOCK DISTRIBUTION, {NODE=*processor-name* CONF=*confname* DB=*dbname* SS=*subschema-name*}	Removes locks on entries in the distribution table

Table 33: DAL commands for the independent DBH (part 6 of 6)

10.1.1 UDSADM

Starting UDSADM

(cf. the "Database Operation" manual, section 4.1)

UDSADM is started with:

```
/START-UDS-ADM
```

Statement selection rules

Statement	Selection rules
END HELP-DAL-CMD MODIFY-MSG-FORMAT MODIFY-MSG-WAIT-TIME	These statements may be entered at any time.
DISCONNECT-CONFIGURATION	This statement is permitted only for a connection that was previously set up using CONNECT-CONFIGURATION.
CONNECT-CONFIGURATION	This statement is permitted for a connection not yet set up.
EXECUTE-DAL-CMD SET-RECEIVE-MODE SHOW-CONNECTION-ATTRIBUTES SHOW-OUTSTANDING-MSG	These statements are permitted only between CONNECT-CONFIGURATION and DISCONNECT.

Table 34: Statement selection rules

UDSADM-statements

(cf. the "Database Operation" manual, section 4.1.1)

The statement formats of the UDSADM administration program comply with SDF (System Dialog Facility) conventions; see the manuals "SDF Dialog Interface" and the commands manuals for "BS2000 OSD/BC". Uppercase letters printed in boldface denote guaranteed abbreviations of keywords.

Overview of the UDSADM statements

Statement	Meaning
CONNECT-**C**ONFIGURATION **C**ONFIGURATION-**N**AME = <name> ,**PAS**SWORD = ***NONE** / <x-string> <c-string> ,**HOST** = ***LOC**AL / <name>	Sets up connection to UDS/SQL configuration
DISCONNECT-**C**ONFIGURATION	Clears connection to UDS/SQL configuration
END	Terminates UDSADM
EXECUTE-**D**AL-**C**MD **CMD** = <dal-cmd>	Executes DAL commands
HELP-**D**AL-**C**MD **?**	Help function for DAL commands
MODIFY-**M**SG-**F**ORMAT **HEADER** = **NO** / **YES**	Sets message output format
MODIFY-**M**SG-**W**AIT-**T**IME **TIME** = **STD** / <integer 1..7200 *seconds*>	Sets wait time for DAL commands
SET-**R**ECEIVE-**M**ODE	Activates permanent receive task
SHOW-**C**ONNECTION-**A**TTRIBUTES	Displays information on connection
SHOW-**O**UTSTANDING-**M**SG	Displays outstanding messages

Table 35: UDSADM statements

UDSADM statements entered incorrectly are rejected with a specific message and can be corrected. Each UDSADM statement entered correctly is executed immediately.

10.2 Database operation using the linked-in DBH

(cf. the "Database Operation" manual, chapter 3)

Load parameters

Parameter	Default value	Meaning
PP 2KB-BUFFER-SIZE=*n*	1	Defines size of the 2-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 4KB-BUFFER-SIZE=*n*	1	Defines size of the 4-Kbyte system buffer pool in Mbytes; *n*=1..2047
PP 8KB-BUFFER-SIZE=*n*	0	Defines size of the 8-Kbyte system buffer pool in Mbytes; *n*=0 or *n*=3..2047
PP CATPASS={STANDARD | *password*}	STD	Defines password for files to be created by DBH, such as ALOG files and temporary realms
PP CONSOLE ={YES | NO}	NO	Outputs linked-in DBH messages to operator console as well
PP DBNAME=[$*userid.*]*dbname* [.*copyname*] [,SHARED-RETRIEVAL] [,{*n* | [*n*],*bufferid*}]	-	Names databases in the DB configuration; max. 222 databases *n*: defines size of user buffer pools in Mbytes; *n*=0..2047 *bufferid*: buffer pool identifier
PP END	-	Terminates input of load parameters
PP LOG={NO | :*catid*: | PUBLIC | (*priv-vsn-1/device-1* [,*priv-vsn-2/device-2*] [,*priv-vsn-3/device-3*]]) | (*vsn-1*[,*vsn-2* [,*vsn-3*]]) }	-	Maintains the RLOG file

Table 36: Load parameters of the linked-in DBH (part 1 of 3)

Parameter	Default value	Meaning
PP LOG-2={:*catid*: PUBLIC (*priv-vsn-1*/*device-1* [,*priv-vsn-2*/ *device-2* [,*priv-vsn-3*/ *device-3*]]) (*vsn-1*[,*vsn-2* [,*vsn-3*]])	-	Maintains the duplicate RLOG file
PP LOG-SIZE=([*primary*] [,[*secondary*]])	192,192	Specifies amount of storage (number of PAM pages) in RLOG files
PP MAXDB=*n*	Sum of PP DBNAME	Defines maximum number of databases in DB configuration; *n*=1..122
PP PARLIST={YES NO}	NO	Lists load parameters used
PP PASSWORD={NONE STANDARD *password* (*password*, *password*,..) *password*,..)}	STD	Defines password that the database must use along with PP CATPASS in order to open files
PP PRIVACY-CHECK={STD NO-KSET OFF}	STD	Controls handling of privacy checks
PP RESERVE={NONE :*catid*: PUBLIC (*priv-vsn-1*/ *device-1* [,*priv-vsn-2*/ *device-2* [,*priv-vsn-3*/ *device-3*]]) (*vsn-1*[,*vsn-2* [,*vsn-3*]])	NONE	Specifies reserve volumes for the RLOG files
PP STDCKPT={YES NO}	NO	Writes standard checkpoints in AFIM logging at a DBH start, DBH end, and a session restart
PP SUBSCHEMA=*n*	1	Defines maximum number of subschemas that can be used at one time in a database; *n*=1..100

Table 36: Load parameters of the linked-in DBH (part 2 of 3)

Parameter	Default value	Meaning
PP SUBTRANSACTION=*n*	0	Defines maximum number of logical file names open at one time per database (KDBS only); *n*=1..254
PP TRANSACTION=1	1	Defines number of simultaneously active transactions
PP WARMSTART={STD / FAST / VERY-FAST}	STD	Determines duration of a warm start

Table 36: Load parameters of the linked-in DBH (part 3 of 3)

Command sequence for starting a linked-in application

(cf. the "Database Operation" manual, section 2.3.1)

Commands	Meaning
/SET-FILE-LINK LINK-NAME=DATABASE ,FILE-NAME=*configuration-name*	Name of the DB configuration in the session
[/CREATE-FILE FILE-NAME=*konfname*.TEMPO.*nnn* ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=*device*[,SPACE=...])]	Temporary user realms on private disk
[/CREATE-FILE FILE-NAME=*konfname*.SLF ,SUPPORT=*PRIVATE-DISK(VOLUME=*priv-vsn* ,DEVICE-TYPE=*device*[,SPACE=...])]	Session log file on private disk
/CREATE-FILE FILE-NAME=*confname*.DBSTAT ,SUPPRESS-ERRORS=*FILE-EXISTING /CREATE-FILE FILE-NAME=*confname*.DBSTAT.SAVE ,SUPPRESS-ERRORS=*FILE-EXISTING	DB status files
[/MODIFY-JOB-SWITCHES OFF=28]	Selection of the linked-in DBH when the UDSLNKA module is integrated
[/ASSIGN-SYSDTA TO={*LIBRARY-ELEMENT(...) | *SYSCMD}]	Input source for the DBH parameters
[/ADD-FILE-LINK LINK-NAME=PPFILE ,FILE-NAME=*filename*]	
[/SET-FILE-LINK LINK-NAME=UDSDBHJV ,FILE-NAME=*jvname*]	Variable part of the name of the session job variable
[/SET-JV-LINK LINK-NAME=UDSPS01 ,JV-NAME=*jvname*]	UDS/SQL pubset declaration
[/ADD-FILE-LINK LINK-NAME=$UDSSSI ,FILE-NAME=*SSITAB*-modlib]	SSITAB modules (CALL-DML)
[/ADD-FILE-LINK LINK-NAME=$UDSPLEX ,FILE-NAME=*PLITAB*-modlib]	PLITAB modules (UDSKDBS)
[/ADD-FILE-LINK LINK-NAME=$UDSKONF ,FILE-NAME=*UDSTRTAB*-modlib]	User-specific UDSTRTAB module

Table 37: DBH- start commands (linked-in DBH) (part 1 of 2)

Commands	Meaning
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL ,VERSION=*nn.nann*,SCOPE=*TASK	Selection of the UDS/SQL product libraries and subsystems
Private installation (additional commands)	
[/ADD-FILE-LINK LINKNAME=$UDSLIB ,FILE-NAME=*UDS/SQL-modlib*]	
[/ADD-FILE-LINK LINKNAME=$UDSKLIB ,FILE-NAME=*KDBS-modlib*]	
Private installation (alternative commands)	Starting the the linked-in application
/START-PROGRAM FROM-FILE=*MODULE(LIB=*lib* ,ELEM=*elem*,RUN-MODE=*ADVANCED [(NAME-COLLISION=*STD)],PROG-MOD=*ANY) [,RESIDENT-PAGES=*PAR(MIN=...,MAX=...)]	

Table 37: DBH- start commands (linked-in DBH) (part 2 of 2)

DAL commands of a linked-in DBH

(cf. the "Database Operation" manual, section 4.4)

DAL command	Meaning
%BIB	Displays the number of base interface blocks processed in this part of the session
%DUMP	Outputs a complete dump of the DBH with edited tables to a file; the program is not aborted

Table 38: DAL commands for the linked-in DBH

10.3 UDSMON

(cf. the "Database Operation" manual, section 11.3)

UDSMON statements for starting

Statement	Meaning
CONFNAME=*confname*	Allocates configuration
HELP	Lists all UDSMON statements
MASK={S | C | ALL | CS | DC | DT | D}	Assigns output mask(s) At the defined interval, UDSMON displays: S : the STATUS mask C : the COUNTER mask ALL : all masks in turn CS : the COUNTER mask and the STATUS mask alternately DC : the CONNECT mask DT : the TRANSACTION mask D : the TRANSACTION mask and the CONNECT mask alternately Default value : S
MEDIUM={T,*n* | L,*n*[,D] | F,*n*[,D] | S,*n*}	Assigns output device and time interval T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 Default value : T in interactive mode L in batch mode *n* : time interval, in seconds, between queries (*n*=5..999) Default value : 10 D : specifies that UDS-D operation is to be evaluated. With output to a file, an extra label and, at the set intervals, an extra data record are written. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is added.
DISPLAY {DBCOUNTERS | DEPRECATED | EXPENSIVE | MESSAGES}	Output to intermediate file (cf. corresponding command during operation, see table 40 on page 160) The command is noted and executed at startup.

Table 39: UDSMON statements for starting (part 1 of 2)

Statement	Meaning
REBASE-COUNTER=NEW	Requests a logical zero point exclusively for the session counters of the counter mask
RUNTIME=*n*	Specifies the time after which UDSMON is to terminate automatically. This statement can be used particularly in batch mode, e.g. to obtain output to printer after a specified time. *n* : runtime in seconds (*n* = 60..86400) The time specified here should be greater than the time interval for the allocated output media.
START	Starts UDSMON Data collection commences.
END	Terminates UDSMON No data is collected.

Table 39: UDSMON statements for starting (part 2 of 2)

UDSMON commands during execution

Command	Meaning
INFORM–**PROG**RAM **MSG**= 'ADD MEDIUM= {T,*n* / L,*n*[,D] / F,*n*[,D] / S,*n*}'	Assigns output device and time interval T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 Default value : T in interactive mode L in batch mode *n* : time interval, in seconds, between queries (*n*=5..999) Default value : 10 D : specifies that UDS-D operation is to be evaluated. With output to a file, an extra label and, at the set intervals, an extra data record are written. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is added.
INFORM–**PROG**RAM **MSG**='DISPLAY'	Lists devices and time intervals currently set
INFORM–**PROG**RAM **MSG**='DISPLAY CP–SIZE'	Lists number and size of common pools
INFORM–**PROG**RAM **MSG**='DISPLAY DB–IO–COUNTER‘	Lists all database I/Os that have occurred since starting the DBH and in the last time interval. The database I/Os are arranged in categories (2-Kbyte, 4-Kbyte, 8-Kbyte, and exclusive I/Os) and further subdivided into logical and physical read/write I/Os per category
INFORM–**PROG**RAM **MSG**=‘DISPLAY DBCOUNTERS‘	Displays the number of DMLs and I/Os per database since starting the DBH and while database is online.
INFORM–**PROG**RAM **MSG**=‘DISPLAY DEPRECATED‘	List of applications with properties which contain a potential risk (limited address space utilization, obsolete loading technique, subschema validation missing) The list is written to the intermediate file (see the manual "Database Operation") and displayed by the monitor by means of a SHOW-FILE command.

