

English



FUJITSU Software BS2000

UDS/SQL V2.8

Recovery, Information and Reorganization

User Guide

Edition March 2016

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

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

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

manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

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

cognitas. Gesellschaft für Technik-Dokumentation mbH

www.cognitas.de

Copyright and Trademarks

Copyright © 2016 Fujitsu Technology Solutions GmbH.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

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

Contents

1	Preface	11
1.1	Structure of the UDS/SQL documentation	11
1.2	Objectives and target groups of this manual	16
1.3	Summary of contents	17
1.4	Changes since the last edition of the manuals	18
1.5	Notational conventions	20
1.5.1	Warnings and notes	20
1.5.2	Non-SDF notational conventions	20
1.5.3	SDF syntax representation	22
1.6	Sample databases	27
2	Updating and reconstructing a database with BMEND	29
2.1	Functions	30
2.2	Statements	31
2.2.1	Rules	31
2.2.2	Permitted functions	32
2.2.3	BMEND statements	33
	Attach realms to a database (ADD-REALM)	34
	Define buffer size (ALLOCATE-BUFFER-POOL)	35
	Disable online backup capability for the database (DISABLE-ONLINE-COPY)	36
	Enable online backup capability for the database (ENABLE-ONLINE-COPY)	37
	Terminate input of statements (END)	38
	Stop logging for an inconsistent database (KILL-LOG)	39
	Open database (OPEN-DATABASE)	40
	Detach realms (REMOVE-REALM)	41
	Show log information (SHOW-LOG-INFORMATION)	42
	Start logging for database original (START-LOG)	48
	Stop logging for database operations (STOP-LOG)	55

	Undo a statement (UNDO)	56
	Apply AFIMs to a database (UPDATE-DATABASE)	57
2.2.4	Command sequence to start BMEND	60
2.3	Supplying the BMEND job variable	61
3	Checking the consistency of a database with BCHECK	67
<hr/>		
3.1	Description of the checking procedure	68
3.1.1	Setting the checking mode	68
3.1.2	Defining the scope of checking	70
3.1.3	Checking for coherence	70
3.2	System environment	71
3.3	Using the results of the summing run in a sort run	78
3.4	Statements for BCHECK	79
	Define size of sort buffer (SORTCORE)	80
	Select checking mode and define extent of checking (CHECK)	81
	Selecting criteria for the global consistency check (TYPE)	82
	Identify schema (SCHEMA NAME)	84
	Specify realms to be checked (REALM NAME)	85
	Specify record types to be checked (RECORD NAME)	86
	Specify sets to be checked (SET NAME)	88
	Specify SEARCH keys to be checked (KEY REF)	90
3.5	Command sequence to start BCHECK	92
3.6	BCHECK examples	93
3.7	Messages	99
3.7.1	Warnings	99
3.7.2	Error messages	100
3.7.3	Execution messages	100
3.7.4	Consistency error messages	106
3.7.4.1	Global consistency errors (without index check)	106
3.7.4.2	Global consistency errors (index check)	116
3.7.4.3	Local consistency errors	119
3.7.5	Usage of job switches	132

4	Printing the schema/subschema information area with BPSIA	133
4.1	System environment	134
4.2	BPSIA statements	135
	Print a schema (DISPLAY SCHEMA)	135
	Print a subschema (DISPLAY SUBSCHEMA)	136
	Terminate statement input (END)	136
4.3	Command sequence for starting BPSIA	137
4.4	Description of the SIA report	138
	SIA PRINT REPORT (general information)	138
	REFERENCE NUMBERS	141
	AREA INFORMATION	142
	RECORD WITHIN LIST	144
	RECORD INFORMATION	145
	DBTT INFORMATION	147
	CALC INFORMATION	149
	SET INFORMATION	151
	KEY INFORMATION (NO CALC SEARCH KEYS)	155
	CALC-SEARCH-KEY INFORMATION	158
4.5	Description of the SSIA report	160
	SSIA PRINT REPORT (general information)	160
	REFERENCE NUMBERS	162
	AREA INFORMATION	163
	RECORD INFORMATION	164
	CALC KEY INFORMATION	166
	ITEM STRING LIST	168
	KEY ITEM LIST	170
	SET INFORMATION	172
	KEY INFORMATION	175
5	Output relational schema information with BPSQLSIA	177
5.1	Overview	177
5.2	System environment	178
5.3	Prerequisites for SQL access to CODASYL definitions	179
5.4	SQL data types	180
	Alphanumeric data type	180
	National data type	180
	Numeric data types	181

	Structured data types	182
5.5	BPSQLSIA statements	185
	Terminate input (END)	185
	Open database (OPEN-DATABASE)	186
	Select subschemas (PRINT-RELATIONAL-SCHEMAINFO)	187
5.6	Command sequence to start BPSQLSIA	188
5.7	Description of the output of BPSQLSIA	189
5.8	Conversion rules	190
5.9	Summary of the SQL access permitted for each base table	192
5.10	Example	193
6	Printing statistics on the occupied storage space with BSTATUS	199
6.1	Functions	200
6.2	System environment	203
	Work files	204
6.3	BSTATUS statements	206
	Designate the subschema (SUBSCHEMA)	207
	Print realm statistics (DISPLAY REALM)	208
	Print set statistics (DISPLAY TABLE FOR SET)	211
	Print owner statistics (DISPLAY TABLE FOR OWNER)	214
	Print record type statistics (DISPLAY RECORD)	217
	Print CALC key statistics (DISPLAY CALC)	220
	Print record number statistics (DISPLAY RECORDNUMBER)	224
	Terminate input (END)	226
6.4	Command sequence to start BSTATUS	227
7	Printing out the contents of realms with BPRECORD	229
7.1	System environment	230
7.2	General description of the output of BPRECORD	232
7.3	BPRECORD statements	235
	Designate the schema (SCHEMA NAME)	237
	Specify the realm to be printed (REALM NAME)	238
	Determine scope of output (PRINT)	239
	Print act-key-0 page (DISPLAY PAGE)	242

	List FPA entries (DISPLAY FPA)	246
	List DBTT entries (DISPLAY DBTT)	247
	Print CALC pages (DISPLAY CALC)	249
	Print data pages (DISPLAY DATA)	254
	Terminate BPRECORD (END)	257
7.4	Command sequence to start BPRECORD	258
8	Executing online services with the UDS online utility	259
8.1	Functions of the UDS online utility	260
8.2	DML RELOCATE - Relocating data pages	262
8.3	DML FPASCAN - Defining the search mode for the free place search	265
8.4	DML PREFRLM - Defining the preferred realm for distributable lists	265
8.5	DML REORGPPP – Reorganizing probable position pointers (PPPs)	266
8.6	Command sequence for starting the UDS online utility	269
8.7	Statements of the UDS online utility	269
8.8	SDF statements of the UDS online utility	270
	Defining a procedure (DECLARE-PROCEDURE)	271
	Defining a variable (DECLARE-VARIABLE)	272
	Deleting a procedure (DELETE-PROCEDURE)	273
	Deleting a variable (DELETE-VARIABLE)	273
	Terminating the UDS online utility (END)	273
	Executing a procedure (REPEAT-PROCEDURE)	274
	Defining the search mode for the free place search (SET-FPA-SCAN-PARAMETERS)	275
	Defining online utility parameters (SET-ONLINE-UTILITY-PARAMETERS)	276
	Defining the preferred realm (SET-PREF-REALM-PARAMETERS)	277
	Defining the properties of a RELOCATE DML (SET-RELOCATE-PARAMETERS)	278
	Defining the properties of a REORGPPP DML (SET-REORGANIZE-PPP-PARAMETERS)	283
	Showing the parameters which are currently valid for DML FPASCAN (SHOW-FPA-SCAN-PARAMETERS)	286
	Showing a procedure (SHOW-PROCEDURE)	286
	Showing the parameters which are currently valid for DML PREFRLM (SHOW-PREF-REALM-PARAMETERS)	287
	Showing the parameters which are currently valid for DML RELOCATE (SHOW-RELOCATE-PARAMETERS)	287

	Showing the parameters which are currently valid for DML REORGPPP (SHOW-REORGANIZE-PPP-PARAMETERS)	287
	Showing the current value of a variable (SHOW-VARIABLE)	288
8.9	Procedure statements of the UDS online utility	289
	Adding a value to a variable (ADD)	290
	Terminating a procedure sequence immediately (BREAK)	291
	Terminating input of procedure statements (END)	291
	Terminating the procedure sequence after the current cycle (EXIT)	291
	Terminating the current transaction (FINISH)	292
	Defining the start page for the free place search (FPASCAN)	292
	Defining the value of a variable (MOVE)	293
	Setting or changing the preferred realm for a distributable list (PREFRLM)	293
	Starting current transactions of the UDS online utility (READY UPDATE)	294
	Performing a relocation (RELOCATE)	294
	Inserting a remark (REMARK)	295
	Performing a PPP reorganization (REORGPPP)	295
	Showing the value of a variable (SHOW)	296
	Defining the wait time (WAIT)	296
8.10	Error handling of the UDS online utility	297
8.11	Predefined variables	298
8.12	Predefined standard procedures	299
8.13	Status codes	299
8.14	Condition syntax	300
8.15	Examples	301
	Example 1: Relocation of a fixed number of pages using the standard procedure	301
	Example 2: Relocation of all possible pages using the standard procedure	301
	Example 3: Preventing conflicts	302
	Example 4: The prevention of conflicts is based on the locks reported	303
	Example 5: Preventing conflicts with waiting	304
	Example 6: Information on the status code	305
	Example 7: Reducing the size of realms	306
	Example 8: Reorganizing PPPs	306
9	Database reorganization with BREORG	307
9.1	Functions	308
9.2	System environment	309
	Work files	311

9.3	Database saving	315
9.4	BREORG statements	316
	Define buffer size (ALLOCATE-BUFFERPOOL)	317
	Terminate input of statements (END)	318
	Modify realm size (MODIFY-REALM-SIZE)	319
	Modify record population (MODIFY-RECORD-POPULATION)	322
	Open database (OPEN-DATABASE)	325
	Reorganize CALC areas (REORGANIZE-CALC)	326
	Reorganize all probable position pointers (PPP) in a realm (REORGANIZE-POINTERS)	333
	Reorganize tables and set constructs (REORGANIZE-SET)	334
	Specify schema (SPECIFY-SCHEMA)	340
	Specify subschema (SPECIFY-SUBSCHEMA)	341
	Undo statement (UNDO)	342
9.5	Command sequence to start BREORG	343
9.6	Examples	345
10	Controlling the reuse of database keys and the free place search with BMODTT	349
10.1	System environment	350
10.2	BMODTT statements	351
10.3	Command sequence to start BMODTT	353
	Glossary	355
	Abbreviations	397
	Related publications	401
	Index	407

1 Preface

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

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

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

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

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

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

1.1 Structure of the UDS/SQL documentation

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

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

Guide through the manuals

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

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

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

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

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

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

Examples

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

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

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

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

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

Manual UDS/SQL Design and Definition

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

Manual UDS/SQL Application Programming

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

Table 1: Guide through the manuals

(part 1 of 3)

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

Manual UDS/SQL Creation and Restructuring

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

Manual UDS/SQL Database Operation

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

Table 1: Guide through the manuals

(part 2 of 3)

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

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

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

Additional Manuals

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

Table 1: Guide through the manuals

(part 3 of 3)

¹ only for COBOL-DML

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

Additional notes on the manuals

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

(see the “[Application Programming](#)” manual, CONNECT)

advises the user to look up CONNECT in the “[Application Programming](#)” manual.

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

UDS/SQL Messages

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

UDS/SQL System Reference Guide

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

SQL for UDS/SQL Language Reference Manual

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

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

1.2 Objectives and target groups of this manual

This manual is intended for the database administrator, i.e. the person responsible for updating and reconstructing databases, reorganizing data, and checking databases for consistency.

The database administrator must be familiar with all the steps involved in creating a database (database design, schema, subschema, and SSL generation) and must know how to write DB application programs.

In addition, the DB administrator should have a comprehensive knowledge of BS2000, be familiar with the UDS/SQL transaction concept and the general security concept of UDS/SQL (see the manual “[Database Operation](#)”), and also be thoroughly acquainted with the files of a UDS/SQL database and the UDS/SQL utility routines (see the manual “[Creation and Restructuring](#)”, Files and realms of a UDS/SQL database).

1.3 Summary of contents

What does this manual contain?

This manual describes all the administrative and operational activities necessary to ensure trouble-free operation of the database. This includes:

- updating and reconstructing the database,
- checking the consistency of the database,
- the output of database information,
- reorganization of the database, and
- controlling the reuse of deallocated database keys.

Illustrative examples are provided to explain these functions.

Readme file

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

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

Information under BS2000

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

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

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

Additional product information

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

1.4 Changes since the last edition of the manuals

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

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

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

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

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

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



General information

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

1.5 Notational conventions

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

1.5.1 Warnings and notes

	Points out particularly important information
 CAUTION!	Warnings

1.5.2 Non-SDF notational conventions

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

Table 3: Notational conventions

(part 1 of 2)

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

Table 3: Notational conventions

(part 2 of 2)

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

1.5.3 SDF syntax representation

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

Table 4: Metasyntax

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

Table 5: Data types

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

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

Table 6: Data type suffixes

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

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

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords. Some keywords begin with *.	OPEN DATABASE COPY-NAME = * <u>NONE</u>
=	The equal sign connects an operand name with the associated operand values.	CONFIGURATION-NAME = <name 1..8>
< >	Angle brackets denote variables whose range of values is described by data types and their suffixes (Tables 5 and 6).	DATABASE = <dbname>
<u>Underscoring</u>	Underscoring denotes the default value of an operand.	SCHEMA-NAME = * <u>STD</u>
/	A slash separates alternative operand values.	CMD = * <u>ALL</u> / <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.	<pre> USER-GROUP-NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host> </pre>
	A vertical bar identifies related operands within structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	<pre> USER-GROUP-NAME = *ALL-EXCEPT(...) *ALL-EXCEPT(...) NAME = *KSET-FORMAT(...) *KSET-FORMAT(...) HOST = <host> ... </pre>
,	A comma precedes further operands at the same structure level.	,SPACE = <u>STD</u>
list-poss(n):	list-poss signifies that the operand values following it may be entered as a list. If a value is specified for (n), the list may contain no more than that number of elements. A list of two or more elements must be enclosed in parentheses.	NAME = list-poss(30): <subschema-name>

Table 4: Metasyntax

(part 2 of 4)

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

Table 5: Data types

(part 1 of 4)

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

Table 5: Data types

(part 2 of 4)

Data type	Character set	Special rules
realm-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that may consist of a number of substrings by hyphens; first character: A..Z
realmref	0..9	1..3 characters
record-name	A..Z 0..9 Structure identifier: hyphen	1..30 characters String that can consist of a number of substrings separated by hyphens; first character: A..Z In the case of record types with a search key it is 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: $\left. \begin{array}{l} hh:mm:ss \\ hh:mm \\ hh \end{array} \right\}$ hh, mm, ss: Leading zeros may be omitted
userid	A..Z 0..9 \$,#,@	1..8 characters, beginning with A..Z or \$,#,@ BPRIVACY: Strings of less than 8 characters are filled internally with underscore characters.
volume	A..Z 0..9 \$,#,@	1..6 characters starting with A..Z or 0..9
x-string	Hexadecimal: 00..FF	1..8 characters Must be enclosed in single quotes and prefixed with the letter X. There may be an odd number of characters

Table 5: Data types

(part 4 of 4)

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

Table 6: Data type suffixes

1.6 Sample databases

The SHIPPING and CUSTOMER databases form the basis for most of the examples and utility routines in this manual.

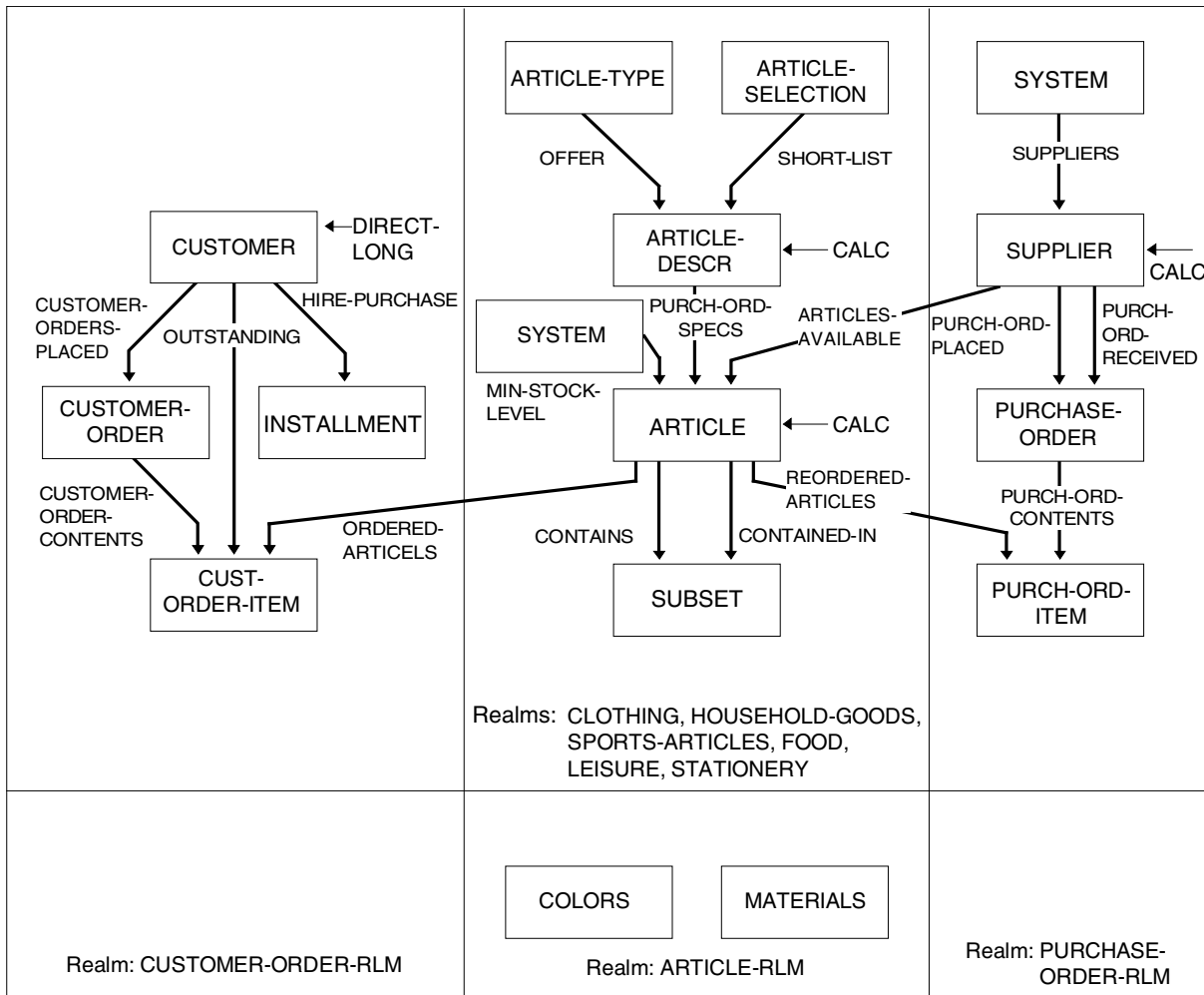


Figure 1: SHIPPING database with schema name MAIL-ORDERS

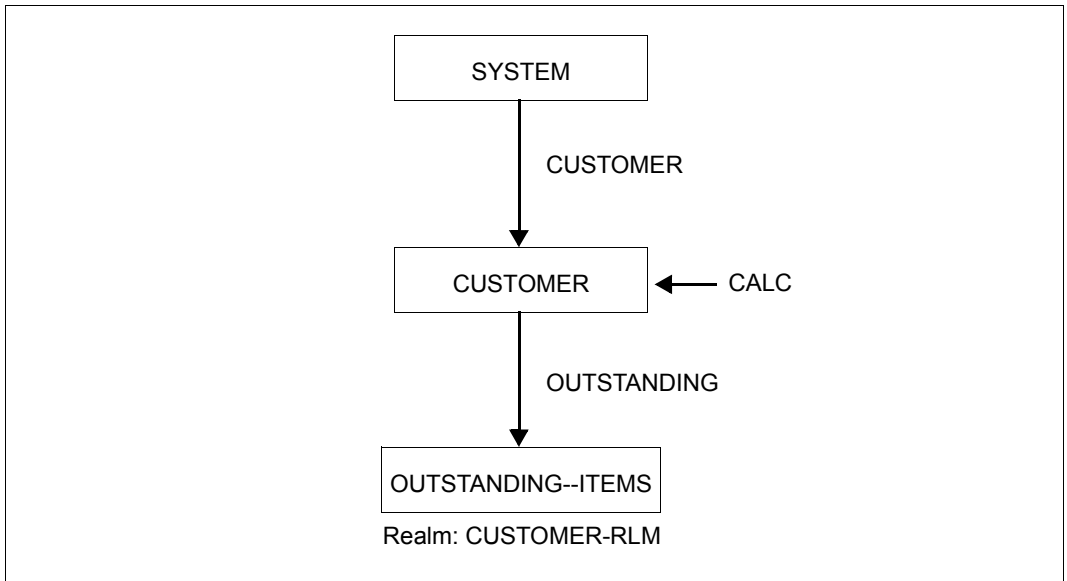


Figure 2: CUSTOMER database with schema name CUSTOMER-FILE

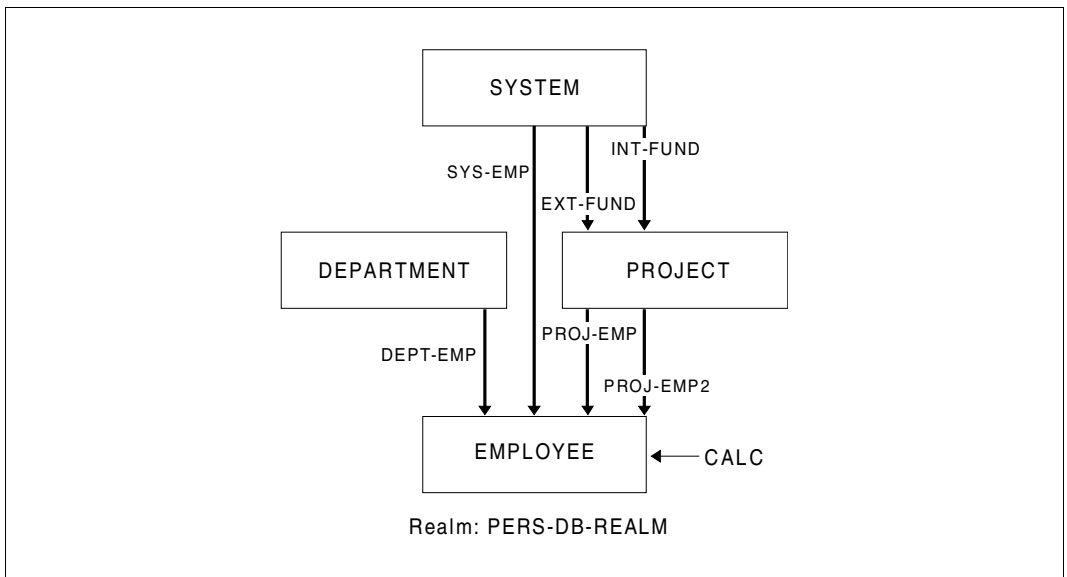


Figure 3: PERSONAL database with schema name PERS-DB

2 Updating and reconstructing a database with BMEND

The BMEND utility routine performs the function of updating and reconstructing a database within the general framework of the UDS/SQL security concept (see the “[Database Operation](#)” manual).

BMEND can be used to update the entire database or to update individual or selected realms with one or more ALOG files. It can be applied on both the original database and the shadow database as well as detached realms.

The database can be updated using ALOG files with the UPDATE-DATABASE statement.

The BMEND statement SHOW-LOG-INFORMATION shows which ALOG files must be specified when updating individual realms in order to make their state consistent with that of the entire database. In other words, you must make these ALOG files available for the update.

You are also provided with information on whether a logging gap exists in the sequence of ALOG files and whether any ALOG file was terminated in an inconsistent state. In such cases, updates can only be applied until that point in time. Later it is only possible using backup.

2.1 Functions

The BMEND utility routine provides the following functions for

- updating databases and shadow databases (or specific realms thereof)
- attaching (ADD) and detaching (REMOVE) realms
- enabling and disabling the online backup capability for a database
- starting and stopping AFIM logging
- obtaining information on the status of databases, shadow databases and ALOG files, and
- concurrently updating detached realms in parallel with DBH operations and calling an information function for realms or ALOG files

At startup BMEND takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

2.2 Statements

2.2.1 Rules

Incorrectly entered statements can be corrected.

Multiple statements of the same type (except for SHOW-LOG-INFORMATION) are combined and only executed once.

If conflicting specifications concerning the function or object are made (e.g. START-LOG/STOP-LOG or ADD-REALM/REMOVE-REALM), the last specification entered applies.

All valid statements, except for the ALLOCATE-BUFFER-POOL, OPEN-DATABASE and UNDO statements, are executed after the END statement:

- The ALLOCATE-BUFFER-POOL statement must be the first statement specified.
- The OPEN-DATABASE statement is only permitted if no ADD-FILE-LINK LINK-NAME=DATABASE has been specified.
- Every correctly entered statement can be reversed with the UNDO statement or with its inverse function (if one exists)

You may enter all other statements in any order. Execution occurs in the following order:

- ALLOCATE-BUFFER-POOL
- OPEN-DATABASE
- UPDATE-DATABASE
- ADD-REALM
- REMOVE-REALM
- START-LOG
- STOP-LOG
- KILL-LOG
- ENABLE-ONLINE-COPY
- DISABLE-ONLINE-COPY
- SHOW-LOG-INFORMATION

2.2.2 Permitted functions

The range of functions permitted depends on the following questions:

- Is the original database or shadow database involved?
- Is the database or shadow database being processed by the DBH?
- Is the database consistent?
- Has AFIM logging been enabled?

Functions that are not permitted are not shown in the SDF mask.

The set of functions permitted may change in the course of a BMEND run, since BMEND statements can have an effect on consistency and AFIM logging.

Consistency point information can always be output at any time.

The KILL-LOG statement is only permitted for inconsistent databases. This statement is required for a database warm start without logging.

List of special cases

- Original database with active DBH:
UPDATE-DATABASE
Concurrent updating with a session is only allowed for realms that are detached according to DBDIR-AK0.
- Inconsistent original database with AFIM logging and without active DBH:
ADD-REALM
REMOVE-REALM
START-LOG
STOP-LOG
Rejected if not preceded by an UPDATE-DATABASE statement to create a consistent database.
- Consistent original database without AFIM logging and without active DBH:
ENABLE-ONLINE-COPY
Only permitted if the START-LOG statement has already been issued
- Shadow database with active DBH:
UPDATE-DATABASE
Concurrent updating with a session is only allowed for realms that are detached according to DBDIR-AK0.

- Inconsistent shadow database without active DBH:

ADD-REALM

REMOVE-REALM

Rejected if not preceded by an UPDATE-DATABASE statement to create a consistent database.

2.2.3 BMEND statements

Statement	Meaning
ADD-REALM	Attaches realms to a database
ALLOCATE-BUFFER-POOL	Defines the buffer size
DISABLE-ONLINE-COPY	Disables the online backup capability for the database
ENABLE-ONLINE-COPY	Enables the online backup capability for the database
END	Terminates the input of statements
KILL-LOG	Stops logging for an inconsistent database
OPEN-DATABASE	Opens the database
REMOVE-REALM	Detaches realms
SHOW-LOG-INFORMATION	Displays logging information
START-LOG	Starts logging for the database original
STOP-LOG	Stops logging for database operations
UNDO	Cancels the effect of a statement
UPDATE-DATABASE	Applies AFIMs to the database

Table 7: BMEND statements

The individual statements of BMEND are described below in alphabetical order.

Attach realms to a database (ADD-REALM)

The ADD-REALM statement can be used to attach one or more realms to a database. The realms to be attached must be consistent and compatible with the current DBDIR.

```
ADD-REALM
```

```
REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realm-name>
```

```
  *ALL-EXCEPT(...)
```

```
    | NAME = list-poss(30): <realm-name>
```

REALM-NAME = *ALL

All detached realms are attached.

REALM-NAME = *ALL-EXCEPT(...)

All detached realms except for those specified are attached.

NAME = list-poss(30): <realm-name>

Name(s) of the realm(s) to be excluded.

REALM-NAME = list-poss(30): <realm-name>

All specified realms are attached.



The realms DBDIR and DBCOM cannot be attached and are therefore rejected.

Define buffer size (ALLOCATE-BUFFER-POOL)

The ALLOCATE-BUFFER-POOL statement defines the size of the used buffer pool in Mbytes.

This statement may be optionally omitted (if the default value is desired); otherwise, it must be specified as the first statement.

ALLOCATE-BUFFER-POOL
BUFFER-SIZE = *STD / <integer 1..2000>

BUFFER-SIZE = *STD

The standard size of the buffer pool is defined as 1 Mbyte.

BUFFER-SIZE = <integer 1..2000>

The size of the buffer pool must lie within the given limits. The maximum value depends on the version of the operating system and the configuration of working memory in the system.



The value for ADDRESS-SPACE-LIMIT must be greater than the value specified here. The appropriate value can be set by the system administrator with the MODIFY-USER-ATTRIBUTES command.

Disable online backup capability for the database (DISABLE-ONLINE-COPY)

The DISABLE-ONLINE-COPY statement disables the online backup capability for all realms of the database.

This statement is also permitted for a shadow database.

DISABLE-ONLINE-COPY

This statement has no operands.

Enable online backup capability for the database (ENABLE-ONLINE-COPY)

The ENABLE-ONLINE-COPY statement enables the online backup capability for all realms of the database.

This statement is also permitted for a shadow database.

The online backup capability for a database cannot be enabled for an original unless the AFIM logging function is active or the START-LOG statement is issued beforehand. Since a consistent database is a prerequisite for the START-LOG statement, an UPDATE-DATABASE statement may also be required.

To create online backups of databases or individual realms, you should use the corresponding HSMS or ARCHIVE statements.

When COPY-FILE is used to generate an online copy, the system does not check whether all requirements for "online backup capability" are fulfilled for the database concerned. In this case you must ensure that AFIM logging is enabled so that the online copy can be made consistent later by applying the changes.

It is never permissible to generate online copies while modifying utility routines are running.

The online backup capability of a database is recorded both in the UDS/SQL administration data and in the DMS catalog entries for the database files. When a database is copied either with COPY-FILE or with HSMS/ARCHIVE, the specifications relating to the online backup capability in the DMS catalog may be lost depending on the selected parameters.

Therefore, before using a copy of a database from an original for which the online backup capability was active, you must ensure that this property remains consistent, for example by running the BMEND utility routine again with the statement ENABLE-ONLINE-COPY.

ENABLE-ONLINE-COPY

This statement has no operands.

Terminate input of statements (END)

The END statement is used to terminate the input of statements. All entered statements are executed after this statement.

The END statement cannot be canceled with the UNDO statement.

END

This statement has no operands.

Stop logging for an inconsistent database (KILL-LOG)

The KILL-LOG statement is used to suppress AFIM logging in order to perform a database warm start without an ALOG file (due to a hardware error on the ALOG file, for instance). The ALOG file is used in a warm start to include the AFIMs from the RLOG file which are not yet in the ALOG file.

The RLOG file is sufficient for a database warm start.

The KILL-LOG statement automatically disables the online backup capability as well.

This statement is only permitted for an inconsistent database.

KILL-LOG

The KILL-LOG statement has no operands.



When a warm start is performed without the ALOG file, a logging gap is created. It is therefore advisable to save the database after a warm start and to begin logging again before any new changes are made.

Open database (OPEN-DATABASE)

The OPEN-DATABASE statement specifies the database to be processed by the statements which follow.

```
OPEN-DATABASE
```

```
DATABASE-NAME = <dbname>
```

```
,COPY-NAME = *NONE / <copy-name>
```

```
,USER-IDENTIFICATION = *OWN / <userid>
```

DATABASE-NAME = <dbname>

Name of the database. You can only process a database that is cataloged under your own user ID. A database under a foreign user ID can only be processed from the system administrator ID TSOS.

COPY-NAME = *NONE

The database original is processed.

COPY-NAME = <copy-name>

The shadow database with the specified copy name is processed.

USER-IDENTIFICATION = *OWN

The database is located under the user's own user ID.

USER-IDENTIFICATION = <userid>

The specification of a foreign user ID is only permitted under the system administrator ID TSOS.



The OPEN-DATABASE statement is not permitted if the database is assigned using ADD-FILE-LINK LINK-NAME=DATABASE.

Detach realms (REMOVE-REALM)

The REMOVE-REALM statement can be used to detach consistent realms.

```
REMOVE-REALM
```

```
REALM-NAME = *ALL-EXCEPT(...) / list-poss(30): <realm-name>
```

```
  *ALL-EXCEPT(...)
```

```
    | NAME = list-poss(30): <realm-name>
```

REALM-NAME = *ALL-EXCEPT(...)

All attached realms except for those specified are detached.

Name = list-poss(30): <realm-name>

Name(s) of the realms that are not to be detached.

REALM-NAME = list-poss(30): <realm-name>

All specified realms are detached.



The realms DBDIR and DBCOM cannot be detached and are therefore rejected.

Show log information (SHOW-LOG-INFORMATION)

The SHOW-LOG-INFORMATION statement can be used to output the following information:

- status of realms of the assigned database with respect to the log interval required in order to reach a common consistency point
- attributes of the database “with logging or without logging” and whether “online copies are allowed”
- information on up to 63 ALOG files (history, sequence numbers in ascending order)

The displayed times reflect the local time.

In addition, useful information for the application of updates is stored in a job variable, provided such a job variable has been created with LINK-NAME=JVBMEND (see [section “Supplying the BMEND job variable” on page 61](#)).