Table 40: UDSMON commands during operation (part 1 of 5)

Command	Meaning
INFORM–**PROG**RAM **MSG**=‘DISPLAY EXPENSIVE‘	List of applications with the greatest resource utilization with regard to transaction duration, number of DMLs per transaction, number of logical inputs/outputs per DML and per transaction, separated according to UTM and TIAM applications The list is written to the intermediate file (see the manual "Database Operation") and displayed by the monitor by means of a SHOW-FILE command.
INFORM–**PROG**RAM **MSG**=‘DISPLAY MESSAGES‘	Edited list of buffered DBH messages The list is written to the intermediate file (see the manual "Database Operation") and displayed by the monitor by means of a SHOW-FILE command. In the case of consecutive commands in a monitor run with explicit allocation of the intermediate files, messages which have been added since the previous DISPLAY MESSAGES command are supplemented. Messages which are overwritten in the buffer and can therefore no longer be listed are tagged with a separate flag. The date change is displayed.
INFORM–**PROG**RAM **MSG**='DISPLAY REX'	Displays total number of transactions opened using READY EXCLUSIVE or READY PROTECTED
INFORM–**PROG**RAM **MSG**=‘DISPLAY SATURATION‘	List of the greatest saturation of central system resources in the DBH session to date (e.g. number of parallel transactions, utilization of non-extensible memory pools)

Table 40: UDSMON commands during operation (part 2 of 5)

Command	Meaning
INFORM–**PROG**RAM **MSG**='DISPLAY TASKS'	Lists all the user tasks associated with UDS/SQL and the number of DMLs already processed in these tasks In the case of the processed DMLs, in contrast to comparable values of the counter mask, for diagnostic reasons requests of the COBOL runtime system which ensure correct transaction processing in the case of a task switch are also counted. For applications which were terminated asynchronously, e.g. as a result of timeouts in UTM applications, the following is also output: – whether this termination has been completed successfully and the internal UDS resources can be reused (RELEASED), – whether termination has not yet been fully completed (PENDING) or – whether the relevant internal UDS resource is locked permanently because of a serious error (BLOCKED).
INFORM–**PROG**RAM **MSG**='END'	Terminates the monitor
INFORM–**PROG**RAM **MSG**= 'FINISH MEDIUM= {T / L,[D] / F,[D] / S}'	Terminates output on an output device T : output to data display terminal L : output to printer F : output to a file S : transfer to SM2 D : Only output for UDS-D operation is terminated. With output to a printer, the information from the masks DC (CONNECT) and DT (TRANSACTION) is omitted. With output to a file, the UDS-D record is omitted.
INFORM–**PROG**RAM **MSG**='HELP'	Lists input options

Table 40: UDSMON commands during operation (part 3 of 5)

Command	Meaning
INFORM–PROGRAM **MSG**=C'MASK={S / C / ALL / CS / DC / DT / D}'	Assigns output mask(s) At the defined interval, UDSMON displays: S : the STATUS mask C : the COUNTER mask ALL : all masks in turn CS : the COUNTER mask and the STATUS mask alternately DC : the CONNECT mask DT : the TRANSACTION mask D : the TRANSACTION mask and the CONNECT mask alternately Default value : S

Table 40: UDSMON commands during operation (part 4 of 5)

Command	Meaning
INFORM–**PROG**RAM **MSG**='REBASE–COUNTER=NEW'	Rebasing of the counters of the COUNTER mask, i.e. requesting a logical zero point exclusively for the session counters of the COUNTER mask The next time the counters are collected, the current statuses of the COUNTER mask will be stored ("rebasing" or "snapshot" of the current counter statuses), and from then all counter statuses are specified with reference to this stored status (logical zero point). Rebasing can be used for the following in the case of extremely long DBH sessions: - to display comparable session counters, for example on a daily or weekly basis - to facilitate recognition of a change to counter statuses - to prevent problems in comparative interpretation when individual counters overrun Rebasing always refers to a single monitor instance. It is not possible to return to a previous rebasing. The values for a rebasing remain valid as long as the corresponding DBH session or the UDS monitor continues to run. The following events invalidate rebasing: - Restart of the DBH session When the DBH session restarts, the DBH's counters once more begin with 0. - Restart of the UDS monitor As the rebasing data is located locally in the monitor memory, it is lost when the monitor restarts.
INFORM–**PROG**RAM **MSG**='REBASE–COUNTER=OFF'	The counter statuses of the COUNTER mask which are used for rebasing are set to 0. As a result, the session counters in the COUNTER mask are once more displayed as absolute values with reference to the start of the DBH session.

Table 40: UDSMON commands during operation (part 5 of 5)

Command sequence for starting and operating UDSMON

(cf. the "Database Operation" manual, section 11.4)

```
01  [/CREATE-FILE FILE-NAME=monitor-file
       [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary,SECONDARY-ALLOCATION=secondary)) or
        ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,DEVICE-TYPE=device[,SPACE=...])]]
    [/ADD-FILE-LINK LINK-NAME=TR,FILE-NAME=monitor-file,ACCESS-METHOD=*SAM]

02  /START-UDS-MONITOR

03  "udsmon statements for starting"

04 [EM], [DUE]

05  "udsmon commands during execution"

06 [EM], [DUE]

07  /INFORM-PROGRAM MSG='END'
```

01 If the monitor file is to be set up explicitly, the CREATE-FILE command must be used for this purpose. In this case specifying :*catid*: is permitted in accordance with BS2000 conventions.

02 The monitor is started.

04/06 An interrupt during monitor execution is initiated by means of [EM], [DUE] .

07 The monitor is terminated.

After a program interrupt, due to [K2] for example, the INFORM-RPGRAM command is required in order to resume operation.

In batch mode, an invalid UDSMON statement causes the monitor start to be terminated. An error message is output to SYSOUT.

10.4 Using pubsets in UDS/SQL

(cf. the "Database Operation" manual, section 9.1)

The following table indicates the user/DBH interfaces at which specifying a catalog ID :*catid*: is and is not permitted:

User interface	Function
/SET-FILE-LINK LINK-NAME=DATABASE ,FILE-NAME=[:*catid*:][$*admuserid*.]*configuration-name*	Defines the configuration name
/ADD-FILE-LINK LINK-NAME=DATABASE ,FILE-NAME=[:*catid*:][$*dbuserid*.]*dbname*[.DBDIR] [1]	Defines the database for the utility run
PP DBNAME=[$*dbuserid*.]*dbname*[.*copy-name*],... ADD DB=[$*dbuserid*.]*dbname*[.*copy-name*],...	Specifies the name and optionally the database identification
PP DISTABLE=[:*catid*:][$*userid*.]*file-name* ADD DIS,FILE=[:*catid*:][$*userid*.]*file-name* SAVE DIS,FILE=[:*catid*:][$userid.]*file-name*	Specifies the name of a DISTABLE file
/ADD-FILE-LINK LINK-NAME=PPFILE ,FILE-NAME=[:*catid*:][$*userid*.]*file-name* /ASSIGN-SYSDTA TO=[:*catid*:][$*userid*.]*file-name*	Assigns the load parameter file

Table 41: Overview of options for the catalog identifier at UDS/SQL user interfaces

1 Specifying :*catid*: is permitted in this case so that an existing DBDIR can be utilized for the ADD-FILE-LINK command.

10.5 Using job variables in UDS/SQL

(cf. the "Database Operation" manual, section 9.2)

The following job variable types are supported by UDS/SQL:

Type	Name	Meaning	See
Pubset declaration	LINK-NAME= UDSPS01	Defines the pubsets which can be used by the DBH and the utility routines	page 168
Session	JV-NAME= UDSSQL.DBH.*session*	Information on the status of the DBH session for automatic control of applications and administration	page 170
Database	JV-NAME= UDSSQL.DB. *database-name*	Information on the status of a database for automatic control of applications and administration	page 173
BMEND	LINK-NAME= JVBMEND	Output data of BMEND for automatic control of backup and recovery	Recovery, Information and Reorganization manual

Table 42: Job variables used by UDS/SQL

In order to work with job variables, the “Job Variables (JV)” subsystem must be installed. If this subsystem is not available, generally no job variables are supplied and no corresponding message is issued.

10.5.1 Pubset declaration job variable

(cf. the "Database Operation" manual, section 9.2.1)

Syntax of the UDS/SQL pubset declaration

The UDS/SQL pubset declaration in the pubset declaration job variable consists of a sequence of catid groups, each of which comprises an FSTAT-compliant catalog specification (see the Introductory Guide to DMS) and separated from each other by one or more blanks.

catid-group[catid-group]...
Specifies one or more catid groups (1- to 4-character catalog IDs without ":") which may contain the wildcards listed in table 43 below:

*	Replaces any string, even an empty one. An * in the first position must be duplicated if the * is followed by further characters and the string entered does not contain at least one more wildcard.
/	Replaces precisely one arbitrary character.
<s_x:s_y>	Replaces a string for which the following applies: – It is at least as long as the shortest string (s_x or s_y) – It is no longer than the longest string (s_x or s_y) – In the alphabetic sort it is between s_x and s_y; digits are sorted after letters (A…Z, 0…9) – s_x may also be the empty string which is in the first position in the alphabetic sort – s_y may also be the empty string which in this position stands for the string with the highest possible coding (contains only the characters X'FF')
<s_1,…>	Replaces all strings which one of the character combinations specified with s matches. s can also be the empty string. Each s string can also be the range specification "s_x:s_y".
-s	Replaces all strings which do not match the specified string s. The minus sign may only be used at the beginning of the string. **This specification cannot be combined with other specifications in a UDS/SQL pubset declaration.**

Table 43: Wildcards for catalog IDs in the UDS/SQL pubset declaration

Up to 100 catid groups may be specified.
A catid group may be up to 26 characters long.
Lowercase letters are treated like the corresponding uppercase letters.
Catalog IDs which do not exist or are not available may be specified.

Multiple specification of catalog IDs is also possible.

Examples

Specification	Stands for the following catalog IDs
A001	A001
X/C	XAC, XBC, XCC, ..., XZC, X0C, ..., X9C (All 3-character catalog IDs which begin with X and end with C)
5*	5, 5A, ..., 59, 5AA, ..., 599, 5AAA, 5999 (All 1- to 4-character catalog IDs which begin with 5)
<C015:C025>	C015, C016, ..., C024, C025 (All 4-character catalog IDs in the range from C015 to C025)
<BE:DC>	BE, BF, BG, ..., B9, CA, ..., C9, DA, ..., DC (All 2-character catalog IDs in the range from BE to DC)
<D015:D025,F015:F045>	D015, D016, ..., D024, D025, F015, F016, ..., F044, F045 (List of ranges: all 4-character catalog IDs from D015 to D025 and from F015 to F045)
<A:D;BE:DC>	A, B, C, D, BE, BF, BG, ..., B9, CA, ..., C9, DA, ..., DC (List of ranges: all 1-character catalog IDs from A to D and all 2-character catalog IDs from BE to DC)
-5*	All (1- to 4-character) catalog IDs which **do not** begin with 5 (exclusion condition).

Table 44: Examples of catalog IDs in the UDS/SQL pubset declaration

Command string for defining and assigning a UDS/SQL pubset declaration:

```
/CREATE-JV JV-NAME=UDS-PUB-DECL
/MODIFY-JV JV-CONTENTS=UDS-PUB-DECL, -
/ SET-VALUE='A001 B001 <C015:C025> <D015:D025,F015:F045> 5*'
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
```

Command string for defining and assigning a UDS/SQL pubset declaration with exclusion condition:

```
/CREATE-JV JV-NAME=UDS-PUB-DECL
/MODIFY-JV JV-CONTENTS=UDS-PUB-DECL,SET-VALUE='-5*'
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
```

The Default Public Volume Set of the execution user ID is always implicitly taken into account by UDS/SQL and consequently does not need to be included in the UDS/SQL pubset declaration. It is not possible to exclude the Default Public Volume Set of the execution user ID from being used.

Assignment of a UDS/SQL pubset declaration which only contains blanks is permissible; only the Default Public Volume Set of the execution user ID is taken into account.

10.5.2 Session job variable

(cf. the "Database Operation" manual, section 9.2.2)

The DBH supports a session job variable that helps to automate administration. This job variable can be used to control user jobs and programs.

The job variable UDSSQL.DBH.*session* is structured as follows:

Column	Contents	Meaning
1-30	UDSDBH_STARTING / UDSDBH_ACTIVE / UDSDBH_CLOSE_INITIATED / UDSDBH_NORMAL_END / UDSDBH_ABNORMAL_END	Status (1)
31-40 41-48	*yyyy-mm-dd* *hh:mm:ss*	DBH start (2) Date Time
49-58 59-66	*yyyy-mm-dd* *hh:mm:ss*	DBH end (3) Date Time
67-68	01	Version ID of the layout of the session job variable
69-76	*session-name*	Configuration name of the DBH session
77-84	*host*	Host der DBH-Session
85-92	*nnnnnnnn* / ␣	Session section number (4)
93	B / ␣	Status of the DBH operational interruption: – Current operational interruption – Currently no operational interruption
94-96	DAL / DBH / ␣	Type of current or last DBH operational interruption (5) – Because of DALs (PERFORM) – Because of internal system activities – No operational interruption has occurred as yet
97	A / – / ␣	Attaching database or realm in current or last operational interruption (5) – Yes – No – No operational interruption has occurred as yet

Table 45: Structure of the session job variable for UDS/SQL (part 1 of 2)

Column	Contents	Meaning
98	D / – / ␣	Detaching database or realm in current or last operational interruption (5) – Yes – No – No operational interruption has occurred as yet
99	A / – / ␣	Activity with respect to ALOG files in current or last operational interruption (5) – Yes – No – No operational interruption has occurred as yet
100	R / – / ␣	Activity with respect to RLOG files in current or last operational interruption (5) – Yes – No – No operational interruption has occurred as yet
101-112	␣	Reserved
113-122 123-130	*yyyy-mm-dd* *hh:mm:ss*	Start of the preparatory phase for the current or last operational interruption (5) Date Time
131-138	*hh:mm:ss*	Start of the Implementation phase for the current or last operational interruption (5) Time
139-146	*hh:mm:ss*	End of the last operational interruption (5) Time
147-162	␣	Reserved
163-172 173-180	*yyyy-mm-dd* *hh:mm:ss*	Last change to the job variable Date Time

Table 45: Structure of the session job variable for UDS/SQL (part 2 of 2)

Comments

(1) *Status*

Content	Set when?
UDSDBH_STARTING	When the DBH starts
UDSDBH_ACTIVE	After successful initialization (e.g. System Ready message of the independent DBH)
UDSDBH_CLOSE_INITIATED	When no further requests are permitted in the DBH because the DBH end has been initiated by the UDS/SQL administration or for internal reasons by the DBH
UDSDBH_NORMAL_END / UDSDBH_ABNORMAL_END	When the DBH ends

(2) *DBH start* is initialized with 0 when the DBH starts and is filled with the current time following successful initialization (e.g. System Ready message of the independent DBH).

(3) *DBH end* is initialized with 0 when the DBH starts and is filled with the current time when the DBH ends.

(4) *Session section number* is initialized with blanks when the DBH starts and during initialization is filled with a value which unambiguously identifies the session section and remains unchanged up to the end of the session section.

The session section number enables the assignment of database job variables to a current session to be checked: only if the session section numbers in the database job variables and in the session job variables match are the contents of the database job variables valid for the current session.

(5) The displays for and times of the various phases of an operational interruption always relate to the current or last operational interruption displayed in *Status of the DBH operational interruption* (byte 93).

Internal operational interruptions in the DBH's start or termination phase are not displayed.

10.5.3 Database job variable

(cf. the "Database Operation" manual, section 9.2.3)

The DBH and the utility routines DDL compiler, SSL compiler, BGSIA, BGSSIA, BPRIVACY, BMEND, BREORG, BCHANGE, BRENAME and BALTER maintain a database job variable to enhance automatic administration. You can use this job variable to control user requests and programs.

The database job variable UDSSQL.DB.*databasename*.[.*copyname*][.*no*] is assigned values as follows:

Column	Contents	Meaning
1-2	01	Version identifier of the session job variable's layout
3-19	*ccccccccccccccc*	Database name (1)
20-26	*cccccc* / ␣	Copy name (2)
27-32	*cccccc*	DB layout version
33	C / I / ␣	Consistency (3)
34-40	␣	Reserved
41-50	UPDATE / RETR/ WARMSTART/ OPEN / DROP / CLOSE / ERROR	Processing status (4): – Attach mode for the DBH – Attach mode for a utility routine – Cause of the last termination in the case of the DBH or a utility routine
51	␣ / A	Activation of ALOG (5)
52	␣ / O	Identifier for online backup capability (6)
53-60	DBH/ LKIN-DBH / UTIL-DBH / *utility*	DBH or utility routine (7) Current or last user of the database – Independent DBH – Linked-in DBH – Utility routine with linked-in DBH (DDL/SSL compiler, BGSIA, BGSSIA, BPRIVACY) – Name of a modifying utility routine
61-68	*cccccccc* / ␣	Configuration name of the DBH session (8)
69-72	*cccc* / ␣	Default catalog of the DBH user ID (9)
73-81	$*cccccccc* / ␣	User ID of the DBH session (with $) (9)
82-89	*nnnnnnnn* / ␣	Session section number (10)
90-99 100-107	*yyyy-mm-dd* *hh:mm:ss*	Start of processing (11) Date Time

Table 46: Structure of the database job variable for UDS/SQL (part 1 of 2)

Column	Contents	Meaning
108-117 118-125	*yyyy-mm-dd* *hh:mm:ss*	End of processing (12) Date Time
126-135 136-143	*yyyy-mm-dd* *hh:mm:ss*	Last ALOG change (13) Date Time
144-152	*nnnnnnnnn* / ␣	ALOG sequence number (14)
153-162	*nnnnnnnnnn* / ␣	Size of the ALOG file (15)
163-182	␣	Reserved
183-192 193-200	*yyyy-mm-dd* *hh:mm:ss*	Last change job variable Date Time

Table 46: Structure of the database job variable for UDS/SQL (part 2 of 2)

Comments

(1) *Database name* is the name of the database which is also contained in the job variable name.

(2) *Copy name* is only filled with values in the shadow database.

(3) *Consistency* is only filled with values by the DBH and shows whether the database is consistent ("C") or inconsistent ("I"). In this sense a database is inconsistent when the DBH is currently modifying the database and possibly not all changes have been written back to the database files.

However, the database can also be inconsistent if processing by the DBH was terminated abnormally. Warm-starting the database generally returns it to a consistent status.

(4) *Processing status* shows the attach mode or the cause of the last termination. Temporary access restrictions (e.g. because of DAL ACCESS) or restrictions to the operating mode (because, for example, the RLOG file cannot be used) are not displayed.

ERROR is set if the database is switched off under control while it is inconsistent or processing of the database has been terminated because of other errors. In the latter case it is possible that the database is still consistent.

However, in some cases it is possible that the DBH session is terminated abnormally without it being possible to update the processing status. The database is then generally still inconsistent and the processing status remains UPDATE. When a subsequent warm start is performed by the DBH, the processing status

WARMSTART is set, and at the end of the warm start the status required for processing (e.g. UPDATE) is entered. Modifying utility routines supply the statuses OPEN, CLOSE and, if required, ERROR.

In some error situations controlled program termination is not possible for the utility routines. In such a case the ERROR status can also not be set correctly. The job variable then remains in the OPEN status even after the utility routine has terminated.

It is also possible that an error occurs in the last phase of utility routine termination after the job variable has already been supplied with the CLOSE status. In this case the job variable is, if possible, filled with the ERROR status.

(5) *Activation of ALOG* shows whether ALOGGING is activated ("A") for the database or not (blank). Current processing in the DBH can take place without ALOGGING if, for example, the database is attached in SHARED-RETRIEVAL mode.

(6) *Identifier for online backup capability*
"O" indicates that an online backup of the database can currently be created, i.e. online backup capability is enabled for the database (BMEND ENABLE-ONLINE-COPY) and the database is attached to a DBH.
In all other cases the item contains blanks.
This property should always be checked before an inconsistent copy is made in parallel to a DBH session using COPY-FILE.
The item is filled with information **only by the DBH** and is deleted when the database is detached. In particular it is **not** set by BMEND if online backup capability has just been enabled.

(7) *DBH or utility routine* is filled with information by the DBH or utility routine when the database is attached and then remains unchanged.

(8) *Configuration name of the DBH session* shows the current or the last name of the DBH configuration. The item is filled with information when the database is attached and then remains unchanged. Utility routines which do not use the linked-in DBH enter blanks. In SHARED-RETRIEVAL mode a database job variable should be created before the database is attached to a DBH session and filled with the configuration name.

(9) *Default catalog of the DBH user ID* and *User ID of the DBH session* contain the values of the relevant DBH session. These values can be used for unique access to the session job variable.

Utility routines which do not use the linked-in DBH enter blanks.

(10) *Session section number* is filled with the current value of the session when the database is attached, and the DBH deletes it with blanks when the database is detached.

Utility routines which do not use the linked-in DBH enter blanks when the database is attached.

(11) *Start of processing* is filled when the database is attached (DBH) or opened (utility routine) and then remains unchanged.

(12) *End of processing* is filled when the database is detached (DBH) or closed (utility routine). When utility routines terminate abnormally, the data provided depends on whether the job value as a whole is updated. In the UPDATE and RETR status the item contains blanks.

(13) *Last ALOG change* shows the time of the last ALOG change or (re)activation of ALOGGING. The date is retained even when ALOGGING is disabled.

(14) *ALOG sequence number* is filled with values by the DBH or utility routines in the event of a change of ALOG if ALOGGING is enabled.

(15) *Size of the ALOG file* shows the size of the used part of the ALOG file in PAM pages. This size can differ slightly from the file size as seen by the DMS. This deviation can be caused either by rounding on account of the size of the allocation unit used (3, 4 or 32 PAM pages) or because as a result of doubling the secondary allocation (cf. class-2 system parameter DMMAXSC or the MAXIMAL-ALLOCATION parameter in ADD-/MODIFY-MASTER-CATALOG-ENTRY) a current file extension is larger than the extension requested by UDS/SQL.

The item is filled with a current value **only by the DBH**. Utility routines which do not use the linked-in DBH enter blanks.

11 Utility routines

11.1 START commands for the UDS/SQL programs

Syntax of the START-UDS-... commands

START-UDS-...

VERSION = ***STD** / <product-version>
,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>
,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767 *seconds*>

,RESIDENT-PAGES = [***PAR**AMETERS](...) *Only for DBH*
[***PAR**AMETERS](...)
MINIMUM = ***STD** / <integer 0..32767 *4Kbyte*>
,**MAX**IMUM = ***STD** / <integer 0..32767 *4Kbyte*>

The following table shows the START commands (and their aliases) that you can use to call the specified UDS/SQL programs

The following prerequisites must be fulfilled:

– UDS/SQL must be installed with IMON and
– the SDF system syntax file must be activated.

Program	START command	Alias
UDSSQL	START-UDS-DBH	UDS, SYSINT
BALTER	START-UDS-BALTER	BALTER
BCALLSI	START-UDS-BCALLSI	BCALLSI
BCHANGE	START-UDS-BCHANGE	BCHANGE

Table 47: Calling UDS/SQL programs using START commands (part 1 of 2)

Program	START command	Alias
BRENAME	START-UDS-BRENAME	BRENAME
BCHECK	START-UDS-BCHECK	BCHECK
BCREATE	START-UDS-BCREATE	BCREATE
BFORMAT	START-UDS-BFORMAT	BFORMAT
BGSIA	START-UDS-BGSIA	BGSIA
BGSSIA	START-UDS-BGSSIA	BGSSIA
BINILOAD	START-UDS-BINILOAD	BINILOAD
BMEND	START-UDS-BMEND	BMEND, START-UDS-REPAIR
BMODTT	START-UDS-BMODTT	BMODTT
BOUTLOAD	START-UDS-BOUTLOAD	BOUTLOAD, START-UDS-OUTLOAD
BPGSIZE	START-UDS-BPGSIZE	BPGSIZE START-UDS-PAGE-RESIZING
BPRECORD	START-UDS-BPRECORD	BPRECORD
BPRIVACY	START-UDS-BPRIVACY	BPRIVACY START-UDS-AUTHORIZATION
BPSIA	START-UDS-BPSIA	BPSIA
BPSQLSIA	START-UDS-BPSQLSIA	BPSQLSIA, START-UDS-PRINT-SQLSIA
BREORG	START-UDS-BREORG	BREORG, START-UDS-REORGANIZATION
BSTATUS	START-UDS-BSTATUS	BSTATUS
DDL	START-UDS-DDL	DDL
DMLTEST	START-UDS-DMLTEST	DMLTEST
SSL	START-UDS-SSL	SSL
ONLINE-PRIVACY	START-UDS-ONLINE-PRIVACY	OPRIVACY
UDSADM	START-UDS-ADM	UDSADM, START-UDS-ADMINISTRATION
UDSMON	START-UDS-UDSMON	UDSMON
UDS online utility	START-UDS-ONLINE-UTILITY	ONLUTIL

Table 47: Calling UDS/SQL programs using START commands (part 2 of 2)

11.2 BALTER

BALTER is used in a restructuring cycle initiated by BCHANGE (cf. the "Creation and Restructuring“ manual, section 6.11) as well as in a renaming cycle initiated by BRENAME (cf. the "Creation and Restructuring“ manual, section 7.8).