The SHOW-LOG-INFORMATION statement can be specified alone and may also be entered in the course of DBH operations.

```
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-no>
,OUTPUT = list-poss: *SYSLST / *SYSOUT
```

REALM-NAME = *ALL

Shows information for all realms of the assigned database.

REALM-NAME = *ALL-EXCEPT(...)

Shows information for all realms except for those specified.

NAME = list-poss(30): <realm-name>

Name(s) of the realm(s) for which no information is to be shown.

REALM-NAME = list-poss(30): <realm-name>

Shows information for all specified realms.

LOG-FILE = *STD

Shows information as of the current ALOG file, i.e. the one specified in the DBDIR.

LOG-FILE = *NONE

No information on ALOG files is output.

LOG-FILE = <alog-seq-no>

Shows information as of the specified ALOG file.

OUTPUT = list-poss: *SYSLST / *SYSOUT

Defines where the information is to be output.

***SYSLST**

Log information is output to SYSLST.

***SYSOUT**

Log information is output to SYSOUT.

Message texts

The following information is output:

1. Details with respect to the processed database

```
***** LOG INFORMATION FOR DATABASE $userid.dbname[.copy-name]
```

dbname

Name of the assigned database

copy-name

is output if a shadow database was assigned

2. List of specified realms indicating the ALOG SEQ NR (e.g. with differing ALOG sequence numbers at the start and end of an online copy)

```
***** LOG INTERVAL OF SPECIFIED REALMS
```

REALM NAME	ALOG SEQ NR		CONSISTENT
	BEGIN	END	
<i>realm-name-1</i>	<i>alog-seq-no</i>	<i>alog-seq-no</i>	YES/NO
...			
<i>realm-name-n</i>	<i>alog-seq-no</i>	<i>alog-seq-no</i>	YES/NO

realm-name

Names of the specified realms

ALOG SEQ NR

BEGIN: *alog-seq-no* of the realm at the start of the backup operation (of the online backup)

END: *alog-seq-no* of the realm at the end of the backup operation (of the online backup)

CONSISTENT

YES: The consistency point date at the start and end of the backup operation is the same, and no SYSTEM BREAK is set.

NO: The status of the realm differs from that of a consistency point (the consistency date at the start and end of the backup operation differ, or a SYSTEM BREAK is set)

3. Specification of the log interval required to apply the update

```
***** TO MAKE THE SPECIFIED REALMS CONSISTENT, THE FOLLOWING
LOG FILES ARE NECESSARY :
FROM ALOG SEQ NR alog-seq-no1 TO ALOG SEQ NR alog-seq-no2
OR FROM LOG INTERVAL BEGIN datetime-1 TO LOG INTERVAL END datetime-2
```

alog-seq-no1

The update must be applied starting with this ALOG SEQ NR. This value is saved in a job variable.

alog-seq-no2

This value represents the DEADLINE the must be given in order to make all specified realms consistent.

datetime-1

Date and time of the first log

datetime-2

Date and time of the last log

If no updates need to be applied, this output is suppressed.

4. Log mode with volume information; Message 4 is not output for shadow databases.

```
***** SUPPORTS OF ACTUAL LOG FILE:
      DEFAULT SUPPORT:{VOLUME = vol-1 DEVICE = dev-1
                      [... ] /
                      PVS ID = {DEFAULT PVS / catid}}
      RESERVE SUPPORT:{VOLUME = vol-4 DEVICE = dev-4
                      [... ] /
                      PVS ID = {DEFAULT PVS / catid}}
```

5. Logging history

***** INFORMATION ABOUT LOG HISTORY:

ALOG SEQ NR	LOG INTERVAL		AFIM	BACKOUT	LOGGING
	BEGIN	END			
<i>alog-seq-nr</i>	<i>time-1</i>	<i>time-2</i>	[*]	[*]	[{*/?}]

ALOG SEQ NR

Starting with the sequence number specified via the LOG-FILE operand, information on a maximum of 63 ALOG files is output (with sequence numbers in descending order)

LOG INTERVAL

BEGIN: *time-1* Time of the first log
 END: *time-2* Time of the last log

If the time is not unique, daylight saving time (summer time) is assumed, and a warning is issued.

AFIM *: AFIMs are recorded in the ALOG file
 _: Gap in the AFIM log

BACKOUT BFIM

*: BFIMs are recorded in the ALOG file
 _: No BFIMs are recorded in the ALOG file

LOGGING GAP

*: The LOG INTERVAL END is less than the LOG INTERVAL BEGIN of the next ALOG file in the sequence or less than the BACK UP DATA of the DBDIR (LOG-FILE=*STD)
 ? : The continuation of logging in the ALOG file with the next sequence number cannot be examined (LOG-FILE=*alog-seq-no*)
 _: Logging was continued to the next ALOG file in the sequence without interruption

If the ALOG file that is specified in the LOG-FILE operand cannot be read, an error message is output instead of the table.

6. Whether online copies are allowed

***** ONLINE COPIES BY ARCHIVE ARE [NOT] ALLOWED

Example

```

/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=SHIPPING,DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.8A00
/START-UDS-BMEND
***** START          BMEND          (UDS/SQL V2.8 0000 )    2015-06-28    09:55:15
//SHOW-LOG-INFORMATION REALM-NAME=*ALL,LOG-FILE=4,OUTPUT=*SYSOUT
SYSTEM_BREAK OCCURRED IN REALM DATABASE-DIRECTORY
***** INCONSISTENT DATABASE DIRECTORY
SYSTEM_BREAK OCCURRED IN REALM PURCHASE-ORDER-RLM
SYSTEM_BREAK OCCURRED IN REALM FOOD
SYSTEM_BREAK OCCURRED IN REALM ARTICLE-RLM
***** INCONSISTENT DATABASE DIRECTORY
        FUNCTION ADD NOT AVAILABLE
        FUNCTION REMOVE NOT AVAILABLE
        FUNCTION START NOT AVAILABLE
        FUNCTION STOP NOT AVAILABLE

//END
***** BEGIN          FUNCTION SHOW LOG INFORMATION AT 09:55:16
***** LOG INFORMATION FOR DATABASE $XXXXXXXXX.SHIPPING _____ 1.
***** LOG INTERVAL OF SPECIFIED REALMS _____ 2.

                !      ALOG SEQ NR      !      !
REALM-NAME      ! BEGIN  ! END    ! CONSISTENT !
-----+-----+-----+-----+
DATABASE-DIRECTORY      !      2 !      2 !      NO      !
DATABASE-COMPILER-REALM !      1 !      1 !      YES     !
CUSTOMER-ORDER-RLM     !      1 !      1 !      YES     !
PURCHASE-ORDER-RLM    !      2 !      2 !      NO      !
CLOTHING               !      1 !      1 !      YES     !
HOUSEHOLD-GOODS        !      1 !      1 !      YES     !
SPORTS-ARTICLES        !      1 !      1 !      YES     !
FOOD                   !      2 !      2 !      NO      !
LEISURE                !      1 !      1 !      YES     !
STATIONERY             !      1 !      1 !      YES     !
ARTICLE-RLM            !      2 !      2 !      NO      !
***** TO MAKE THE SPECIFIED REALMS CONSISTENT, THE FOLLOWING LOG FILES ARE _____ 3.
        NECESSARY :
        FROM ALOG SEQ NR      2 TO ALOG SEQ NR      2
        OR FROM LOG INTERVAL BEGIN 20150628095452 TO LOG INTERVAL END
        20150628095452
***** LOG MODE : AFIM LOGGING _____ 4.
***** SUPPORTS OF ACTUAL LOG FILE :
        DEFAULT SUPPORT :PVS ID = DEFAULT PVS
        RESERVE SUPPORT :PVS ID = :SQL2:
    
```

***** INFORMATION ABOUT LOG HISTORY : _____ 5.

ALOG SEQ NR!	LOG INTERVAL		! AFIM	!BACKOUT!	LOGGING !
	BEGIN	END	!	!	!
4	!20150628095505!	20150628095506!	* !	!	? !
3	!20150628095504!	20150628095505!	* !	!	!
2	!20150628095452!	20150628095504!	* !	!	!
1	!20150628095451!	20150628095452!	* !	!	!

***** ONLINE COPIES BY ARCHIVE ARE ALLOWED _____ 6.

***** NORMAL END FUNCTION SHOW LOG INFORMATION AT 09:55:16

***** DIAGNOSTIC SUMMARY OF BMEND

NO WARNINGS
 NO ERRORS
 NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY

***** NR OF DATABASE ACCESSES : 51

***** NORMAL END BMEND (UDS/SQL V2.8 0000) 2015-06-28 09:55:16

Start logging for database original (START-LOG)

The START-LOG statement is used to activate AFIM logging for the original database and to optionally define new volumes on which subsequent ALOG files are to be created.

START-LOG

```

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

```


DEFAULT-SUPPORT = *PUBLIC (...)

Any new ALOG file to be created will be created on public disk unless an explicit CREATE-FILE command is issued for the new ALOG file.



A specification other than *STD is only permitted for one of the operands CATID, VOLUME-SET or VOLUME.

The specifications are not checked for compatibility with the current UDS/SQL pubset declaration for the BMEND run. However, you must bear in mind that the specifications made here must be compatible with the UDS/SQL pubset declarations of the later application environments (DBH operation, utility routines).

CATID = *STD / *OWN / <catid>

Determines the catalog ID of the pubset.

CATID = *STD

The catalog ID is determined by the system from the specifications made in the VOLUME or VOLUME-SET operand. If nothing is specified for either of these operands or *STD was specified, the default catalog ID of the configuration user ID is used.

CATID = *OWN

The default catalog ID of the configuration user ID is used.

CATID = <catid>

The specified catalog ID is used.

VOLUME-SET = *STD / <catid>

Specifies the volume set of an SM pubset on which the ALOG file is to be configured. Non-privileged users can only specify volumes sets explicitly if they are authorized to perform physical allocations on the pubset concerned.

VOLUME-SET = *STD

The volume set for ALOG files on an SM pubset is selected by the system.

VOLUME-SET = <catid>

Explicit specification of the volume set on which the ALOG file is to be configured.

VOLUME = *STD / list-poss(15): <volume>

Specifies the public disks on which the ALOG file is to be configured.

VOLUME = *STD

The disks on which the ALOG file is configured are selected by the system.

VOLUME = list-poss(15): <volume>

A new ALOG file is created on the specified disks. These can be assigned to an SF pubset or to a volume set of an SM pubset. Up to 15 VSNs can be specified, no two of which may be the same.

The VSN can be specified in PUB notation (PUBpxx) or in dot notation (pp[pp].[xy]z). All the disks specified must belong to the same volume set, i.e. they must have the same catalog ID.

Non-privileged users may only use this specification if they are authorized to perform physical allocation of public storage space on the pubset concerned.

DEFAULT-SUPPORT = list-poss(15): *PRIVATE (...)

Any new ALOG file to be created will be created on the specified disk or disks unless an explicit CREATE-FILE command is issued for the new ALOG file. A maximum of 15 disks may be specified.

VOLUME = list-poss(15): <volume>

Specifies the private disks on which ALOG files can be created.

DEVICE = <device>

Defines the device type of the private disks.

DEFAULT-SUPPORT = *UNCHANGED

Existing values remain in effect. *UNCHANGED is only possible if logging has already been started.

RESERVE-SUPPORT = *PUBLIC (...)

The ALOG file will be created on public disk if no explicit specification for the new ALOG file to be created exists, and if the file cannot be created on the volumes specified with DEFAULT-SUPPORT.



A specification other than *STD is only permitted for one of the operands CATID, VOLUME-SET or VOLUME.

The specifications are not checked for compatibility with the current UDS/SQL pubset declaration for the BMEND run. However, you must bear in mind that the specifications made here must be compatible with the UDS/SQL pubset declarations of the later application environments (DBH operation, utility routines).

CATID = *STD

The catalog ID is determined by the system from the specifications made in the VOLUME or VOLUME-SET operand. If nothing is specified for either of these operands or *STD was specified, the default catalog ID of the configuration user ID is used.

CATID = *OWN

The default catalog ID of the configuration user ID is used.

CATID = <catid>

The specified catalog ID is used.

VOLUME-SET = *STD / <catid>

Specifies the volume set of an SM pubset on which the ALOG file is to be configured. Non-privileged users can only specify volumes sets explicitly if they are authorized to perform physical allocations on the pubset concerned.

VOLUME-SET = *STD

The volume set for ALOG files on an SM pubset is selected by the system.

VOLUME-SET = <catid>

Explicit specification of the volume set on which the ALOG file is to be configured.

VOLUME = *STD / list-poss(15): <volume>

Specifies the public disks on which the ALOG file is to be configured.

VOLUME = *STD

The disks on which the ALOG file is configured are selected by the system.

VOLUME = list-poss(15): <volume>

A new ALOG file is created on the specified disks. These can be assigned to an SF pubset or to a volume set of an SM pubset. Up to 15 VSNs can be specified, no two of which may be the same.

The VSN can be specified in PUB notation (PUBp_{xx}) or in dot notation (pp[pp].[xy]z).

All the disks specified must belong to the same volume set, i.e. they must have the same catalog ID.

Non-privileged users may only use this specification if they are authorized to perform physical allocation of public storage space on the pubset concerned.

RESERVE-SUPPORT = list-poss(15): *PRIVATE (...)

Any new ALOG file to be created will be created on the specified disk if the file cannot be created on the volumes specified with DEFAULT-SUPPORT.

VOLUME = list-poss(15): <volume>

Specifies the private disks on which ALOG files can be created.

DEVICE = <device>

Defines the device type of the private disks.

RESERVE-SUPPORT = *UNCHANGED

Existing values remain in effect. *UNCHANGED is only possible if logging has already been started.



If the DEFAULT and RESERVE operands are assigned the same value, an error occurs. This error is detected on creating the ALOG files, but not during the syntax analysis of the START-LOG statement. It is not possible to switch to an alternative medium.

SPACE = *STD

The new ALOG file to be created is assigned a primary allocation value of 192 and a secondary allocation value of 576.

SPACE = *RELATIVE (...)

The specified values are assigned as primary and secondary allocations for the new ALOG file to be created.

(The values specified here must be ≥ 192 and ≤ 50331645 for the primary allocation, and ≥ 576 and ≤ 32767 for the secondary allocation.)

PRIMARY-ALLOCATION = <integer 192..50331645>

Number of PAM pages for the primary allocation.

SECONDARY-ALLOCATION = <integer 576..32767>

Number of PAM pages for the secondary allocation.

SPACE = *UNCHANGED

Existing values remain in effect. *UNCHANGED is only possible if logging has already been started.

USER-ACCESS = *OWNER-ONLY

Restricts access to the ALOG file to the user ID under which it was created.

USER-ACCESS = *ALL-USERS

Permits the ALOG file to be accessed by other user IDs as well.

RESET-LOG-POOL = *NO

The new ALOG file is created with an ALOG sequence number that is obtained by incrementing the ALOG SEQ NR from the Act-Key-0 of the DBDIR by 1.

RESET-LOG-POOL = *YES

The ALOG file begins with ALOG SEQ NR = 1.

This operand is typically used to create the log pool of a duplicated database - starting with ALOG SEQ NR = 1.



You can specify up to 15 disks in a PRIVATE operand and up to 15 variants of the PRIVATE operand. However, if the number of disks that are specified in a statement exceeds 15, the last 15 entries apply.

This means that it is only worthwhile specifying multiple variants of the PRIVATE operand if the disks are assigned to different device types.

If an ALOG file cannot be accessed (because it was deleted, for example), a new ALOG file with an ALOG sequence number incremented by 1 is created by the utility routine. If the deleted ALOG file had no relevant information (no deviation from consistency point), the update could be applied in two steps despite the logging gap produced as a result of the deletion (step 1 up to the gap; step 2 after it).

Example

1. Activate AFIM logging and the online backup capability. New ALOG files are to be created on different public disks.

```

/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.67A00
/START-UDS-BMEND
***** START  BMEND  (UDS/SQL V2.8 0000 ) 2015-09-18   09:54:50
//ALLOCATE-BUFFER-POOL BUFFER-SIZE=*STD
//OPEN-DATABASE DATABASE-NAME=SHIPPING
***** DATABASE ORIGINAL WITHOUT AFIM LOGGING
        FUNCTION ENABLE NOT AVAILABLE
        FUNCTION KILL NOT AVAILABLE
***** CONSISTENT  DATABASE DIRECTORY
//START-LOG  DEFAULT-SUPPORT=*PUBLIC(CATID=*OWN), -
//          RESERVE-SUPPORT=*PUBLIC(CATID=SQL2), -
//          SPACE=*STD, RESET-LOG-POOL=*NO
***** LOGGING WILL BE ACTIVATED
        FUNCTION ENABLE AVAILABLE FROM NOW ON
//ENABLE-ONLINE-COPY
//END
***** BEGIN          FUNCTION START LOGGING AT 09:54:50
ALOG FILE CREATED ACCORDING TO DEFAULT-SUPPORT
***** NORMAL  END FUNCTION START LOGGING AT 09:54:50
***** BEGIN          FUNCTION ENABLE ONLINE COPY AT 09:54:50
***** ONLINE COPY FOR DATABASE $XXXXXXXXX.SHIPPING ALLOWED
***** NORMAL  END FUNCTION ENABLE ONLINE COPY AT 09:54:50

***** DIAGNOSTIC SUMMARY OF BMEND

                NO WARNINGS
                NO ERRORS
                NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES   :           147
***** NORMAL END  BMEND  (UDS/SQL V2.8 0000 ) 2015-09-18   09:54:50

```

2. Reset the ALOG SEQ NR to 1 and specify three disks of the same device type on which new ALOG files can be subsequently created.

```

/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=02.8A00
/START-UDS-BMEND
//OPEN-DATABASE DATABASE-NAME=dbname
//START-LOG
//          DEFAULT=*PRIVATE(VOLUME=(G3200A,G3200B,G3200C),-
//                                DEVICE=D3468)                , -
//          RESERVE=*PRIVATE(VOLUME=G3400A                    , -
//                                DEVICE=D3468)                , -
//          RESET-LOG-POOL=*YES
//END
***** BEGIN FUNCTION START LOGGING AT timestamp

***** NORMAL END FUNCTION START LOGGING AT timestamp

```

3. Assign three disks of different device types on which new ALOG files can be subsequently created.

```

/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=02.8A00
/START-UDS-BMEND
//OPEN-DATABASE DATABASE-NAME=dbname
//START-LOG DEFAULT=( *PRIVATE(VOLUME=G3030M,                , -
//                                DEVICE=D3468)                , -
//                                *PRIVATE(VOLUME=(G2065C,G2065D) , -
//                                DEVICE=D5804)                ), -
//          RESERVE=*PUBLIC
//END
***** BEGIN FUNCTION START LOGGING AT timestamp

***** NORMAL END FUNCTION START LOGGING AT timestamp

```

timestamp indicates the current time.

Stop logging for database operations (STOP-LOG)

The STOP-LOG statement deactivates logging. Before logging is stopped, BMEND disables the online backup capability for all realms if required (i.e. if previously enabled).

STOP-LOG

This statement has no operands.

Undo a statement (UNDO)

The UNDO statement cancels the last correctly entered statement (except for UNDO itself). In other words, that statement is not executed.

Each subsequent UNDO statement cancels the preceding statement in the chain (except for the UNDO statement itself).

The UNDO statement does not cancel the ALLOCATE-BUFFER-POOL and END statements.

UNDO

This statement has no operands.

Apply AFIMs to a database (UPDATE-DATABASE)

The UPDATE-DATABASE statement can be used to apply AFIMs from ALOG files to realms and thus update them.

The required realms, which represent an older status of the database, must be first copied, read in from ARCHIVE backups, or recataloged and made available.

Detached realms can also be processed if BMEND is run in parallel with the DBH.

Inconsistent realms must be made consistent with the UPDATE-DATABASE statement before they are attached or detached.

UPDATE-DATABASE

REALM-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(30): <realm-name>

*ALL-EXCEPT(...)

 | NAME = list-poss(30): <realm-name>

,DEADLINE = *STD / *BREAK-POINT / <alog-seq-no> / *TIME-STAMP(...)

*TIME-STAMP(...)

 | DATE = <date>

 | ,TIME = <time>

,DELETE = *NO / *YES

REALM-NAME = *ALL

All realms are updated.

REALM-NAME = *ALL-EXCEPT(...)

All realms except for those specified are updated.

NAME = list-poss(30): <realm-name>

 Name of the realm that is not to be updated.

REALM-NAME = list-poss(30): <realm-name>

All specified realms are updated.

DEADLINE = *STD

The database is updated to the end of the last ALOG file closed. A consistency point recorded in an ALOG file that is not closed yet cannot be reached with DEADLINE = *STD.

DEADLINE = *BREAK-POINT

All consistent ALOG files, including the last, current, and possibly inconsistent ALOG file are used for the update. This allows an update up until the point at which failure occurred. If the last ALOG file is inconsistent, a warm start of the database is required afterwards. Realms of the shadow database are only updated with closed ALOG files.

DEADLINE = <alog-seq-no>

Sequence number of the ALOG file up to and including which AFIMs are to be applied. Only closed ALOG files are used for the update (leading zeros in the *alog-seq-no* may be omitted in the specification.)

DEADLINE = *TIME-STAMP(...)

Updates are applied to the database up to and including the last closed ALOG file for which the LOG_INTERVAL_END is less than or equal to the given *TIME-STAMP(...).

DATE = <date>

Date that limits the applied updates.

TIME = <time>

Time that limits the applied updates. If the time is not unique, daylight saving time (summer time) is assumed, and a warning is issued.

DELETE = *NO

The applied ALOG files are retained on disk.

DELETE = *YES

With the exception of the current file, all applied ALOG files (i.e. the files read in for the update) are automatically deleted, assuming that no error has occurred when executing this statement.

The DBDIR and the DBCOM are treated like any other realm. The realm DBCOM is implicitly addressed by the *ALL and *ALL-EXCEPT options.

If all the realms specified using the REALM-NAME operand are not available, the missing realms are not processed. Furthermore, no deletion of the ALOG file takes place even if requested.

The DEADLINE = *BREAK-POINT operand is meaningless for shadow databases. *BREAK-POINT is treated as *STD in such cases.

The sequence number of the next ALOG file required to reach the desired DEADLINE is saved in a job variable after each individual ALOG file has been applied (see the section on "Supplying job variables" on [page 61](#)).

If a logging gap is contained in the sequence of ALOG files to be applied, the update is terminated at that point.

A warning is issued if the DEADLINE (specified as an *alog-seq-no* or *TIME-STAMP(...)) could not be reached.

A realm other than the DBDIR can only be updated if it was addressed in the update session or matches the DBDIR, which is also being updated.

Statistics and summary report

On completion of the BMEND run, internal counters are evaluated and output for the SUMMARY REPORT:

```
***** DIAGNOSTIC SUMMARY OF BMEND

      { NO }   WARNINGS
      {   }   ERRORS
      { num }  SYSTEM ERROR

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES:  number
```

2.2.4 Command sequence to start BMEND

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

```
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
05 /START-UDS-BMEND
06 [//OPEN-DATABASE DATABASE-NAME=dbname
      [,COPYNAME=*NONE/copyname]
      [,USER-IDENTIFICATION=*OWN/userid]
07 bmend statements
08 //END
```

01, 02 Creates a job variable.

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

04 The version of the utility routine is selected.
 Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.

05 Alias names for the call are START-UDS-REPAIR and BMEND.

Examples on the use of BMEND can be found in the “[Database Operation](#)” manual.

2.3 Supplying the BMEND job variable

In order to implement automatic database saving and recovery operations, the BMEND utility routine stores information in a job variable. This job variable can be used by other programs or procedures for control purposes.

The job variable is supplied with information if a job variable has been created with LINK-NAME=JVBMEND.

BMEND does not use the contents of the job variable as input, but simply updates it with relevant values at the end of certain functions.

The process of supplying this job variable with information is internally organized in two parts:

1. initialization with SHOW-LOG-INFORMATION
2. updating with UPDATE-DATABASE

The initialization of the job variable (SHOW-LOG-INFORMATION statement) provides an initial decision support system for the initiation of recovery procedures.

As the user, you must then decide whether the information returned is sufficient. If necessary, you may have to repeat the initialization by modifying the SHOW-LOG-INFORMATION statement with the LOG-FILE operand.

The job variable contains the following information areas:

- status of the processed DBDIR
- status of a log pool segment (common log data)
- status of a log pool segment (AFIM log data)
- sequence numbers to apply further updates

Statement sequence

```
/CREATE-JV JV-NAME=JOBVAR, PROTECTION=*STD
/SET-JV-LINK LINK-NAME=JVBMEND, JV-NAME=JOBVAR
/ADD-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=SHIPPING.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=version, SCOPE=*TASK
/START-UDS-BMEND
//SHOW-LOG-INFORMATION LOGFILE=*STD, OUTPUT=*SYSOUT
//END
.
.
```

Structure of the job variable

DISPL

(0) DBDIR_DATA	Describes the status of the accessed DBDIR; deleted by UPDATE
(0) ALOG_SEQ_NR	ALOG_SEQ_NR from the DBDIR
(4) CONSISTENCY_DATA	BACK_UP_DATA from the DBDIR; time at which the database was last updated
(18) CONSISTENT	CONSISTENCY / C'Y' or C'N' C'N': if SYSTEM_BREAK in AK0 or AKn is set or if AK0 is not equal to AKn
(19) FILLER	
(20) COMMON_LOG_DATA	Indicates the limits of the examined ALOG file sequence; the oldest and most recent entry in the history is deleted by UPDATE
(20) HIGHEST_ALOG_SEQ_NR	Highest sequence number for which the history is stored in the ALOG file
(24) LOWEST_ALOG_SEQ_NR	Lowest sequence number for which the history is stored in the ALOG file
(28) LOG_POOL_PART_END_DATA	YYYYMMDDHHMMSS; LOG_INTERVAL_END of the ALOG file assigned in the LOG-FILE operand
(42) LOG_POOL_PART_BEGIN_DATA	YYYYMMDDHHMMSS
(56) AFIM_LOG_DATA	Details on most recent log interval with contiguous AFIM logging from the history; equal to 0 if examined area has no AFIMs; is deleted by UPDATE
(56) UPPER_ALOG_SEQ_NR	Highest sequence number of examined log pool segment with AFIMs
(60) LOWER_ALOG_SEQ_NR	Lowest sequence number of examined log pool segment with AFIMs
(64) UPPER_ALOG_DATA	LOG_INTERVAL_END of ALOG file
(78) LOWER_ALOG_DATA	LOG_INTERVAL_BEGIN of ALOG file

(92) BACKOUT_LOG_DATA	Indicates the limits of the most recent BACKOUT area without gaps
(92) UPPER ALOG_SEQ_NR	Reserved for future extensions
(96) LOWER ALOG_SEQ_NR	Reserved for future extensions
(100) UPPER ALOG_DATA	Reserved for future extensions
(114) LOWER ALOG_DATA	Reserved for future extensions
(128) NEXT_SEQ_NR	Contains the sequence number of the next ALOG file to be applied
(128) UPDATE_START_SEQ_NR	Lowest ALOG number of all examined realms initialized by SHOW-LOG; is incremented by 1 after reading in all AFIMs of an ALOG file; equal to 0 if the DEADLINE is reached
(132) RESET_START_SEQ_NR	
(136) CHAR_TYPE_LOG_DATA	Details in character representation
(136) DBNAME	Name of processed database
(153) COPYNAME	CHAR (8); COPYNAME of processed shadow database
(161) ALOG_SEQ_CHAR	CHAR (9); ALOG_SEQ_NR from DBDIR
(170) UPDATE_START_SEQ_CHAR	CHAR (9); UPDATE_START_SEQ_NR
(179) END	

Results of initialization

Information from the ALOG BOTTOM PAGE of the assigned ALOG file and from the DBDIR are used for initialization.

The difference between the lowest and highest sequence number of the ALOG files sequence can also be less than 63. This situation is possible with a smaller number of ALOG files or after an inconsistent switch in the ALOG file (old ALOG file no longer accessible).

If an **original database** is involved, initialization of the job variable returns the following information:

- consistency of the original DBDIR
- time at which the DBDIR was last updated
- highest sequence number of the ALOG file sequence
- the most recent interval with AFIM logging is output (limit values as sequence numbers and with time stamps)

The time of a maximum DEADLINE will have also been stored (LOG END of the most recent AFIM logging interval).

The returned value can be used to read in a suitable recovery log for the UPDATE function.

Since only the last 63 log files are examined, it is not always clear whether the logging period with no gap extends further in the past.

If the ALOG file sequence needs to be examined by going back further in the past, the job variable will need to be reinitialized. When this is done, it is generally advisable to begin with the lowest *alog-seq-no* of the examined sequence in order to obtain overlapped segments.

If a **shadow database** was assigned, it is not possible to obtain any information from the DBDIR with regard to which ALOG file sequence numbers were used when the database was last processed. The highest sequence numbers of the used ALOG files can only be determined if the original is assigned. In the case of shadow databases, only the sequence number at the time of saving the DBDIR can be output.

Job variable fields are supplied with information in the same way for a shadow database as when an original is assigned.

Updating with UPDATE-DATABASE

The UPDATE-DATABASE statement results in the deletion of all values of job variables that affect the DBDIR or backout logging.

When an ALOG file has been read in (i.e. applied), the sequence number of the next ALOG file to be applied is entered. When the DEADLINE is reached, a binary zero is entered.

Statement sequence

```
/CREATE-JV JV-NAME=JOBVAR, PROTECTION=*STD
/SET-JV-LINK LINK-NAME=JVBMEND, JV-NAME=JOBVAR
/ADD-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=SHIPPING.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=version, SCOPE=*TASK
/START-UDS-BMEND
//UPDATE-DATABASE REALM-NAME=*ALL, DEADLINE=STD, DELETE=NO
//END
.
.
```

3 Checking the consistency of a database with BCHECK

For the database to operate correctly it is vital that the physical structures of the database be correct. It is, however, not always possible to preclude violations of physical consistency due to system errors. In contrast to inconsistencies, such violations are termed consistency errors from the DBH point of view. The UDS/SQL utility routine BCHECK allows the user to check the physical structures of the database when problems occur or within the course of periodic data saving and thus uncover potential consistency errors at an early stage. Since BCHECK can pinpoint each error precisely, it is possible to correct a database containing errors and thus avoid further errors.

BCHECK uses the redundancy principle when checking UDS/SQL databases. First of all, it checks the local data in a page on the one hand and, on the other, the predefined physical structures and the DBDIR metadata on the basis of the SIA (local consistency). Next, BCHECK checks the consistency of logically associated system data located on different pages, again on the basis of the SIA (global consistency).

It is not possible to check the logical consistency of user data, since no information concerning the reference data or the validity of the user data in the database is available to UDS/SQL.

The following can be checked: the user realms of the database, the PRIVACY-AND-IFQ database in the DBDIR, and the compiler database in the DBCOM. BCHECK only checks realms on disk.

It is not always necessary to check the entire database: it is also possible to restrict the check range to individual realms, record types, sets or search keys and consistency criteria, thereby saving time. The depth of the check can be likewise restricted by excluding the management structures for the storage of records or the key values of ASC keys, DESC keys or search keys from the check.

As of UDS/SQL V2.6 BCHECK uses I/O transfer lengths of over 32 K when reading in database pages. The minimum possible I/O transfer length of all disks on which the relevant database realms are located is used here.

3.1 Description of the checking procedure

To optimize the number of accesses to the database, BCHECK checks the database in a single sweep.

BCHECK initially makes local checks in the specified realms on:

- the Act-Key-0 page and Act-Key-N page,
- FPA pages, DBTT anchor pages and DBTT pages,
- the page header of all database pages,
- the formal structure of CALC and table pages including lists of level 0,
- the agreement of the key values in the records with those of the CALC table line in direct CALC pages,
- the record displacements in the page index entries, and
- the sort sequence of the keys or the record sequence numbers in the table pages

i.e. it checks the internal page structures of the realms.

Subsequently BCHECK checks the global relationships of the specified check objects, i.e. those relationships which transcend page structures. To do this it reads the relevant DBTT entries, SCD entries, table headers, table entries and records from the record types, sets and search keys to be checked and generates what are known as the check records from this information.

3.1.1 Setting the checking mode

For the following error analysis, you specify the mode in which BCHECK further processes the check records. Two checking modes are possible: a *summing check* and a *sort check*.

For both modes, BCHECK makes use of the fact that in the various page types (DBTT pages, data pages etc.), information on the check objects is stored in redundant form. For example, from a DBTT page, the page address of the record can be determined for each DB key. If the database is consistent, both DB key and page address are also in the data page in which the record is stored. This means that from the various pages two identical check records are generated for each check object, and, depending on their origin, these will have either a positive or a negative sign.

BCHECK has two procedures for detecting consistency errors in the database:

1. The counter procedure

BCHECK sets up three counters:

- It uses one to count whether precisely the same number of positive and negative check records exist, which is the case if the database is consistent.
- In the other two, it adds the check records compressed to 6 bytes and the square of these values, respectively.

This technique, which is based on the theory of error detecting codes, ensures with a high degree of accuracy that an existing error will be detected because the corresponding sums do not match.

The counter procedure can detect and roughly locate consistency errors in the database but not pinpoint them. This checking method is also called summing check.

2. Sorting procedure

BCHECK collects all the check records in a file and sorts them so that associated sets of check records must be located next to one another if the database is correct. It then compares each set in turn; if sets do not match, there is a consistency error at that point in the database.

This means that BCHECK can use this checking method not only to detect consistency errors in the database but also to pinpoint them so that the consistency errors can be rectified. However, because the check records have to be sorted, this procedure is far more time-consuming than the counter procedure.

Summing check

Depending on the BCHECK statements, BCHECK uses the counter procedure, the sorting procedure or, as part of a summing check, a combination of the two. Since the sorting procedure in the summing check is only used for a restricted set of check records, the summing check takes very little time and is ideally suited for use during daily data saving activities. If global summation consistency errors are detected, a sort check would be needed to pinpoint the error location.

Sort check

Here BCHECK uses only the sorting procedure. If run unmodified, this check is considerably more time-consuming than the summing check.

You can greatly reduce this time requirement by performing an incremental check. This procedure is described below and can also be used for summing checks.

3.1.2 Defining the scope of checking

It is possible to define both the checking mode and the scope of checking. BCHECK has facilities for *overall checking* and *incremental checking*.

Overall checking

BCHECK checks the database in its entirety for consistency.

The following can be checked: the original database or a shadow database.

Overall checking must always be performed if there is no consistent copy of the database or the existing copy has not yet been checked.

It is also required if the database has been restructured or reorganized, and there is either no consistent copy of the changed database or the existing copy has not yet been checked.

Incremental checking

BCHECK only checks the pages that have changed with respect to a shadow database of an earlier database status. BCHECK can determine which page contents have changed by comparing the previous shadow database with the original database or with a more recent shadow database. The older shadow database then only needs the DBDIR if this has been changed.

Incremental checking saves a considerable amount of time when compared to overall checking, since in most cases BCHECK does not have to check all pages in the database. The extra time involved in reading every non-empty page twice is counterbalanced by the saving involved in checking only the modified pages and - in SORTING mode - by what is generally a far smaller volume of data to be sorted.

In SUMMING mode, the performance difference between overall checking and incremental checking is not very large. However, there is no general rule concerning the performance of the various checking modes in combination with the scope of checking.

3.1.3 Checking for coherence

BCHECK checks whether realms which have been specified as part of a group actually belong to the same version, i.e. whether they are coherent. To do this, it reads from each realm specified the internal version number entered in the DBDIR and the time at which the last change was made (cf. "Consistency time stamp") and compares these values with the values entered in the realm itself. Only those realms for which BCHECK establishes consistency with the DBDIR are coherent.

Following utility routine runs with BALTER and BREORG, coherency within an incremental check is not ensured.

3.2 System environment

Effects on database operation

If BCHECK accesses the original database during the check run, this is referred to as an online check run and can be executed in parallel with SHARED-RETRIEVAL database operation.

Check runs in which BCHECK does not access the original database are known as offline check runs and can be executed in parallel with any database operation mode.

More than one check run can be executed in parallel.

Before each BCHECK run, the database to be checked or the shadow database must be assigned with the following command:

```
/ADD-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=dbname.DBDIR[.copy-name]
```

At startup BCHECK takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

Coherence checking

During each check run, BCHECK performs a coherence check on the databases for which it can access the DBDIR. The following databases are checked for coherence:

- **For overall checking:**
the original database or the shadow database
- **For incremental checking of original ↔ shadow database:**
the original database and the shadow database, provided its DBDIR is available
- **For incremental checking of new shadow database ↔ old shadow database:**
the newer shadow database and the older shadow database, provided its DBDIR is available

The following diagrams show the files needed for the various check runs:

Overall check of original

Original of each realm to be checked

Original of DBDIR

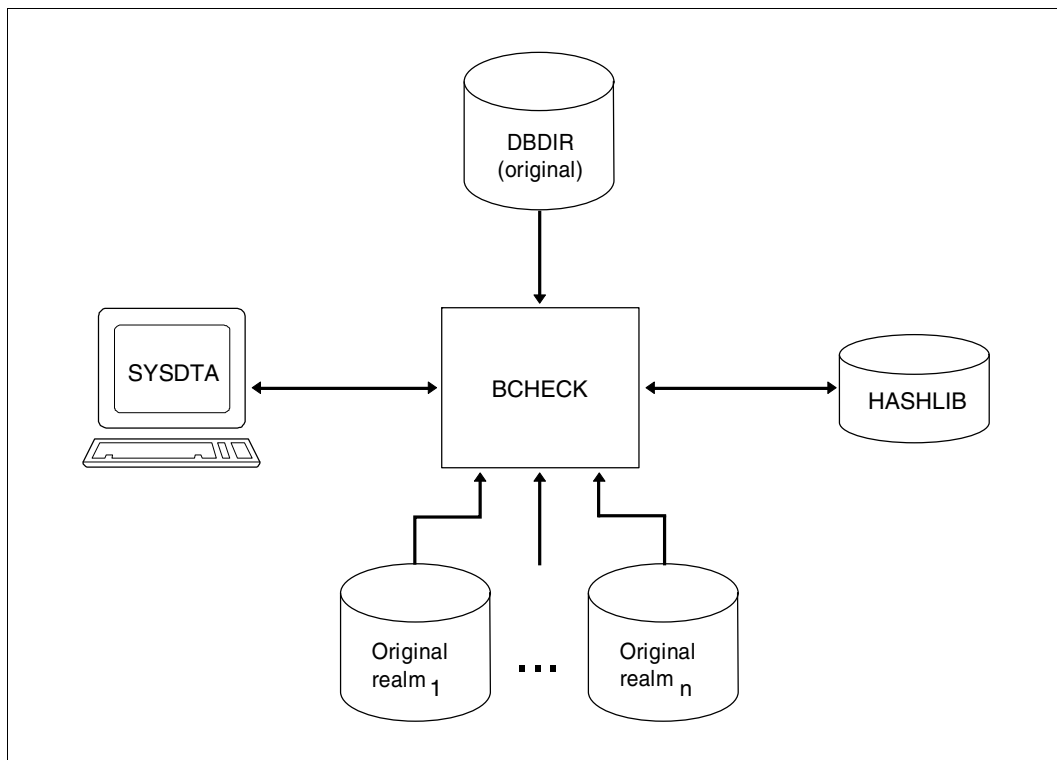


Figure 4: System environment for an overall check of original realms

Overall check of shadow database

Realms of the shadow database that are to be checked

DBDIR of the shadow database

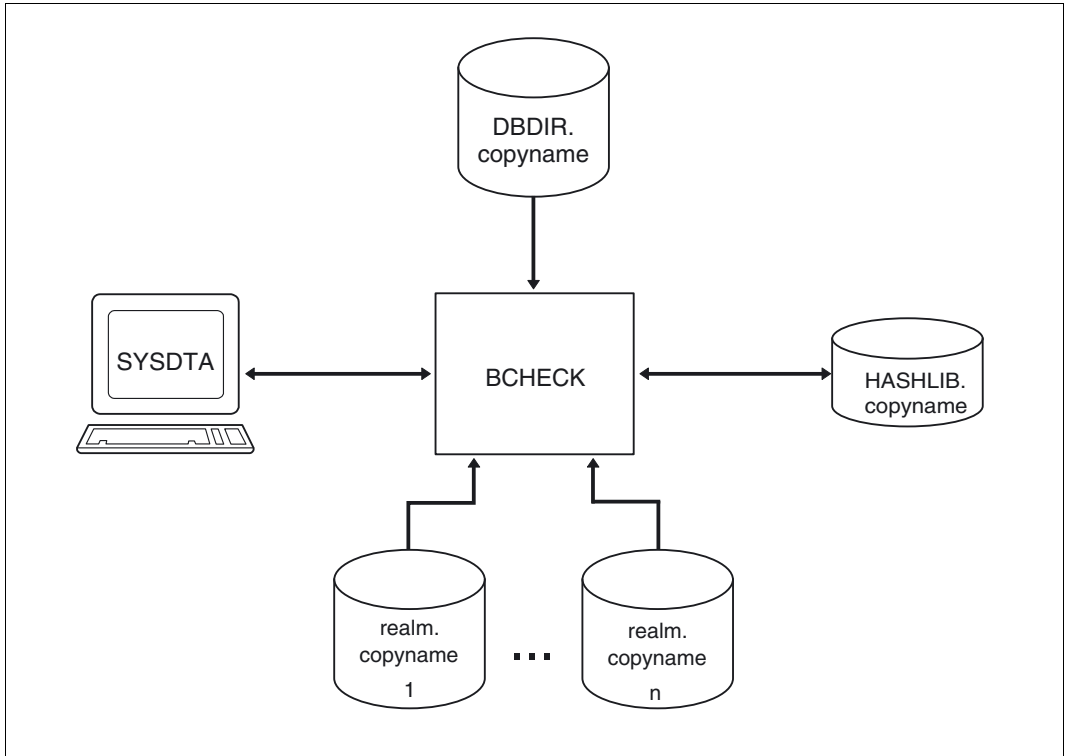


Figure 5: System environment for an overall check of the shadow database

Incremental check of original ↔ shadow database

Original of each realm to be checked

Original of the DBDIR

Realms of the shadow database

Possibly, DBDIR of the shadow database (see [page 71](#))

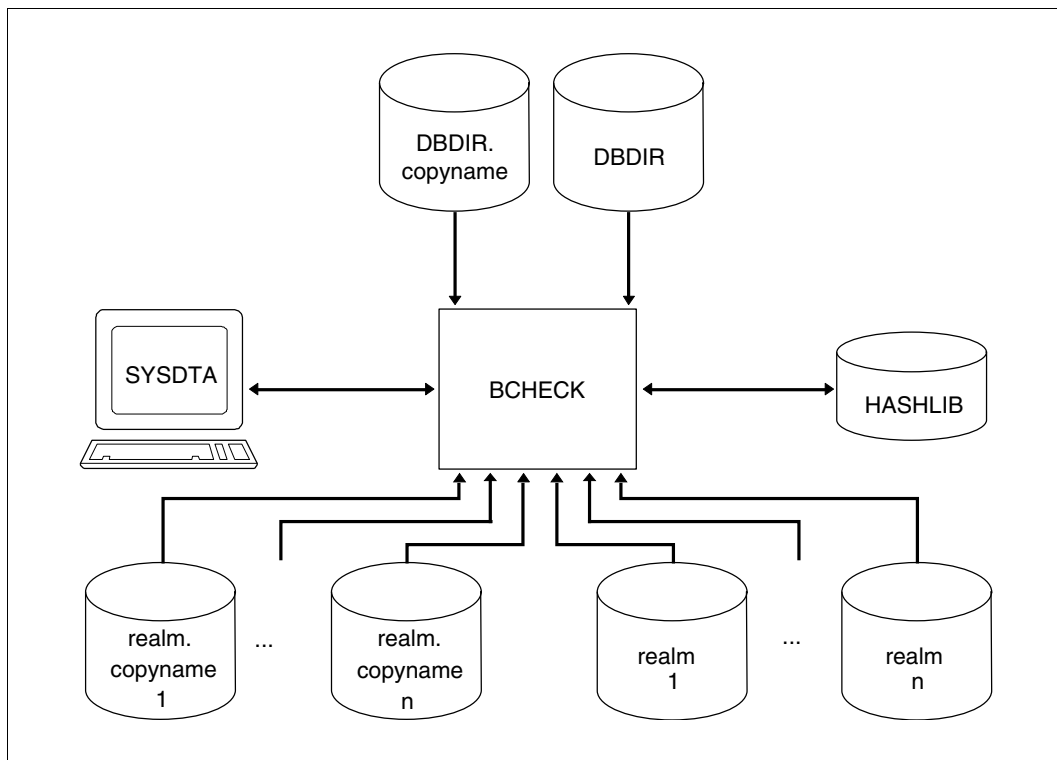


Figure 6: System environment for an incremental check of original ↔ shadow database

Incremental check of new shadow database ↔ old shadow database

Realms of each shadow database that are to be checked

DBDIR for older shadow database or both DBDIRs (see [page 71](#))

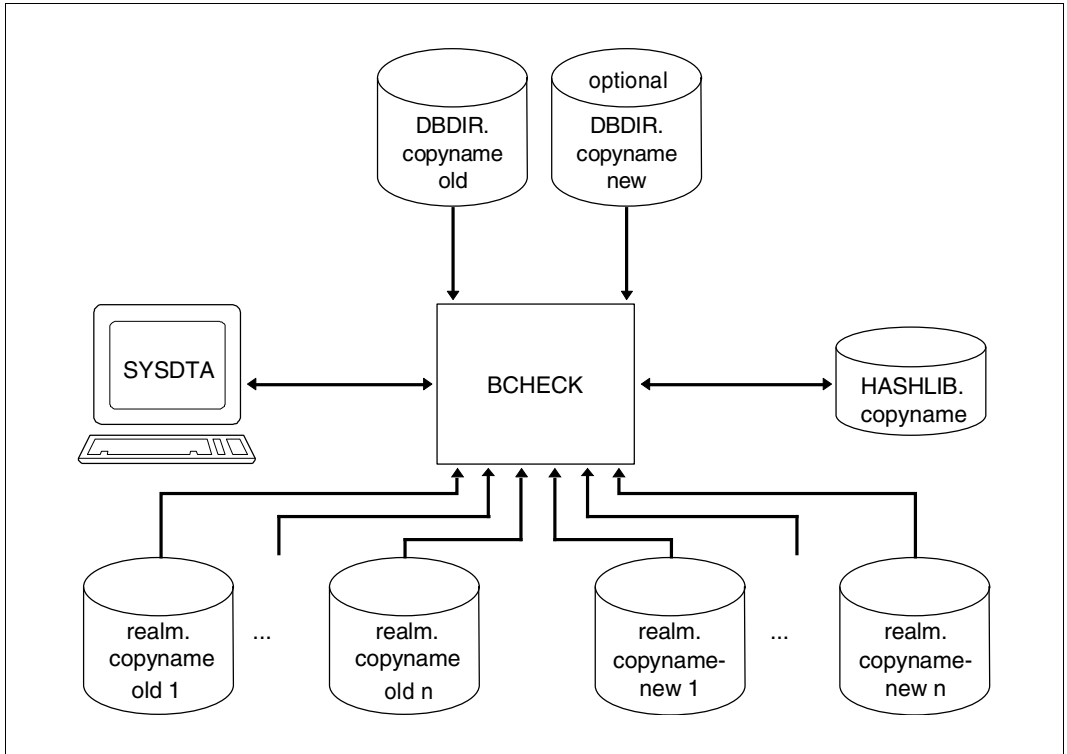


Figure 7: System environment for an incremental check of shadow database ↔ shadow database

Required work files

BCHECK requires two work files for a check run which it sets up automatically on public disk under the user identification under which it was started and which it deletes again after the run has terminated normally. The default link names of these files are SCRTCH1 and SORTWK:

SCRTCH1

BCHECK requires this file during each check run to store a page directory.

SORTWK

Requires the SORT used by BCHECK in the case of a sort check and when checking index values in order to collate and sort the check records. See also the manual "[SORT \(BS2000\)](#)".

If the two work files are to be created explicitly, they must be assigned the following attributes:

Work-file-1

File link name: SCRTCH1

Access method = SAM; fixed record length

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

$$\textit{number-of-pages} \times 16 \text{ Bytes}$$

number-of-pages

Number of relevant pages of all realms to be checked, i.e. the sum of the realm sizes in PAM pages minus the act-key-0 pages, FPA pages, and empty pages.

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.

Work-file-2

File link name: SORTWK

Access method = PAM

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

- RSQ check formula:

$26 \times \text{number-of-check-records}$ Bytes

- Index value check and key value check formula:

$\text{key-length} \times \text{number-of-check-records}$ Bytes

You can ascertain the population involved in a SORTING run by first performing CHECK SUMMING, calculating twice the total amount of check objects from “RECORD/TABLE-OCCURRENCES, CHAIN-SET-MEMBERSHIPS, REFERENCES BETWEEN TABLE-OCCURRENCES and REFERENCES FROM TABLES TO MEMBER-RECORDS as given by DIAGNOSTIC SUMMARY OF BCHECK” and entering this value as *number-of-check-records* in the RSQ check formula.

When performing a SORTING run with key value checking, you add the value ascertained from the key value check formula to the value thus obtained, using double the value of USERKEYS BETWEEN TABLES AND MEMBER-RECORDS as *number-of-check-records*.

When performing a SORTING run with index value checking, you add the value ascertained from the index value check formula to the value obtained so far, using double the value of REFERENCES BETWEEN TABLE-OCCURRENCES as *number-of-check-records*.

As the key length you use the average key length of all keys (weighted with the number of records); in the case of the index value check for keys with duplicates you must take into account an additional 6 bytes. To simplify matters, you can use the maximum key length and, if duplicates are permitted, the additional 6 bytes to ensure that no resource bottleneck occurs when sorting takes place.

In the case of incremental checks, the population refers to the changes compared to the comparison state.

SORT needs Work-file-2 if the main memory affected by the SORTCORE statement is inadequate. The primary allocation should therefore be based on the data population that is to be sorted. There should always be an appropriate secondary allocation in case it is necessary to extend the storage space.

3.3 Using the results of the summing run in a sort run

Using internal results of summing checks

On detecting global summation consistency errors BCHECK generates an output file named

`UTI.tsn.time-stamp.BCHECK`

tsn Four-digit task sequence number

time-stamp

Date and time of file generation

Format: *ddhhmmss*

This file holds the internal results from the summing run, which BCHECK can then evaluate as input information for a subsequent sort run.

In order to have BCHECK evaluate the output file from the summing run, the following actions must be completed before the sort run:

the output file `UTI.tsn.time-stamp.BCHECK` must be assigned the file link name BCHECK, and GENERATE SORTING must be specified in the CHECK statement for the sort run.

If there is no usable data for a SORTING run in the UTI file, the file will be deleted when global errors occur.

Evaluating the output log

In a summing run BCHECK writes the following information to SYSLST:

- all messages and, on detecting global summation consistency errors,
- the command sequence for a sort run coordinated with the summing run (excluding the CREATE-FILE and ADD-FILE-LINK commands for the two work files SORTWK and SCRTCH1).

Editing the SYSLST file

If you assign SYSLST to a file prior to a summing run, you can edit the BCHECK output using EDT. The edited file with the generated SORTING statements can be used for the sort run.

Within the summing run, BCHECK identifies the record types, sets or keys in which errors are present, and the consistency criteria involved. The SORTING statements generated contain only these objects and the corresponding TYPE clauses. Realm selection is limited to the minimum necessary.

3.4 Statements for BCHECK

Statement	Meaning
SORTCORE	Optional; define size of sort buffer
CHECK	Select checking mode and define scope of checking
TYPE	Optional; specify consistency criteria
SCHEMA NAME	Identify schema
REALM NAME	Specify realms to be checked
RECORD NAME	Optional; specify record types to be checked
SET NAME	Optional; specify sets to be checked
KEY REF	Optional; specify search keys to be checked
END	Terminate statement input

Table 8: BCHECK statements

Entering BCHECK statements interactively

The BCHECK statements comprise

- the control statements
SORTCORE, CHECK, TYPE and SCHEMA
- the object selection statements
REALM, RECORD, SET and KEY.

You may enter statements within each group in any desired order. You must enter a CHECK statement prior to your first object selection statement. Once you have entered an object you may not enter any more control statements, nor may you correct any control statements entered.

Statements can be corrected and re-entered interactively. However, note that syntax errors in the keywords which follow the object selection statements may lead to default values being generated which cannot be corrected.

Define size of sort buffer (SORTCORE)

[SORTCORE IS *n*]

n Number of main memory pages for the sort buffer in 4-Kbyte units;
the value range for *n* is predefined by the SORT.

Default value: 150

The BS2000 utility routine SORT is used by BCHECK in sort checking to sort the check records (see the “[SORT \(BS2000\)](#)” manual).

The SORTCORE statement is used to determine the amount of main memory space used for the sort buffer of the SORT utility routine (see the “[SORT \(BS2000\)](#)” manual). It is optional for sort checking.

The data population for sorting is the same as that on which the size of Work-file-2 is based (see “[Work-file-2](#)” on page 77).

Select checking mode and define extent of checking (CHECK)

```
CHECK [ { [GENERATE ]SORTING } ] [ AGAINST COPY NAME IS copy-name ]
```

GENERATE SORTING

Sort check with evaluation of internal results of earlier summing run

SORTING

Sort check

SUMMING

Summing check

AGAINST COPY

Incremental check

copy-name

Copy name of the shadow database

Default value:

SUMMING

The CHECK statement is used to define whether BCHECK is to perform a sort check or a summing check and an overall check or an incremental check.

You must enter the CHECK statement among the control statements.



If a preceding check detected local consistency errors, these should be corrected before a sort check, otherwise a very large number of global consistency errors may be reported which are only due to the local inconsistencies.

Selecting criteria for the global consistency check (TYPE)

```
[TYPE IS { ALL [EXCEPT type-no-1][, type-no-2]... }
0
type-no-1[, type-no-2]... }]
```

ALL BCHECK checks under all consistency criteria.

type-no

Number of consistency criterion (see below) *type-no*=1...11

ALL EXCEPT *type-no-1*[, *type-no-2*]...

BCHECK checks all consistency criteria other than those listed after EXCEPT.

0 BCHECK checks only locally, i.e. within one page. Related check objects in other pages are not checked.

type-no-1[, *type-no-2*]...

BCHECK checks only under the given consistency criteria.

Default value:

ALL

BCHECK recognizes the following consistency criteria:

<i>type-no</i>	Consistency criterion
1	Correct referencing of record or top/first table occurrence by Act-key in DBTT column
2	Correct chaining of records, including owner record, in a MODE IS CHAIN set
3	Correct chaining of records, including owner record, in a MODE IS CHAIN LINKED TO PRIOR set
4	Correct chaining between first and last table occurrences at level 0
5	Correct chaining of table occurrences between levels
6	Correct chaining of table occurrences between levels in terms of index values
7	Correct chaining of table occurrences within one level
8	Correct referencing of records by level 0 table entries and key values in table and record match
9	Correct referencing of records by indirect CALC table entries and key values in table and record match
10	Correct chaining in CALC table overflow chain
11	Correct chaining in duplicates table overflow chain

Table 9: BCHECK consistency criteria

The table below shows which consistency criteria are checked for the various check objects.

BCHECK statement		Type	Consistency criterion													
			0 1	0 2	0 3	0 4	0 5	0 6	0 7	0 8	0 9	1 0	1 1			
RECORD	WITH LOCATION CHECK or WITH KEYVALUE CHECK	Not-CALC	X													
		Direct-CALC	X											X		
		Indirect-CALC	X										X	X		
	WITHOUT LOCATION ..	All														
SET	WITH INDEX CHECK	Table	X			X	X	X	X	X						
	WITHOUT INDEX CHECK	Table	X			X	X		X	X						
		CHAIN		X												
		CHAIN PRIOR			X											
		SYSTEM set CHAIN	X	X												
		SYSTEM set CH-PR	X		X											
KEY	WITH INDEX CHECK	Table	X			X	X	X	X	X						
		Duplicates table	X			X	X	X	X	X				X		
	WITHOUT INDEX CHECK	Table	X			X	X		X	X						
		Duplicates table	X			X	X		X	X				X		
		Indirect-CALC											X	X		

Table 10: Combinations of check objects and consistency criteria



The specification “ALL” is recommended in all cases. Only in this way is it guaranteed that the check will be complete. BCHECK ignores unavailable DB structures even if the corresponding consistency criteria are specified. Performance is therefore not improved by omitting these criteria.

Identify schema (SCHEMA NAME)

`SCHEMA NAME IS schema-name`

schema-name

Name of the schema whose SIA is to be used by BCHECK in the checking of the database. The following can be specified for *schema-name*:

- *user schema name* to check user realm
- COMPILER-SCHEMA to check the DBCOM
- PRIVACY-AND-IQF-SCHEMA to check the DBDIR

The SCHEMA statement specifies the SIA from which BCHECK is to fetch the schema information for checking the database. If the SCHEMA statement is not specified, the user database is checked.



When checking the DBDIR, BCHECK does not check the record type SSIA-RECORD on account of its special structure (spanned records).

Specify realms to be checked (REALM NAME)

```
REALM NAME IS { ALL [ EXCEPT realm-name-1, ... ]
                realm-name-2, ... }
```

realm-name

Name of a realm in the database; enter information as follows

- for user realms: the realm name defined in the schema DDL per AREA clause
- for the database directory: DATABASE-DIRECTORY
- for the database compiler realm: DATABASE-COMPILER-REALM

ALL BCHECK checks all non-temporary user realms identified as EXISTENT and SWITCHED-ON in the database catalog.

The realms are checked by realm numbers in ascending order.

ALL EXCEPT *realm-name-1,...*

Same meaning as for ALL; however, BCHECK excludes the listed realms from the check.

realm-name-2,...

Names of the realms - listed individually - which BCHECK is to check.

If multiple REALM statements are specified between the control statements and the END statement, BCHECK checks in the sequence specified. At least one realm must be specified.

All HASH routines belonging to CALC pages in the specified realms must be entered in the module UDSHASH and must be present in the HASHLIB; otherwise, further checking will be declined.

Specify record types to be checked (RECORD NAME)

```
[RECORD NAME IS { ALL [ EXCEPT satlname-1, ... ] } [ { WITH } { LOCATION CHECK } ] ]
                { satlname-2, ... } [ { WITHOUT } { KEYVALUE CHECK } ] ]
```

recordname

Name of a record type of the specified schema;

recordname must be located within the specified realm or realms.

ALL BCHECK checks all record types of the named schema which are contained in the realms to be checked.

ALL EXCEPT *recordname-1*,...

Same meaning as for ALL; however, BCHECK excludes the listed record types from the check.

recordname-2,...

Names of the record types - listed individually which BCHECK is to check.

WITH LOCATION CHECK

BCHECK also checks all management structures associated with the storage of a record type, i.e. in the DBTT the references to the records. No key value check takes place.

BCHECK checks for record types defined with LOCATION MODE IS CALC:

- the results of hashing (for primary pages and first overflow pages only)
- the chaining of overflow pages
- the internal structure of the CALC tables and
- the references in the CALC table entries to the records

WITH KEYVALUE CHECK

BCHECK also checks all key fields of a record which occur in any access path.

WITH KEYVALUE CHECK also includes the functionality of WITH LOCATION CHECK.

WITH KEYVALUE CHECK is the default value as of UDS/SQL V2.6.

WITHOUT LOCATION CHECK

BCHECK only checks local information on record types, i.e. it checks whether the record addresses in the DBTT are plausible and whether the record lengths match the entries in the SIA, but it does not generate any check records.

WITHOUT KEYVALUE CHECK

This clause is not permitted.

The RECORD statement specifies for BCHECK the record types to be checked. The statement may be specified as often as desired in any position after the control statements. BCHECK ignores duplicate entries.

Information on CALC keys:

If two or more CALC keys with the same key length exist in a realm (indirect CALC keys or CALC search keys), BCHECK cannot carry out a global check on them unless all CALC keys in this realm have been specified with this key length.

If such a key is selected for checking, BCHECK refuses to perform a global check on it but carries out a local check for the hash area concerned. Since for this local check BCHECK uses all those CALC pages which contain CALC keys of the same length as the CALC keys to be checked, it may be the case that BCHECK will report local consistency errors in CALC pages which do not belong to any of the CALC keys specified.

The key value check of indirect CALC tables for consistency criterion 9 can generally also not be performed for a specified record type. To prevent alleged inconsistencies being flagged in a consistent database - in particular in summing mode - you are recommended to always include all record types in the check.

In a sort check it is generally not possible to pinpoint consistency errors in CALC keys. BCHECK is only able to determine which realm they are in, the RSQ, and the key length.

The occurrence of illegal duplicates of a CALC key in various pages of a hash overflow chain cannot be determined by BCHECK.

Information on the key value check:

Missing SET or KEY clauses containing a member record type which is to be checked lead to the check for the access path concerned not being performed, and in particular also to a required key value check not being performed.

In order to position key fields in a record, BCHECK needs a subschema which contains the complete record type concerned or member record type concerned, including the set.

If key values for a record type which do not occur in any subschema are to be checked, a warning is issued and the check in the record types concerned is then performed without a key value check.

The metadata for key value checking is always generated for all record types. As a result, the warning may also be issued when one of the record types is not to be checked.

Information on distributable lists:

When an indirect CALC key or a CALC search key which belongs to a distributable list is checked, all realms in which the records of the associated member set type can reside must always be specified.

Specify sets to be checked (SET NAME)

```
[SET NAME IS { ALL [ EXCEPT setname-1, ... ] } [ { WITH } ] [ { WITHOUT } ] INDEX CHECK ]
```

setname

Name of a set of the specified schema;

setname must be located within the specified realm or realms.

ALL BCHECK checks all sets of the named schema and their owner record type or member record type, or tables contained in the realms to be checked.

ALL EXCEPT *setname-1*,...

Same meaning as for ALL;

however, BCHECK excludes the listed sets from the check.

setname-2,...

Names of the sets - listed individually - which BCHECK is to check.

WITH INDEX CHECK

Check index values (key value and/or record sequence number or either one of the two, depending on the type of table).

WITH INDEX CHECK refers only to the consistency criteria 5, 6, 7 and 11, and not to the key value check of consistency criteria 8 and 9.

Table checks:

BCHECK checks the index values for correctness of inequality relationships which must exist within the chains of table pages of the same level and between the individual table levels in sort key tables, indexed pointer arrays and indexed lists.

This means that BCHECK performs the following checks:

- For chains of table pages of the same level: whether, in forward chaining (forward pointer), the lowest index value of the successor page is greater than the highest index value of the page in question, and vice-versa for backward chaining (with indexed lists, only for levels > 0)
- For table pages of various levels: whether the index value of a table entry at a level > 0 is always greater than or equal to the greatest index value in the next lowest level of the table to which it points and is less than the smallest index value of the table which follows it (with indexed lists, only for levels > 1)

WITHOUT INDEX CHECK

BCHECK does not check index values

With the SET statement the sets to be checked are specified to BCHECK. For sets, BCHECK checks

- the owner/member relationship via the set connection data (SCD)
- the chaining of member records in the case of MODE IS CHAIN,
- in the case of non-indexed and indexed lists, sort key tables, and non-indexed and indexed pointer arrays:
 - the table header,
 - the address chaining of table pages of the same level,
 - in the case of indexed tables: the pointers between the table pages of various levels and
 - the references in the lowest level table entries to the records.

For each set that BCHECK checks, it also automatically checks the associated owner record type and member record type without LOCATION CHECK. It is thus only necessary to name these record types in the RECORD statement if it is desired to extend the check depth to the extent of WITH LOCATION CHECK.

The SET statement may be specified as often as desired in any position after the control statements. BCHECK ignores duplicate entries.



Implicit sets, i.e. search keys at record type level and search keys belonging to a set are only checked by BCHECK if this is explicitly requested by the KEY statement.

For index checking (WITH INDEX CHECK), BCHECK sorts the check records even for a summing check.

Index values are only checked by BCHECK in an overall check.

Index checking does not include checking of sort sequences in chains and checking of non-indexed lists.

When a distributable list is checked, all realms in which the records of the associated member record type can reside must be specified.

Specify SEARCH keys to be checked (KEY REF)

```
[KEY REF IS { ALL [ EXCEPT keyref-1, ... ] } [ { WITH } ] INDEX CHECK ]
```

keyref Key number of a search key; this can be taken from the SIA PRINT REPORT (see [page 138](#)).

ALL Of the schema named, BCHECK checks all search keys and the relevant record type on which the search key is defined as long as they are contained in the realms to be checked.

ALL EXCEPT *keyref-1*, ...
Same meaning as for ALL
however, BCHECK excludes the listed keys from the check.

keyref-2, ...
Key numbers of the search keys - listed individually - which BCHECK is to check.

WITH INDEX CHECK
Check index values for indexed search key tables (see [page 88](#)).
WITH INDEX CHECK refers only to the consistency criteria 5, 6, 7 and 11, and not to the key value check of consistency criteria 8 and 9.

WITHOUT INDEX CHECK
BCHECK does not check any index values.

The KEY statement is used to indicate to BCHECK the keys to be checked. The search keys may be defined on the record type level or the set level as INDEX-SEARCH or CALC-SEARCH keys. BCHECK also checks duplicate tables. The record type on which a search key is defined is automatically checked by BCHECK at the same time, but without LOCATION CHECK.

To check search keys, BCHECK carries out the same checks as for

- CALC-SEARCH keys:
in a LOCATION CHECK of record types which are defined with LOCATION MODE IS CALC (see [page 87](#));
- INDEX-SEARCH keys:
in the multi-level tables of a set (see [page 88](#)).

In the case of duplicate tables, BCHECK also checks

- the reference to the duplicate header in the table header for plausibility,
- the duplicate table header,
- the index values in the table index entries of a table page for ascending sequence,
- the references to the table entries (DB key lists) from the table index entries for plausibility,
- the chaining of the overflow pages of a duplicates table and
- the record sequence numbers of a table entry for ascending sequence.

The KEY statement may be specified as often as desired in any position after the control statements. BCHECK ignores duplicate entries.



ASC keys or DESC keys cannot be specified in the KEY statement. They are automatically included in the check if the appropriate set is specified in the SET statement.

If the number of an ASC or DESC key is specified in the KEY statement, BCHECK rejects the checking of this key.

For CALC-SEARCH keys having the same key length, the same restriction applies as for indirect CALC keys (see [page 87](#)).

The occurrence of illegal duplicates of a CALC-SEARCH key in various pages of a hash overflow chain cannot be determined by BCHECK.

For index checking (WITH INDEX CHECK), BCHECK sorts the check records even for a summing check.

Index values are only checked by BCHECK in an overall check.

When search keys or CALC search keys which belong to a distributable list are checked, all realms in which the records of the associated member set type can reside must always be specified.