Statements in a restructuring cycle

(cf. the "Creation and Restructuring" manual, section 6.11.4)

Statement	Default value	Meaning
[SORTCORE IS *nnn*.]	150	Specifies size of sort area
EXECUTION IS {YES / NO}.	-	Starts/does not start restructuring phase
REPORT IS {YES / NO}.	-	Requests/suppresses logging
[FILLING IS *nnn* PERCENT [IN SET NAME IS {*setname*,... / *ALL[EXCEPT *setname*,..]}].]	-	Specifies table occupancy level (Format 1)
[FILLING WITH POPULATION [IN SET NAME IS {*setname*,... / *ALL[EXCEPT *setname*,..]}].]	-	Specifies table occupancy level (Format 2)
END.	-	Terminates entry of statement

Table 48: Statements for BALTER in a restructuring cycle

Statements in a renaming cycle

(cf. the "Creation and Restructuring" manual, section 7.8)

Statement	Default value	Meaning
REPORT IS { YES / NO }.	-	Requests/suppresses logging
END.	-	Terminates entry of statement

Table 49: Statements for BALTER in a renaming cycle

Command sequence to start BALTER in a restructuring phase

(cf. the "Creation and Restructuring" manual, section 6.11.5)

Analysis phase

```
01   /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR

02   /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

03   /START-UDS-BALTER

04   EXECUTION IS NO.

05   REPORT IS YES.

06   END.
```

Restructuring phase

```
01   /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR

02   [/CREATE-FILE FILE-NAME=work-file-1 ...
     /ADD-FILE-LINK LINK-NAME=SCRTCH1,FILE-NAME=work-file-1
         ,ACCESS-METHOD=*UPAM]

03   [/CREATE-FILE FILE-NAME=work-file-2 ...
     /ADD-FILE-LINK LINK-NAME=SORTWK,FILE-NAME=work-file-2
         ,ACCESS-METHOD=*UPAM]

04   /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

05   /START-UDS-BALTER

06   [SORTCORE IS nnn.]

07   [FILLING IS nnn PERCENT
                              { setname,...                 }
            [ IN SET NAME IS  {                             }].]
                              { *ALL [EXCEPT setname,...]   }

08   EXECUTION IS YES.

                 { YES }
09   REPORT IS   {     }.
                 { NO  }

10   END
```

Creating the working files for restructuring

(cf. the "Creation and Restructuring" manual, section 6.11.3)

- SCRTCH1

  The data population for buffering can be calculated approximately using the following formula:

  max(*key-length* x *no.-of-records*) x 3 Bytes

- SORTWK

  The sort data population can be calculated approximately using the following formula:

  max(*rec-length* x *no.-of-records*) Bytes

If you do not set up the two work files yourself, BALTER creates them with the following names and sizes:

UTI.*tsn*.SCRTCH1 (360,360)

UTI.*tsn*.SORTWK (120,120)

11.3 BCALLSI

(cf. the "Creation and Restructuring" manual, section 3.5)

Statements

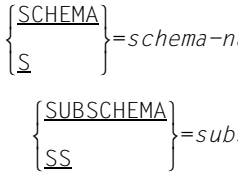
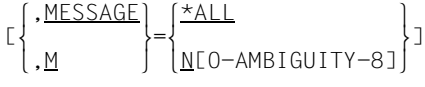
Statement	Default value	Meaning
	-	Mandatory; Assigns the name of the schema and subschema to BCALLSI: *schema-name* Name of the schema assigned in the Schema DDL. *subschema-name* Name of the subschema assigned in the Subschema DDL.
	*ALL	*ALL All cases of ambiguity, including those in the first 8 characters, are output individually to SYSLST. NO-AMBIGUITY-8 Cases of ambiguity in the first 8 characters of a name, are not output individually to SYSLST.

Table 50: Statements for BCALLSI

Command sequence for starting BCALLSI

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-BCALLSI
04  bcallsi statement
```

Entering the SSITAB module in the module library

```
05  /START-LMS
06  //OPEN-LIB LIB=modlib,MODE=*UPDATE
07  //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)
08  //END
```

04 There is no END statement for BCALLSI!

11.4 BCHANGE

(cf. the "Creation and Restructuring" manual, section 6.7)

Command sequence for starting BCHANGE

The BCHANGE utility routine is started by the following commands in the identification under which the database is cataloged:

```
01  [/CREATE-FILE FILE-NAME=dbname.DBCOM.0 ...]
02  [/CREATE-FILE FILE-NAME=dbname.COSSD.0 ...]
03  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05  /START-UDS-BCHANGE
```

01,02 cf. "Setting up the compiler database" in the "Creation and Restructuring" manual, section 3.1.1.

There are no BCHANGE statements!

11.5 BCHECK

(cf. the "Recovery, Information and Reorganization" manual, chapter 3)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 3.4)

Statement	Default value	Meaning
[SORTCORE IS *n*]	150	Define size of sort buffer
CHECK[{[GENERATE]SORTING / SUMMING}] [AGAINST COPY NAME IS *copy-name*]	SUMMING	Select checking mode and define extent of checking
[TYPE IS {ALL [EXCEPT *type-no-1*][,*type-no-2*]...] / 0 / *type-no-1*][,*type-no-2*]...}]	ALL	Select criteria for the global consistency check
SCHEMA NAME IS *schema-name*	-	Identify schema
REALM NAME IS{ALL[EXCEPT *realm-name-1*,...] / *realm-name-2*,...}	-	Specify realms to be checked
[RECORD NAME IS {ALL[EXCEPT *record-name-1*,...] / *record-name-2*,...} [{WITH / WITHOUT} {LOCATION CHECK / KEYVALUE CHECK}]]	-	Specify record types to be checked

Table 51: Statements for BCHECK (part 1 of 2)

Statement	Default value	Meaning
[SET NAME IS {ALL[EXCEPT *setname-1*,...]} {*setname-2*,...} [{WITH}{WITHOUT} INDEX CHECK]]	WITHOUT INDEX CHECK	Specify sets to be checked
[KEY REF IS {ALL[EXCEPT *keyref-1*,...]} {*keyref-2*,...} [{WITH}{WITHOUT} INDEX CHECK]]	WITHOUT INDEX CHECK	Specify SEARCH keys to be checked

Table 51: Statements for BCHECK (part 2 of 2)

Command sequence to start BCHECK

(cf. the "Recovery, Information and Reorganization" manual, section 3.5)

Depending on the check run the following commands are required to start BCHECK:

```
01  [/CREATE-FILE FILE-NAME=work-file-1 ...
     /ADD-FILE-LINK LINK-NAME=SCRTCH1,FILE-NAME=work-file-1,
                    ACCESS-METHOD=*SAM ]

02  [/CREATE-FILE FILE-NAME=arbeitsdatei-2 ...
     /ADD-FILE-LINK LINK-NAME=SORTWK,FILE-NAME=work-file-2,
                    ACCESS-METHOD=*UPAM ]

03   /ADD-FILE-LINK LINK-NAME=DATABASE,
                    FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]

04  [/ADD-FILE-LINK LINK-NAME=BCHECK,
                  FILE-NAME=[:catid:]UTI.tsn.time-stamp.BCHECK ]

05  [/ASSIGN-SYSLST TO=dateiname]

06   /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

07   /START-UDS-BCHECK

08   bcheck-statements

09   END

10  [/ASSIGN-SYSLST TO=*PRIMARY]
```

Creating the work files

(cf. the "Recovery, Information and Reorganization" manual, section 3.2)

- SCRTCH1

  The data population for buffering can be calculated using the following formula:

  number-of-pages x 16 Bytes

- SORTWK

  In the case of an overall check, the data population for sorting can be calculated using the following two formulae:

  – RSQ check formula:

    26 x *number-of-check-records* Bytes

  – Index value check and key value check formula:

    key-length x *number-of-check-records* Bytes

11.6 BCREATE

(cf. the "Creation and Restructuring" manual, section 3.2.1)

Command sequence to start BCREATE

```
01  /CREATE-FILE FILE-NAME=dbname.DBDIR ...
02  /CREATE-FILE FILE-NAME=dbname.DBCOM ...
03  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05  /START-UDS-BCREATE
06  [DATABASE-PAGE-LENGTH IS {2/4/8}KB]
07  END
```

01/02 cf. "Setting Up DBDIR and DBCOM" in the "Creation and Restructuring" manual, section 3.1.1.

11.7 BFORMAT

(cf. the "Creation and Restructuring" manual, section 3.3)

Statements

The BFORMAT statement REALM identifies the realms which are to be formatted. Realms can be formatted in several BFORMAT runs, but each realm can only be formatted once.

Database creation cannot be continued until all realms have been formatted.

Statement	Default value	Meaning
REALM NAME IS {ALL [EXCEPT] [*realm-name1*,..] {*realm-name1*,.. }	ALL	Optional; The specified realms are/are not to be formatted ALL all realms defined in the Schema DDL are to be formatted ALL EXCEPT *realm-name* exclusion list, i.e. all realms other than those specified are to be formatted *realm-name* specifies a user realm
END	-	Mandatory; Terminates statement input

Table 52: Statements for BFORMAT

It is advisable to format realms one at a time. If a BFORMAT run formatting more than one realm aborts without the normal termination procedures owing to an operating system failure, realms that have already been successfully formatted will also be affected. The BFORMAT run will then have to be repeated for them.

The BFORMAT run executes very quickly, since it only formats hash areas and FPA and DBTT pages.

Command sequence to start BFORMAT

```
01   /CREATE-FILE FILE-NAME=dbname.realm-name ...
02 /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04 /START-UDS-BFORMAT
05  [bformat-statement]
06   END
```

01 cf. the "Creation and Restructuring" manual, section 3.1.2.

05 If the REALM statement is omitted, all realms are formatted

11.8 BGSIA

(cf. the "Creation and Restructuring" manual, section 3.2.4)

Statements

Statement	Default value	Meaning
GENERATE SCHEMA *schema-name*	-	Mandatory; Checks and generates the SIA. *schema-name* name of schema as specified in Schema DDL
RENAME { AREA | RECORD | SET } {'*name-old*' TO '*name-new*'} [, ...] .		May only be specified in the renaming cycle; changes the names of record types, sets and user realms *name-old*: name which is to be changed name-new new name The renaming of and changes to items in record types cannot be specified here.
DISPLAY [SCHEMA *schema-name*]	-	Optional; Prints the SIA generated by BGSIA *schema-name* name of schema as specified in GENERATE statement It is sufficient to specify DISPLAY.
END	-	mandatory; terminates statement input

Table 53: Statements for BGSIA

Command sequence for starting BGSIA

```
01   /DELETE-SYSTEM-FILE FILE-NAME=OMF
02  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
03  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04  /START-UDS-BGSIA
05   bgsia-statements
06   END
```

Entering the UDSHASH module in the HASHLIB

```
01  /START-LMS
02  //OPEN-LIB LIB=dbname.HASHLIB,MODE=*UPDATE(STATE=*NEW)
03  //ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=*LIBRARY-ELEMENT(TYPE=R)
04  //END
```

11.9 BGSSIA

(cf. the "Creation and Restructuring" manual, section 3.4.2)

Statements

Statement	Default value	Meaning
GENERATE SUBSCHEMA *subschema-name* OF SCHEMA *schema-name*	-	Optional; *subschema-name*: name of the subschema *schema-name*: name of the schema Checks whether an SSIA is available for a specific subschema and generates – an SSIA with information on realms, record types and sets – lists of individual items – lists of all names contained in the subschema.
DELETE SUBSCHEMA *subschema-name* OF SCHEMA *schema-name*	-	Optional; Deletes a previously generated SSIA from the DBDIR.
REGENERATE SUBSCHEMA *subschema-name* OF SCHEMA *schema-name*	-	Optional; Deletes the old SSIA and generates a new SSIA (combines DELETE and GENERATE functions). Suitable for correction of a subschema.
DISPLAY[SUBSCHEMA *subschema-name* OF SCHEMA *schema-name*]	-	Optional; Can only be used in conjunction with the GENERATE or REGENERATE statement. To print out SSIA, DISPLAY by itself is sufficient.
END	-	Mandatory; Terminates entry of statements.

Table 54: Statements for BGSSIA

Command sequence to start BGSSIA

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03  /START-UDS-BGSSIA
04   bgssia-statements
05   END
```

11.10 BINILOAD

(cf. the "Creation and Restructuring" manual, section 5.1)

Statements

(cf. the "Creation and Restructuring" manual, section 5.1.4)

Control statements

control execution of the UDS/SQL utility routine BINILOAD.

Statement	Default value	Meaning
[EXECUTION {WITH | WITHOUT} CHECK.]	WITH	Checks/does not check input data
[SORTCORE IS *nnn*.]	150	Specifies size of main memory for sort/merge routine

Table 55: Control statements for BINILOAD

Program statements

determine the schema, subschema, input file, and the occupancy level of tables.

Statement	Default value	Meaning
SCHEMA NAME IS *schema-name*. SUBSCHEMA NAME IS *subschema-name*.	-	Name of schema and subschema
FILLING IS *nnn* PERCENT.	-	Specifies occupancy level for table pages
USER FILE RECORD LENGTH IS *n*.	-	Length of input records in bytes
USER FILE BUFFER LENGTH IS *n*.	-	Block length of input file; must be a multiple of 2048
INPUT FILE NAME '*file-name*'.	-	File name of input file
INPUT RECORDNUMBER IS n.	-	None, only tolerated for reasons of compatibility

Table 56: Program statements for BINILOAD

STORE statements

provide BINILOAD with information on the record type and its relation to the input records.