3.5 Command sequence to start BCHECK

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

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=SCRATCH1, FILE-NAME=work-file-1,
      ACCESS-METHOD=*SAM]
02  [/CREATE-FILE FILE-NAME=work-file-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[.copy-name]
04  [/ADD-FILE-LINK LINK-NAME=BCHECK, -
      FILE-NAME=[:catid:]UTI.tsn.time-stamp.BCHECK]
05  [/ASSIGN-SYSLST TO=filename]
06  /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=version
07  /START-UDS-BCHECK
08  bcheck statements
09  END
10  [/ASSIGN-SYSLST TO=*PRIMARY]

```

- 03 In this case specifying *:catid:* is permitted (see the “[Database Operation](#)” manual).
- 04 specified in a sort run to make BCHECK evaluate the summing run output file *UTI.tsn.time-stamp.BCHECK*
- 05 specified in a summing run if the output log is to be used in a subsequent sort run.
- 06 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.
- 07 The UDS/SQL utility routine can also be started with the alias BCHECK.

3.6 BCHECK examples

The database CUSTOMER (see [page 27](#)) appears as shown below after the restructuring operation (see the “[Creation and Restructuring](#)” manual):

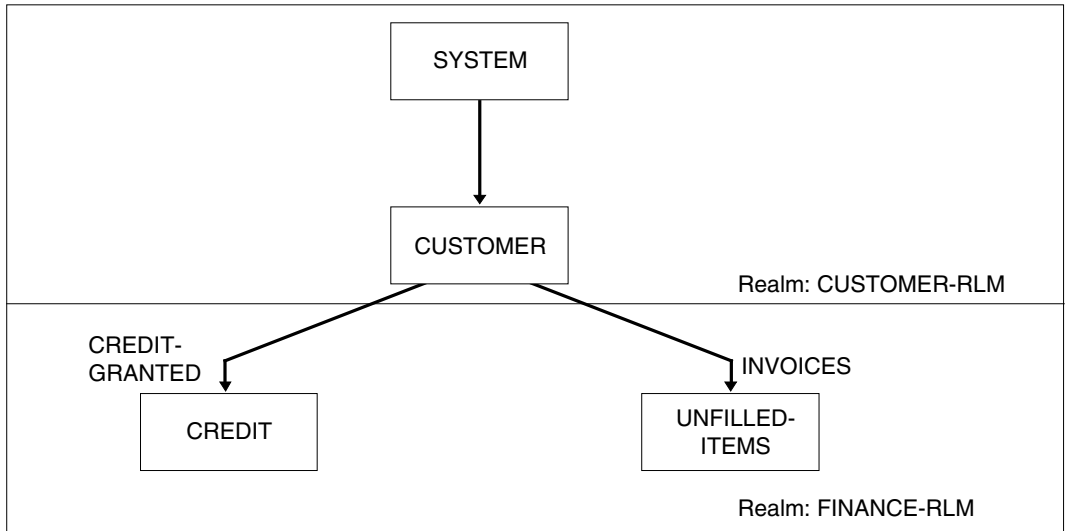


Figure 8: Database CUSTOMER after restructuring

After a short period of operation with the restructured database CUSTOMER, it should be checked for consistency using BCHECK. The following states of the database exist:

- shadow database directly after restructuring (*copynome.AFTRESTR*)
- original database (earlier state than AFTRESTR)

Example 1

Complete summing check of the shadow database after restructuring (AFTRESTR):

```

/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CUSTOMER.DBDIR.AFTRESTR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.8A00
/START-UDS-BCHECK
***** START          BCHECK          (UDS/SQL V2.8 0000 )    2015-06-28  11:40:26
CHECK SUMMING
SCHEMA CUSTOMER-LIST
REALM ALL
RECORD ALL
SET ALL WITH INDEX CHECK
KEY ALL WITH INDEX CHECK
END
***** FOR INDEX CHECKS A SORTING MUST BE PERFORMED EVEN FOR THE SUMMING
          PROCEDURE

***** THE INPUT IS CORRECT, THE CHECK RUN IS STARTING NOW
***** ANALYSING NEW STATE OF REALM CUSTOMER-RLM FROM FILE
          :SQL2:$XXXXXXXXX.CUSTOMER.CUSTOMER-RLM.AFTRESTR
          100 NON EMPTY PAGES HAVE BEEN ANALYSED
***** ANALYSING NEW STATE OF REALM FINANCE-RLM FROM FILE
          :SQL2:$XXXXXXXXX.CUSTOMER.FINANCE-RLM.AFTRESTR
          6 NON EMPTY PAGES HAVE BEEN ANALYSED
***** SORTING   STARTED          DATE AND TIME 2015-06-28  11:40:26
***** SORTING COMPLETED        DATE AND TIME 2015-06-28  11:40:26

***** DIAGNOSTIC SUMMARY OF BCHECK

          NO WARNINGS
          NO ERRORS
          NO SYSTEM-ERRORS

11379 LOCAL CHECKS HAVE BEEN DONE
126 RECORD/TABLE-OCCURRENCES HAVE BEEN CHECKED AGAINST DBTT
68 CHAIN-SET-MEMBERSHIPS HAVE BEEN CHECKED
52 REFERENCES BETWEEN TABLE-OCCURRENCES HAVE BEEN CHECKED
72 REFERENCES FROM TABLES TO MEMBER-RECORDS HAVE BEEN CHECKED
72 USERKEYS BETWEEN TABLES AND MEMBER-RECORDS HAVE BEEN CHECKED

          NO EASY (MINOR) LOCAL CONSISTENCY ERRORS
          NO FATAL (SERIOUS, STRUCTURAL) LOCAL CONSISTENCY ERRORS

          NO GLOBAL CONSISTENCY ERRORS
          NO GLOBAL INDEX CHECK HAS BEEN DONE

```

```

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :           26
***** NORMAL END   BCHECK      (UDS/SQL V2.8 0000 )   2015-06-28  11:40:26

```

Example 2

The original database is checked against the shadow database that was checked by BCHECK before restructuring and found to be error-free (AFTRESTR). While this incremental summing check is performed, BCHECK also checks both databases for coherence:

```

/ADD-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=CUSTOMER.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,   VERSION=02.8A00
/START-UDS-BCHECK
***** START      BCHECK      (UDS/SQL V2.8 0000 )   2015-06-28  11:51:11
CHECK SUMMING AGAINST COPY NAME AFTRESTR
SCHEMA CUSTOMER-LIST
REALM ALL
RECORD ALL
SET ALL
KEY ALL
END

***** THE INPUT IS CORRECT, THE CHECK RUN IS STARTING NOW
***** ANALYSING NEW STATE OF REALM CUSTOMER-RLM FROM FILE
      :SQL2:$XXXXXXXXX.CUSTOMER.CUSTOMER-RLM
      AGAINST OLD STATE FROM FILE :SQL2:$XXXXXXXXX.CUSTOMER.CUSTOMER-RLM.AFTRESTR
      50 CHANGED PAGES HAVE BEEN ANALYSED
***** ANALYSING NEW STATE OF REALM FINANCE-RLM FROM FILE
      :SQL2:$XXXXXXXXX.CUSTOMER.FINANCE-RLM
      AGAINST OLD STATE FROM FILE :SQL2:$XXXXXXXXX.CUSTOMER.FINANCE-RLM.AFTRESTR
      2 CHANGED PAGES HAVE BEEN ANALYSED

+++++ GLOBAL CONSISTENCY ERRORS IN TYPE-NR :           1

+++++ GLOBAL CONSISTENCY ERRORS IN REF-NR  :           2

***** DIAGNOSTIC SUMMARY OF BCHECK

      NO WARNINGS
      NO ERRORS
      NO SYSTEM-ERRORS

14474 LOCAL CHECKS HAVE BEEN DONE
      246 RECORD/TABLE-OCCURRENCES HAVE BEEN CHECKED AGAINST DBTT
      136 CHAIN-SET-MEMBERSHIPS HAVE BEEN CHECKED
      102 REFERENCES BETWEEN TABLE-OCCURRENCES HAVE BEEN CHECKED
      144 REFERENCES FROM TABLES TO MEMBER-RECORDS HAVE BEEN CHECKED

```

```

144 USERKEYS BETWEEN TABLES AND MEMBER-RECORDS HAVE BEEN CHECKED

NO EASY (MINOR) LOCAL CONSISTENCY ERRORS
NO FATAL (SERIOUS, STRUCTURAL) LOCAL CONSISTENCY ERRORS

+++++          GLOBAL CONSISTENCY ERRORS. DO A SORTING CHECK
                NO GLOBAL INDEX CHECK HAS BEEN DONE

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :           48
***** NORMAL END  BCHECK      (UDS/SQL V2.8 0000 )   2015-06-28  11:51:11

```

Extracts from the SYSLST log:

```

.
.
.
/REMARK  START GENERATED BCHECK-STATEMENTS
/ADD-FILE-LINK LINK-NAME=BCHECK,              -
/  FILE-NAME=$XXXXXXXX.UTI.1GZE.01132838.BCHECK
/ADD-FILE-LINK LINK-NAME=DATABASE,            -
/  FILE-NAME=$XXXXXXXX.CUSTOMER.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=XX.XXXX
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
/START-UDS-BCHECK
CHECK GENERATE SORTING AGAINST COPY AFTRESTR
TYPE      IS 1
SCHEMA NAME IS CUSTOMER-LIST
REALM NAME IS CUSTOMER-RLM
REALM NAME IS FINANCE-RLM
RECORD NAME IS CUSTOMER                      WITH LOCATION CHECK
SET NAME IS APPROVED-CREDITS                 WITHOUT INDEX CHECK
KEY REF IS 1                                 WITHOUT INDEX CHECK
END
.
.
.

```

Since errors were detected in the summing check, a sort check is performed to localize them. The command sequence for the limited sort check appears after REMARK in the SYSLST output of the summing check.

Example 3

A limited sort check is performed to localize the errors more precisely. This is done by using the statements listed after REMARK.. in the log of the summing check.

```

/ADD-FILE-LINK BCHECK, $XXXXX.UTI.1GZE.01132838.BCHECK
/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=CUSTOMER.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.8A00
/SET-JV-LINK LINK-NAME=UDSPS01,JV-NAME=UDS-PUB-DECL
/START-UDS-BCHECK
***** START          BCHECK          (UDS/SQL V2.8 0000 )    2015-06-28   09:54:00
CHECK GENERATE SORTING AGAINST COPY AFTRESTR
TYPE          IS 1
SCHEMA NAME IS CUSTOMER-LIST
REALM NAME IS CUSTOMER-RLM
REALM NAME IS FINANCE-RLM
RECORD NAME IS CUSTOMER          WITH LOCATION CHECK
SET NAME IS APPROVED-CREDITS          WITHOUT INDEX CHECK
KEY REF IS 1          WITHOUT INDEX CHECK
END

***** THE INPUT IS CORRECT, THE CHECK RUN IS STARTING NOW.
***** ANALYSING NEW STATE OF REALM CUSTOMER-RLM FROM FILE
:SQL2:$XXXXXXXXX.CUSTOMER.CUSTOMER-RLM
AGAINST OLD STATE FROM FILE :SQL2:$XXXXXXXXX.CUSTOMER.CUSTOMER-RLM.AFTRESTR
50 CHANGED BLOCKS HAVE BEEN ANALYSED.
***** ANALYSING NEW STATE OF REALM FINANCE-RLM FROM FILE
:SQL2:$XXXXXXXXX.CUSTOMER.FINANCE-RLM
AGAINST OLD STATE FROM FILE :SQL2:$XXXXXXXXX.CUSTOMER.FINANCE-RLM.AFTRESTR
2 CHANGED BLOCKS HAVE BEEN ANALYSED.

.
.
.
***** DIAGNOSTIC SUMMARY OF BCHECK

NO WARNINGS
NO ERRORS
NO SYSTEM-ERRORS

10062 LOCAL CHECKS HAVE BEEN DONE
144 RECORD/TABLE-OCCURRENCES HAVE BEEN CHECKED AGAINST DBTT
0 CHAIN-SET-MEMBERSHIPS HAVE BEEN CHECKED
0 REFERENCES BETWEEN TABLE-OCCURRENCES HAVE BEEN CHECKED
0 REFERENCES FROM TABLES TO MEMBER-RECORDS HAVE BEEN CHECKED

NO EASY (MINOR) LOCAL CONSISTENCY ERRORS
NO FATAL (SERIOUS, STRUCTURAL) LOCAL CONSISTENCY ERRORS

```

```
+++++          1 GLOBAL CONSISTENCY ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :          67
***** NORMAL END  BCHECK      (UDS/SQL V2.8 0000 )    2015-06-28  09:54:01
```

Extracts from the SYSLST log:

```
.
.
.
***** THE FOLLOWING CHANGE OF DATABASE IS INCONSISTENT
CHECK-ELEMENT : 0002 00000021 0002 FFFF 03000066 01 00 0000 00000000
FOR OWNER-DBKEY 0002-000000000021 DBTT-COL 2
      A TABLE-OCCURRENCE AT TABLE-LEVEL MAX. WAS ADDED TO PAGE 03000066
      WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE DBTT.

.
.
.
```

3.7 Messages

The messages output by BCHECK can be divided into four groups:

- **Warnings**

Warnings indicate circumstances that do not impair normal execution of BCHECK. However, it is possible that not all of the objects specified by the user are checked. Job switch 30 is set.

- **Error messages (errors, system errors)**

Error messages indicate circumstances that impair or prevent normal execution of BCHECK or result in BCHECK aborting before completion. Job switch 31 is set.

- **Execution messages**

Execution messages provide information on BCHECK execution and are output to SYSOUT and SYSLST.

- **Consistency error messages**

Consistency error messages indicate consistency errors in the examined database. They are output to SYSLST. The job switches 26 (MINOR LOCAL), 27 (SERIOUS LOCAL and STRUCTURAL LOCAL) and/or 28 (GLOBAL) are set.

3.7.1 Warnings

Following input, BCHECK performs an analysis comparing the objects specified with the objects of the schema.

If this analysis shows that not all of the realms needed for the checking requested have been specified, and only partial checking is possible, a corresponding warning is issued.

Warnings in a BCHECK run with the “CHECK GENERATE SORTING” statement which refer to missing specification of objects to be checked can be ignored as a proper check already took place in the associated “CHECK SUMMIMG” run.

Conditions that cause warnings to be issued are, for example:

- An indirect CALC area is to be checked and not all CALC areas of the realm with keys with the same key length have been specified.
- A set or key is to be checked and not all realms containing owner or member records or tables have been specified.
- A record type is to be checked and not all realms with records of this type have been specified.

3.7.2 Error messages

If the analysis of the specified objects, described above under warnings, indicates that specified objects cannot be checked, corresponding error messages are issued and the BCHECK run is aborted.

Conditions that cause error messages to be issued are, for example:

- No realm has been specified.
- A set or key is to be checked and the realms for owner or member records or tables have not been specified.
- A record type, set or key is to be checked and the DBTT realm has not been specified.

These and other error messages that do not result from user input are explained in the “[Messages](#)” manual. The error messages also cover any system errors that may occur.

3.7.3 Execution messages

Messages relating to execution during the analysis phase

```
CHECK OF INDICES IS IMPOSSIBLE WITHOUT TOTAL CHECK OF THE DATABASE: THE INDEX  
CLAUSE IS IGNORED.
```

Meaning

During incremental checking, consistency criterion 6 cannot be checked (correct chaining of table pages between levels with respect to their index values).

```
FOR INDEX CHECKS A SORTING MUST BE PERFORMED EVEN FOR THE SUMMING PROCEDURE.
```

Meaning

In an index check the check records must be sorted, even for a summing check. During input analysis it is determined that consistency criterion 6 is to be checked (correct chaining of table pages between levels with respect to their index values).

```
***** THE INPUT IS CORRECT, THE CHECK RUN IS STARTING NOW.
```

Meaning

Input analysis did not generate any error messages. Actual consistency checking is beginning.

Messages relating to the realm currently being processed

***** ANALYSING NEW STATE OF REALM *realm-name* FROM FILE *file-name*. *number* NON EMPTY BLOCKS HAVE BEEN ANALYSED.

Meaning

Total checking, with statistical information. If no DBTT anchor pages, no DBTT pages, no CALC pages and only empty data pages exist, only administration data must be checked in the realm. In this case only this check is performed, but no message is issued for the realm.

***** ANALYSING NEW STATE OF REALM *realm-name* FROM FILE *file-name* AGAINST OLD STATE FROM FILE *file-name*. *number* CHANGED BLOCKS HAVE BEEN ANALYSED.

Meaning

Incremental check ORIG/COPY ↔ COPY, with statistical note

Messages for the sort phase

***** SORTING STARTED DATE AND TIME YYYY-MM-DD HH:MM:SS

***** SORTING COMPLETED DATE AND TIME YYYY-MM-DD HH:MM:SS

Meaning

These messages are output to SYSOUT and SYSLST immediately before and after all check records have been sorted. The following actions are performed between the two messages:

- DB access with the local consistency checks and generation of the check records
- Sorting of the check records
- Evaluation of the check records and, if required, generation of the consistency error messages

Messages relating to consistency checking in summing checks

```

+++++ GLOBAL CONSISTENCY ERRORS IN {TYPE-NR} {type-no}
                                   {REF-NR} {ref-no}
[ - SYSTEM-ANCHOR-DBKEY = 0000-rsq          ]
[ - KEY-LENGTH           = key-length       ]
[ - DBTT-COL-NR         = dbtt-column-no    ]
[ - AREA-REF            = area-ref          ]

```

type-no Designates the consistency criteria for which BCHECK has detected consistency errors (see [page 82](#))

ref-no Designates the record types that are faulty with respect to consistency criteria 1-8

type-no =1 Record type for the faulty DBTT

type-no =2-8 Owner record type for the corresponding set

0000-rsq

Output in conjunction with *ref-no* =0; designates the DB key of an anchor record.

key-length

Output with consistency criteria 9 and 10; designates the length of the CALC key.

dbtt-column-no

Output with consistency criterion 11; designates the column number in the DBTT

area-ref

Output with consistency criteria 9-11

type-no = 9,10 Realm in which the indirect CALC table is located.

type-no = 11 Realm in which the duplicates table is located.

SUMMARY report

***** DIAGNOSTIC SUMMARY OF BCHECK

{NO/ *number*} WARNINGS refers to the warnings
 {NO/ *number*} ERRORS and
 {NO/ *number*} SYSTEM-ERRORS refers to the error messages

If a consistency check was possible, additional statistical messages are issued:

number RECORD/TABLE ...

Number of references checked from the actual key in a DBTT column to the record or to the highest table page or first table page (consistency criterion 1; see [page 82](#))

number CHAIN-SET-...

Number of records checked

- in a set MODE IS CHAIN including the owner record (consistency criterion 2)
- in a set MODE IS CHAIN LINKED TO PRIOR including the owner record (consistency criterion 3)

number REFERENCES BETWEEN ...

Number of references checked between

- the first table page and the last table page at level 0 (consistency criterion 4)
- the table pages between levels (consistency criterion 5)
- the table pages between levels with respect to their index values (consistency criterion 6)
- the table pages within one level (consistency criterion 7)
- a CALC table overflow chain (consistency criterion 10)
- a duplicates table overflow chain (consistency criterion 11)

number REFERENCES FROM ...

Number of references checked from

- table entries at level 0 to the records (consistency criterion 8)
- the indirect table entries to the records (consistency criterion 9)

number LOCAL CHECKS HAVE BEEN DONE ...

Total number of local checks performed in all the realms to be checked.

number USERKEYS BETWEEN TABLES AND MEMBER-RECORDS HAVE BEEN CHECKED

Number of checks output under REFERENCES FROM TABLES ... which were executed with key value check.

- **Local check:**

$$\left. \begin{array}{l} \{ \text{NO} \\ \{ \textit{number} \} \end{array} \right\} \left. \begin{array}{l} \{ \text{EASY (MINOR)} \\ \{ \text{FATAL (SERIOUS, STRUCTURAL)} \} \end{array} \right\} \text{LOCAL CONSISTENCY ERRORS.}$$

Number of consistency errors which BCHECK has discovered in the database during the check run

EASY (MINOR) LOCAL CONSISTENCY ERRORS

Less important consistency errors, i.e. local errors that do not impede database operation.

FATAL (SERIOUS, STRUCTURAL) LOCAL CONSISTENCY ERRORS

Severe consistency errors, i.e. local errors that must be rectified since they impair or prevent correct database operation.

SERIOUS identifies an error which generally only affects a single object contained on a page.

STRUCTURAL indicates that several objects on a page are affected, or the page as a whole; as a result a large number of global consistency errors can be reported just for this one page.

- **Summing check:**

$$\left. \begin{array}{l} \{ \text{NO GLOBAL CHECK HAS BEEN DONE} \\ \{ \text{NO GLOBAL CONSISTENCY ERRORS} \\ \{ \text{GLOBAL CONSISTENCY ERRORS. DO A SORTING CHECK} \} \} \end{array} \right\}$$

These messages refer to consistency criteria 1-5 and 7-11; see [page 82](#).

$$\left. \begin{array}{l} \{ \text{NO GLOBAL INDEX CHECK HAS BEEN DONE} \\ \left\{ \begin{array}{l} \{ \text{NO} \\ \{ \textit{number} \} \} \text{GLOBAL CONSISTENCY ERRORS IN INDEX CHECK} \end{array} \right\} \end{array} \right\}$$

These messages refer only to consistency criterion 6 ([Correct chaining of table occurrences between levels in terms of index values](#)).

- **Sort check:**

```
{ NO GLOBAL CHECK HAS BEEN DONE }
{ { NO } GLOBAL CONSISTENCY ERRORS }
{ { number } }
```

Number of global consistency errors with respect to consistency criteria 1-11; see also [page 82](#).

*****END OF DIAGNOSTIC SUMMARY

Meaning

End of the SUMMARY report

3.7.4 Consistency error messages

If BCHECK has detected local or global consistency errors, it reports these on SYSLST; however, global consistency errors are only reported during a sort check. Additionally it sets a job switch (see [section “Usage of job switches” on page 132](#)).

3.7.4.1 Global consistency errors (without index check)

Structure of the message key:

Messages concerning global consistency errors begin with a five-character number, starting with “G” for “Global” and followed by two two-digit numbers:

The first number defines the consistency criterion of the TYPE statement. In the display of consistency criterion 1 a distinction is still made between DBTT column 0 (value 00) and DBTT column > 0 (value 01).

The second number has the following meaning:

- 00 Existence record from the original database (or of the newer shadow database in the case of an incremental check)
- 01 Reference record from the original database (or of the newer shadow database in the case of an incremental check)
- 02 Existence record from the (older) shadow database in the case of an incremental check
- 03 Reference record from the (older) shadow database in the case of an incremental check

Error messages for global consistency errors which are not related to index checking, start with the header line

```
THE FOLLOWING CHANGE OF DATABASE IS NOT CONSISTENT
```

If BCHECK discovers additional entries relevant to an actually consistent global relationship (e.g. two record entries with the same DB key, with the DBTT entry referring to one of the two), BCHECK lists the consistent relationship after the consistency error message. The header line:

```
THE FOLLOWING DATABASE CHANGES MAY BE CONNECTED WITH THE CONSISTENCY ERROR
```

is followed in this case by the relevant messages without the additional information:

```
WITH NO CORRESPONDING CHANGE .....
```

In the following, all messages for global consistency errors - except for those related to index checking - are explained. Note that messages containing the text

..... DELETED FROM

are only output during incremental checking. These messages always relate to the older shadow database; however, the explanations given for them refer to the original (or the newer shadow database) and **are only relevant** if the older shadow database is consistent. Should the older shadow database not be consistent, the explanation for the corresponding message with ADDED TO applies to the consistency errors in this shadow database.

G0000 FOR DBKEY *dbkey* THE RECORD-OCCURRENCE WAS ADDED TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF A REFERENCE FROM DBTT-COL-0.

Meaning

The DBTT entry for record *dbkey* in page *act-key* is incorrect.

G0001 FOR DBKEY *dbkey* AN ENTRY WAS ADDED TO DBTT-COL-0 POINTING TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF LOCATION OF THE RECORD-OCCURRENCE.

Meaning

For record *dbkey* there exists a DBTT entry which refers to page *act-key*. The record is not contained in this page.

G0002 FOR DBKEY *dbkey* THE RECORD-OCCURRENCE WAS DELETED FROM PAGE *act-key* WITH NO CORRESPONDING CHANGE OF A REFERENCE FROM THE DBTT-COL-0.

Meaning

For record *dbkey* there exists a DBTT entry which refers to page *act-key*. The record is not contained in this page.

G0003 FOR DBKEY *dbkey* AN ENTRY WAS DELETED FROM DBTT-COL-0 POINTING TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF LOCATION OF THE RECORD-OCCURRENCE.

Meaning

The DBTT entry for record *dbkey* in page *act-key* is incorrect.

G0100 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL {*levno*|MAX} WAS ADDED TO PAGE *act-key-1* (POINTING TO NEXT HIGHER TABLE IN PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE {DBTT|SYSTEM-RECORD}.

Meaning

In a table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the DBTT or from the anchor record to the table page with level *levno* in page *act-key-1* is missing. For table pages of the highest level, MAX replaces *levno*.

G0101 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY REFERRING TO A TABLE-OCCURRENCE AT LEVEL *levno* WAS ADDED TO THE {DBTT|SYSTEM-RECORD} POINTING TO A PAGE *actkey-1* (WITH THE REFERENCE COMING FROM PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF LOCATION OF THE TABLE-OCCURRENCE POINTED AT.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, a table page of level *levno*, which is referenced from the DBTT or from the anchor record, is missing from page *act-key-1*.

G0102 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL {*levno*|MAX} WAS DELETED FROM PAGE *act-key-1* (POINTING TO NEXT HIGHER TABLE IN PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE {DBTT|SYSTEM-RECORD}.

Meaning

In a table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, a table page of level *levno* in page *act-key-1*, which is referenced from the DBTT or from the anchor record, is missing. For table pages of the highest level, MAX replaces *levno*.

G0103 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY REFERRING TO A TABLE-OCCURRENCE AT LEVEL *levno* WAS DELETED FROM THE TABLE AT NEXT HIGHER LEVEL POINTING TO A PAGE *act-key-1* (WITH THE REFERENCE COMING FROM PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF LOCATION OF THE TABLE-OCCURRENCE POINTED AT.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the next higher level (table entry in page *act-key-2*) to a table page with level *levno* in page *act-key-1* is missing.

G0200

G0300 FOR OWNER-DBKEY *dbkey-1* SET-REF *setref* A RECORD-OCCURRENCE WAS ADDED TO THE OWNER'S SET CHAIN WITH MEMBER-DBKEY *dbkey-2* (POINTING TO PRIOR DBKEY *dbkey-3*) WITH NO CORRESPONDING CHANGE OF A FORWARD POINTER IN THE SET CHAIN.

Meaning

In set *setref*, no reference to record *dbkey-2* exists in the set occurrence of owner. For backward chaining, member record *dbkey-3* which should have contained the reference is also given.

G0201

G0301 FOR OWNER-DBKEY *dbkey-1* SET-REF *setref* A FORWARD POINTER WAS ADDED TO THE OWNER'S SET CHAIN POINTING TO MEMBER-DBKEY *dbkey-2* (FROM DBKEY *dbkey-3*) WITH NO CORRESPONDING CHANGE OF THE MEMBER-RECORD POINTED AT.

Meaning

In set *setref*, there exists in the set occurrence of owner *dbkey-1* a reference to record *dbkey-2* which is not, however, a member of the chain. For backward chaining, member record *dbkey-3* which contains the reference is also given.

G0202
G0302

FOR OWNER-DBKEY *dbkey-1* SET-REF *setref* A RECORD-OCCURRENCE WAS DELETED FROM THE OWNER'S SET CHAIN WITH MEMBER-DBKEY *dbkey-2* (POINTING TO PRIOR DBKEY *dbkey-3*) WITH NO CORRESPONDING CHANGE OF A FORWARD POINTER IN THE SET CHAIN.

Meaning

In set *setref*, there exists in the set occurrence of owner *dbkey-1* a reference to record *dbkey-2* is not, however, a member of the chain. For backward chaining, member record *dbkey-3* which contains the reference is also given.

G0203
G0303

FOR OWNER-DBKEY *dbkey-1* SET-REF *setref* A FORWARD POINTER WAS DELETED FROM THE OWNER'S SET CHAIN POINTING TO MEMBER-DBKEY *dbkey-2* (FROM DBKEY *dbkey-3*) WITH NO CORRESPONDING CHANGE OF THE MEMBER-RECORD POINTED AT.

Meaning

In set *setref*, no reference to record *dbkey-2* exists in the set occurrence of the owner. For backward chaining, member record *dbkey-3* which should have contained the reference is also given.

G0400

FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A LAST TABLE-OCCURRENCE WAS ADDED TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE TOP-TABLE.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the table page of the highest level corresponding to the last table page of the lowest level in page *act-key* is missing.

Or:

In the flat table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the first table page corresponding to the last table page (*act-key*) is missing.

G0401

FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* THE REFERENCE TO THE LAST TABLE WAS ADDED TO THE TOP-TABLE POINTING TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF LOCATION OF THE LAST TABLE-OCCURRENCE.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, page *act-key* which is referenced in the table header of the highest level table does not contain the last table page of the lowest level.

Or:

In the flat table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, page *act-key*, which is referenced in the table header of the first page, does not contain the last table page.

G0402 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A LAST TABLE-OCCURRENCE WAS DELETED FROM PAGE *act-key* WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE TOP-TABLE.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, page *act-key*, which is referenced in the table header of the highest level, does not contain the last table page of the lowest level.

Or:

In the flat table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, page *act-key*, which is referenced in the table header of the first page, does not contain the last table page.

G0403 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* THE REFERENCE TO THE LAST TABLE WAS DELETED FROM THE TOP-TABLE POINTING TO PAGE *act-key* WITH NO CORRESPONDING CHANGE OF LOCATION OF THE LAST TABLE-OCCURRENCE.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the table page of the highest level corresponding to the last table page of the lowest level in page *act-key* is missing.

Or:

In the flat table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the first table page corresponding to the last table page (*act-key*) is missing.

G0500 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL {*levno*|MAX} WAS ADDED TO PAGE *act-key-1* (POINTING TO NEXT HIGHER TABLE IN PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE TABLE AT NEXT HIGHER LEVEL.

Meaning

In an indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the next higher level (table entry in page *act-key-2*) to the table page with level *levno* in page *act-key-1* is missing. For table pages of the highest level, MAX replaces *levno*.

G0501 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY REFERRING TO A TABLE-OCCURRENCE AT LEVEL *levno* WAS ADDED TO THE TABLE AT NEXT HIGHER LEVEL POINTING TO A PAGE *actkey-1* (WITH THE REFERENCE COMING FROM PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF LOCATION OF THE TABLE-OCCURRENCE POINTED AT.

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, a table page of level *levno*, which is referenced from the next higher level (table entry in page *act-key-2*), is missing from page *actkey-1*.

G0502 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL $\{levno|MAX\}$ WAS DELETED FROM PAGE *act-key-1* (POINTING TO NEXT HIGHER TABLE IN PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF THE REFERENCE IN THE TABLE AT NEXT HIGHER LEVEL.

Meaning

In an indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, a table page of level *levno* in page *act-key-1*, which is referenced from the next higher level (table entry in page *act-key-2*), is missing. For table pages of the highest level, MAX replaces *levno*.

G0503 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY REFERRING TO A TABLE-OCCURRENCE AT LEVEL *levno* WAS DELETED FROM THE $\{DBTT|SYSTEM-RECORD\}$ POINTING TO A PAGE *act-key-1* (WITH THE REFERENCE COMING FROM PAGE *act-key-2*) WITH NO CORRESPONDING CHANGE OF LOCATION OF THE TABLE-OCCURRENCE POINTED AT.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the reference from the DBTT or from the anchor record to a table page with level *levno* in page *act-key-1* is missing.

G0700 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-1* POINTING TO PRIOR TABLE IN PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF A FORWARD POINTER IN THE TABLE CHAIN.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the forward pointer from page *act-key-2* corresponding to the backward pointer in page *act-key-1* on the same level *levno* is missing.

G0701 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-1* POINTING TO NEXT-TABLE IN PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF A BACKWARD POINTER IN THE TABLE CHAIN.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the backward pointer from page *act-key-2* corresponding to the forward pointer in page *act-key-1* on the same level *levno* is missing.

G0702 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS DELETED FROM PAGE *act-key-1* POINTING TO PRIOR TABLE IN PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF A FORWARD POINTER IN THE TABLE CHAIN.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the backward pointer in page *act-key-1* corresponding to the forward pointer in page *act-key-2* on the same level *levno* is missing.

G0703 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS DELETED FROM PAGE *act-key-1* POINTING TO NEXT-TABLE IN PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF A BACKWARD POINTER IN THE TABLE CHAIN.

Meaning

In the table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the forward pointer from page *act-key-1* corresponding to the backward pointer in page *act-key-2* on the same level *levno* is missing.

G0800 FOR OWNER-DBKEY *dbkey-1* DBTT-COL *col-no* THE MEMBERSHIP-INDICATOR WAS ADDED TO THE RECORD WITH MEMBER-DBKEY *dbkey-2* (LOCATED IN PAGE *act-key*) WITH NO CORRESPONDING CHANGE OF A POINTER IN SOME BOTTOM TABLE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*.

Meaning

In the SEARCH key or sort key table which is determined unambiguously by owner record *dbkey-1* and the DBTT column number *colnr* a table entry at level 0 exists for record *dbkey-2* although this record is not a member of the associated set.

G0801 FOR OWNER-DBKEY *dbkey-1* DBTT-COL *col-no* AN ENTRY WAS ADDED TO A BOTTOM TABLE POINTING TO MEMBER-RECORD WITH DBKEY *dbkey-2* (WITH THE TABLE BEING LOCATED IN PAGE *act-key*) WITH NO CORRESPONDING CHANGE OF THE RECORD-OCCURRENCE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*.

Meaning

In the search key or sort key table which is uniquely identified by owner record *dbkey-1* and DBTT column number *col-no*, there exists for record *dbkey-2* a table entry at level 0 (stored in page *act-key*), although this record is not a member of the corresponding set.

G0802 FOR OWNER-DBKEY *dbkey-1* DBTT-COL *col-no* THE MEMBERSHIP-INDICATOR WAS DELETED FROM THE RECORD WITH MEMBER-DBKEY *dbkey-2* (LOCATED IN PAGE *act-key*) WITH NO CORRESPONDING CHANGE OF A POINTER IN SOME BOTTOM TABLE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*.

Meaning

In the search key or sort key table which is uniquely identified by owner record *dbkey-1* and DBTT column number *col-no*, the table entry at level 0 corresponding to member record *dbkey-2* (stored in page *act-key*) is missing.

G0803 FOR OWNER-DBKEY *dbkey-1* DBTT-COL *col-no* AN ENTRY WAS DELETED FROM A BOTTOM TABLE POINTING TO A MEMBER-RECORD WITH *dbkey-2* (WITH THE TABLE BEING LOCATED IN PAGE *act-key*) WITH NO CORRESPONDING CHANGE OF THE RECORD-OCCURRENCE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*.

Meaning

In the search key or sort key table which is uniquely identified by owner record *dbkey-1* and DBTT column number *col-no*, the table entry at level 0 (stored in the page *act-key*) corresponding to member record *dbkey-2* is missing.

G0900 A RECORD-OCCURRENCE WITH RSQ *rsq* WHICH HAS AN (INDIRECT) CALC-(SEARCH-)KEY WITH KEYLENGTH *key-length* WAS ADDED TO THE PAGE *act-key* WITH NO CORRESPONDING CHANGE IN THE CALC-TABLE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*

Meaning

In the CALC table of the indirect CALC key or CALC-SEARCH key with key length *key-length*, the reference to the record with record sequence number *rsq* stored in page *act-key* is missing.

G0901 FOR AN (INDIRECT) CALC-(SEARCH-)KEY WITH KEYLENGTH *key-length* AN ENTRY WITH RSQ *rsq* WAS ADDED TO A CALC TABLE AT PAGE *act-key* OF THE DATABASE WITH NO CORRESPONDING CHANGE IN THE RELATED RECORD-OCCURRENCE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*

Meaning

In the CALC table of the indirect CALC key or CALC-SEARCH key with key length *key-length*, the entry in page *act-key* with record sequence number *rsq* refers to a non-existent record.

G0902 A RECORD-OCCURRENCE WITH RSQ *rsq* WHICH HAS AN (INDIRECT) CALC-(SEARCH-)KEY WITH KEYLENGTH *key-length* WAS DELETED FROM THE PAGE *act-key* WITH NO CORRESPONDING CHANGE IN THE CALC TABLE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*

Meaning

In the CALC table of the indirect CALC key or CALC-SEARCH key with key length *key-length*, the entry with record sequence number *rsq* refers to a non-existent record.

G0903 FOR AN (INDIRECT) CALC-(SEARCH-)KEY WITH KEYLENGTH *key-length* AN ENTRY WITH RSQ *rsq* WAS DELETED FROM A CALC TABLE AT PAGE *act-key* OF THE DATABASE WITH NO CORRESPONDING CHANGE IN THE RELATED RECORD-OCCURRENCE. CONTENT OF KEYVALUE OR TABLE ENTRY: *keyvalue*

Meaning

In the CALC table of the indirect CALC key or CALC-SEARCH key with key length *key-length*, the reference to the stored record with record sequence number *rsq* is missing.

G1000 IN A HASH OVERFLOW CHAIN FOR KEYLENGTH *key-length* A POINTER TO PRIOR OVERFLOW PAGE *act-key-1* WAS ADDED TO PAGE *actkey-2* WITH NO CORRESPONDING CHANGE OF THE FORWARD POINTER IN THE OVERFLOW CHAIN

Meaning

In a hash overflow chain of the CALC key with key length *key-length*, there exists in page *act-key-2* a backward pointer to page *act-key-1* with no corresponding forward pointer in the chain of overflow pages.

G1001 IN A HASH OVERFLOW CHAIN FOR KEYLENGTH *key-length* A POINTER TO NEXT OVERFLOW PAGE *act-key-1* WAS ADDED TO PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF THE BACKWARD POINTER IN THE OVERFLOW CHAIN.

Meaning

In a hash overflow chain of the CALC key with key length *key-length*, there exists in page *act-key-2* a forward pointer to page *act-key-1* with no corresponding backward pointer in the chain of overflow pages.

G1002 IN A HASH OVERFLOW CHAIN FOR KEYLENGTH *key-length* A POINTER TO PRIOR OVERFLOW PAGE *act-key-1* WAS DELETED FROM PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF THE FORWARD POINTER IN THE OVERFLOW CHAIN.

Meaning

In a hash overflow chain of the CALC key with key length *key-length*, there exists in page *act-key-1* a forward pointer to page *act-key-2* with no corresponding backward pointer in the chain of overflow pages.

G1003 IN A HASH OVERFLOW CHAIN FOR KEYLENGTH *key-length* A POINTER TO NEXT OVERFLOW PAGE *act-key-1* WAS DELETED FROM PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF THE BACKWARD POINTER IN THE OVERFLOW CHAIN.

Meaning

In a hash overflow chain of the CALC key with key length *key-length*, there exists in page *act-key-1* a backward pointer to page *act-key-2* with no corresponding forward pointer in the chain of overflow pages.

G1100 IN THE OVERFLOW CHAIN OF A DUPLICATE TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A BACKWARD POINTER TO PAGE *act-key-2* WAS ADDED TO PAGE *act-key-3* WITH NO CORRESPONDING CHANGE IN THE PREDECESSOR.

Meaning

In the overflow chain of a duplicate table with the main level in page *act-key-1* and DBTT column number *col-no*, there exists in page *act-key-3* a backward pointer to page *act-key-2* with no corresponding forward pointer in the chain of overflow pages.

G1101 IN THE OVERFLOW CHAIN OF A DUPLICATE TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A FORWARD POINTER TO PAGE *act-key-2* WAS ADDED TO PAGE *act-key-3* WITH NO CORRESPONDING CHANGE IN THE SUCCESSOR.

Meaning

In the overflow chain of a duplicate table with the main level in page *act-key-1* and DBTT column number *col-no*, there exists in page *act-key-3* a forward pointer to page *act-key-2* with no corresponding backward pointer in the chain of overflow pages.

G1102 IN THE OVERFLOW CHAIN OF A DUPLICATE TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A BACKWARD POINTER TO PAGE *act-key-2* WAS DELETED FROM *act-key-3* WITH NO CORRESPONDING CHANGE IN THE PREDECESSOR.

Meaning

In the overflow chain of a duplicate table with the main level in page *act-key-1* and DBTT column number *col-no*, there exists in page *act-key-2* a forward pointer to page *act-key-3* with no corresponding backward pointer in the chain of overflow pages.

G1103 IN THE OVERFLOW CHAIN OF A DUPLICATE TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A FORWARD POINTER TO PAGE *act-key-2* WAS DELETED FROM PAGE *act-key-3* WITH NO CORRESPONDING CHANGE IN THE SUCCESSOR.

Meaning

In the overflow chain of a duplicate table with the main level in page *act-key-1* and DBTT column number *col-no*, there exists in page *act-key-2* a backward pointer to page *act-key-3* with no corresponding forward pointer in the chain of overflow pages.

3.7.4.2 Global consistency errors (index check)

Errors in the index value relationships between two chained table pages of an indexed table or between the pages of the overflow chain of a duplicate table are reported by BCHECK with the header:

THE FOLLOWING CHANGE OF DATABASE IS NOT CONSISTENT DUE TO WRONG KEY VALUE RELATIONS IN MATCHING CHECK-RECORDS

If, however, BCHECK reports the following in the header:

THE FOLLOWING CHANGE OF DATABASE IS NOT CONSISTENT DUE TO DIFFERENT KEY LENGTHS IN MATCHING CHECK-RECORDS

then the index values to be compared were of different lengths. In such a case, BCHECK does not perform the actual index check.

The index check messages are listed below.

***THE FOLLOWING DATABASE CHANGES MAY BE CONNECTED WITH THE CONSISTENCY ERROR:

G0500 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-1* POINTING TO NEXT HIGHER TABLE IN PAGE *act-key-2*.

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-2
rsq-2

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, in table page *act-key-2* of level *levno+1* the index value *keyvalue-1 rsq-1* of an entry is smaller than the highest index value *keyvalue-2 rsq-2* in the table page *act-key-1* of the next lower level *levno* to which it refers.

G0501 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY REFERRING TO A TABLE-OCCURRENCE AT LEVEL *levno* WAS ADDED TO THE TABLE AT NEXT HIGHER LEVEL POINTING TO A PAGE *act-key-1* WITH THE REFERENCE COMING FROM PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF LOCATION OF THE TABLE-OCCURRENCE POINTED AT.

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-1
rsq-1

***THE FOLLOWING DATABASE CHANGES MAY BE CONNECTED WITH THE CONSISTENCY ERROR:

G0600 IN AN INDEXED TABLE FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY AT TABLE LEVEL *levno+1* WAS ADDED TO PAGE *act-key-3* POINTING TO A TABLE AT NEXT-LOWER LEVEL *act-key-2* WITH NO CORRESPONDING CHANGE OF THE SUCCESSOR OF THE REFERENCED TABLE

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-2
rsq-2

Meaning

In the indexed table which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, in table page *act-key-3* of level *levno+1* the index value *keyvalue-2 rsq-2* of an entry is not smaller than the lowest index value *keyvalue-1 rsq-1* of successor page *act-key-1* of that table page *act-key-2* to which the entry refers.

G0601 IN AN INDEXED TABLE FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* AN ENTRY AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-1* POINTING TO ITS PREDECESSOR IN THE TABLE CHAIN *act-key-2* WITH NO CORRESPONDING CHANGE OF THE TABLE AT NEXT HIGHER LEVEL.

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-1
rsq-1

***THE FOLLOWING DATABASE CHANGES MAY BE CONNECTED WITH THE CONSISTENCY ERROR:

G0700 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-2* POINTING TO PRIOR TABLE IN PAGE *act-key-1*

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-2
rsq-2

Meaning

In the table chain which is uniquely identified by owner record *dbkey* and DBTT column number *col-no*, the highest index value *keyvalue-1 rsq-1* in table page *act-key-1* is not smaller than the lowest index value *keyvalue-2 rsq-2* in successor page *act-key-2* of the table chain at the same level *levno*.

G0701 FOR OWNER-DBKEY *dbkey* DBTT-COL *col-no* A TABLE-OCCURRENCE AT TABLE LEVEL *levno* WAS ADDED TO PAGE *act-key-1* POINTING TO NEXT-TABLE IN PAGE *act-key-2* WITH NO CORRESPONDING CHANGE OF A BACKWARD POINTER IN THE TABLE CHAIN.

CONTENT OF KEYVALUE OR TABLE ENTRY:

keyvalue-1
rsq-1

G1100 IN THE OVERFLOW CHAIN OF A DUPLICATE-TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A BACKWARD POINTER TO PAGE *act-key-2* WAS ADDED TO PAGE *act-key-3* WITH NO CORRESPONDING CHANGE IN THE PREDECESSOR.

CONTENT OF RSQ:

rsq-1

***THE FOLLOWING DATABASE CHANGES MAY BE CONNECTED WITH THE CONSISTENCY ERROR:

G1101 IN THE OVERFLOW CHAIN OF A DUPLICATE-TABLE WITH MAIN LEVEL *act-key-1* DBTT-COL *col-no* A FORWARD POINTER TO PAGE *act-key-3* WAS ADDED TO PAGE *act-key-2*.

CONTENT OF RSQ:

rsq-2

Meaning

In the overflow chain of the duplicates table with the main level in page *act-key-1* and DBTT column number *col-no*, the highest RSQ value *rsq-2* of the duplicates table in page *act-key-2* is not smaller than the lowest RSQ value *rsq-1* of successor page *act-key-3* of the duplicates table.

3.7.4.3 Local consistency errors

Structure of the message key:

Messages relating to local consistency errors begin with a five-character number starting with “L” for “Local”, followed by a four-digit number which has no semantic meaning.

L0027 STRUCTURAL CONSISTENCY ERROR: FREE-SPACE.DISPL IN THE PAGE HEADER DOES NOT OCCUR AS SMALLEST RECORD-DISPL IN A PAGE INDEX

Meaning

The FPA displacement in the page header does not occur as the smallest record displacement in at least one page index.

L0033 SERIOUS CONSISTENCY ERROR: FPA VALUE FOR {OLD/NEW} {ACT-KEY-0/DBTT ANCHOR PAGE/DBTT PAGE/CALC PAGE/FPA PAGE/TABLE PAGE/DUPLICATE TABLE OVERFLOW PAGE} *act-key* CLAIMS NO FULL PAGE, VALUE = *fpa-value*

Meaning

The central FPA value for an ACT-KEY-0 page, DBTT anchor page, DBTT page, CALC page, FPA page, table page or for a duplicates table overflow page is non-zero.

L0049 MINOR CONSISTENCY ERROR: BNR OF PPP IN TABLE-ENTRY *te-no* IS WRONG

Meaning

The page number in the (indirect CALC) table entry is zero, while the realm ref is non-zero.

L0056 SERIOUS CONSISTENCY ERROR: BNR OF PPP IN TABLE-ENTRY *te-no* IS WRONG

Meaning

The page number in the (indirect CALC) table entry is zero, while the realm ref is non-zero.

L0057 STRUCTURAL CONSISTENCY ERROR: DB-KEY-LIST IN DUPLICATE TABLE FOR TABLE ENTRY *te-no* HAS AN INVALID LENGTH

Meaning

The length of the duplicate table entry is not a multiple of 3 (for a page length of 2048 bytes) or 6 (for a page length of 4000 or 8096 bytes).

L0070 SERIOUS CONSISTENCY ERROR: FPA VALUES FOR {DBTT/CALC}-PAGE(S) BETWEEN *pagenumber* AND *pagenumber* INDICATE EMPTY PAGES OR ARE NOT RECOGNIZED AS {DBTT/CALC}-PAGE(S), REALM IS *realm-name*

Meaning

The central FPA value for the DBTT pages or CALC pages is X'07EC' for a page length of 2048 bytes, X'0F8C' for a page length of 4000 bytes or X'1F8C' for a page length of 8096 bytes or the page layout is not the correct DBTT or CALC page layout.

L0081 STRUCTURAL CONSISTENCY ERROR: TABLE FOR PAGE INDEX *pageindex* IN TABLE LIST DOES NOT OCCUPY A WHOLE PAGE

Meaning

The list alone is intended for the page, but the central FPA value or the number of page index entries or the number of entries reserved for the list contradicts this.

L0084 SERIOUS CONSISTENCY ERROR: FIRST MEMBER DB-KEY *dbkey* IS INVALID (SET-REF = *setref*, SCD-DISPL = *displacement*)

Meaning

The DB key of the first member record in the owner's SCD is invalid.

L0085 SERIOUS CONSISTENCY ERROR: LAST MEMBER DB-KEY *dbkey* IS INVALID (SET-REF = *setref*, SCD-DISPL = *displacement*)

Meaning

The DB key of the last member record in the owner's SCD is invalid.

L0086 SERIOUS CONSISTENCY ERROR: NEXT MEMBER DB-KEY *dbkey* IS INVALID (SET-REF = *setref*, SCD-DISPL = *displacement*)

Meaning

The DB key of the next member record in a member's SCD is invalid.

L0087 SERIOUS CONSISTENCY ERROR: PRIOR MEMBER DB-KEY *dbkey* IS INVALID (SET-REF = *setref*, SCD-DISPL = *displacement*)

Meaning

The DB key of the preceding member record in a member's SCD is invalid.

L0088 SERIOUS CONSISTENCY ERROR: OWNER SEQUENCE NUMBER *number* IS INVALID (SET-REF = *setref*, SCD-DISPL = *displacement*)

Meaning

The owner's RSQ in the SCD is higher than the permitted maximum.

- L0089 SERIOUS CONSISTENCY ERROR: RECORD-LENGTH IS INVALID OR PAGE INDEX DISPL WRONG
- Meaning**
The record length in the SIA is different from the length calculated locally in the page, or the displacement in the page index is incorrect.
- L0090 SERIOUS CONSISTENCY ERROR: COMPRESSION FORMAT IS INVALID
- Meaning**
The SIA record length differs from the length of a compression record as calculated locally in the page.
- L0091 SERIOUS CONSISTENCY ERROR: SET RELATION IS NOT CONSISTENT (SET-REF = *setref*, SCD-DISPL = *displacement*)
- Meaning**
The SCD does not tally with the set description in the SIA.
- L0092 MINOR CONSISTENCY ERROR: PPP *act-key* IS INVALID, REALM IS WRONG (SET-REF = *setref*, SCD-DISPL = *displacement*)
- Meaning**
The specified pointer contains either a realm number not defined in the schema or it points to a realm in which the record is not permitted (severity EASY (MINOR) LOCAL ERROR).
- L0093 SERIOUS CONSISTENCY ERROR: ACT-KEY *actkey* IN SYSTEM-RECORD IS INVALID, COL-NR=*col-no*
- Meaning**
The act-key in a DBTT column of the SYSTEM record and the act-key of the page are different, or the reference to a table is missing.
- L0094 SERIOUS CONSISTENCY ERROR: THE DB-KEYS IN THE OWNER-RECORD POINTING TO ITS SET DON'T MATCH (SET-REF = *setref*, SCD-DISPL = *displacement*)
- Meaning**
In the SCD of an owner of an empty set, only the DB key of the first member record or only the DB key of the last member record is the same as the DB key of the set.
- L0095 SERIOUS CONSISTENCY ERROR: DB-KEY OF THE RECORD IS INVALID
- Meaning**
The DB key of the SYSTEM record and that of the SIA (anchor) are at variance.
- L0098 STRUCTURAL CONSISTENCY ERROR: NUMBER OF ALLOWED ENTRIES = 0. THE CALC TABLE WILL NOT BE ANALYZED
- Meaning**
The maximum number of entries in the table is set to 0, so the table cannot be used.

L0099 NOT CONSISTENT CALC HEADER: *calc-header*

- L0063 MINOR CONSISTENCY ERROR: NUMBER OF PRESENT ENTRIES = 0 IN CALC OVERFLOW PAGE

Meaning

The number of table entries on a CALC overflow page is zero.

- L0096 SERIOUS CONSISTENCY ERROR: NUMBER OF ACTUAL ENTRIES *value* DIFFERS FROM NUMBER OF PAGE INDEX ENTRIES *value*

Meaning

The number of existing table entries and the specified number of page indexes on the direct CALC page are different.

- L0122 STRUCTURAL CONSISTENCY ERROR: NUMBER OF ACTUAL ENTRIES *value* IS TOO BIG

Meaning

The number of current table entries is greater than the number of reserved entries.

- L0125 STRUCTURAL CONSISTENCY ERROR: PAGE NUMBERS OF SOME POINTERS ARE TOO BIG, MAXIMUM PAGE NUMBER IS *max*

Meaning

The backward or forward pointer value in the CALC table header is greater than the permitted maximum.

- L0158 SERIOUS CONSISTENCY ERROR: PRIMARY BUCKET *bnr* IN REALM *realm-name* HAS A BACKWARD-POINTER

Meaning

The backward pointer value for a primary bucket in the CALC table header is non-zero.

- L0163 SERIOUS CONSISTENCY ERROR: KEYLENGTH AND/OR NUMBER OF TABLE ENTRIES DO NOT CORRESPOND WITH LENGTH OF TABLE

Meaning

The table lengths calculated from the page control information using 'FREE SPACE.DISPL' and from the entries in the CALC table header do not match.

L0100 SERIOUS CONSISTENCY ERROR: FPA VALUE OF {OLD/NEW} PAGE *act-key* IS IMPOSSIBLE

Meaning

The FPA value is greater than the page length minus the length of the page control information.

L0101 MINOR CONSISTENCY ERROR: THE {OLD/NEW} FPA-VALUE = "*value*" IS WRONG, PAGE HEADER CONTENTS *pageheader*

Meaning

1. A list takes up an entire page, but there is a remainder.
2. The central FPA indicates less free space than is actually available.
3. The remainder is more than one entry, and the central FPA value and the local FPA value are at variance.

The page header (20 bytes) is output.

L0102 DUMP OF {OLD/NEW} PAGE CONTENTS:

Meaning

Checking of the page contents of the "old" shadow database or of the "new" shadow database or the original has stopped: the page contents have therefore been dumped to printer.

L0103 STRUCTURAL CONSISTENCY ERROR: THE PAGE HEADER IS NOT CONSISTENT *explanation*
 - WRONG ACT-KEY

Meaning

The page's act-key is incorrect.

- Miscellaneous message extension

Meaning

Error in bytes 5-20, the name of the field is entered in *explanation*.

L0104 SERIOUS CONSISTENCY ERROR: THE PAGE INDEX IS NOT CONSISTENT *erläuterung*

Meaning

See "[L0105 STRUCTURAL CONSISTENCY ERROR: THE PAGE INDEX IS NOT CONSISTENT *explanation*](#)" on page 123.

L0105 STRUCTURAL CONSISTENCY ERROR: THE PAGE INDEX IS NOT CONSISTENT *explanation*
 - CALC-REC IN NON-CALC-PAGE

Meaning

The status byte in the page index of a page outside the CALC bucket shows a value of 3.

- DBTT-COL = 0 FOR TABLE-REC

Meaning

The DBTT column number in the page index of a table record indicates a value of zero.

- DISPL IMPOSSIBLE

Meaning

The displacement in the page index has an invalid value.

– DISPL IMPOSSIBLE FOR TABLE REC

Meaning

The displacement in the page index of the table record is greater than the permitted maximum.

– IMPOSSIBLE LENGTH

Meaning

The specified record or table length is not possible.

– MISSING BLOCK INDEX FOR EXPECTED LIST TABLE HEADER

Meaning

A list record without an associated list header exists.

– NON-CALC-REC IN CALC PAGE

Meaning

The status byte in the page index of a page from the CALC bucket shows a value $\neq 3$.

– REC-REFS FOR CALC RECORDS NOT EQUAL

Meaning

The rec refs in the page indices of the CALC records are not all the same.

– RSQ IS 0

Meaning

The RSQ in the page index indicates a value of 0.

– STATUS-BYTE AND REC-REF DON'T MATCH

Meaning

The status byte in the page index points to a spanned record, but the rec ref is $\neq 1$.

– WRONG DBTT-COL FOR GIVEN TABLE-REC

Meaning

The DBTT column number of a SYSTEM record is higher than the permitted maximum.

– WRONG DBTT-COL OR STATUS

Meaning

The status byte in the page index shows a value $\neq 3$, but the DBTT column number indicates a value higher than 0.

– WRONG REC-REF

Meaning

The rec ref (record reference) in the page index is distorted or higher than the permitted maximum.

- WRONG REC-REF FOR EXPECTED LIST RECORD

Meaning

Wrong Rec Ref of a list record.

- WRONG RSQ FOR GIVEN REC-REF

Meaning

The RSQ of a SYSTEM record is higher than the permitted maximum.

- WRONG STATUS-BYTE

1. The status byte in the page index is not 0,1,2,3 or 4.
2. The status byte in page index of a SYSTEM record which is not a table indicates a non-zero value.

- WRONG STATUS-BYTE FOR EXPECTED LIST-RECORD

Meaning

Too few list records exist for the associated list header.

L0106 SERIOUS CONSISTENCY ERROR: THE TABLE APPEARS TWICE IN THE SAME PAGE

Meaning

The DB key of a table record appears in at least two page indices.

L0120 NOT CONSISTENT TABLE-HEADER: *table-header*

- L0121 STRUCTURAL CONSISTENCY ERROR: NUMBER OF TABLE-ENTRIES DOES NOT CORRESPOND WITH LENGTH OF TABLE

Meaning

The table length is not equal to the table length calculated from the number of reserved entries and the entry length.

- L0122 STRUCTURAL CONSISTENCY ERROR: NUMBER ACTUAL ENTRIES *value* TOO BIG

Meaning

The number of current table entries is greater than the number of reserved entries.

- L0123 STRUCTURAL CONSISTENCY ERROR: TABLE-DESCRIPTION-BYTE DIFFERS FROM SIA

Meaning

The table types entered in the table page header and in the SIA are not the same.

- L0124 STRUCTURAL CONSISTENCY ERROR: INDEX-LEVEL FOR A NON-INDEXED TABLE

Meaning

A level higher than 0 is entered in the page header of a table other than an index table.

- L0125 STRUCTURAL CONSISTENCY ERROR: PAGE NUMBERS OF SOME POINTERS ARE TOO BIG, MAXIMUM PAGE NUMBER IS *max*

Meaning

At least one pointer value in the table header is greater than the permitted maximum.

- L0126 SERIOUS CONSISTENCY ERROR: RSQ *rsq* IN TABLE ENTRY *te-no* IS TOO BIG, MAXIMUM RSQ IS *max*
- Meaning**
The RSQ in the (CALC) table entry is higher than the permitted maximum.
- L0127 NOT CONSISTENT DUPLICATE-TABLE-HEADER: *d-table-header*
- L0128 STRUCTURAL CONSISTENCY ERROR: DUPLICATE TABLE HEADER IS OUT OF TABLE RANGE
- Meaning**
The duplicate table header value is outside the table storage range.
- L0129 STRUCTURAL CONSISTENCY ERROR: FREE-TABLE-SPACE IS WRONG
- Meaning**
The FREE-TABLE.SPACE value in the duplicate table header of an overflow page is at variance with the calculated free space.
- L0168 STRUCTURAL CONSISTENCY ERROR: POINTER TO PRIOR OVERLFLOW PAGE IN LEVEL-0 TABLE NOT ALLOWED
- Meaning**
The backward pointer value in the duplicate table header of a primary page is non-zero.
- L0169 STRUCTURAL CONSISTENCY ERROR: POINTER TO PRIOR OVERFLOW PAGE IN DTOB MISSING
- Meaning**
The backward pointer value in a duplicate table header of an overflow page is zero.
- L0170 STRUCTURAL CONSISTENCY ERROR: POINTER TO NEXT OVERFLOW PAGE IN A PAGE WITH MORE THAN ONE DATABASE KEY LIST NOT ALLOWED
- Meaning**
The forward pointer in the duplicate table header is non-zero even though there is still at least one more duplicates table on the page.
- L0172 STRUCTURAL CONSISTENCY ERROR: FREE SPACE IN DTOB NOT ALLOWED BETWEEN PAGE INDEX AND DUPLICATE TABLE HEADER
- Meaning**
The duplicate table header of an overflow page does not come immediately after the page index.
- L0130 STRUCTURAL CONSISTENCY ERROR: DB-KEY-LIST IN DUPLICATE TABLE FOR TABLE ENTRY *te-no* IS OUT OF TABLE RANGE
- Meaning**
The duplicate table entry is outside the duplicates table storage range.

L0132 SERIOUS CONSISTENCY ERROR: USER KEY OF TABLE ENTRY *te-no* IS NOT IN SEQUENCE

Meaning

1. The CALC keys are not sorted in ascending sequence.
2. The table entry is out of sequence in a sorted table.

L0133 SERIOUS CONSISTENCY ERROR: TABLE WAS STORED INTO WRONG REALM

Meaning

The table is in a realm which is not as specified in the SIA.

L0134 SERIOUS CONSISTENCY ERROR: REALM-REF OF PPP IN TABLE ENTRY *te-no* IS WRONG

Meaning

The realm ref in the (indirect CALC) table entry is zero or is higher than the permitted maximum.

L0135 INCORRECT *act-key-0/act-key-N* OF REALM *realm-name*:

- L0137 MINOR CONSISTENCY ERROR: REALMNAME *realm-name* DIFFERS FROM SIA

Meaning

The realm name in the act-key-0 page or act-key-N page does not match the realm name in the SIA.

- L0138 SERIOUS CONSISTENCY ERROR: SYSTEM-BREAK HAD OCCURRED

Meaning

The system break bit in the act-key-0 page or act-key-N page of the DBDIR is set.

- L0139 SERIOUS CONSISTENCY ERROR: FPA-VALUE FOR *act-key-0/act-key-N* CLAIMS NO FULL PAGE, VALUE = *fpa-value*

Meaning

The FPA value for the act-key-0 page or act-key-N page is non-zero.

L0151 SERIOUS CONSISTENCY ERROR: USER KEY OF TABLE ENTRY *te-no* IS A NOT ALLOWED DUPLICATE

Meaning

1. The CALC key of this entry already exists in the same page, and duplicates are not allowed.
2. An entry in a table other than a duplicates table already exists once, and duplicates are not allowed.

- L0152 SERIOUS CONSISTENCY ERROR: ACT-KEY *actkey* IN DBTT-ENTRY *dbttentry* COLUMN *col-no* IS WRONG
- Meaning**
In one column of the DBTT entry, the realm ref is zero (record deleted) but the page number is greater than 1.
- L0153 DBTT-ENTRY FOR NON EXISTING RECORD
- Meaning**
In a column with a number higher than zero, the realm ref of a deleted record is non-zero.
- L0154 REALM-REF NOT ALLOWED IN SCHEMA
- Meaning**
The realm ref of the DBTT entry is higher than the permitted maximum number.
- L0155 REALM-REF INVALID FOR RECORD OR TABLE
- Meaning**
The realm ref of the DBTT entry identifies a realm which has not been released for storage.
- L0156 PAGE NUMBER TOO BIG, MAXIMUM PAGE NUMBER IS *max*
- Meaning**
The page number in the DBTT entry is higher than the permitted maximum.
- L0159 SERIOUS CONSISTENCY ERROR: INDEX POINTER OF TABLE ENTRY *te-no* DOES NOT POINT TO A PAGE INDEX
- Meaning**
The displacement to the page index in the direct CALC table entry is outside the storage range of the page indices.
- L0160 SERIOUS CONSISTENCY ERROR: USER CALC KEY OF TABLE ENTRY *te-no* WAS STORED INTO WRONG BUCKET (CORRECT BUCKET: *bucket-no*)
- Meaning**
A CALC record in a primary bucket or the first overflow bucket has been stored in the wrong bucket.
- L0162 SERIOUS CONSISTENCY ERROR: RSQ *rsq* OF TABLE ENTRY *te-no* FOR EQUAL USER KEYS NOT IN SEQUENCE
- Meaning**
1. The RSQs for identical CALC keys are not sorted in ascending sequence.
 2. A table other than a duplicates table contains more than one RSQ.

L0171 SERIOUS CONSISTENCY ERROR: MAIN LEVEL POINTER IN DTOB *act-key* REFERS PAGE *pagenumber of main level* WHICH IS NO LEVEL-0 DUPLICATE TABLE BLOCK TABLE PAGE WITH ONE INDEX ENTRY

Meaning

Two or more records are stored on the main level of a duplicates table.

L0173 SERIOUS CONSISTENCY ERROR: RSQS OF DB-KEY-LIST IN DUPLICATE TABLE FOR TABLE ENTRY *te-no* NOT IN SEQUENCE

Meaning

The RSQs in the duplicates table are not sorted in ascending sequence.

L0176 SERIOUS CONSISTENCY ERROR: RSQ OF TABLE ENTRY *te-no* DIFFERS FROM RSQ OF PAGE INDEX

Meaning

In a direct CALC area reference is made to a page index with a differing RSQ.

L0177 SERIOUS CONSISTENCY ERROR: BNR OF MAIN LEVEL-POINTER IN DTOB *actkey* IS IMPOSSIBLE

Meaning

The page number of the main level of a duplicates table overflow page is higher than the permitted maximum.

L0179 SERIOUS CONSISTENCY ERROR: THE {OLD/NEW} FPA VALUE "*value*" IS WRONG, PAGE HEADER CONTENTS *pageheader*

Meaning

1. The central FPA value for an *act-key-0* page, a DBTT page or a CALC page is non-zero.
2. A list takes up an entire page; the free space is less than one entry, but the remainder is more than 8 bytes for a page length of 2048 bytes or more than 12 bytes for a page length of 4000 or 8096 bytes
3. The central FPA value is greater than the value calculated on the page.

L0180 STRUCTURAL CONSISTENCY ERROR: DBTT ANCHOR PAGE *act-key* NOT CONSISTENT: VALUE "*content*" FROM *itemname* IS WRONG

Meaning

An error was discovered in a DBTT anchor page. The incorrect content of the underlying field is output together with the *act-key* of the DBTT anchor page.

L0181 STRUCTURAL CONSISTENCY ERROR: DBTT ANCHOR PAGES NOT CONSISTENT: VALUE "*content*" FROM *itemname* IS WRONG, RECORD IS *recordname*

Meaning

An error was discovered during the processing of DBTT anchor pages. The incorrect content of the underlying field is output together with the record type name.

L0182 STRUCTURAL CONSISTENCY ERROR: DBTT ANCHOR PAGES NOT CONSISTENT: THE DBTT PAGE(S) BETWEEN *pagenumber1* AND *pagenumber2* OF RECORD *recordname1* ARE NOT SEPARATED FROM THE DBTT PAGE(S) BETWEEN *pagenumber3* AND *pagenumber4* OF RECORD *recordname2*

Meaning

An overlap between two DBTT areas was identified after the sorting of the DBTT areas administered in the DBTT anchor pages in accordance with the act-key of the first DBTT page in question.

L0183 STRUCTURAL CONSISTENCY ERROR: DBTT ANCHOR PAGE *act-key* NOT CONSISTENT: VALUES "*content*" FROM *itemnames* DON'T MATCH

Meaning

An error was discovered in a DBTT anchor page. The incompatible content of the two underlying fields is output together with the act-key of the DBTT anchor page.

L0184 STRUCTURAL CONSISTENCY ERROR: DBTT ANCHOR PAGES NOT CONSISTENT: VALUES "*content*" FROM *itemnames* DON'T MATCH, RECORD IS *recordname*

Meaning

An error was discovered during the processing of DBTT anchor pages. The incompatible content of the two underlying fields is output together with the record type names.

L0187 SERIOUS CONSISTENCY ERROR: TABLE HEADER NOT CONSISTENT: {ACTKEY STRUCTURE/TOP-OR-FIRST-INDICATOR} NOT ALLOWED

Meaning

Table header in ACTKEY format/with the Top-or-First indicator is not permitted.

L0188 SERIOUS CONSISTENCY ERROR: TABLE HEADER NOT CONSISTENT: LEVEL_BACK_KEY = 0, BUT TOP-OR-FIRST-INDICATOR IS NOT ON

Meaning

When Level-Back = 0 in ACTKEY format, the Top-or-First indicator must be set.

L0189 SERIOUS CONSISTENCY ERROR: TABLE HEADER NOT CONSISTENT: TOP-OR-FIRST-INDICATOR IS ON, BUT FORWARD-KEY <> 0 OR LEVEL-BACK-KEY <> 0

Meaning

When the Top-or-First indicator is set, the forward key and Level-Back = 0 must be set

L0190 SERIOUS CONSISTENCY ERROR: TABLE HEADER NOT CONSISTENT: INVALID AREA-REF IN {FORWARD/BACKWARD}-KEY

Meaning

An AREA-REF which is greater than the maximum AREA-REF in the database is entered in the forward or backward key in ACT-KEY.

L0206 SERIOUS CONSISTENCY ERROR: COMPRESSION ENTRY *c-e* DOES NOT MATCH WITH RECORD-LENGTH

Meaning

The displacements noted in the record's compression entry do not match the the record length.

3.7.5 Usage of job switches

When an error occurs BCHECK sets job switches (using the BS2000 command or macro MODIFY-JOB-SWITCHES). You can use these job switches in your procedures, but you should only provide a control automatism in the case that no job switch is set. In any other case it is necessary to consult the SYSLST protocol.

The following table describes which job switches BCHECK sets and which measures are necessary:

Type of error	Job switch	Measure
WARNING	30	see the “Messages” manual
ERROR	31	see the “Messages” manual
SYSTEM-ERROR	31	see the “Messages” manual
EASY (MINOR) LOCAL CONSISTENCY ERROR	26	Send error report to systems support; database operation is possible without restrictions
FATAL (SERIOUS, STRUCTURAL) LOCAL CONSISTENCY ERROR	27	Send error report to systems support; database operation may possible depending on the kind of consistency error. Please contact systems support.
GLOBAL CONSISTENCY ERROR	28	Send error report to systems support; database operation may possible depending on the kind of consistency error. Please contact systems support.

Table 11: Usage of job switches together in BCHECK

4 Printing the schema/subschema information area with BPSIA

BPSIA prints the schema SIA report or the subschema SSIA report in the form of a table.

The SIA report may be of assistance to the user, for instance:

- when storing records in the database using BINILOAD,
- when printing out certain tables using BSTATUS, or
- when printing out the contents of the database using BPRECORD.

The SSIA report, by contrast, is intended more as an aid for the programmers of DB applications.

BPSIA can be used to print the following:

- the user schema and the user subschemata
- the compiler schema and the compiler subschema
- the PRIVACY-AND-IQF schema and the PRIVACY-AND-IQF subschema

In order to print a schema or subschema, BPSIA selects the appropriate Schema Information Area (SIA) or Subschema Information Area (SSIA) from the DBDIR and lists them in the form of tables.

A printout of the user SIA or a user SSIA in the same form can also be obtained by entering a DISPLAY statement at the time of SIA/SSIA generation with the BGSIA utility routine, or with the BGSSIA utility routine, respectively.

In addition to outputting the data to SYSLST, you can also output it to a file in CSV format. The use of CSV format facilitates the further processing of the data in other system environments (e.g. in spreadsheet applications). The output in CSV format is described in the manual "[Database Operation](#)", section "Outputting database information in a neutral format".

4.1 System environment

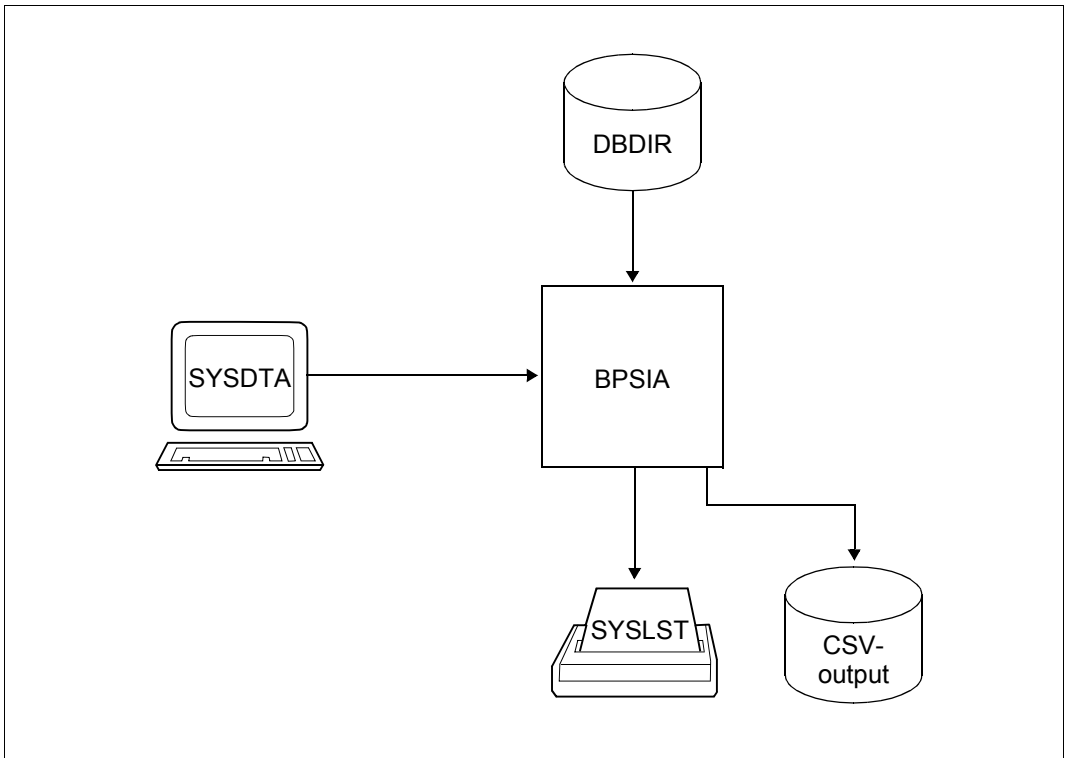


Figure 9: BPSIA system environment



BPSIA can also be started online. The DBDIR is then flagged as being inconsistent. A warning is issued, saying that the SYSTEM-BREAK-BIT in DBDIR is set.

When you run BPSIA online, you must assume that the data output is not current since the DBH has not copied all the data from the buffer to the database yet. In order to obtain as much current data as possible, you should force a database update just before the BPSIA run using the DAL command CHECKPOINT or NEW RLOG. However, the BPSIA output may still differ from the actual contents of the database if an update is running parallel to this task.

At startup BPSIA takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

4.2 BPSIA statements

BPSIA recognizes the following statements:

Statement	Meaning
DISPLAY SCHEMA	Print a schema
DISPLAY SUBSCHEMA	Print a subschema
END	Terminate statement input

Table 12: BPSIA statements

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

The statements can extend over several lines. However, each line is limited to 72 characters. A continuation character is not required in multiple-line notation.

Print a schema (DISPLAY SCHEMA)