Statement	Default value	Meaning
STORE RECORD NAME IS *record-name*.	-	Record type to be stored
RECORD-DBKEY IS DISPL IS *n*,LENGTH IS {4 | 8}.	-	Assigns database key value; – displacement and length of the database key value
RECORD-RSQ IS DISPL IS *n*,LENGTH IS {3 | 6}.	-	Assigns database key value; – displacement and length of the record sequence number (RSQ) The associated record reference number (REC-REF) is determined by BINILOAD.
RECORD-DISPL IS *n*, {DISPL IS *n*,LENGTH IS *n* | VALUE IS '*literal*'}.	-	Structures database record. For specified record type; – displacement and length of the items of this record – character string to be inserted in the database records
RECORD-AREA NAME IS *realm-name*.	-	Realm into which the records are to be loaded.

Table 57: STORE statements for BINILOAD

INSERT statements

indicate to BINILOAD the sets into which the records are to be inserted.

Statement	Default value	Meaning
INSERT INTO SET NAME IS *set-name*.	-	Specifies the set into which the records are to be inserted as member records.
SET ORDER {USING DISPL IS *n*,LENGTH IS *n* VIA USER FILE SEQUENCE}.	VIA USER FILE SEQUENCE	Specifies sort sequence of records within the sets with ORDER IS FIRST, LAST, NEXT, PRIOR, IMMATERIAL; specifies length of sort item.
OWNER CALCKEY IS {DISPL IS *n*, LENGTH IS *n* VALUE IS '*literal*'}, AREA NAME IS *realm-name*.	-	Selects set occurrence by selecting the owner – displacement and length of the CALC key values in the input file records by means of which the owner is to be selected – character string with CALC key – name of the realm in which the owner record is stored.
OWNER SEARCHKEY IS {DISPL IS *n*, LENGTH IS *n* VALUE IS '*literal*'}, [VIA SET NAME IS *set-name*,] SEARCHKEY TABLE {COLUMN-NR IS *n* ORDER-NR IS *keyref*}.	-	Selects set occurrence by selecting the owner via SEARCH key – displacement and length of the SEARCH key values in the input file records by means of which the owner is to be selected – character string with SEARCH key table – name of the SYSTEM set in which the owner is a member – DBTT column number of SEARCH key table – key reference number.

Table 58: INSERT statements for BINILOAD (part 1 of 2)

Statement	Default value	Meaning
OWNER DBKEY IS { DISPL IS *n*, LENGTH IS { 4 | 8 } . { VALUE IS *dbkey*	-	Selects set occurrence by selecting the owner via its data base key value: – displacement and length of the database key value in the input file records by means of which the owner is to be selected – character string with data base key value.
OWNER RSQ IS { DISPL IS *n*, LENGTH IS { 3 | 6 } . { VALUE IS *rsq*	-	Selects set occurrence by selecting the owner via its data base key value: – displacement and length of the record sequence number (RSQ) in the input file records by means of which the owner is to be selected – character string with record sequence number (RSQ). The associated record reference number (REC-REF) is determined by BINILOAD.
OWNER KEY IS DISPL IS *n*, LENGTH IS 1.	-	Position of the item in the input records, which specifies whether or not the record is to be inserted in the SYSTEM sets.

Table 58: INSERT statements for BINILOAD (part 2 of 2)

Command sequence for starting BINILOAD

(cf. the "Creation and Restructuring" manual, section 5.1.5)

```
01  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
02  [/CREATE-FILE FILE-NAME=eingabebanddatei,...]
03  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
04  /START-UDS-BINILOAD
05  biniload-statements
06  END
```

The BINILOAD statements are read in via SYSDTA! A file generated by BOUTLOAD can also be used for this purpose.

Creating work files

(cf. the "Creation and Restructuring" manual, section 5.1.6)

If you wish to create the work files explicitly, you must issue the appropriate CREATE-FILE commands. If you specify too little storage space (with SPACE), the value you specify will be corrected internally by BINILOAD.

```
/CREATE-FILE FILE-NAME=workfile-n [,SUPPORT ...]

                          ⎧SCRTCH1  ⎫
                          ⎪SCRTCH2  ⎪
                          ⎪SCRTCH3  ⎪
                          ⎪SCDnnnnn ⎪                                    ⎧PAM⎫
/SET-FILE-LINK LINK-NAME=⎨STKnnnnn⎬,FILE-NAME=workfile-n[,ACCESS-METHOD=⎨   ⎬]
                          ⎪KEYnnnnn ⎪                                    ⎩SAM⎭
                          ⎪KSTnnnnn ⎪
                          ⎪SORTWK   ⎪
                          ⎩SRT1WK   ⎭
```

SCRTCH1

(total key lengths + 12) x *number of input records* Bytes

SCRTCH2

12 x *number of input records* Bytes

SCRTCH3

3 x *number of input records* Bytes

SCD*nnnnn*

with 2048-byte page length:

40 x *number of input records* Bytes

with 4000/8096-byte page length:

50 x *number of input records* Bytes

STK*nnnnn*

with 2048-byte page length:

8 + *reclength_1*) x *number of input records* Bytes

with 4000/8096-byte page length:

(12 + *reclength_1*) x *number of input records* Bytes

KEY*nnnnn* and for SEARCH key

with 2048-byte page length:

(16 + *keylength_1*) x *number of input records* Bytes

with 4000/8096-byte page length:

(24 + *keylength_1*) x *number of input records* Bytes

KEY*mmmmm* and KST*nnnnn* for SORT-Key

with 2048-byte page length:

(*keylength_1* + 12 + *keylength_2*) x *number of input records* Bytes

with 4000/8096-byte page length:

(*keylength_1* + 16 + *keylength_2*) x *number of input records* Bytes

SORTWK and SRT1WK

The volume of the data that is to be sorted is calculated using the formula:

(*reclength_2*+SCD+12) x *number of input records* Bytes

11.11 BMEND

(cf. the "Recovery, Information and Reorganization" manual, chapter 2)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 2.2.3)

Statement	Meaning
ADD-REALM	Attach realms to a database
REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realm-name> *ALL-EXCEPT(...) | NAME = list-poss(30): <realm-name>	
ALLOCATE-BUFFER-POOL	Define buffer size
BUFFER-SIZE = *STD / <integer 1..2000>	
DISABLE-ONLINE-COPY	Disable online-copy facility for the database
ENABLE-ONLINE-COPY	Enable online-copy facility for the database
END	Terminate input of statements
KILL-LOG	Stop logging for an inconsistent database
OPEN-DATABASE	Open database
DATABASE-NAME = <dbname> ,COPY-NAME = *NONE / <copy-name> ,USER-IDENTIFICATION = *OWN / <userid>	
REMOVE-REALM	Detach realms
REALM-NAME = *ALL-EXCEPT(...) / list-poss(30): <realm-name> *ALL-EXCEPT(...) | NAME = list-poss(30): <realm-name>	

Table 59: Statements for BMEND (part 1 of 3)

Statement	Meaning
SHOW-LOG-INFORMATION	Show log information
REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realm-name> *ALL-EXCEPT(...) | NAME = list-poss(30): <realm-name> ,LOG-FILE = *STD / NONE / <alog-seq-nr> ,OUTPUT = list-poss: *SYSLST / *SYSOUT	
START-LOG	Start logging for database original
DEFAULT-SUPPORT = *PUBLIC(...) / *UNCHANGED / list-poss(15): *PRIVATE(...) *PUBLIC(...) | CATID = *STD / *OWN / <catid> | ,VOLUME-SET = *STD / <catid> | ,VOLUME = *STD / list-poss(15): <volume> *PRIVATE(...) | VOLUME = list-poss(15): <volume> | ,DEVICE = <device> ,RESERVE-SUPPORT = *PUBLIC(...) / *UNCHANGED / list-poss(15): *PRIVATE(...) *PUBLIC(...) | CATID = *STD / *OWN / <catid> | ,VOLUME-SET = *STD / <catid> | ,VOLUME = *STD / list-poss(15): <volume> *PRIVATE(...) | VOLUME = list-poss(15): <volume> | ,DEVICE = <device> ,SPACE = STD / *RELATIVE(...) / *UNCHANGED *RELATIVE(...) | PRIMARY-ALLOCATION = <integer 192..50331645> | ,SECONDARY-ALLOCATION = <integer 576..32767> ,USER-ACCESS = *OWNER-ONLY / *ALL-USERS ,RESET-LOG-POOL = *NO / *YES	
STOP-LOG	Stop logging for database operations

Table 59: Statements for BMEND (part 2 of 3)

Statement	Meaning
UNDO	Undo a statement
UPDATE-DATABASE	Apply AFIMs to a database
REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realmname> *ALL-EXCEPT(...) | NAME = list-poss(30): <realmname> ,DEADLINE = *STD / BREAK-POINT / <alog-seq-nr> / *TIME-STAMP(...) *TIME-STAMP(...) | DATE = <date> | ,TIME = <time> ,DELETE = *NO / *YES	

Table 59: Statements for BMEND (part 3 of 3)

Command sequence to start BMEND

(cf. the "Recovery, Information and Reorganization" manual, section 2.2.4)

```
01 [/CREATE-JV-LINK JV-NAME=JOBVAR,PROTECTION=*STD]

02 [/SET-JV-LINK LINK-NAME=JVBMEND,JV-NAME=JOBVAR]

03 [/ADD-FILE-LINK LINKNAME=DATABASE,
              FILE-NAME=[:catid:][$userid.]dbname.DBDIR]

04 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

05 /START-UDS-BMEND

06 [//OPEN-DATABASE DATABASE-NAME=dbname
                    [,COPYNAME=*NONE/copyname]
                      [,USER-IDENTIFICATION=*OWN/userid]

07 bmend-statements

08 //END
```

03, 06 You must specify one of the two statements.

11.12 BMODTT

(cf. the "Recovery, Information and Reorganization" manual, chapter 10)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 10.2)

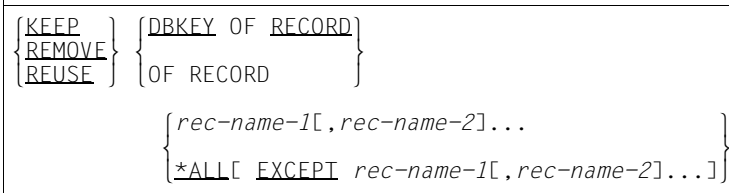
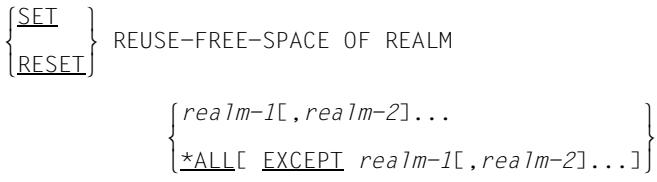
Statement	Meaning
	– KEEP: Lock deallocated database keys – REMOVE: Release locked database keys – REUSE: Deallocate database keys for reuse
	– SET: in the search for free space, the search begins with the first page of the realm – RESET: in the free place search, the search starts with the first free page which is not followed by any partially filled pages up to the end of the realm, but only pages which are still free or full.

Table 60: Statements for BMODTT

Command sequence to start BMODTT

(cf. the "Recovery, Information and Reorganization" manual, section 10.3)

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,
             FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]

02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.SCOPE=*TASK

03 /START-UDS-BMODTT

04 bmodtt-statements

05 END
```

03 BMODTT may only be invoked under the database administrator's user ID.

The BMODTT utility routine is restartable.

11.13 BOUTLOAD

(cf. the "Creation and Restructuring" manual, section 5.2)

Statements

(cf. the "Creation and Restructuring" manual, section 5.2.5)

Statement	Meaning

COPY-RECORD
  RECORD-NAME = *ALL / list-poss(20): <record-name>/
  *ALL-EXCEPT(...)
 ,REALM-NAME = *ALL / <realm-name>
 ,SET-INFORMATION = YES / NO
 ,CSV-OUTPUT = *NO / *YES
	Copy all records of the specified record types to output files

END	Terminate BOUTLOAD

EXPORT-RECORD
  RECORD-NAME = *ALL / list-poss(20): <record-name>/
  *ALL-EXCEPT(...)
 ,SET-INFORMATION = YES / NO
	Unload all records of the specified record types to output files

OPEN-DATABASE DATABASE-NAME = <dbname> ,COPY-NAME = *NONE / <copyname> ,USER-IDENTIFICATION = *OWN / <userid>	Assign the database

REMOVE-RECORD
  RECORD-NAME = *ALL / list-poss(20): <record-name>/
  *ALL-EXCEPT(...)
	Delete all records of the specified record types

Table 61: Statements for BOUTLOAD

Command sequence to start BOUTLOAD

(cf. the "Creation and Restructuring" manual, section 5.2.6)

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR]
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
03 /START-UDS-BOUTLOAD
04 [OPEN-DATABASE DATABASE-NAME=dbname]
05 boutload-statements
06 END
```

01,04 One of the two statements must be used to assign the database.