```
[DISPLAY [IN CSV [csv-filename]] SCHEMA schema-name]
```

IN CSV

BPSIA also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPSIA run (e.g. DISPLAY IN CSV 'BPSIA.CSV' ...).

For a detailed description of CSV format output, see the manual "[Database Operation](#)", section "Outputting database information in a neutral format".

schema-name

Name of the schema whose SIA is to be printed; the options are:

user-schema-name

COMPILER-SCHEMA

PRIVACY-AND-IQF-SCHEMA

Print a subschema (DISPLAY SUBSCHEMA)

[DISPLAY [IN CSV [*csv-filename*]] SUBSCHEMA *subschema-name*]

IN CSV

BPSIA also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPSIA run (e.g. DISPLAY IN CSV 'BPSIA.CSV' ...).

For a detailed description of CSV format output, see the manual "[Database Operation](#)", section "Outputting database information in a neutral format".

subschema-name

Name of the subschema whose SSIA is to be printed; the options are:

user-subschema-name

COMPILER-SUBSCHEMA

PRIVACY-AND-IQF-SS

Terminate statement input (END)

END

4.3 Command sequence for starting BPSIA

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

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

- 01 In this case, specifying *:catid:* is permitted (see the “[Database Operation](#)” manual).
- 02 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel..
- 03 The UDS/SQL utility routine can also be started with the alias BPSIA.
- 04 A period is treated as an end criterion. It may be followed by another statement.

4.4 Description of the SIA report

The SIA report consists of a printout of an SIA in the form of a table. It contains the most important information from a schema.

SIA PRINT REPORT (general information)

Example

```
*** SIA PRINT REPORT ***  
  
DATABASE ID           = XXXXXXXX  
DATABASE NAME        = SHIPPING  
DATABASE-LAYOUT-VERSION = 004.00  
SCHEMA NAME          = MAIL-ORDERS  
SCHEMA TYPE           = USER SCHEMA  
SIA VALIDATION DATE  = 2015-06-28 11:26:15  
LENGTH OF SIA        = 6576  
MAXIMUM AREA REF     = 12  
MAXIMUM RECORD REF   = 15  
MAXIMUM SET REF      = 34  
MAXIMUM KEY REF      = 19  
TCUA LENGTH          = 3424  
MAXIMUM RECORD LENGTH = 580  
MAXIMUM ENTRY LENGTH = 52  
MAXIMUM NR MEMBERSHIPS = 4  
MAXIMUM SPLIT PARAMETER = 5  
LENGTH KEY-BIT-STRING = 0  
LENGTH PHYSICAL BLOCK = 4000  
FPA-ENTRIES MAIN-BLOCK = 1990
```

Under the header SIA PRINT REPORT, BPSIA prints the following general information:

DATABASE ID

Identification the database is stored under

DATABASE NAME

Name of the database

DATABASE-LAYOUT-VERSION

Layout version of the database

SCHEMA NAME

Name of the schema

SCHEMA TYPE

Type of schema

SIA VALIDATION DATE

Validation date of the schema (date and time)

LENGTH OF SIA

Self-explanatory

MAXIMUM AREA REF

Highest realm number in the user's database;
after reorganization, it is not necessarily identical with the actual number of realms

MAXIMUM RECORD REF

Highest number of a record type in the user's database;
after reorganization, it is not necessarily identical with the actual number of record types

MAXIMUM SET REF

Highest set number in the user's database;
after reorganization, it is not necessarily identical with the actual number of sets

MAXIMUM KEY REF

Highest number of an ASC, DESC, or SEARCH key in the user's database;
after reorganization, it is not necessarily identical with the actual number of corresponding keys

TCUA LENGTH

Length of the Transaction Currency Area of a subschema

MAXIMUM RECORD LENGTH

Length of the longest record type in the schema including set connection data (SCD)

MAXIMUM ENTRY LENGTH

Length of the longest key in the schema (CALC key, ASC or DESC key, or SEARCH key), incremented by 7 bytes and rounded up to an integral multiple of 4

MAXIMUM NR MEMBERSHIPS

The maximum number of sets in the schema containing the same record type as member record type

MAXIMUM SPLIT PARAMETER

Maximum value specified in the DYNAMIC REORGANIZATION clause of the SSL

LENGTH-KEY-BIT-STRING

Check byte for the MODIFY function; if 0 is specified, 4 bytes are reserved

LENGTH PHYSICAL BLOCK

Length of the database pages

FPA ENTRIES MAIN-BLOCK

Number of possible FPA entries per FPA base page

REFERENCE NUMBERS

Example

```

*** REFERENCE NUMBERS ***

11 AREAS : 1 3 4 5 6 7 8 9 10 11
           12

15 RECORDS : 1 2 3 4 5 6 7 8 9 10
             11 12 13 14 15

34 SETS : 1 2 3 4 5 6 7 8 9 10
          11 12 13 14 15 16 17 18 19 20
          21 22 23 24 25 26 27 28 29 30
          31 32 33 34

19 KEYS : 1 2 3 4 5 6 7 8 9 10
          11 12 13 14 15 16 17 18 19

```

Under the header REFERENCE NUMBERS, BPSIA provides a printout listing the total number of realms (areas), record types, records and keys defined in the schema and their respective numbers.

The numbers printed reflect the current status of the database. They need not necessarily be the maximum values printed under the header SIA PRINT REPORT (see [page 138](#)).

AREAS

- 1: Database directory
- 2: Database compiler realm
- 3,...: User realms

The database compiler realm is part of the compiler database; its reference number is therefore not included in the printout of a user SIA.

RECORDS

- 1: SSIA RECORD
- 2,...: User record types

AREA INFORMATION

Example

*** AREA INFORMATION ***

REF	AREA-NAME	TEMP	D/T	FPA-BEGIN ¹	ENTRIES	EXTENTS	INCR	CURRENT	FREE	FULL	PART-FILLED	SCAN	REUSE
				REF	BNR			-ACT	BNR	NR-PAGES	NR-PAGES	NR-PAGES	FIL%
1	DATABASE-DIRECTORY		D	1	1	100	0	YES	49	50	43	7	75 1
3	CUSTOMER-ORDER-RLM		D	3	1	38	0	YES	13	24	13	1	2 1
4	PURCHASE-ORDER-RLM		D	4	1	60	0	YES	24	33	26	1	2 1
5	CLOTHING		D	5	1	54	0	YES	20	32	10	12	65 1
6	HOUSEHOLD-GOODS		D	6	1	24	0	YES	7	16	8	0	- 1
7	SPORTS-ARTICLES		D	7	1	46	0	YES	8	37	9	0	- 1
8	FOOD		D	8	1	18	0	YES	8	8	5	5	92 1
9	LEISURE		D	9	1	44	0	YES	8	35	9	0	- 1
10	STATIONERY		D	10	1	24	0	YES	6	17	7	0	- 1
11	ARTICLE-RLM		D	11	1	62	0	YES	46	14	44	4	9 1
12	SEARCH-RLM	*	D						0	0			1

Under the header AREA INFORMATION, BPSIA prints information on the realms of the database:

REF Realm number

AREA NAME

Realm name

TEMP Marker for temporary realms

*: Realm is temporary

D/T Marker indicating whether the realm is stored on disk or on tape

D: Disk

T: Tape

FPA BEGIN

Address (realm number (REF) and page number (BNR)) of the first page of the free place administration in this realm

ENTRIES

Total number of database pages in this realm

EXTENTS

Number of FPA extents in this realm

¹ The fields in bold print contain the value '0' before the BFORMAT utility routine is run.

INCR-ACT

Denotes the online extensibility of this realm

YES: The online extensibility is activated for the realm.

NO: The online extensibility is not activated for the realm.

The reliable information about online extensibility is stored in the Act-key-0 of the particular realm and can be displayed using BPRECORD statement DISPLAY PAGE ZERO. Reliable information is available in the SIA only if the DBH was the last program which modified the database.

CURRENT BNR

Number of the page in which free storage space was last located

0: Realm has not yet been accessed

FREE NR-PAGES

Number of completely free pages

FULL NR-PAGES:

Number of full pages

PART-FILLED NR-PAGES

Number of partially filled pages; on average FIL% occupied

SCAN Indicates which search mode is used for the free place search

1: First Scan (space for new data which is to be stored is searched for in the free area at the end of the realm).

2: Second Scan (space for new data which is to be stored is searched for from the beginning of the realm).

REUSE

Flag indicating whether the BMODTT statement SET REUSE-FREE-SPACE has been specified for this realm

_ not specified

* specified



When a user schema or compiler schema is output, the FPA-BEGIN and ENTRIES values for the DBDIR (line 1) may not be up-to-date. This may be the case after BREORG has been executed, for example.

RECORD WITHIN LIST

Example

*** RECORD WITHIN LIST ***

REF	AREA-NAME	NR-WITHIN	LIST OF RECORDS				
1	DATABASE-DIRECTORY	1	1				
3	CUSTOMER-ORDER-RLM	4	2	3	4	5	
4	PURCHASE-ORDER-RLM	3	13	14	15		
5	CLOTHING	4	6	7	8	9	
6	HOUSEHOLD-GOODS	5	6	7	8	9	10
7	SPORTS-ARTICLES	5	6	7	8	9	10
8	FOOD	4	6	7	8	9	
9	LEISURE	4	6	7	8	9	
10	STATIONERY	4	6	7	8	9	
11	ARTICLE-RLM	2	11	12			
12	SEARCH-RLM	0					

Under the header RECORD WITHIN LIST, BPSIA prints out the record types contained in the individual realms.

REF Realm number

AREA-NAME

 Name of the realm

NR-WITHIN

 Number of record types stored in the realm

LIST OF RECORDS

 Numbers of the record types

RECORD INFORMATION

Example

*** RECORD INFORMATION ***

REF	RECORD-NAME	LOC-MODE	LENGTH	SYSINFO	COMPR	IMPL-SET	LIST-SET	LOC-VIA	OPT-CLAIM
1	SSIA-RECORD		0	0					0
2	CUSTOMER	DIR-LG	116	48					588
3	CST-ORDERS		17	6				1	0
4	ORD-ITEM		54	46					0
5	INSTALLMENT		44	18		28			0
6	ART-TYPE		29	4		29			0
7	ART-SELECTION		25	0		30			0
8	ART-DESCR	CALC	580	20	V				0
9	ARTICLE	CALC	215	128		31	7	7	0
10	SUBSET		58	56					0
11	COLORS		22	0		32			0
12	MATERIALS		21	0		33			0
13	SUPPLIER	CALC	167	37					0
14	PURCHASE-ORDER		50	40					0
15	P-ORD-ITEM		36	28			17	17	0

Under the header RECORD INFORMATION, BPSIA prints a table with the most important information on the individual record types.

REF Number of the record type

RECORD-NAME

Name of the record type

LOC-MODE

Type of LOCATION MODE clause defined in DDL

CALC: LOCATION MODE IS CALC

DIRECT: LOCATION MODE IS DIRECT

DIR-LG: LOCATION MODE IS DIRECT-LONG

LENGTH

Total length of the record type including set connection data (SCD)

SYSINFO

Length of set connection data (in bytes)

COMPR

Marker indicating whether the records of the record type are compressed

*: Compression

V: The record type includes a variable-length item

IMPL-SET

Number of the implicit set of this record type if a SEARCH key has been defined at record type level

LIST-SET

Number of a set defined by MODE IS LIST which contains the record types as member record type

LOC-VIA

Number of a set for which MODE IS LIST or PLACEMENT OPTIMIZATION has been defined and which contains the record type as member record type

OPT-CLAIM

Number of bytes reserved when a record of this record type is stored (important only if the record type in question is included as an owner record within a set defined with PLACEMENT OPTIMIZATION or ATTACHED TO OWNER).

The value specifies the number of bytes required to store the owner, the expected member and the administration information and tables.

DBTT INFORMATION

Under the header DBTT INFORMATION, BPSIA prints out two tables:

- The first table contains information on the database key translation tables (DBTTs) of the individual record types in the schema.
- The second table lists the sets in which the record types in the schema are member record types or owner record types.

Example

*** DBTT INFORMATION ***

REF	RECORD-NAME	ANCHOR	CURRENT	NR	EXTENTS	LENGTH	PER-BLOCK	NO REUSE
1	SSIA-RECORD	1	2 ¹	11	995	0	4	995
2	CUSTOMER	3	2	1	331	0	12	331
3	CST-ORDERS	3	4	1	497	0	8	497
4	ORD-ITEM	3	6	1	1990	0	4	995
5	INSTALMENT	3	9	1	995	0	4	995
6	ART-TYPE	11	2	4	497	0	8	497
7	ART-SELECTION	11	4	5	497	0	8	497
8	ART-DESCR	11	6	13	497	0	8	497
9	ARTICLE	11	8	63	995	0	4	995
10	SUBSET	6	5	1	995	0	4	995
11	COLORS	11	10	25	995	0	4	995
12	MATERIALS	11	12	10	995	0	4	995
13	SUPPLIER	4	2	1	662	0	12	331
14	PURCHASE-ORDER	4	18	1	497	0	8	497
15	P-ORD-ITEM	4	20	1	995	0	4	995

REF	RECORD-NAME	OWNERSHIPS				MEMBERSHIPS			
2	CUSTOMER	1	3	4					
3	CST-ORDERS	2				1			
4	ORD-ITEM					2	3	13	
5	INSTALLMENT					28	4		
6	ART-TYPE	5				29			
7	ART-SELECTION	6				30			
8	ART-DESCR	7				5	6		
9	ARTICLE	9	10	13	14	31	7	8	
10	SUBSET					9	10		
11	COLORS					32			
12	MATERIALS					33			
13	SUPPLIER	12	15	16		11			
14	PURCHASE-ORDER	17				15	16		
15	P-ORD-ITEM					14	17		

¹ The items printed in bold contain the value '0' before the BFORMAT utility routine is run.

REF Number of a record type

RECORD-NAME
Name of the record type

ANCHOR
First page of the DBTT anchor table (realm number and page number)

CURRENT
Record sequence number of the last stored record of this record type

NR Maximum number of records of this record type which can be stored taking into account the current size of the DBTT

EXTENTS
Number of DBTT extents that currently exist

LENGTH
Length of a DBTT entry

PER BLOCK
Number of DBTT entries that can be accommodated in one page

NO REUSE
*: Database keys of deleted records cannot be reused.
└ Database keys of deleted records can be reused.

OWNERSHIPS
Numbers of the sets in which the record type is owner

MEMBERSHIPS
Numbers of the sets in which the record type is member



When a user schema or compiler schema is output, the BEGIN, NR and LENGTH values may not be up to date. This may be the case after BREORG has been executed, for example.

CALC INFORMATION

Example

*** CALC INFORMATION ***

REC	REF	RECORD-NAME	DIR	DUPL	LENGTH	HASH-REF	PPP-BITS O M TAB	PHYSICAL_CALC_INFO FIRST-BUCKET	NR-BUCKETS	
8		ART-DESCR		*	40			5	2 ¹	3
								6	2	2
								7	2	3
								8	2	1
								9	2	3
								10	2	2
9		ARTICLE			10			5	5	3
								6	4	1
								7	5	3
								8	3	1
								9	5	3
								10	4	2
13		SUPPLIER		*	35			4	5	13

Under the header CALC INFORMATION, BPSIA prints out information on the record types defined with LOCATION MODE IS CALC.

REC REF

Number of the record type

RECORD-NAME

Name of the record type

DIR Marker indicating whether a direct or indirect hash area was created

*: Direct hash area

DUPL Marker indicating whether key values may or may not occur more than once

*: Duplicates allowed

LENGTH

Total length of the CALC key

HASH-REF

Marker for a user-defined hash routine; If no marker is printed in this column, the standard UDS/SQL hash routine is used.

¹ The items printed in bold contain the value '0' before the BFORMAT utility routine is run.

PPP-BITS

* means the following with

- O Owing to relocation of owner records, e.g. with BREORG, probable position pointers (PPPs) which refer to these owner records are very probably no longer up to date
- M Owing to relocation of member records, e.g. with BREORG, probable position pointers which refer to these member records are very probably no longer up to date
- TAB Owing to relocation of tables, e.g. with BREORG, probable position pointers which refer to these tables are very probably no longer up to date

Probable position pointers can also exist without these displays being set because the DBH does not maintain these bits when individual records are relocated. The displays are as a rule set when all records have been relocated by utility routines. It may be useful to perform a BREORG run.

PHYSICAL CALC INFO

Physical information on the hash area:

FIRST-BUCKET

Address (realm number and page number) of the first CALC page in the primary area

NR-BUCKETS

Number of CALC pages reserved for the primary area

Information listed under the header PHYSICAL CALC INFO is repeated by BPSIA for each realm in which a hash area for the relevant record type is located.

SET INFORMATION

Under the header SET INFORMATION, BPSIA prints out two tables containing information on the sets defined in the schema.

Example

*** SET INFORMATION ***

REF	SET-NAME	TYPE	MODE	ORDER	INSERT	REMOVE	SOS	INIT	INCR	T-AREA	P-AREA
1	CST-ORD-PLACED		PTRAY	SORT	AUTO	TRAN	OWN	10		1	
2	CST-O-CONTENTS		PTRAY	SORT	AUTO	PERM	CUR	0		1	
3	OUTSTANDING		CHAIN	LAST	AUTO	TRAN	OWN	0		1	
4	HIRE-PURCHASE		CHAIN	LAST	AUTO	PERM	OWN	0		1	
5	OFFER		PTRAY	SORT	AUTO	PERM	CUR	100		5	
6	SHORT-LIST		PTRAY	SORT	AUTO	PERM	CUR	100		20	
7	P-ORD-SPEC		LIST	SORT	AUTO	PERM	CUR	15		1	
8	MIN-STOCK-LEVEL	SING	CH-PR	SORT	MANL	TRAN		0		1	
9	CONTAINING		CHAIN	NEXT	AUTO	PERM	OWN	10		1	
10	CONTAINED-IN		CH-PR	NEXT	AUTO	PERM	OWN	0		1	
11	SUPPLIERS	SING	PTRAY	SORT	AUTO	PERM		0		1	
12	ARTICLES-AVAILABLE		PTRAY	SORT	AUTO	PERM	CUR	500		1	
13	ORDERED-ARTICLES		CHAIN	LAST	AUTO	PERM	OWN	0		1	
14	REORDERED-ARTICLES		CHAIN	LAST	AUTO	PERM	OWN	0		1	
15	P-ORD-PLACED		CHAIN	LAST	AUTO	PERM	CUR	0		1	
16	P-ORD-RECEIVED		CHAIN	FIRST	MANL	PERM	CUR	0		1	
17	P-ORD-CONTENTS		LIST	NEXT	AUTO	PERM	CUR	20		1	
18	RESULT-SET	DYN	PTRAY	SORT	MANL	TRAN		0		1	
19	LIMITED-SET	DYN	PTRAY	SORT	MANL	TRAN		0		1	
20	IQL-DYN1	DYN	PTRAY	SORT	MANL	TRAN		0		1	
21	IQL-DYN2	DYN	PTRAY	SORT	MANL	TRAN		0		1	
22	IQL-DYN3	DYN	PTRAY	SORT	MANL	TRAN		0		1	
23	IQL-DYN4	DYN	PTRAY	SORT	MANL	TRAN		0		1	
24	IQL-DYN5	DYN	PTRAY	SORT	MANL	TRAN		0		1	
25	IQL-DYN6	DYN	PTRAY	SORT	MANL	TRAN		0		1	
26	IQL-DYN7	DYN	PTRAY	SORT	MANL	TRAN		0		1	
27	IQL-DYN8	DYN	PTRAY	SORT	MANL	TRAN		0		1	
28	SYS_INSTALLMENT	IMPL		SORT	AUTO	PERM		0		1	
29	SYS_ART-TYPE	IMPL		SORT	AUTO	PERM		0		1	
30	SYS_ART-SELECTION	IMPL		SORT	AUTO	PERM		0		1	
31	SYS_ARTICLE	IMPL		SORT	AUTO	PERM		0		1	
32	SYS_COLORS	IMPL		SORT	AUTO	PERM		0		1	
33	SYS_MATERIALS	IMPL		SORT	AUTO	PERM		0		1	
34	IMPLICIT_RESULT_SET	DYN	PTRAY	SORT	MANL	TRAN		0		1	

The first table contains the following entries:

REF Set number

SET NAME

Set name

TYPE Set type

DYN: Dynamic set

IMPL: Implicit set

SING: SYSTEM set (singular set)

.....: Standard set

MODE

Set mode

CHAIN: Chain

CH-PR: Chain with backward chaining (CHAIN LINKED TO PRIOR)

LIST: List

PTRAY: Pointer array

.....: Implicit set

ORDER

Sort sequence within the set occurrences of the set

INSERT

Insertion of new member records in the set

AUTO: AUTOMATIC

MANL: MANUAL

REMOVE

Type of set membership

PERM: Permanent (MANDATORY member)

TRAN: Transient (OPTIONAL member)

SOS Set occurrence selection (in the case of non-singular sets only)

CUR: THRU CURRENT OF SET

OWN: THRU LOCATION MODE OF OWNER

INIT Initial number of set occurrences according to the POPULATION clause for this set

INCR Number of entries by which a set occurrence can be increased according to the INCREASE clause for this set

T-AREA

Realm number of the table part (pages with levels >0) of a distributable list which is distributed over more than one realm ("table realm").

P-AREA

Realm number of the realm in which the DBH is currently searching for free pages to store member records of a distributable list which is distributed over more than one realm ("preferred realm").

SET	OWNER	MEMBER	LNK	RSQ	OWN	MEM	OWN	MEM	OWN	MEM	ANCHOR-ACT	ANCHOR-DBK		
1 CST-ORD-PLACED	2	3		0	0	0	0	6	3					
2 CST-O-CONTENTS	3	4		0	0	0	0	6		3				
3 OUTSTANDING	2	4		12	0	6	24	18	4	13				
4 HIRE-PURCHASE	2	5		12	24	0	24	18						
5 OFFER	6	8		0	0	4	4	6		6				
6 SHORT-LIST	7	8		0	0	10	0	6						
7 P-ORD-SPEC	8	9 *		0	0	84	4	10		8				
8 MIN-STOCK-LEVEL	0	9		0	8	94	24	24	11	12	5	9	0	1
9 CONTAINING	9	10 *		12	0	0	12	22	10	10				
10 CONTAINED-IN	9	10 *		24	12	22	24	34	13					
11 SUPPLIERS	0	13		0	0	36	0	1	28		4	23	0	2
12 ARTICLES-AVAILABLE	13	9 *		0	0	118	0	10	15					
13 ORDERED-ARTICLES	9	4 *		12	36	24	24	22	14					
14 REORDERED-ARTICLES	9	15 *		12	60	0	24	22		17				
15 P-ORD-PLACED	13	14		12	0	4	24	18	16	16				
16 P-ORD-RECEIVED	13	14		12	24	22	12	18						
17 P-ORD-CONTENTS	14	15		0	0	22	4	6						
18 RESULT-SET	0	0		0	0	0	0	0	19		12	8	0	0
19 LIMITED-SET	0	0		0	0	0	0	0	20		12	8	0	0
20 IQL-DYN1	0	0		0	0	0	0	0	21		12	8	0	0
21 IQL-DYN2	0	0		0	0	0	0	0	22		12	8	0	0
22 IQL-DYN3	0	0		0	0	0	0	0	23		12	8	0	0
23 IQL-DYN4	0	0		0	0	0	0	0	24		12	8	0	0
24 IQL-DYN5	0	0		0	0	0	0	0	25		12	8	0	0
25 IQL-DYN6	0	0		0	0	0	0	0	26		12	8	0	0
26 IQL-DYN7	0	0		0	0	0	0	0	27		12	8	0	0
27 IQL-DYN8	0	0		0	0	0	0	0			12	8	0	0
28 SYS_INSTALMENT	0	5		0	0	0	0	0	29	4	3	12	0	3
29 SYS_ART-TYPE	0	6		0	0	0	0	0	30		5	10	0	4
30 SYS_ART-SELECTION	0	7		0	0	0	0	0	31		11	32	0	5
31 SYS_ARTICLE	0	9		0	0	0	0	0	32	7	5	11	0	6
32 SYS_COLORS	0	11		0	0	0	0	0	33		11	33	0	7
33 SYS_MATERIALS	0	12		0	0	0	0	0			11	36	0	8
34 IMPLICIT_RESULT_SET	0	0		0	0	0	0	0			0	0	0	0

The second table contains the following entries:

SET REF

Set number

SET-NAME

Set name

OWNER

Number of the owner record type in the case of a non-singular set

0: indicates a singular set

MEMBER

Number of the member record type

0: indicates a dynamic set

PHYS LNK

Marker for additional link between member records and their owner record (physical linked to owner)

*: Additional link defined

OWNER RSQ

Within the set connection data of the member record type: displacement of the item with the record sequence number of the owner record relative to the beginning of the set connection data (see the “[Design and Definition](#)” manual).

SCD-DISPL

Displacement of the set connection data relative to the beginning of the record:

OWN in the owner record type

MEM in the member record type

SCD-LNGTH

Length of the set connection data of the set:

OWN in the owner record type

MEM in the member record type

CHAIN-LNK

Number of the next set with:

OWN the same owner record type

MEM the same member record type

ANCHOR-ACT

Address (realm number and page number) of the anchor record of a SYSTEM set.

For dynamic sets, the realm number points to the temporary realm, since dynamic sets are stored in temporary realms.

ANCHOR-DBK

Database key of the anchor record of ANCHOR-ACT

(number of the record type = 0)

KEY INFORMATION (NO CALC SEARCH KEYS)

Example

*** KEY INFORMATION (NO CALC-SEARCH KEYS) ***

SET	KEY	DUPL	DUPL	LIST	PPP-BITS	DBTT	SSIA	OWNER	TABLE-ACTKEY								
REF SET-NAME	REF TYPE	LNTH	NOT	TABLE	TABLE	INDEX	ATT	SET	O	M	TAB	SPLIT	COLUM	DISPL	DISPL	AREA	BNR ¹
1	CST-ORD-PLACED	1	ASC	4	*		*						2	1	0		0
		2	SEARCH	6			*						2	2	28		0
2	CST-O-CONTENTS	3	ASC	2	*		*						2	1	0		0
3	OUTSTANDING				*								2	0	0		
4	HIRE-PURCHASE				*								2	0	0		
5	OFFER	4	ASC	40			*	*					5	1	0	0	11
6	SHORT-LIST	5	ASC	40			*	*					5	1	0		11
7	P-ORD-SPEC	6	ASC	4	*		*	*	*				2	1	0	0	0
8	MIN-STOCK-LEVEL	7	ASC	10	*		*	*					2	1	0		5
9	CONTAINING				*								2	0	0		
10	CONTAINED-IN				*								2	0	0		
11	SUPPLIERS	8	ASC	35	*		*	*					2	1	0		4
12	ARTICLES-AVAILABLE	9	ASC	40			*	*					5	1	0		0
		10	SEARCH	1	*		*	*					2	2	28		0
13	ORDERED-ARTICLES				*								2	0	0		
14	REORDERED-ARTICLES				*								2	0	0		
15	P-ORD-PLACED				*								2	0	0		
16	P-ORD-RECEIVED				*								2	0	0		
17	P-ORD-CONTENTS				*			*					2	1	0	0	0
18	RESULT-SET		ASC	1	*		*	*					2	1	0		12
19	LIMITED-SET		ASC	1	*		*	*					2	1	0		12
20	IQL-DYN1		ASC	1	*		*	*					2	1	0		12
21	IQL-DYN2		ASC	1	*		*	*					2	1	0		12
22	IQL-DYN3		ASC	1	*		*	*					2	1	0		12
23	IQL-DYN4		ASC	1	*		*	*					2	1	0		12
24	IQL-DYN5		ASC	1	*		*	*					2	1	0		12
25	IQL-DYN6		ASC	1	*		*	*					2	1	0		12
26	IQL-DYN7		ASC	1	*		*	*					2	1	0		12
27	IQL-DYN8		ASC	1	*		*	*					2	1	0		12
28	SYS_INSTALMENT	11	SEARCH	6	*		*	*					2	1	0		3
30	SYS_ART-SELECTION	13	SEARCH	25			*	*					5	1	0		11
33	SYS_MATERIALS	18	SEARCH	1	*		*	*					5	1	0		11
		19	SEARCH	20	*		*	*					5	2	28		11
34	IMPLICIT_RESULT_SET		ASC	1	*		*	*					0	1	0		12

Under the header KEY INFORMATION (NO CALC-SEARCH KEYS), BPSIA prints out information about the keys defined on the record type level and set level in the schema. The table does not contain any information on CALC keys and CALC SEARCH keys (see pages [149](#) and [158](#)).

¹ The items printed in bold contain the value '0' before the BFORMAT utility routine is run.

SET REF

Number of the set to which the key belongs

SET-NAME

Name of the set

KEY REF

Number of the key

TYPE Type of key

ASC: ASCENDING key

DESC: DESCENDING key

SEARCH: SEARCH key (USING INDEX)

DBKEY: sorted by record sequence number

LNGTH

Overall length of the key item

DUPL NO

Marker indicating whether duplicate key values are allowed or not

*: Duplicates are not allowed

DUPL TABLE

Marker for duplicates tables

*: Duplicates tables set up (TYPE IS DATABASE-KEY-LIST)

┌: No duplicates tables set up (TYPE IS REPEATED-KEY)

TABLE

Marker for table

*: Table set up

INDEX

Marker indicating whether a single-level or multi-level table has been set up

*: Multi-level table

ATT Marker indicating whether a set occurrence is to be stored in close proximity to the owner

*: ATTACHED TO OWNER

LIST SET

Marker indicating whether the set occurrence table has been set up as a list

*: List

PPP-BITS

* means the following with

- O Owing to relocation of owner records, e.g. with BREORG, probable position pointers (PPPs) which refer to these owner records are very probably no longer up to date
- M Owing to relocation of member records, e.g. with BREORG, probable position pointers which refer to these member records are very probably no longer up to date
- TAB Owing to relocation of tables, e.g. with BREORG, probable position pointers which refer to these tables are very probably no longer up to date

Probable position pointers can also exist without these displays being set because the DBH does not maintain these bits when individual records are relocated. The displays are as a rule set when all records have been relocated by utility routines. It may be useful to perform a BREORG run.

SPLIT Number of pages specified in the REORGANIZATION clause

2: Default value; also applies when no table exists

DBTT COLUM

Column number in the DBTT of the owner record type in which the address of the table has been entered

0: No table

SSIA DISPL

Displacement of key description in the SSIA relative to the beginning of all key information for this set

OWNER DISPL

Within the set connection data for this set in the owner record type:
displacement of the item containing the address of the table; only applicable if the table has been defined with WITH PHYSICAL LINK option

TABLE-ACTKEY

For standard sets:

AREA: Number of the realm in which the table is stored (DETACHED WITHIN);
0: Table is stored in the realm of the owner record type (DETACHED or ATTACHED)

BNR: No entry for standard sets

In the case of non-standard sets: address of the table

AREA: Realm number

BNR: Page number; 0 for dynamic sets

CALC-SEARCH-KEY INFORMATION

Example

*** CALC-SEARCH-KEY INFORMATION ***

SET REF	SET-NAME	KEY REF	DUPL LN	HASH NOT	PPP-BITS REF O M TAB	SSIA DISPL	PHYSICAL_CALC_INFO FIRST-BUCKET	NR-BUCKETS
29	SYS_ART-TYPE	12	25			0 11	14	1
31	SYS_ARTICLE	14	8	*		0 11	15	3
		15	40			68 11	18	11
32	SYS_COLORS	16	20	*		0 11	29	1
		17	2	*		28 11	30	1

Under the header CALC-SEARCH-KEY INFORMATION, BPSIA prints out information on the SEARCH keys which are defined on record or set level (for SYSTEM sets) by USING CALC:

SET REF

Set number (implicit set or SYSTEM set)

SET-NAME

Name of the set

KEY REF

Number of the key

LN

Total length of the key

DUPL NOT

Marker indicating whether duplicate key values may occur or not

*: Duplicates not allowed

HASH REF

Marker for the hash routine used

_: Standard UDS/SQL hash routine

> 0: Number of user-defined hash routine

PPP-BITS

* means the following with

O Owing to relocation of owner records, e.g. with BREORG, probable position pointers (PPPs) which refer to these owner records are very probably no longer up to date

M Owing to relocation of member records, e.g. with BREORG, probable position pointers which refer to these member records are very probably no longer up to date

TAB Owing to relocation of tables, e.g. with BREORG, probable position pointers which refer to these tables are very probably no longer up to date

Probable position pointers can also exist without these displays being set because the DBH does not maintain these bits when individual records are relocated. The displays are as a rule set when all records have been relocated by utility routines. It may be useful to perform a BREORG run.

SSIA DISPL

Displacement of this key description in the SSIA relative to the beginning of all key information for this set

PHYSICAL CALC INFO

Physical information on the hash area

FIRST-BUCKET

Address of the first CALC page reserved for table entries

NR-BUCKETS

Number of CALC pages reserved for table entries

4.5 Description of the SSIA report

The SSIA report is a copy of a subschema in the form of a table.

SSIA PRINT REPORT (general information)

Example

```
*** SSIA PRINT REPORT ***  
  
DATABASE NAME      = SHIPPING  
SCHEMA NAME        = MAIL-ORDERS  
SUBSCHEMA NAME     = ADMIN  
SIA VALIDATION DATE = 2015-06-28  11:26:15  
LENGTH OF SSIA     = 4296  
TCUA LENGTH        = 2952  
CRA LENGTH         = 200  
CRR LENGTH         = 336  
CRS LENGTH         = 2240  
MAXIMUM AREA REF   = 12  
MAXIMUM RECORD REF = 15  
MAXIMUM SET REF    = 34  
NR AREAS           = 10  
NR RECORDS         = 14  
NR SETS            = 34
```

Under the header SSIA PRINT REPORT, BPSIA prints out the following general information:

DATABASE NAME
Self-explanatory

SCHEMA NAME
Self-explanatory

SUBSCHEMA NAME

Self-explanatory

SIA VALIDATION DATE

Validation date of the associated schema (with date and time)

LENGTH OF SSIA

Self-explanatory

TCUA LENGTH

Length of the transaction currency area of the subschema

CRA LENGTH

Length of CURRENT OF AREA table within the TCUA

CRR LENGTH

Length of CURRENT OF RECORD table within the TCUA

CRS LENGTH

Length of CURRENT OF SET table within the TCUA

MAXIMUM AREA REF

Highest realm number within the subschema

MAXIMUM RECORD REF

Highest record type number within the subschema

MAXIMUM SET REF

Highest set number within the subschema

NR AREAS

Number of user realms within the subschema

NR RECORDS

Number of record types within the subschema

NR SETS

Number of sets within the subschema

REFERENCE NUMBERS

Example

```
*** REFERENCE NUMBERS ***  
  
AREAS   :    3    4    5    6    7    8    9   10   11   12  
  
RECORDS :    2    3    4    5    6    7    8    9   10   11  
          12   13   14   15  
  
SETS    :    1    2    3    4    5    6    7    8    9   10  
          11   12   13   14   15   16   17   18   19   20  
          21   22   23   24   25   26   27   28   29   30  
          31   32   33   34  
  
KEYS    :    1    2    3    4    5    6    7    8    9   10  
          11   12   13   14   15   16   17   18   19
```

Under the header REFERENCE NUMBERS, BPSIA lists the numbers of the realms, record types, sets and keys contained in the subschema.

The numbers in the printout reflect the current status of the database. They need not necessarily be the maximum values listed under the header SSIA PRINT REPORT.

AREA INFORMATION

Example

*** AREA INFORMATION ***

REF	AREA-NAME	CRA-DISPL
3	CUSTOMER-ORDER-RLM	0
4	PURCHASE-ORDER-RLM	20
5	CLOTHING	40
6	HOUSEHOLD-GOODS	60
7	SPORTS-ARTICLES	80
8	FOOD	100
9	LEISURE	120
10	STATIONERY	140
11	ARTICLE-RLM	160
12	SEARCH-RLM	180

Under the header AREA INFORMATION, BPSIA prints out information on the CURRENT-OF-AREA table of the TCUA.

The rest of the realm information is contained in the SIA report.

REF Realm number

AREA-NAME

 Name of the realm

CRA-DISPL

 Displacement of the associated realm entry in the CURRENT OF AREA table of the TCUA relative to the beginning of the table

RECORD INFORMATION

Example

*** RECORD INFORMATION ***

REC REF	RECORD-NAME	REC DISPL	CRR DISPL	NR KEYS	DB-KEY-LOCATION		AREA-WITHIN-LIST							
					IMPL SET	REC DISPL	ITEM DISPL	AREA-ID DISPL	LIST-OF-AREAS					
2	CUSTOMER	0	0	0		0	60		3					
3	CST-ORDERS	72	24	2					3					
4	ORD-ITEM	88	48	1					3					
5	INSTALLMENT	96	72	1	28				3					
6	ART-TYPE	128	96	1	29			1056	5	6	7	8	9	
									10					
7	ART-SELECTION	160	120	1	30			1086	5	6	7	8	9	
									10					
8	ART-DESCR	192	144	3				1116	5	6	7	8	9	
									10					
9	ARTICLE	752	168	7	31			1146	5	6	7	8	9	
									10					
10	SUBSET	840	192	0				1176	6	7				
11	COLORS	848	216	2	32				11					
12	MATERIALS	872	240	2	33				11					
13	SUPPLIER	896	264	2					4					
14	PURCHASE-ORDER	1032	288	0					4					
15	P-ORD-ITEM	1048	312	0					4					

Under the header RECORD INFORMATION, BPSIA prints out information on the record types of the subschema.

REC-REF

Number of the record type

RECORD-NAME

Name of the record type

REC DISPL

Displacement of the record type within the UWA relative to the beginning of the RECORD AREA

CRR DISPL

Displacement of the record type entry in the CURRENT-OF-RECORD table of the TCUA relative to the beginning of the table

NR KEYS

Number of keys defined for the record type; the keys defined on set level are included by BPSIA only if they are contained in a subschema set in which the record type is a member record type

IMPL SET

Number of the implicit set, if a SEARCH key has been defined on record type level

DB-KEY-LOCATION

Is printed by BPSIA only if a record defined with LOCATION MODE IS DIRECT or DIRECT-LONG occurs in the schema:

REC DISPL

Displacement of the area containing the database key, relative to the beginning of the RECORD AREA; if the database key item is contained in the record type, this area is the record area itself

ITEM DISPL

Displacement of the database key item relative to REC DISPL

AREA-WITHIN-LIST

Information on the realms in which records of the record type can be stored:

AREA-ID DISPL

Displacement of the AREA ID item in the WITHIN clause relative to the beginning of the RECORD AREA

LIST-OF-AREAS

Numbers of the realms in which – in accordance with the subschema description – records of the record type can be stored

CALC KEY INFORMATION

Example

*** CALC KEY INFORMATION ***

REF	RECORD-NAME	NR-ITEMS	ITEM-REF	REC-DISPL	LENGTH	TYPE	NEXT-SET	NEXT-KEY
8	ARTICLE-DESCR	1	6	26	40	4	5	4
9	ARTICLE	3	0	128	6	5	8	7
			6	134	2	5	7	6
			54	182	2	5	7	6
13	SUPPLIER	2	0	37	5	5	11	8
			5	42	30	4	11	8

Under the header CALC KEY INFORMATION, BPSIA prints out a table with information on the CALC keys (LOCATION MODE IS CALC) included in the subschema:

REF Number of the associated record type

RECORD-NAME

Name of the record type

NR-ITEMS

Number of items making up the CALC key

ITEM-REF

Displacement of a CALC key item within the record type in accordance with the subschema format

_: The item is not contained in the subschema format of the record type

REC-DISPL

Displacement of a CALC key item within the database record (schema format including set connection data)

LENGTH

Length of the CALC key item

TYPE Item type:

0: Database key item

1: Numeric item (packed)

2: Binary item

4: Alphanumeric item

5: Numeric item (unpacked)

8: Floating-point item

15: Various item types (only when the CALC key is made up of several items)

NEXT-SET

Number of the next set in the subschema, in which this item is defined as a key item

NEXT-KEY

Number of the key from NEXT-SET

The entries from ITEM-REF to NEXT-KEY are repeated for each CALC key item (NR-ITEMS).

ITEM STRING LIST

Under the header ITEM STRING LIST, BPSIA prints out a table showing the differences between the subschema format and schema format of a record type.

An "Item String" is a series of items defined in the subschema format of the record type in the same contiguous order as those in the schema format. If the subschema format is exactly the same as the schema format, the subschema record is an item string.

Example

*** ITEM STRING LIST ***

REC REF	RECORD-NAME	COMPL REC	USER-REC DISPL	DB-REC DISPL	LENGTH
2	CUSTOMER	*	0	48	68
3	CST-ORDERS	*	0	6	11
4	ORD-ITEM	*	0	46	8
5	INSTALLMENT	*	0	18	26
6	ART-TYPE	*	0	4	25
7	ART-SELECTION	*	0	0	25
8	ART-DESCR	*	58	78 V	502
			0	20	58
9	ARTICLE	*	0	128	87
10	SUBSET	*	0	56	2
11	COLORS	*	0	0	22
12	MATERIALS	*	0	0	21
13	SUPPLIER	*	0	37	130
14	PURCHASE-ORDER	*	0	40	10
15	P-ORD-ITEM	*	0	28	8

The table contains the following entries:

REC REF

Number of the record type

RECORD-NAME

Name of the record type

COMPL REC

Marker for identical subschema and schema record:

*: Identical

USER-REC DISPL

Displacement of an item string in the subschema record relative to the beginning of record (in descending order of displacement)

DB-REC DISPL

Displacement of the item string relative to the beginning of the schema record including the set connection data

LENGTH

Length of the item string in the subschema record

V: preceding the length indicates that the item in question is a variable item

The entries from USER-REC DISPL to LENGTH are repeated for each item string in the subschema record.

KEY ITEM LIST

Example

*** KEY ITEM LIST ***

REF	RECORD-NAME	ITEM-REF	REC-DISPL	LENGTH	SET-REF	KEY-REF						
3	CST-ORDERS	0	6	4	1	1						
		4	10	2	1	2						
		6	12	2	1	2						
		8	14	2	1	2						
4	ORD-ITEM	0	46	2	2	3						
		20	38	2	28	11						
5	INSTALLMENT	22	40	2	28	11						
		24	42	2	28	11						
		0	4	25	29	12						
7	ARTICLE-SELECTION	0	0	25	30	13						
8	ARTICLE-DESCR	6	26	40	CALC KEY ITEM							
					5	4						
					6	5						
9	ARTICLE	0	128	6	CALC KEY ITEM							
					8	7						
		6	134	2	CALC KEY ITEM							
					7	6						
					8	7						
		8	136	40	12	31	15					
								48	176	4	31	14
								52	180	2	31	14
								54	182	2	CALC KEY ITEM	
											7	6
86	214	1	12	31	14							
						11	17					
						2	2	20	32	16		
12	MATERIALS	0	0	1	33	18						
		1	1	20	33	19						
13	SUPPLIER	0	37	5	CALC KEY ITEM							
					11	8						
		5	42	30	CALC KEY ITEM							
					11	8						

Under the header KEY ITEM LIST, BPSIA prints out a table with information on all key items of the record types of the subschema.

REF Number of the associated record type

RECORD-NAME

 Name of the associated record type

ITEM-REF

 Displacement of the key item relative to the beginning of the subschema record

REC-DISPL

 Displacement of the key item relative to the beginning of the schema record, including set connection data

LENGTH

 Length of the key item

SET-REF

 Numbers (in ascending order) of the sets in which the key item is also contained;

 CALC KEY ITEM

 identifies the CALC key items of the LOCATION MODE clause, which are not related to a set or key and thus do not have any set numbers or key numbers

KEY-REF

 Number of the key to which the key item belongs

SET INFORMATION

Example

*** SET INFORMATION ***

SET REF SET-NAME	NR KEYS	CRS DISPL	CRS-SORT KEY-DISPL	SOS-OWNER-INFO		REC DISPL	(DIR)	(CALC)				SET-READY-LIST	
				LOC OWN	ALIAS		ITEM DISPL	NR ITEMS	ITEM REF	DB-REC DISPL	ITEM LENGTH		ITEM TYPE
1 CST-ORD-PLACED	2	0	0	DIR		0	60						3
2 CST-O-CONTENTS	1	80	4										3
3 OUTSTANDING	0	160	0	DIR		0	60						3
4 HIRE-PURCHASE	0	240	0	DIR		0	60						3
5 OFFER	1	320	6										5 6 7 8
													10 11
6 SHORT-LIST	1	400	46										5 6 7 8
													10 11
7 P-ORD-SPEC	1	480	86										5 6 7 8
													10 11
8 MIN-STOCK-LEVEL	1	560	90										5 6 7 8
													10 11
9 CONTAINING	0	640	0	CALC		752		3	0	128	6	5	5 6 7 8
									6	134	2	5	10 11
									54	182	2	5	
10 CONTAINED-IN	0	720	0	CALC	*	1056		3	150	128	6	5	5 6 7 8
									156	134	2	5	10 11
									158	182	2	5	
11 SUPPLIERS	1	800	100										4
12 ARTICLES-AVAILABLE	2	880	135										4 5 6 7
													9 10 11
13 ORDERED-ARTICLES	0	960	0	CALC		752		3	0	128	6	5	3 5 6 7
									6	134	2	5	9 10 11
									54	182	2	5	
14 REORDERED-ARTICLES	0	1040	0	CALC		752		3	0	128	6	5	4 5 6 7
									6	134	2	5	9 10 11
									54	182	2	5	
15 P-ORD-PLACED	0	1120	0										4
16 P-ORD-RECEIVED	0	1200	0										4
17 P-ORD-CONTENTS	0	1280	0										4
.													
.													
.													
28 SYS_INSTALLMENT	1	0	0										3
29 SYS_ARTICLE-TYPE	1	0	0										5 6 7 8
													10 11
30 SYS_ARTICLE-SELECTION	1	0	0										5 6 7 8
													10 11
31 SYS_ARTICLE	2	0	0										5 6 7 8
													10 11
32 SYS_COLORS	2	0	0										11
33 SYS_MATERIALS	2	0	0										11
34 IMPLICIT_RESULT_SET	0	2160	0										

Under the header SET INFORMATION, BPSIA prints out a table with information about all sets of the subschema.

SET-REF

Set number

SET-NAME

Name of the set

NR KEYS

Number of keys of the set's member record

CRS DISPL

Displacement of the set entry in the CURRENT OF SET table of the TCUA relative to the beginning of the table

CRS-SORT KEY-DISPL

Displacement of the ASC key entry or DESC key entry relative to the beginning of the KEY AREA in the TCUA

SOS-OWNER-INFO

Information on the owner of the set, if the SET OCCURRENCE SELECTION clause has been specified with THRU LOCATION MODE OF OWNER in the DDL:

LOC OWN:

LOCATION MODE of the owner record type

CALC: With hash routine

DIR: DIRECT or DIRECT-LONG, i.e. with database key

ALIAS

*: ALIAS clause defined

REC DISPL

Displacement of the record type relative to the beginning of the RECORD AREA in which the item for locating the owner is situated

If *identifier* or the ALIAS clause has been specified, this is an implicit record containing all implicitly defined items.

(DIR) Only if the LOCATION MODE of the owner is DIRECT or DIRECT-LONG:

ITEM DISPL

Displacement of the database key item, relative to REC DISPL

(CALC)

Only if the LOCATION MODE of the owner is CALC:

NR ITEMS

Number of items making up the CALC key

ITEM REF

Displacement of the CALC key item within the subschema record.

If an ALIAS clause has been specified: displacement of the ALIAS item within the record for implicitly defined items

DB-REC-DISPL

Displacement of the CALC key item in the schema record including set connection data

ITEM LENGTH

Length of the CALC key item or of the ALIAS item including set connection data

ITEM TYPE

Type of the CALC key item or ALIAS item (see [page 166](#))

SET READY LIST

Numbers of the realms which can be referenced when accessing via set

KEY INFORMATION

Example

*** KEY INFORMATION ***

SET REF SET-NAME	KEY REF	NEXT KEY	NR DESC	ITEM ITEMS	DB-REC REF	ITEM DISPL	ITEM LENGTH	ITEM TYPE	KEY-CHAIN SET	KEY
1 CST-ORD-PLACED	1	28		1	0	6	4	5		
	2	0		3	4	10	2	5		
					6	12	2	5		
					8	14	2	5		
2 CST-O-CONTENTS	3	0		1	0	46	2	5		
5 OFFER	4	0		1	6	26	40	4	6	5
6 SHORT-LIST	5	0		1	6	26	40	4		
7 P-ORD-SPEC	6	0		2	6	134	2	5	8	7
					54	182	2	5	8	7
8 MIN-STOCK-LEVEL	7	0		3	0	128	6	5		
					6	134	2	5		
					54	182	2	5	31	14
11 SUPPLIERS	8	0		2	5	42	30	4		
					0	37	5	5		
12 ARTICLES-AVAILABLE	9	28		1	8	136	40	4	31	15
	10	0		1	86	214	1	4		
28 SYS_INSTALLMENT	11	0		3	20	38	2	5		
					22	40	2	5		
					24	42	2	5		
29 SYS_ARTICLE-TYPE	12	0		1	0	4	25	4		
30 SYS_ARTICLE-SELECTION	13	0		1	0	0	25	4		
31 SYS_ARTICLE	14	68		3	48	176	4	5		
					52	180	2	5		
					54	182	2	5		
					15	0		1	8	136
32 SYS_COLORS	16	28		1	2	2	20	4		
	17	0		1	0	0	2	5		
33 SYS_MATERIALS	18	28		1	0	0	1	4		
	19	0		1	1	1	20	4		

Under the header KEY INFORMATION, BPSIA prints out a table with information on all keys in the subschema except for those defined in the LOCATION MODE clause:

SET REF

Number of the associated set

SET-NAME

Name of the associated set

KEY REF

Number of the key

NEXT KEY

Displacement of the next key defined for this set within the SSIA

0: The current key is the last key or the only key

DESC Marker indicating whether the key is ASCENDING or DESCENDING

*: DESCENDING key

NR ITEMS

Number of items making up the key

ITEM REF

Displacement of each item within the subschema record

65535: The item is not contained in the subschema format of the record

DB-REC DISPL

Displacement of each item within the schema record including set connection data

ITEM LENGTH

Self-explanatory

ITEM TYPE

Type of item (see [page 166](#))

KEY-CHAIN

If the item is defined as a key item in other sets:

SET: Is the number of the next set with the same key item

KEY: Is the number of this key

5 Output relational schema information with BPSQLSIA

The data in a UDS/SQL database can be accessed in relational terms.

BPSQLSIA prints the relational schema information for an existing UDS/SQL subschema that was defined in accordance with the CODASYL model. The relational schema information serves as a programming aid for the SQL user.

5.1 Overview

Relational access can, for example, take place via the SQL interface of DRIVE V2.1 (see the “[DRIVE/WINDOWS \(BS2000\)](#)” manuals).

To aid the SQL user when working in this way, BPSQLSIA can be used to print out a relational representation of existing UDS/SQL data structures that have been defined in accordance with the CODASYL model. In the following sections, this relational description of the data structures will be called **relational schema information**.

The CODASYL schema remains unchanged, however, and can continue to be used by CODASYL applications.

The relational schema information includes all the information needed, such as table names and field definitions, to permit the SQL user to work with a CODASYL database on a relational basis. It also indicates whether an existing CODASYL subschema can be processed on a completely relational basis or whether such access is limited.

BPSQLSIA generates separate relational schema information for each CODASYL subschema.

5.2 System environment

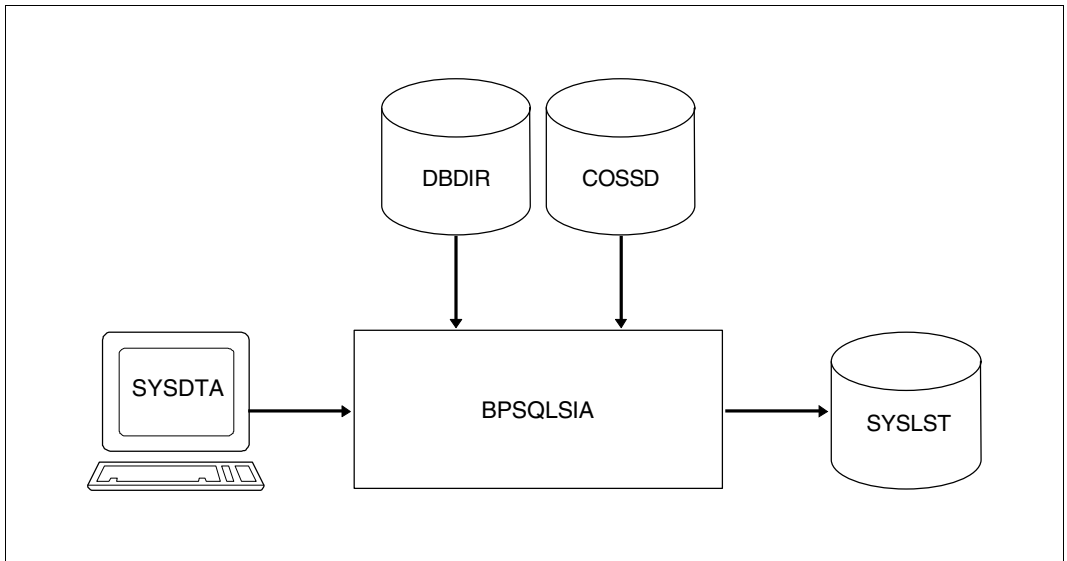


Figure 10: BPSQLSIA system environment

BPSQLSIA can be run in parallel with database operations and is restartable.

If you are working with the DATABASE application area, you can call BPSQLSIA with the BS2000 command `START-UDS-PRINT-SQLSIA` or the alias `BPSQLSIA`.

The UDS/SQL user syntax file sets the SDF user prompting mode to `EXPERT`. You can change the user prompting with the command:

```
MODIFY-SDF-OPTIONS GUIDANCE=*EXPERT/*NO/*MAXIMUM/*MEDIUM/*MINIMUM
```

At startup BPSQLSIA takes into account any assigned UDS/SQL pubset declaration (see the [“Database Operation”](#) manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

5.3 Prerequisites for SQL access to CODASYL definitions

In order for a CODASYL subschema to be processed on a completely relational basis with SQL it must satisfy the following conditions:

Condition	SQL limitation if the condition is not satisfied
The temporary realm must be present.	No SQL access permitted.
All sets whose member record types are included in the subschema must likewise be included in the subschema.	No SQL access permitted for the record type involved.
All ASCENDING/DESCENDING/CALC and SEARCH keys must be completely contained in the subschema.	No SQL access permitted for the record type involved.
Sets may not be defined with ORDER IS NEXT or ORDER IS PRIOR.	Neither INSERT nor UPDATE permitted for the member record type of the set.
Record types may not be distributed among multiple realms.	INSERT not permitted for the record type involved.
A record type may not be defined with LOCATION MODE IS DIRECT or DIRECT-LONG	INSERT not permitted for the record type involved.
A record type may not be defined with the SSL clause COMPRESSION.	UPDATE not permitted for the record type involved.
A record type may not contain variable length fields.	No SQL access permitted for the record type involved.
A record type may not contain a national item (Unicode).	No SQL access permitted for the record type involved.
A record type may not include packed or unpacked numerical fields for which the following apply with respect to the number of storage locations and the scale factor: number of storage locations > 15 or scale factor < 0 or scale factor > number of storage locations A positive scale factor gives the number of positions to the right of the decimal point. A negative scale factor specifies how many zeroes UDS/SQL must add to the field contents when performing calculations.	No SQL access permitted for the field involved, and no INSERT permitted for the record type involved.

Table 13: Prerequisites for SQL access

5.4 SQL data types

A distinction is made between the following data types:

- Alphanumeric data type
- National data type
- Numeric data types
 - Fixed-point data types: DECIMAL, NUMERIC
 - Integral data types: INTEGER, SMALLINT

Data types which are formed from these elementary data types also exist. These are called

- Structured data types

The tables below provide information on the permitted content of a record element, and the value range and length of a record element for the various data types.

Alphanumeric data type

Data type	Permitted content of a record element, value range	Length of a record element in bytes
CHARACTER[(n)]	Any EBCDIC characters, e.g. digits, letters or special characters <i>n</i> = Number of characters $1 \leq n \leq 255$ <i>Example</i> FIRST-NAME CHARACTER(20)	<i>n</i> Default: <i>n</i> =1

National data type

Data type	Permitted content of a record element, value range	Length of a record element in bytes
NCHAR[(n)]	Any Unicode or NATIONAL characters <i>n</i> = Number of characters $1 \leq n \leq 127$ <i>Example</i> SURNAME NCHAR(20)	$2n$ Default: <i>n</i> =1

Numeric data types

Data type	Permitted content of a record element, value range	Length of a record element in bytes
DEC[IMAL][(<i>n</i> [, <i>m</i>])]	<p>Positive or negative fixed-point number (packed) with sign. <i>n</i> corresponds to the number of digits; of these <i>m</i> are decimal places. $1 \leq n \leq 15.0 \leq m \leq n$</p> <p><i>Example</i> SUBTOTAL DECIMAL(6,4)</p>	$\frac{n+1}{2}$ rounded up Default: <i>n</i> =15; <i>m</i> =0
NUMERIC[(<i>n</i> [, <i>m</i>])]	<p>Positive or negative fixed-point number (unpacked) with <i>n</i> digits; of these <i>m</i> are decimal places. $1 \leq n \leq 15.0 \leq m \leq n$</p> <p><i>Example</i> TOTAL NUMERIC(8,2)</p>	<i>n</i> Default: <i>n</i> =8; <i>m</i> =0
SMALLINT	<p>Integer in the range from -32768 to 32767.</p> <p><i>Example</i> DB-PAGE SMALLINT</p>	2
INTEGER	<p>Integer in the range from -2147483648 to 2147483647.</p> <p><i>Example</i> COUNTER INTEGER</p>	4

Structured data types

Record elements of a structured data type consist in turn of record elements.

Record elements of a structured data type are

- vectors,
- structures and
- vectors with structured elements.

A structured record element can be referenced as a complete record element or you can reference individual record elements from it in SQL statements.

Vectors

A vector is a record element of a structured data type which comprises a fixed number of components with the same data type. In the case of vector A a single variant can be referenced in the form $A(l)$ or a range of variants in the form $A(l..m)$ or the entire range of variants in the form A or $A(1..n)$ ($n = \text{number of variants}$; $1 \leq l \leq m \leq n$).

Example

```
FOREIGN-LANGUAGE (3) CHARACTER(10)
```

└─ Number of variants

The vector *FOREIGN-LANGUAGE* contains 3 variants with a length of 10 of the alphanumeric data type.

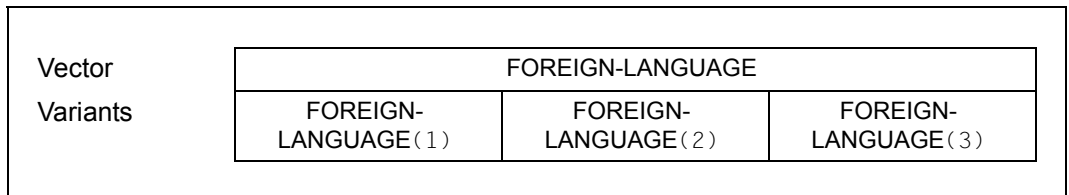


Figure 11: Structured data type- vector

Structure

A structure is a group of record elements.

A structure can contain the following elements:

- simple record elements of the non-structured data type,
- vectors,
- structures or
- vectors with structured elements.

Example

CUST-ADDRESS	STRUCTURE	Structure
STREET	CHARACTER(20)	Simple record element
ZIP	NUMERIC(4)	Simple record element
CITY	CHARACTER(20)	Simple record element

The structure *CUST-ADDRESS* consists of record elements of the alphanumeric and numeric data types.

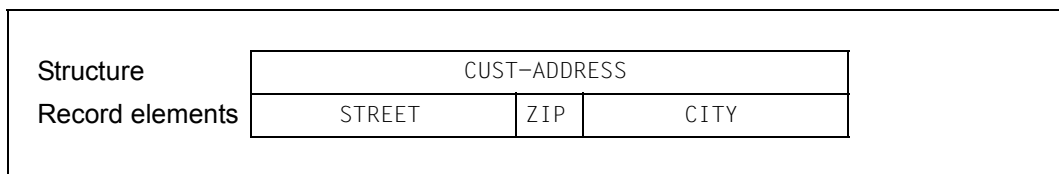


Figure 12: Structured data type- structure

National structure

The national structure is a special case among the structures. It may only contain the following national elements:

- simple record elements of the national data type,
- vectors from components of the national data type,
- national structures or
- vectors from national structures.

A national structure is treated as a complete record element like a record element of the national data type.

Example

NAME	NCHARSTRU	National structure
FIRST-NAME	NCHAR(20)	Simple record element
SURNAME	NCHAR(20)	Simple record element

The structure *NAME* consists of record elements of the national data type.

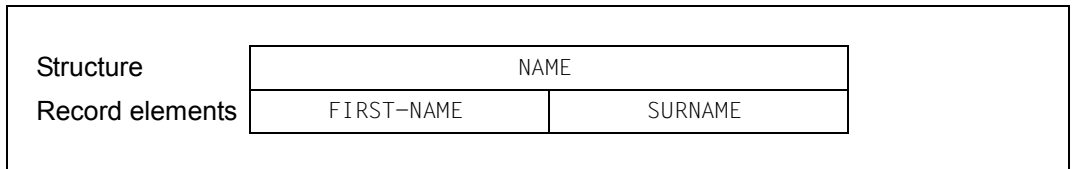


Figure 13: Structured data type- national structure

Vector with structured elements

A vector with structured elements is a structure with a repeating factor. The repeating factor specifies how many variants of the structure are grouped.

Example

CUST-ADDRESS	STRUCTURE(2)	Vector with structured elements
STREET	CHARACTER(20)	Record element
ZIP	NUMERIC(4)	Record element
CITY	CHARACTER(20)	Record element

The structure *CUST-ADDRESS* occurs twice and is therefore a vector with structured elements. The record elements *STREET*, *ZIP* and *CITY* also occur twice.

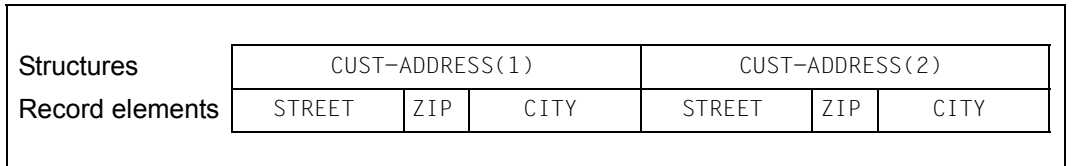


Figure 14: Structured data type- vector with structured elements

5.5 BPSQLSIA statements

Statement	Meaning
END	Terminates input
OPEN-DATABASE	Opens the database
PRINT-RELATIONAL-SCHEMAINFO	Selects subschemas

Table 14: BPSQLSIA statements

The individual statements of BPSQLSIA are described below in alphabetical order.

Terminate input (END)

This statement is used to terminate input and start the program run.

END

The END statement has no operands.

Open database (OPEN-DATABASE)

The OPEN-DATABASE statement must be given as the first statement if you have not assigned the database with

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

If you have already issued a ADD-FILE-LINK statement, the OPEN-DATABASE statement is rejected as an error and not offered in the SDF mask.

The file link name remains in effect until it is released with the command REMOVE-FILE-LINK. The OPEN-DATABASE statement, by contrast, is effective only till the end of the BPSQLSIA run.

OPEN-DATABASE

```
DATABASE-NAME = <dbname>
,COPY-NAME = *NONE / <copy-name>
,USER-IDENTIFICATION = *OWN / <userid>
```

DATABASE-NAME = <dbname>

Name of the database with which you wish to work.

COPY-NAME = *NONE

Selects the database original for processing.

COPY-NAME = <copy-name>

Selects the database copy with the specified copy name for processing.

USER-IDENTIFICATION = *OWN

BPSQLSIA runs under the same user ID as the one under which the database is cataloged.

USER-IDENTIFICATION = <userid>

User ID under which the database is cataloged. The user ID is specified without the '\$' character.



A database copy can be assigned explicitly with

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

but may also be specified in the OPEN-DATABASE statement.

Select subschemas (PRINT-RELATIONAL-SCHEMAINFO)

Up to 30 subschemas can be explicitly specified in one BPSQLSIA run. With PRINT *ALL, however, BPSQLSIA generates relational schema information for all subschemas of the database, regardless of how many exist.

BPSQLSIA outputs the relational schema information in the order in which the subschemas appear in the COSSD, even if a different order is specified in the PRINT command.

The PRINT statement may be specified more than once.