Preparing the output files

(cf. the "Creation and Restructuring" manual, section 5.2.2)

The individual output files for BOUTLOAD can be created using the following commands:

```
/CREATE-FILE FILE-NAME=dbname.RECnnnnn[.mmmmm] [,SUPPORT= ...]
/ADD-FILE-LINK LINK-NAME=linkname,FILE-NAME=dbname.RECnnnnn[.mmmmm]
   [,BUFFER-LENGTH=xxx][,FILE-SEQUENCE=*NEW]
```

The volume of data for output is calculated as follows:

```
number of records x reclength Bytes
```

The record length is calculated as follows

– for records containing set information in a 2-Kbyte database:

```
reclength = record length as per SIA report - length of system information
            + 4 * (number of non-singular sets in which the record is a
                   member + 1)

            + 1 * (number of singular sets in which the record is a
                   member, except for MANDATORY AUTOMATIC members)
```

– for records containing set information in a 4-Kbyte or 8-Kbyte database:

```
reclength = record length as per SIA report - length of system information
            + 8 * (number of non-singular sets in which the record is a
                   member + 1)

            + 1 * (number of singular sets in which the record is a
                   member, except for MANDATORY AUTOMATIC members)
```

– for records not containing set information:

```
reclength = record length as per SIA report - length of system information
```

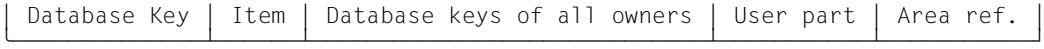
In the case of record types which are distributed to realms, five bytes for the area reference are added to the record length when the records are copied or extracted from multiple realms.

The records are always copied into one output file per record type. An origin from more than one realm is therefore required for the area reference to be specified.

Creating the output record

(cf. the "Creation and Restructuring" manual, section 5.2.2)

If BOUTLOAD has also output the set information on account of the SET-INFORMATION=YES statement, the output record is created with the following structure:



– The record's database key

– A one-byte long item with the content
X'00' = Member inserted
X'FF' = Member not inserted
(for all singular sets in which the record is a member, except for MANDATORY AUTOMATIC members)

– The owners' database keys are not singular sets in which the set is a member

– If the record is not inserted in the set, the owner's database key is set to High Value (X'FFFFFFFF' in the case of a 2-KB database, or X'FFFFFFFFFFFFFFFF' in the case of a 4/8-KB database)

– User part

– The five-byte area reference (realm reference) in the case of record types which are distributed to realms if their records are copied from multiple realms. The records are always copied into one output file per record type. An origin from more than one realm is therefore required for the area reference to be specified.

When BOUTLOAD outputs set information on the individual sets, the length of the database key values is specified in the BOUTLOAD log which contains the statements for a subsequent BINILOAD run (length "4" in the case of a 2-Kbyte database, length "8" in the case of a 4-Kbyte/8-Kbyte database).

Without any set information, the output record consists of the user part only.

11.14 BPGSIZE

(cf. the "Creation and Restructuring" manual, section 8.2)

Statements

(cf. the "Creation and Restructuring" manual, section 8.2.4)

Statement	Function
ALLOCATE-BUFFER-POOL	Define buffer size
BUFFER-SIZE = STD / <integer 1..2000>	
CONVERT-DATABASE	Convert database
REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realm-name> *ALL-EXCEPT(...) | NAME = list-poss(30): <realm-name> ,DATABASE-PAGE-LENGTH = *UNCHANGED / 2KB / 4KB / 8KB ,TABLE-FILLING = *UNCHANGED / *MAXIMUM / <integer 1..100>	
END	Terminate input of statements
OPEN-DATABASE	Open database
DATABASE-NAME = <dbname> ,COPY-NAME = *NONE / <copyname> ,USER-IDENTIFICATION = *OWN / <userid>	
UNDO	Cancel effect of statement

Table 62: Statements for BPGSIZE

Command sequence to start BPGSIZE

(cf. the "Creation and Restructuring" manual, section 8.2.5)

```
01  [/ADD-FILE-LINK LINK-NAME=DATABASE
          ,FILE-NAME=[:catid:][$userid.].DBDIR.[copyname]]

02  [/CREATE-FILE FILE-NAME=[:catid:][$userid.]dbname.realm-name.NEW
        [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary
         ,SECONDARY-ALLOCATION=576)) or
         ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn, -
           DEVICE-TYPE=device[,SPACE=...])]]

03  [... further CREATE-FILE statements for files of the converted realm]

04  [/CREATE-FILE FILE-NAME= -
      [:catid:][$userid.]UTI.BPGSIZE.dbname.realm-number.recref-number
        [,SUPPORT=*PUBLIC-DISK(SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary
         ,SECONDARY-ALLOCATION=secondary)) or
         ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn, -
           DEVICE-TYPE=device[,SPACE=...])]]

05  [...  further CREATE-FILE statements for work files of BPGSIZE]

06   /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

07   /START-UDS-BPGSIZE

08  [//OPEN-DATABASE DATABASE-NAME = ...]

09   //BPGSIZE-statements

10   //END
```

01,08 You must specify only one of these two statements.

11.15 BPRECORD

(cf. the "Recovery, Information and Reorganization" manual, chapter 7)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 7.3)

Statement	Default value	Meaning
SCHEMA NAME IS *schema-name*.	User schema	Designate the schema
REALM NAME IS *realm-name*.	-	Specify the realm to be printed
PRINT[{WITH | WITHOUT} PAGEINFO][{WITH | WITHOUT} PAGEINDEX] [{WITH | WITHOUT} SCD][DBTT {DEC | HEX | BOTH}];	WITHOUT and DEC	Determine scope of output
DISPLAY [IN CSV [*csv-filename*]] PAGE ZERO;	-	Print Act-Key-0 page
DISPLAY [IN CSV [*csv-filename*]] FPA OF *page-selection*;	-	List FPA entries
DISPLAY [IN CSV [*csv-filename*]] DBTT OF {ALL RECORDS | RECORD *satzname* FOR *rsq-auswahl*};	-	List DBTT entries
DISPLAY [IN CSV [*csv-filename*]] CALC PAGES *page-selection* {ALL [{RECORDS | CALC SEARCHKEYS}] | ONLY {RECORD *record-name* | CALC SEARCHKEY *keyref*} *rsq-selection*};	-	Print CALC pages

Table 63: Statements for BPRECORD (part 1 of 2)

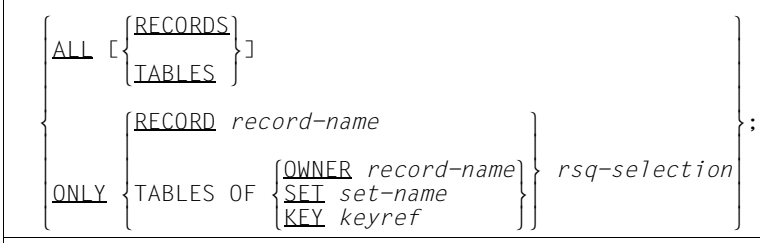
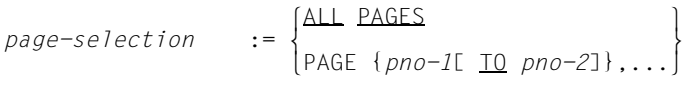
Statement	Default value	Meaning
DISPLAY [IN CSV [*csv-filename*]] DATA PAGES *page-selection* 	-	Print data pages
END		Terminate run

Table 63: Statements for BPRECORD (part 2 of 2)

Physical selection (page selection)



ALL PAGES
: The total collection of pages defined by the logical selection

PAGE *pno-1*,...
: List of page numbers

PAGE {*pno-1* TO *pno-2*},...
: Range from page number *pno-1* to page number *pno-2*, etc.

Logical selection (RSQ selection)

```
rsq-selection     := {ALL RSQS                          }
                     {RSQ {rsq-1[ TO rsq-2]},...        }
```

ALL RSQS
: All record sequence numbers

RSQ *rsq-1*,...
: List of record sequence numbers

RSQ {*rsq-1* TO *rsq-2*},...
: Range of record sequence numbers from *rsq-1* to *rsq-2*, etc.

Command sequence to start BPRECORD

(cf. the "Recovery, Information and Reorganization" manual, section 7.4)

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,
             FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]

02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.SCOPE=*TASK

03 /START-UDS-BPRECORD

04 bprecord-statements

05 END
```

11.16 BPRIVACY

(cf. the "Creation and Restructuring" manual, chapter 4; see “ONLINE-PRIVACY” on page 232 to find out how to specify the access rights when the database is running).

Structure of user group names

(cf. the "Creation and Restructuring" manual, section 4.3)

Configuration	Value			Definition in the ADD-USER-GROUP statement
	host	appl	grp	
openUTM Appl. w/o KSET	*host*	*appl*	-	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*,KSET=*NONE)
openUTM Appl. with KSET	*host*	*appl*	*kset*	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*,KSET=*kset*)
TIAM	*host*	'_'	*id*	*FREE-FORMAT(HOST=*host*,USER-ID=*id*)
linked-in	*host*	'_'	*id*	*FREE-FORMAT(HOST=*host*,USER-ID=*id*)

Table 64: Structure of user group names

host Name of the host computer on which the UDS/SQL-UTM application or the UDS/SQL application program runs.
Here you must specify the name of your processor from the standpoint of DCAM. If no DCAM is available in the TIAM case, you specify HOST=LOCAL.
In non-productive *open*UTM (UTM-T) operation, you specify HOST=UTM.

appl Name of the UTM application

kset KSET name associated with the corresponding UTM user ID

id BS2000 user ID

Statements

(cf. the "Creation and Restructuring" manual, section 4.7)

Statement	Meaning

ADD-USER-GROUP
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OBJECT = NONE / list-poss(6): *REALM(...) /
  *RECORD(...) / *SET(...)
	Defines a user group, possibly with access rights

END	Terminates command input

GRANT-ACCESS
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OBJECT = list-poss(6): *REALM(...) / *RECORD(...) /
  *SET(...)
	Assigns access rights to a user group

OPEN-DATABASE DATABASE-NAME = <dbname>	Opens a database

REMOVE-USER-GROUP
  USER-GROUP-NAME = ALL / *ALL-EXCEPT(...) /
  list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
	Deletes one or more user group(s)

REVOKE-ACCESS
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
  ,OBJECT = list-poss(6): *REALM(...) / *RECORD(...) /
  *SET(...)
	Withdraws access rights from a user group

SHOW-USER-GROUP
  USER-GROUP-NAME = ALL / *ALL-EXCEPT(...) /
  list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OUTPUT = list-poss: SYSLST / SYSOUT
	Outputs information on one or more user group(s)

UNDO	Undoes a statement

Table 65: Statements for BPRIVACY

Command sequence for starting BPRIVACY

(cf. the "Creation and Restructuring" manual, section 4.9)

```
01  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
02  [/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR]
03  /START-UDS-BPRIVACY
04  bprivacy-statements
05  END
```

02 If you assign the database using LINK-NAME=DATABASE, the BPRIVACY statement OPEN-DATABASE is not allowed and must not be specified.
If you do not assign the database using LINK-NAME=DATABASE, the BPRIVACY statement OPEN-DATABASE is mandatory, i.e. must be specified.

11.17 BPSIA

(cf. the "Recovery, Information and Reorganization" manual, chapter 4)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 4.2)

Statement	Meaning
[DISPLAY [IN CSV [*csv-filename*]] SCHEMA *schema-name*]	Print a schema
[DISPLAY [IN CSV [*csv-filename*]] SUBSCHEMA *subschema-name*]	Print a subschema
END	Terminate statement input

Table 66: Statements for BPSIA

The two DISPLAY statements are optional; they may be repeated as often as desired.

Command sequence for starting BPSIA

(cf. the "Recovery, Information and Reorganization" manual, section 4.3)

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,
             FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]

02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK

03 /START-UDS-BPSIA

04 bpsia-statements

05 END
```

04 A period is treated as an end criterion. It may be followed by another statement.

11.18 BPSQLSIA

(cf. the "Recovery, Information and Reorganization" manual, chapter 5)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 5.5)

Statement	Meaning
END	Terminate input
OPEN-DATABASE	Open database
DATABASE-NAME = <dbname> ,COPY-NAME = *NONE / <copy-name> ,USER-IDENTIFICATION = *OWN / <userid>	
PRINT-RELATIONAL-SCHEMAINFO	Select subschemas
SUBSCHEMA-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(20): <subschema-name> *ALL-EXCEPT(...) NAME = list-poss(20): <subschema-name>	

Table 67: Statements for BPSQLSIA

Command sequence to start BPSQLSIA

(cf. the "Recovery, Information and Reorganization" manual, section 5.6)

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,
              FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]]
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.SCOPE=*TASK
03 /START-UDS-BPSQLSIA
04 [//OPEN-DATABASE DATABASE-NAME = ...]
05 //PRINT-statements
06 //END
```

01, 04 You must use one of the two assignments for the database.

11.19 BRENAME

(cf. the "Recovery, Information and Reorganization" manual, section 7.4)

Command sequence for starting BRENAME

The BRENAME utility routine is started by the following commands in the identification under which the database is cataloged:

```
01  [/CREATE-FILE FILE-NAME=dbname.DBCOM.0 ...]
02  [/CREATE-FILE FILE-NAME=dbname.COSSD.0 ...]
03  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=dbname.DBDIR
04  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version,SCOPE=*TASK
05  /START-UDS-BRENAME
06  END
```

01,02 See the "Creation and Restructuring".manual, section 3.1.1

The END statement is the only BRENAME statement.

11.20 BREORG

(cf. the "Recovery, Information and Reorganization" manual, chapter 9)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 9.4)

Statement	Meaning
ALLOCATE-BUFFERPOOL	Define buffer size
BUFFER-SIZE = *STD / <integer 1..2000>	
END	Terminate input of statements
MODIFY-REALM-SIZE	Modify realm size
REALM-NAME = <realm-name> ,REALM-SIZE = <integer 1..16777216> / *RELATIVE(...) / *MINIMUM *RELATIVE(...) DIFFERENCE = <integer -16777216..16777216>	
MODIFY-RECORD-POPULATION	Modify record population
RECORD-NAME = <record-name> ,RECORD-POPULATION = <integer 1..2147483647> / *RELATIVE(...) / *MINIMUM *RELATIVE(...) DIFFERENCE = <integer -2147483647..2147483647>	
OPEN-DATABASE	Open database
DATABASE-NAME = <dbname> ,SCHEMA-NAME = *STD / <schema-name> ,USER-IDENTIFICATION = *OWN / <userid>	

Table 68: Statements for BREORG (part 1 of 2)

Statement	Meaning
REORGANIZE-CALC	Reorganize CALC areas
RECORD-NAME = <record-name> ,CALC-RECORD = NONE / list-poss(6): *WITHIN-POPULATION(...) *WITHIN-POPULATION(...) REALM = *ALL / <realm-name> ,POPULATION = *UNCHANGED / <integer 1..2147483647> ,CALC-SEARCHKEY = NONE / list-poss(30): *KEY-POPULATION(...) *KEY-POPULATION(...) KEY-REF = *ALL / <integer 1..65535> ,POPULATION = *STD / *UNCHANGED / <integer 1..2147483647>	
REORGANIZE-POINTERS	Reorganize all probable position pointers (PPP) of a realm
REALM-NAME = <realname>	
REORGANIZE-SET	Reorganize tables and set constructs
SET-NAME = <set-name> ,OWNER-SELECTION = ALL / list-poss(30): <integer 1..2147483647> / *RANGE(...) *RANGE(...) FROM-RSQ = <integer 1..2147483647> ,TO-RSQ = <integer 1..2147483647> ,KEY-SELECTION = *ALL / list-poss(30): <integer 1..32767> ,FILLING = *UNCHANGED / <integer 1..100>	
SPECIFY-SCHEMA	Specify schema
SCHEMA-NAME = *STD / <schema-name>	
SPECIFY-SUBSCHEMA	Specify subschema
SUBSCHEMA-NAME = <subschema-name>	
UNDO	Undo statement

Table 68: Statements for BREORG (part 2 of 2)

The following overview shows which probable position pointers (PPP) and tables can be reorganized using the REORGANIZE SET function (cf. the "Design and Definition" manual).

DDL and SSL statements		Probable Position Pointer (PPP)		Table	
		Explanation	Updating possible	Type	Restructuring possible
MODE IS CHAIN	ORDER IS FIRST/NEXT/PRIOR SORTED	Owner record includes PP of 1st member record in chain [1]	yes	-	-
		Forward chaining of member records with RSQ and PPP [1]	yes		
	ORDER IS LAST or LINKED TO PRIOR	Owner record includes PPP of last member record in chain	yes		
	LINKED TO PRIOR	Backward chaining of member records with RSQ and PPP	yes		
	ORDER IS SORTED INDEXED BY DEFINED KEYS... ASC/DESC KEY IS	Every table entry includes PPP of member record	yes	multi-level sort key table	yes
	ORDER IS SORTED INDEXED BY DATABASE-KEY	Every table entry includes PPP of member record	yes	multi-level sort key table	yes
MODE IS POINTER-ARRAY	ORDER IS FIRST/LAST/NEXT/PRIOR	Every table entry includes PPP of member record	yes	single-level pointer array	yes
	ORDER IS SORTED INDEXED BY DEFINED KEYS... ASC/DESC KEY IS...	Every table entry includes PPP of member record	yes	multi-level pointer array	yes
	ORDER IS SORTED INDEXED BY DATABASE-KEY or ORDER IS IMMATERIAL	Every table entry includes PPP of member record	yes	multi-level pointer array	yes

Table 69: Overview of options in the REORGANIZE SET function (part 1 of 2)

DDL and SSL statements		Probable Position Pointer (PPP)		Table	
		Explanation	**Updating possible**	**Type**	**Restructuring possible**
MODE IS LIST	ORDER IS FIRST/LAST/ NEXT/PRIOR	PPP not included	-	single-level list	yes
	ORDER IS SORTED INDEXED (DB key or ASC/DESC key)	PPP not included	-	multi-level list	yes
SEARCH KEY ..USING INDEX	TYPE IS REPEATED-KEY	Every table entry includes PPP of member record	yes	multi-level SEARCH key table	yes
	TYPE IS DATABASE-KEY-LIST	PPP not included	-	duplicates table	yes
MEMBER IS PHYSICALLY LINKED TO OWNER		Member record includes pointer to owner record (PPP)	yes	-	-

Table 69: Overview of options in the REORGANIZE SET function (part 2 of 2)

[1] These PPPs are standard with MODE IS CHAIN.

Command sequence to start BREORG

(cf. the "Recovery, Information and Reorganization" manual, section 9.5)

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,
              FILE-NAME=[:catid:][$userid.]dbname.DBDIR)]

02 [/CREATE-FILE FILE-NAME=arbeitsdatei-1[,SUPPORT=*PUBLIC-DISK
              (SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary,
              SECONDARY-ALLOCATION=secondary))/
              ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,
              DEVICE-TYPE=gerät[,SPACE=...])]
   /ADD-FILE-LINK LINK-NAME=SCRTCH1,FILE-NAME=work-file-1,
              ACCESS-METHOD=*UPAM]

03 [/CREATE-FILE FILE-NAME=arbeitsdatei-2[,SUPPORT=*PUBLIC-DISK
              (SPACE=*RELATIVE(PRIMARY-ALLOCATION=primary,
              SECONDARY-ALLOCATION=secondary))/
              ,SUPPORT=*PRIVATE-DISK(VOLUME=priv-vsn,
              DEVICE-TYPE=device[,SPACE=...])]
   /ADD-FILE-LINK LINK-NAME=SORTWK,FILE-NAME=work-file-2,
              ACCESS-METHOD=*UPAM]

04 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.SCOPE=*TASK

05 /START-UDS-BREORG

06 [//ALLOCATE-BUFFERPOOL BUFFER-SIZE = ...]

07 [//OPEN-DATABASE DATABASE-NAME = ...]

08 [//SPECIFY-SCHEMA SCHEMA-NAME = ...]

09 [//SPECIFY-SUBSCHEMA SUBSCHEMA-NAME = ...]

10... further breorg-statements

11 //END
```

01, 07 You must specify one of the two statements.

Work files

(cf. the "Recovery, Information and Reorganization" manual, section 9.2)

Work files for the REORGANIZE-CALL and REORGANIZE-SET statements

- Link name SCRTCH1

  The data population for buffering can be calculated using the following formulae

  – for the reorganization of indirect hash areas:

  `(12 + keylength ) * number of entries` Bytes

  – for the reorganization of direct hash areas:

  `8 * number of entries` Bytes

  – for the reorganization of multi-level tables:

  `12 * number of entries` Bytes

- Link name SORTWK

  The data population for sorting can be calculated using the following formulae

  – for the reorganization of indirect hash areas:

  `(12 + key length ) * number of entries` Bytes

  – for the reorganization of direct hash areas:

  `(record length + key length + 7) * number of entries` Bytes

  – for the reorganization of multi-level tables:

  `12 * number of entries` Bytes

If you do not create the two work files yourself, BREORG sets them up automatically with the following names and sizes:

`UTI.tsn.SCRTCH1`	(360,360) for REORGANIZE-SET and for REORGANIZE-CALC
`UTI.tsn.SORTWK`	(120,120)

Work files for the REORGANIZE-POINTERS statement

- File names UTI.BREORG.*dbname*.*xxx*.*yyyyy*

dbname	Name of the database
xxx	Realm number of the specified realm
yyyyy	Number of the record type whose probable position pointers (PPP) are updated in the realm; yyyyy=0 is used for probable position pointers (PPP) in SYSTEM sets if required.

  The data population for buffering can be calculated using the following formula

  ```
  number of ppps * 11 Bytes
  ```

- File names UTI.BREORG.*dbname*.*xxx*.00001

dbname	Name of the database
xxx	Realm number of the specified realm

  The user schema does not contain any record type with record type number 1. All the updated probable position pointers (PPP), sorted by their position in the realm, are stored in the work file with record type number 1. The size of this file therefore depends on the total number of required individual files UTI.BREORG.*dbname*.xxx.yyyyy (yyyyy=0 bzw. yyyyy>1).

- File link name: SRT1WK

  Approximate size: maximum size of all files UTI.BREORG.*dbname*.*xxx*.00001

11.21 BSTATUS

(cf. the "Recovery, Information and Reorganization" manual, chapter 6)

Statements

(cf. the "Recovery, Information and Reorganization" manual, section 6.3)

Statement	Meaning
SUBSCHEMA IS *subschema-name*	Designate the subschema
DISPLAY [IN CSV [*csv-filename*]] REALM STATISTICS FOR { *realm-name-1*,... | ALL }	Print realm statistics
DISPLAY [IN CSV [*csv-filename*]] TABLE STATISTICS FOR SET { *set-name-1*,... | *ALL[EXCEPT *set-name-1*,...] }	Print set statistics
DISPLAY [IN CSV [*csv-filename*]] TABLE STATISTICS FOR OWNER IN SET { *set-name-1 [rsq-selection-1]*,... | *ALL[EXCEPT *set-name-1*,...] }	Print owner statistics
DISPLAY [IN CSV [*csv-filename*]] RECORD STATISTICS FOR { *record-name-1*,... | ALL }	Print record type statistics
DISPLAY [IN CSV [*csv-filename*]] CALC KEY STATISTICS FOR { RECORD { *record-name-1*,... | ALL } | SEARCHKEY { *keyref-1*,... | ALL } } IN REALM { *realmname-1*,... | ALL }	Print CALC key statistics

Table 70: Statements for BSTATUS (part 1 of 2)

Statement	Meaning
DISPLAY [IN CSV [*csv-filename*]] RECORDNUMBER STATISTICS FOR { REALM { *realm-name-1*,... | ALL } | RECORD { *record-name-1*,... | ALL } }	Print record number statistics
END	Terminate statement input

Table 70: Statements for BSTATUS (part 2 of 2)

All DISPLAY statements are optional. They can be specified in any sequence as often as desired.

Every BSTATUS statement may be terminated with a period (.).

Command sequence to start BSTATUS

(cf. the "Recovery, Information and Reorganization" manual, section 6.4)

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,
             FILE-NAME=[:catid:][$userid.]dbname.DBDIR[.copyname]

02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.SCOPE=*TASK

03 /START-UDS-BSTATUS

04 SUBSCHEMA IS subschema-name

05 display-statements

06 END
```

Setting up work files

(cf. the "Recovery, Information and Reorganization" manual, section 6.2)

- SCRTCH1

  The data population for buffering can be calculated using the following formula:

  ```
  132 * (no. of sets + no. of keys) Bytes
  ```

  The primary allocation for work file 1 should be based on the data population that is to be buffered. There should always be an appropriate secondary allocation in case the storage space proves to be insufficient.

- SORTWK

  The data population for sorting can be calculated using the formula:

  ```
  16 * no. of sort records Bytes
  ```

If the two work files are not created explicitly, BSTATUS generates them automatically with the following names and sizes:

UTI.SAMWORK.*tsn.time-stamp.nnnn*	(33,33)
UTI.*tsn*.SORTWK	(120,120)

If execution terminates normally, any work files created by BSTATUS and their file link names are deleted. Any work files that you have set up explicitly are not deleted and their file link names are not released.

11.22 BTRANS24

(See the "Creation and Restructuring“ manual, chapter 9)

Statements

(See the "Creation and Restructuring“ manual, section 9.3):

Statement	Meaning
CHECK-DATABASE	Start check run
TRANSFORM-DATABASE	Transform database
END	End statement input and start execution

Table 71: Statements for BTRANS24

Command sequence for starting BTRANS24

(See the "Creation and Restructuring“ manual, section 9.4)

```
/START-EXECUTABLE-PROGRAM FROM-FILE=(LIB=UDS/SQL-T-modlib,
                                     ELEM=BTRANS24)

//BTRANS24 statements ...

//END
```

BTRANS24 is a component part of the UDS-SQL-T package and is by default contained in the SIPPRG.UDS-SQL-T.028 library.

11.23 ONLINE-PRIVACY

(See the "Creation and Restructuring“ manual, chapter 4; see “BPRIVACY” on page 215 to find out how to specify the access rights in the offline mode).

Structure of the user group specifications

(See the "Creation and Restructuring“ manual, section 4.3)

UDS/SQL V2.3 requires *open*UTM V4.0 or higher.

Configuration	Value			Definition in the ADD-USER-GROUP statement
	host	appl	grp	
*open*UTM App. without KSET	*host*	*appl*	*kset*	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*,KSET=*NONE)
*open*UTM App. without KSET	*host*	*appl*	*kset*	*KSET-FORMAT(HOST=*host*,APPLICATION=*appl*,KSET=*kset*)
TIAM	*host*	' _'	*UID*	*FREE-FORMAT(HOST=*host*,USER-ID=*user ID*)
linked-in	*host*	' _'	*UID*	*FREE-FORMAT(HOST=*host*,USER-ID=*user ID*)

Table 72: Structure of the user group specifications

host Name of the host processor computer on which the UDS/SQL-UTM application or the UDS/SQL user program is running.
You must specify the standard name of your own processor from the DCAM point of view. If there is no DCAM available in the TIAM case, enter HOST=LOCAL.

appl Name of the UTM application

kset KSET name that is assigned to the corresponding UTM application

UID BS2000 user ID

Statements

(See the "Creation and Restructuring“ manual, section 4.7)

Statement	Meaning

ADD-USER-GROUP
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OBJECT = NONE / list-poss(6): *REALM(...) /
  *RECORD(...) / *SET(...)
	Define user group, possibly with access rights

END	Terminate command input

GRANT-ACCESS
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OBJECT = list-poss(6): *REALM(...) / *RECORD(...) /
  *SET(...)
	Assign access rights to a user group

OPEN-DATABASE DATABASE-NAME = <dbname>	Open a database

REMOVE-USER-GROUP
  USER-GROUP-NAME = ALL / *ALL-EXCEPT(...) /
  list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
	Delete one or more user group(s)

REVOKE-ACCESS
  USER-GROUP-NAME = list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
  ,OBJECT = list-poss(6): *REALM(...) / *RECORD(...) /
  *SET(...)
	Revoke access rights for a user group

SHOW-USER-GROUP
  USER-GROUP-NAME = ALL / *ALL-EXCEPT(...) /
  list-poss(6): *KSET-FORMAT(...) /
  *FREE-FORMAT(...)
 	
  ,OUTPUT = list-poss: SYSLST / SYSOUT
	Output information on one or more user group(s)

UNDO	Undo the statement

Table 73: Statements for ONLINE-PRIVACY

Command sequence for starting ONLINE-PRIVACY

(See the "Creation and Restructuring“ manual, section 4.8)

```
01  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version
02  /ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=configuration_name
03  /START-UDS-ONLINE-PRIVACY
04  OPEN-DATABASE DATABASE-NAME=dbname
05  other online-privacy statements
06  END
```

02 You assign the configuration name FILE-NAME=*configurationname* via the link name DATABASE with the SET-FILE-LINK command.
The UDS/SQL configuration that is to work with ONLINE-PRIVACY and that the database to be processed is attached to must be made known to the system using this command.

11.24 UDS online utility

(See the "Recovery, Information and Reorganization" manual, chapter 8)

Statements

(See the "Recovery, Information and Reorganization" manual, section 8.6, 8.7, 8.8)

SDF statements of the UDS online utility:

Statement	Meaning
DECLARE-PROCEDURE	Open procedure declaration
PROCEDURE-NAME = <structured-name 1..20> ,CODE = <c-string 1..1800> / <filename> / *SYSDTA	
DECLARE-VARIABLE	Define variable
VARIABLE-NAME = <structured-name 1..20> ,TYPE = *STRING(...) / *INTEGER *STRING LENGTH = <integer 1..20> ,INITIAL-VALUE = <integer 1..16777215> / <c-string> / *STD / *NONE	
DELETE-PROCEDURE	Delete procedure
PROCEDURE-NAME = <structured-name 1..20>	
DELETE-VARIABLE	Delete variable
VARIABLE-NAME = <structured-name 1..20>	
END	Terminate UDS online utility
REPEAT-PROCEDURE	Execute procedure
PROCEDURE-NAME = *STDRELOC / *STDFPASCAN / *STDREPPP / <structured-name 1..20> ,CYCLE-LIMIT = <integer 1..16777215> / *MAX	
SET-FPA-SCAN-PARAMETERS	Define search mode for free place search
SUBSCHEMA-NAME = <structured-name 1..30> ,REALM-NAME = <structured-name 1..30> ,SEARCH-MODE = *REUSE / *NOREUSE	
SET-ONLINE-UTILITY-PARAMETERS	Define online utility parameters
DBH = *INDEPENDENT / *LINKED-IN ,CONFIGURATION-NAME = <structured-name 1..17>	

Table 74: SDF statements for the UDS online utility (part 1 of 3)

Statement	Meaning
SET-PREF-REALM-PARAMETERS SUBSCHEMA-NAME= <structured-name 1..30> ,SET-NAME= <structured-name 1..30> ,PREFERRED-REALM-NAME= <structured-name 1..30>	Set or change the preferred realm for a distributable list
SET-RELOCATE-PARAMETERS SUBSCHEMA-NAME = <structured-name 1..30> ,REALM-NAME = <structured-name 1..30> ,RELOCATE-TYPE = *RECORD-PAGES(...) / *BASE-LEVEL-TABLE-PAGES(...) / *INDEX-LEVEL-TABLE-PAGES(...) / *DISTRIBUTABLE-TABLE-PAGES(...) *RECORD-PAGES INITIALIZE= *ANY / *YES / *NO ,PAGES-PER-DML= <integer 1..16777215> ,SKIP-ABOVE-FILLING= <integer 1..100> ,CLASH-HANDLING= *BREAK-DML / *SKIP-PAGE *BASE-LEVEL-TABLE-PAGES INITIALIZE= *ANY / *YES / *NO ,PAGES-PER-DML= <integer 1..16777215> ,CLASH-HANDLING= *BREAK-DML / *SKIP-PAGE *INDEX-LEVEL-TABLE-PAGES INITIALIZE= *ANY / *YES / *NO ,PAGES-PER-DML= <integer 1..16777215> *DISTRIBUTABLE-TABLE-PAGES INITIALIZE= *ANY / *YES / *NO ,PAGES-PER-DML= <integer 1..16777215> ,CLASH-HANDLING= *BREAK-DML / *SKIP-PAGE ,SET-NAME= <structured-name 1..30> ,TARGET-REALM-NAME= <structured-name 1..30>	Define properties of a RELOCATE DML
SET-REORGANIZE-PPP-PARAMETERS SUBSCHEMA-NAME = <structured-name 1..30> ,REALM-NAME = <structured-name 1..30> ,INITIALIZE = *ANY / *YES / *NO ,PAGES-PER-DML = <integer 1..16777215> ,CLASH-HANDLING = *BREAK-DML / *SKIP-PAGE	Define properties of a RORGPPP DML

Table 74: SDF statements for the UDS online utility (part 2 of 3)

Statement	Meaning
SHOW-FPA-SCAN-PARAMETERS	Show parameters which are currently valid for DML FPASCAN
SHOW-PROCEDURE	Show procedure
PROCEDURE-NAME = <structured-name 1..20> / *STDRELOC / *STDFPASCAN / *STDREPPP	
SHOW-PREF-REALM-PARAMETERS	Show parameters which are currently valid for DML PREFRLM
SHOW-RELOCATE-PARAMETERS	Show parameters which are currently valid for DML RELOCATE
SHOW-REORGANIZE-PPP-PARAMETERS	Show parameters which are currently valid for DML REORGPPP
SHOW-VARIABLE	Show current value of a variable
VARIABLE-NAME = <structured-name 1..20>	

Table 74: SDF statements for the UDS online utility (part 3 of 3)

Procedure statements of the UDS online utility:

Statement	Meaning
ADD *name*,{*value* | *variable*}[,*condition*]	Add value to a variable
BREAK *condition*	Terminate procedure sequence immediately
END	Terminate input of procedure statements
EXIT *condition*	Terminate procedure sequence after current cycle
FINISH [WITH CANCEL]	Terminate current transaction
FPASCAN	Define start page for free place search
MOVE *name*,{*value* | *variable*}[,*condition*]	Define value of a variable
PREFRLM	Set or change preferred realm for a distributable list
READY [EXCLUSIVE] UPDATE	Start current transaction of the UDS online utility
RELOCATE	Perform relocation
{REMARK | *}	Insert remark
REORGPPP	Reorganize probable position pointers
SHOW *name*[,*condition*]	Show value of a variable

Table 75: Procedure statements of the UDS online utility

WAIT *n*[,*condition*]	Define wait time

Table 75: Procedure statements of the UDS online utility

Command sequence for starting the UDS online utility

(See the "Recovery, Information and Reorganization" manual, section 8.6)

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=...
/START-UDS-ONLINE-UTILITY ...
```

The UDS online utility must execute in the user ID of the database to be processed. The requests cannot be distributed via UDS-D.

The program name with which the UDS online utility appears in the outputs of DAL and on the UDS monitor is $UDSOUTI.

12 Function codes of the DML statements

(cf. the "Database Operation" manual, chapter 15)

The following table shows the function codes in decimal notation.

Function code	Short function name	Meaning
0	START	Start a user program
1	FINISH	FINISH
2	ERASUN	ERASE *record-name*
3	ERASQU	ERASE *record-name* {PERMANENT/SELECTIVE/ALL}
4	FND1	FIND-1
5	FND5	FIND-5
6	FND4R	FIND-4 without WITHIN or with WITHIN *realm-name*
7	FND4S	FIND-4 WITHIN *set-name*
8	FND6	FIND-6
9	FND2	FIND-2
10	FND7IS	FIND-7 USING *record-element-name,...* (sequential search)
11	FND7IK	FIND-7 USING *record-element-name,...* (search with key)
12	FND3IS	FIND-3 USING *record-element-name,...* (sequential search)
13	FND3IK	FIND-3 USING *record-element-name,...* (search with key)
15	CONNEC	CONNECT
16	MODIFY	MODIFY
17	ACCEPT	ACCEPT
18	READY	READY
19	DISCON	DISCONNECT
20	STORE	STORE
21	STOP	End a user program
22	FREE	FREE
23	KEEP	KEEP

Table 76: Function codes of the DML statements (part 1 of 3)

Function code	Short function name	Meaning
24	IF_	IF
25	FND7SE	FIND-7 without USING or with USING *search-expression*
26	ONLINE	UDS online utility
28	FND3SE	FIND-3 without USING
30	LONAM	LOOK for a name
31	LOREC	LOOK for a record
32	LOSET	LOOK for a set
33	LOITM	LOOK for an item
34	LOKEY	LOOK for a key
36	LORLM	LOOK for a realm
37	RDYGLO	Start a distributed transaction[1]
38	PTC	End initiation phase for distributed transaction[1]
125	RELBAS	RELOCATE RELOCATION-TYPE = *BASE-LEVEL-TABLE-PAGES
126	RELINX	RELOCATE RELOCATION-TYPE = *INDEX-LEVEL-TABLE-PAGES
127	RELDST	RELOCATE RELOCATION-TYPE = *DISTRIBUTABLE-TABLE-PAGES
128	RELREC	RELOCATE RELOCATION-TYPE = *RECORD-PAGES
129	FPASCA	FPASCAN
130	PRERLM	PREFRLM
131	REOPPP	REORGPPP
132	FTC1	FETCH-1
133	FTC5	FETCH-5
134	FTC4R	FETCH-4 without WITHIN or with WITHIN *realmname*
135	FTC4S	FETCH-4 with WITHIN *setname*
136	FTC6	FETCH-6
137	FTC2	FETCH-2
138	FTC7IS	FETCH-7 USING *record-element-name,...* (sequential search)
139	FTC7IK	FETCH-7 USING *record-element-name,...* (search with key)
140	FTC3IS	FETCH-3 USING *record-element-name,...* (sequential search)
141	FTC3IK	FETCH-3 USING *record-element-name,...* (search with key)

Table 76: Function codes of the DML statements (part 2 of 3)

Function code	Short function name	Meaning
142	GET	GET
153	FTC7SE	FETCH-7 without USING or with USING *search-expression*
156	FTC3SE	FETCH-3 without USING
158	NLONAM	next LOOK for a name
160	NLOSET	next LOOK for a set
161	NLOITM	next LOOK for an item
162	NLOKEY	next LOOK for a key
164	NLORLM	next LOOK for a realm

Table 76: Function codes of the DML statements (part 3 of 3)

[1] Only in conjunction with UDS-D

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)
Application Programming
User Guide

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 (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