```
PRINT-RELATIONAL-SCHEMAINFO
```

```
SUBSCHEMA-NAME = *ALL / *ALL-EXCEPT(...) / list-poss(20): <subschema-name>
```

```
  *ALL-EXCEPT(...)
```

```
    | NAME = list-poss(20): <subschema-name>
```

SUBSCHEMA-NAME = *ALL

Relational schema information is generated for all subschemas of the database. All further PRINT statements are ignored.

SUBSCHEMA-NAME = *ALL-EXCEPT(...)

Relational schema information is generated for all subschemas of the database except for those listed following *ALL-EXCEPT.

NAME = list-poss(20): <subschema-name>

Names the subschemas for which no relational schema information is to be generated.

SUBSCHEMA-NAME = list-poss(20): <subschema-name>

Relational schema information is generated for all the named subschemas.

5.6 Command sequence to start BPSQLSIA

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,  
    FILE-NAME=[ :catid: ][ $userid. ] dbname.DBDIR[ .copy-name ]]  
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=version  
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.
- 02 The version of the utility routine is selected.
 Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.
- 03 BPSQLSIA can be called from any user ID. The UDS/SQL utility routine can also be started with the alias BPSQLSIA or START-UDS-PRINT-SQLSIA.

5.7 Description of the output of BPSQLSIA

BPSQLSIA outputs to SYSLST:

1. Information on the base tables

Heading: INFORMATION ABOUT RELATIONAL SCHEMA *subschema-name*

This includes:

- Descriptions of the fields of each base table, including field name, data type, null-value condition, default value and additional information. Additional information may be: PRIMARY KEY SYSTEMDEFINED, UNIQUE or REFERENCES...
- A summary of all unique keys at the set level and all unique keys at the record type level consisting of more than one field (UNIQUE summary).
- A summary of all simple and compound keys (INDEX summary).
The INDEX output is divided in two for all keys (e.g. compound key): the left-hand column contains the indexes that SQL can use; the right-hand column contains the indexes as defined in UDS/SQL (e.g. decomposed into items).
Keys that have not been fully taken over into the subschema are indicated. Missing key fields are identified by three question marks.

2. Table of all limitations

Heading: SHORT INFORMATION ABOUT RELATIONAL SCHEMA

This includes:

- A message indicating whether or not the subschema can be processed with SQL.
- If the subschema can be processed with SQL, a summary of the types of SQL access permitted for each base table, under the heading:
SHORT INFORMATION ABOUT TABLES.

3. Messages, if one or more conditions that restrict SQL access are satisfied (see [page 179](#))

(Heading: DIAGNOSTIC SUMMARY FOR SUBSCHEMANAME *subschema-name*)

5.8 Conversion rules

The CODASYL definitions are converted to relational schema information according to the following rules.

No.	CODASYL subschema	Relational schema description
1	Record type	Base table identified with TABLE Name of the base table: <i>record-type-name</i>
2	Record type that is owner record type in at least one set	Primary key: <i>record-type-name_</i> of data type INTEGER Additional information: PRIMARY KEY SYSTEMDEFINED Null-value condition: NOT NULL
3	Item of a record type	Field of the base table of the corresponding data type with null-value condition: NOT NULL Default value: 0 for numeric fields '_' for alphanumeric fields
4	Group item, repeating group	Data type STRUCTURE
5	Repeating factor	(<i>repeating factor</i>)
6	Item of type DATABASE-KEY	Field of the base table with the same name Data type: INTEGER Default value: 0
7	Item of type DATABASE-KEY-LONG	Field of the base table with the same name Data type: CHARACTER ; length 8 Additional information: ATTRIBUTE <i>item-name</i> IS DEFINED AS DATABASE-KEY-LONG; default value: X'00..00'
8	One or more SYSTEM sets not defined MANDATORY AUTOMATIC	A base table with the name SYSTEM and primary key SYSTEM_ , of data type INTEGER
9	Set relationship	Foreign key in the base table that corresponds to the member record type Field name: <i>set-name_</i> Data type: INTEGER Reference condition: REFERENCES <i>owner-record-type-name</i>
10	Set: MANDATORY AUTOMATIC	Foreign key with null-value condition: NOT NULL
11	Set: MANDATORY MANUAL	Foreign key with null-value condition: NOT NULL ON UPDATE Default value: NULL

Table 15: Conversion rules for BPSQLSIA

(part 1 of 2)

No.	CODASYL subschema	Relational schema description
12	Set: OPTIONAL AUTOMATIC	Foreign key with null-value condition: NOT NULL ON INSERT
13	Set: OPTIONAL MANUAL	Foreign key without null-value condition Default value: NULL
14	Key consisting of one or more items. e.g. for compound keys:	Additional information: left-hand column: INDEX ([<i>set-name_</i>]group-name) right-hand column: INDEX ([<i>set-name_</i>]field-1,...field-n)
15	Unique key at record type level consisting of one item	Additional information UNIQUE for the key field
16	Unique key at set level consisting of one item for a SYSTEM set defined with MANDATORY AUTOMATIC	Additional information UNIQUE for the key field
17	Unique key at record type level consisting of multiple items	Additional information UNIQUE for the corresponding base table: UNIQUE (<i>item-1</i> ,..., <i>item-n</i>)
18	Unique key at set level consisting of multiple items for a SYSTEM set defined with MANDATORY AUTOMATIC	Additional information UNIQUE for the base table corresponding to the member record type: UNIQUE (<i>item-1</i> ,..., <i>item-n</i>)
19	Unique key at set level consisting of one or more items for a set defined with MANDATORY AUTOMATIC, or for a SYSTEM set not defined with MANDATORY AUTOMATIC	Additional information UNIQUE for the base table corresponding to the member record type: UNIQUE (<i>set-name_</i> <i>item-1</i> ,..., <i>item-n</i>) where <i>set-name_</i> is the associated foreign key attribute

Table 15: Conversion rules for BPSQLSIA

(part 2 of 2)

The name of the CODASYL subschema becomes the name of the relational schema. Hyphens in names in the CODASYL schema are replaced by underscores in the derived names in the relational schema information.

Condition names (level number 88) are not output in the relational schema information.

Examples of the individual rules can be found on [page 193ff](#).

The meanings of the relational terms and concepts are explained with examples in the “[SQL for UDS/SQL](#)” manual.

5.9 Summary of the SQL access permitted for each base table

For each CODASYL subschema processed, BPSQLSIA outputs a summary as follows:

*** SHORT INFORMATION ABOUT TABLES

TABLE	RET	INS	UPD	ATR
<i>base-table-1</i>	y/n	y/n	y/n	y/n
.
.
.
<i>base-table-n</i>	y/n	y/n	y/n	y/n

RET: y: The SQL retrieval access SELECT is permitted for this base table.
 n: No SQL access is permitted for this base table.

INS: y: The SQL access INSERT is permitted for this base table.
 n: The SQL access INSERT is not permitted for this base table.

UPD: y: The SQL access UPDATE is permitted for this base table.
 n: The SQL access UPDATE is not permitted for this base table.

ATR: y: SQL access is permitted for all fields (attributes) of this base table.
 n: In this base table there is at least one field (attribute) to which no SQL access is possible.

5.10 Example

BPSQLSIA execution

```

/ADD-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=STAFF.DBDIR
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL, VERSION=02.8A00
/START-UDS-BPSQLSIA
***** START      BPSQLSIA      (UDS/SQL V2.8 0000 )      2015-06-28  11:40:32
//PRINT-RELATIONAL-SCHEMAINFO SUBSCHEMA-NAME=STAFF-DB
//END

***** DIAGNOSTIC SUMMARY FOR SUBSCHEMA STAFF-DB

                NO ERRORS
+++++          4 WARNINGS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :           11
***** NORMAL END  BPSQLSIA      (UDS/SQL V2.8 0000 )      2015-06-28  11:40:33

```

DDL of the CODASYL subschema

The numbers shown are the numbers of the conversion rules and refer to the corresponding parts in the relational schema information.

```

SCHEMA PERS-DB.
AREA PERS-DB-REALM.
AREA TEMPO TEMPORARY.

```

1/2) — RECORD NAME IS DEPARTMENT WITHIN PERS-DB-REALM.

```

3) ———— 02 NAME           PIC X(30).
           02 LOCATION      PIC X(30).
           02 EXTERN        PIC X.
           02 DEPARTMENT-MGR TYPE IS BIN 31.

```

RECORD NAME IS EMPLOYEE

15) ———— LOCATION MODE IS CALC USING PERSONNEL-NR
DUPLICATES ARE NOT ALLOWED

14) ———— SEARCH KEY IS M-NAME
USING CALC
DUPLICATES ARE ALLOWED

17) ———— SEARCH KEY IS P-CODE,CITY,STREET

USING INDEX
 DUPLICATES ARE NOT ALLOWED.

```

5) ————— 03 E-NAME                PIC X(30).
                03 FIRSTNAME          PIC X(30) OCCURS 5.
                03 E-AGE                TYPE IS BIN 15.
                03 MARITAL-STATUS      PIC X.

4) ————— 03 CHILDREN                OCCURS 10.
                04 C-NAME              PIC X(30).
                04 C-AGE                TYPE IS BIN 15.
                03 CUST-ADDRESS        OCCURS 2.
                04 P-CODE              PIC X(5).
                04 CITY                PIC X(15).
                04 STREET              PIC X(30).
                03 PERSONNEL-NO        TYPE IS BIN 31.
                03 OCCUPATION          PIC X(10).
                03 SALARY              PIC S9(8)V9(2).
                03 BONUSSES            PIC S9(8)V9(2).
                03 MGR-NO              TYPE IS BIN 31.

RECORD NAME IS PROJECT WITHIN PERS-DB-REALM.

                02 PROJ-NAME          PIC X(30).
                02 BUDGET              PIC S9(10)V9(2).
                02 PROJ-MGR            TYPE IS BIN 31.

9) ——— SET NAME IS DEPT-EMP
        ORDER IS FIRST
        OWNER IS DEPARTMENT

10) — MEMBER IS EMPLOYEE
        MANDATORY AUTOMATIC
        SEARCH KEY IS MGR-NO
        USING INDEX DUPLICATES NOT ALLOWED

19) ————— SEARCH KEY IS OCCUPATION,SALARY
        USING INDEX DUPLICATES NOT ALLOWED
        SELECTION CURRENT.

        SET NAME IS PROJ-EMP
        ORDER IS FIRST
        OWNER IS PROJECT.
        MEMBER IS EMPLOYEE

13) ————— OPTIONAL MANUAL
        SELECTION CURRENT.

8) ——— SET NAME IS INT-FUND
        ORDER IS FIRST
        OWNER IS SYSTEM.
        MEMBER IS PROJECT
  
```

- 12) ————— OPTIONAL AUTOMATIC
 SEARCH KEY IS PROJ-NAME
 USING CALC DUPLICATES NOT ALLOWED.
- SET NAME IS PROJ-EMP-2
 ORDER IS FIRST
 OWNER IS PROJECT.
 MEMBER IS EMPLOYEE
- 11) ————— MANDATORY MANUAL
 SELECTION CURRENT.
- SET NAME IS EXT-FUND
 ORDER IS FIRST
 OWNER IS SYSTEM.
 MEMBER IS PROJECT
 OPTIONAL MANUAL.
- SET NAME IS SYS-EMP
 ORDER IS FIRST
 OWNER IS SYSTEM.
 MEMBER IS EMPLOYEE
 MANDATORY AUTOMATIC
- 16) ————— SEARCH KEY SALARY
 USING CALC DUPLICATES NOT ALLOWED
- 18) ————— SEARCH KEY E-AGE,MARITAL-STATUS
 USING INDEX DUPLICATES NOT ALLOWED.

Output of BPSQLSIA to SYSLST

The numbers shown are the numbers of the conversion rules and refer to the corresponding parts in the DDL.

INFORMATION ABOUT RELATIONAL SCHEMA STAFF

1) — TABLE DEPARTMENT

	ATTRIBUTE	TYPE	NOT NULL	DEFAULT	
2) —	DEPARTMENT_	INTEGER	NOT NULL		PRIMARY KEY SYSTEMDEFINED
3) —	D_NAME	CHARACTER (30)	NOT NULL	' '	
	D_LOCATION	CHARACTER (30)	NOT NULL	' '	
	EXTERN	CHARACTER (1)	NOT NULL	' '	
	DEPARTMENT_MGR	INTEGER	NOT NULL	0	

TABLE EMPLOYEE

	ATTRIBUTE	TYPE	NOT NULL	DEFAULT
--	-----------	------	----------	---------

5) —	E_NAME	CHARACTER (30)	NOT NULL	' '	
	FIRSTNAME	(5) CHARACTER (30)	NOT NULL	' '	
	E_AGE	SMALLINT	NOT NULL	0	
	MARITAL_STATUS	CHARACTER (1)	NOT NULL	' '	
4) —	CHILDREN	(10) STRUCTURE			
	C_NAME	CHARACTER (30)	NOT NULL	' '	
	C_AGE	SMALLINT	NOT NULL	0	
	CUST-ADDRESS	(2) STRUCTURE			
	P_CODE	CHARACTER (5)	NOT NULL	' '	
	CITY	CHARACTER (15)	NOT NULL	' '	
	STREET	CHARACTER (30)	NOT NULL	' '	
	PERSONNEL_NO	INTEGER	NOT NULL	0	UNIQUE
	OCCUPATION	CHARACTER (10)	NOT NULL	' '	
16) —	SALARY	NUMERIC (10, 2)	NOT NULL	0	UNIQUE
	BONUSES	NUMERIC (10, 2)	NOT NULL	0	
	MGR_NO	INTEGER	NOT NULL	0	
9/10) —	DEPT_EMP_	INTEGER	NOT NULL		REFERENCES DEPARTMENT
13) —	PROJ_EMP_	INTEGER		NULL	REFERENCES PROJECT
11) —	PROJ_EMP_2_	INTEGER	NOT NULL ON UPDATE	NULL	REFERENCES PROJECT

- 17) — UNIQUE (P_CODE
CITY,
STREET)
- UNIQUE (DEPT_,EMP_,
MGR_NO)
- 19) — UNIQUE (DEPT_,EMP_,
OCCUPATION,
SALARY)
- 18) — UNIQUE (E_AGE,
MARITAL_STATUS)

INDEX TO BE USED BY SQL INDEX DEFINITION IN UDS

15) — INDEX (PERSONNEL_NO)

WARNING 4018 *** INDEX CAN BE USED ONLY WITHIN 'IN PREDICATE' OR WITHIN
'COMPARISON PREDICATE' WITH 'EQUALS OPERATOR'

14) — INDEX (E_NAME)

WARNING 4018 *** INDEX CAN BE USED ONLY WITHIN 'IN PREDICATE' OR WITHIN
'COMPARISON PREDICATE' WITH 'EQUALS OPERATOR'

INDEX (CUST-ADDRESS) INDEX (P_CODE,
CITY,
STREET)

INDEX (DEPT_,EMP_,
MGR_NO) INDEX (DEPT_,EMP_,
OCCUPATION,
SALARY)

WARNING 4017 *** INDEX CANNOT BE USED IN SQL (COMPOUND KEY)

INDEX (SALARY)

WARNING 4018 *** INDEX CAN BE USED ONLY WITHIN 'IN PREDICATE' OR WITHIN
'COMPARISON PREDICATE' WITH 'EQUALS OPERATOR'

INDEX (E_AGE,
MARITAL-STATUS)

WARNING 4018 *** INDEX CAN BE USED ONLY WITHIN 'IN PREDICATE' OR WITHIN

'COMPARISON PREDICATE' WITH 'EQUALS OPERATOR'

TABLE PROJECT

ATTRIBUTE	TYPE	NOT NULL	DEFAULT	
PROJECT_	INTEGER	NOT NULL		PRIMARY KEY SYSTEMDEFINED
PROJ_NAME	CHARACTER (30)	NOT NULL	' '	
BUDGET	NUMERIC (12, 2)	NOT NULL	0	
PROJ_MGR	INTEGER	NOT NULL	0	
12) — INT_FUND_	INTEGER	NOT NULL ON INSERT		REFERENCES SYSTEM
EXT_FUND_	INTEGER		NULL	REFERENCES SYSTEM

UNIQUE (INT_FUND_, PROJ_NAME)

INDEX TO BE USED BY SQL INDEX DEFINITION IN UDS

INDEX (INT_FUND_, PROJ_NAME)

WARNING 4018 *** INDEX CAN BE USED ONLY WITHIN 'IN PREDICATE' OR WITHIN 'COMPARISON PREDICATE' WITH 'EQUALS OPERATOR'

TABLE SYSTEM

ATTRIBUTE	TYPE	NOT NULL	DEFAULT	
8) — SYSTEM_	INTEGER	NOT NULL	0	PRIMARY KEY SYSTEMDEFINED

*** SHORT INFORMATION ABOUT RELATIONAL SCHEMA

RELATIONAL SCHEMA CAN BE PROCESSED WITH SQL

*** SHORT INFORMATION ABOUT TABLES

RET = Y : TABLE CAN BE PROCESSED WITH SQL
 = N : TABLE CANNOT BE PROCESSED WITH SQL
 INS = N : NO INSERT ALLOWED ON TABLE
 UPD = N : NO UPDATE ALLOWED ON TABLE
 ATR = Y : ALL ATTRIBUTES CAN BE PROCESSED WITH SQL

TABLE	RET	INS	UPD	ATR
DEPARTMENT	Y	Y	Y	Y
EMPLOYEE	Y	Y	Y	Y
PROJECT	Y	Y	Y	Y

***** DIAGNOSTIC SUMMARY FOR SUBSCHEMA STAFF

NO ERRORS

+++++ 4 WARNINGS

***** END OF DIAGNOSTIC SUMMARY

6 Printing statistics on the occupied storage space with BSTATUS

BSTATUS prints tabular overviews showing the status of a user's realms.

These overviews contain the following information:

- the free storage space in the realms,
- DBTT sizes and the number of possible/existing DBTT entries,
- the occupancy level of the tables and the storage space they require,
- the utilization of reserved hash areas and the number of overflow pages, and
- the distribution of record types over realms,

The status reports thus permit optimum use of the storage space reserved for the user's realms.

In addition to outputting the data to SYSLST, you can also output it to a file in CSV format. The use of CSV format facilitates the further processing of the data in other system environments (e.g. in spreadsheet applications). The output in CSV format is described in the manual "[Database Operation](#)", section "Outputting database information in a neutral format".

6.1 Functions

The amount of storage space required for the data in the database varies during processing and is dependent on the nature of the DB applications storing and erasing records.

BSTATUS can be used to obtain an overview of the occupied storage space, giving the DB administrator complete control over storage space allocation. He is thus able to:

- adapt the storage space occupied by the realms of his database to immediate needs so as not to take up more space than is required
- prepare for DB applications which insert new records by allocating additional space to those DB elements (realms, tables) which have become too small to accommodate new records.

There is online access to the original database, i. e. in parallel with database operation, or to a shadow database.



When you run BSTATUS online, you must assume that the data output is not current since the DBH has not copied all the data from the buffer to the database yet. In order to obtain as much current data as possible, you should force a database update just before the BSTATUS run using the DAL command CHECKPOINT or NEW RLOG. However, the BSTATUS output may still differ from the actual contents of the database if an update is running parallel to this task

The tables printed by BSTATUS provide the following specific information:

- Realm statistics - used/unused storage space per realm:
 - size of the realm in pages
 - number of unused pages
 - number of pages partially filled
 - number of full pages
 - total number of unused bytes in the realm

- Set statistics - storage space (per set) occupied by tables
 - number of set occurrences
 - number of stored member records in the largest and smallest set occurrences, and average number of member records in each set occurrence

The following information is recorded for each table in the set:

- column numbers in the owner DBTT containing the addresses of the tables
 - filling ratio (= occupancy level) on index level 0 (main level)
 - filling ratio on all index levels other than the main level
 - maximum and average number of index levels other than the main level
 - number of set occurrences in which reorganization by BREORG results in a reduction in the number of index levels.
-
- Owner statistics - storage space (per set) occupied by the tables for one owner
 - number of member records
 - for set occurrence table of the owner, the DBTT column number containing the address of the table
 - filling ratio (= occupancy level) on index level 0 (main level)
 - filling ratio on all index levels other than the main level
 - number of index levels other than the main level
 - flag indicating whether BREORG can reduce the number of index levels.
-
- Record type statistics - used/unused DBTT entries per record type:
 - number of used DBTT entries, i.e. number of records stored
 - highest record sequence number
 - highest record sequence number possible, i.e. maximum number of records of this record type which can be stored
 - DBTT filling ratio as a percentage

- CALC key statistics - the storage space used by the primary pages and overflow pages per hash area:
 - number of reserved primary pages
 - number of records (for a direct hash area) or number of pointers (for an indirect hash area) which can still be added
 - number of empty primary pages
 - occupancy level of the primary pages
 - number of overflow pages
 - number of records or pointers in the overflow pages
 - occupancy level of the overflow pages
 - depth factor, i.e. the average number of accesses required to locate a record

- Record number statistics - the records of a record type stored in a realm
 - for one or more specified realms: Number of records per record type which have been stored in this realm
 - for one or more specified record types: Number of records of specified record types stored per realm

BSTATUS can also be used to obtain a printout of the statistics on the storage space occupied by

- the database directory (DBDIR) and/or
- the database compiler realm (DBCOM).

6.2 System environment

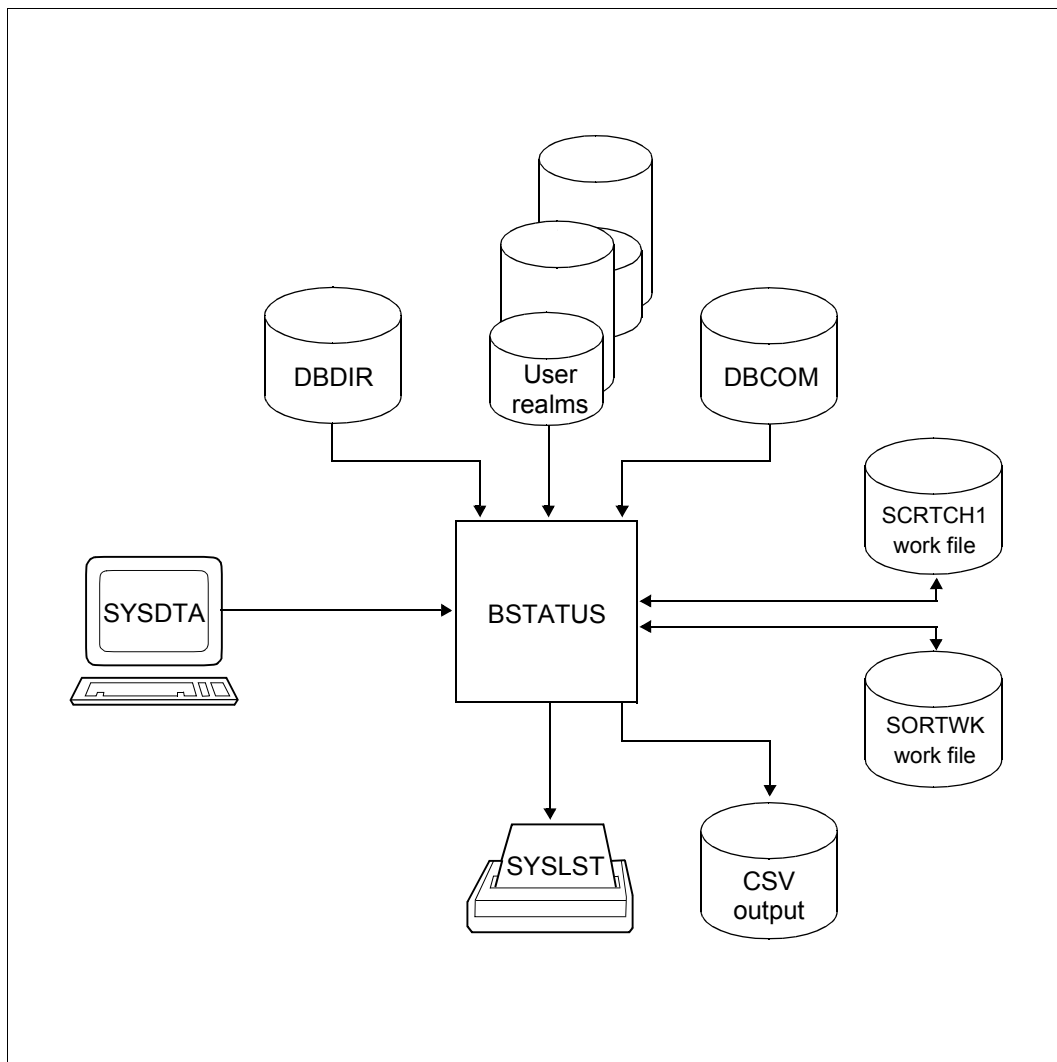


Figure 15: BSTATUS system environment

At startup BSTATUS takes into account any assigned UDS/SQL pubset declaration (see the [“Database Operation”](#) manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

Work files

In order to buffer and sort the set and owner statistics, BSTATUS requires two work files. These work files are set up automatically on public disk by BSTATUS under the user ID under which it was started. The default link names of these files are SCRTCH1 and SORTWK.

SCRTCH1

BSTATUS requires this file as temporary storage for the set statistics when the output of set and owner statistics is requested.

SORTWK

Requires the SORT used by BSTATUS for sorting internal evaluation records (see manual "[SORT \(BS2000\)](#)").

If the work files are to be created explicitly, they must be assigned the following attributes:

Work-file-1

File link name SCRTCH1

Access method = SAM

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

$$132 * (\text{no. of sets} + \text{no. of keys}) \text{ Bytes}$$

no. of sets

Number of sets in the subschema to be checked

no. of keys

Number of keys in the subschema to be checked

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.

Work-file-2

SORT needs Work-file-2 if there is not enough virtual memory for pre-sorting. The primary allocation should be based on the data population that is to be sorted while taking account of the safety factor recommended by SORT (see the discussion of work files in the manual “[SORT \(BS2000\)](#)”). There should always be an appropriate secondary allocation in case it is necessary to extend the storage space.

File link name SORTWK

Access method=PAM

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

$$16 * \text{no. of sort records Bytes}$$

no. of sort records

Number of records to be processed in the table statistics.

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

UTI.SAMWORK. <i>tsn.time-stamp.nnnn</i>	(33,33)
UTI. <i>tsn</i> .SORTWK	(120,120)

tsn task sequence number of the current task.

time-stamp

date and time (yyyymmddhhmmss) on which the file was created.

nnnn four-digit sequential number.

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.

6.3 BSTATUS statements

Statement	Meaning
SUBSCHEMA	Print realm statistics
DISPLAY REALM	Designate subschema
DISPLAY TABLE FOR SET	Print set statistics
DISPLAY TABLE FOR OWNER	Print owner statistics
DISPLAY RECORD	Print record type statistics
DISPLAY CALC	Print CALC key statistics
DISPLAY RECORDNUMBER	Print record number statistics
END	Terminate statement input

Table 16: BSTATUS statements

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

The statements can extend over several lines. However, each line is limited to 72 characters. A continuation character is not required in multiple-line notation.

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

Designate the subschema (SUBSCHEMA)

SUBSCHEMA IS *subschema-name*

subschema-name

Name of subschema for which statistics are to be printed.

The following can be specified:

- *user-subschema-name* For statistics on user realms
- COMPILER-SUBSCHEMA For statistics on the DBCOM
- PRIVACY-AND-IQF-SS For statistics on the DBDIR

The SUBSCHEMA statement must be the first statement entered. All realms, record types and sets whose statistics are to be printed out using BSTATUS must be contained in the specified subschema.

Print realm statistics (DISPLAY REALM)

```
DISPLAY [IN CSV [csv-filename]] REALM STATISTICS FOR { realm-name-1, ... }  
{ ALL }
```

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

realm-name

Name of a realm for which realm statistics are to be printed.

You must specify names as follows:

- for user realms: the realm name defined in the Schema DDL by the AREA clause
- for the DBDIR: DATABASE DIRECTORY
- for the DBCOM: DATABASE COMPILER REALM

ALL BSTATUS prints realm statistics on all the realms contained in the specified subschema.

Example

DISPLAY REALM STATISTICS FOR ALL

R E A L M S	NR OF	MAX NR OF	NR PAGES WITH FILLING PERCENTAGES BETWEEN					NR FULL	TOTAL	FREE	
	EMPTY	CONTIGUOUS	0- 19	20- 39	40- 59	60- 79	80- 99	USER	NR OF	OCTADS	
	PAGES	EMPTY PAGES						PAGES	PAGES	TOTAL	
CUSTOMER-ORDER-RLM	22	22	1	0	0	0	0	10	36	91 520	
PURCHASE-ORDER-RLM	33	33	1	0	0	0	0	23	60	135K	
CLOTHING	21	20	4	0	0	0	8	6	42	101K	
HOUSEHOLD-GOODS	15	14	0	0	0	0	0	6	24	59 700	
SPORTS-ARTICLES	36	36	0	0	0	0	0	5	44	143K	
FOOD	7	6	0	0	0	0	5	3	18	29 630	
LEISURE	36	36	0	0	0	0	0	5	44	143K	
STATIONERY	17	17	0	0	0	0	0	4	24	67 660	
ARTICLE-RLM	1982	1982	3	1	0	0	0	41	2062	7 903K	
SEARCH-RLM			T E M P O R A R Y R E A L M								
T O T A L S	2 169		9	1	0	0	13	103	2 354	8 674K	

BSTATUS does not provide statistics on temporary realms.

REALMS

Names of the realms

NR OF EMPTY PAGES

Number of empty pages

MAX NR OF CONTIGUOUS EMPTY PAGES

Maximum number of contiguous empty pages

NR PAGES WITH FILLING PERCENTAGES BETWEEN

Number of partially filled pages, divided into groups with the specified occupancy level (excluding FPA pages, Act-key-0 pages, and Act-key-N pages)

NR FULL USER PAGES

Number of full pages (excluding FPA pages, Act-key-0 pages, and Act-key-N pages)

TOTAL NR OF PAGES

Size of the realm in pages (including FPA pages, Act-key-0 pages, and Act-key-N pages)

FREE OCTADS TOTAL

Total number of unused bytes per realm

K: If the number is greater than 100,000 bytes, BSTATUS rounds it off to a multiple of 1,000 =1 kbyte

M: If the number is greater than 100,000 kbytes, BSTATUS rounds it off to a multiple of 1,000 kbyte =1 Mbyte

TOTALS

If ALL is specified in the DISPLAY REALM statement, BSTATUS prints the sums of all columns in the TOTALS line.

In this case it calculates the total number of unused bytes before rounding off the numbers for the individual realms.

Print set statistics (DISPLAY TABLE FOR SET)

DISPLAY [IN CSV [*csv-filename*]] TABLE STATISTICS FOR SET

{ *set-name-1*, ... }
 { *ALL[EXCEPT *setname-1*, ...] }

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

set-name

Name of the set for which statistics are to be printed

*ALL Statistics on all sets of the subschema are to be printed

*ALL EXCEPT *set-name-1*,...

Statistics are to be printed for all sets other than those named at *set-name*.

Example

DISPLAY TABLE STATISTICS FOR SET *ALL

SET IDENTIFICATION	OWNER OCCURRENCE	MEMBER OCCURRENCE			TABLE STATISTICS						
		MIN	MAX	AVG	COL	FILLING RATIO		LEVEL	NR	NR OCCUR-	
					NR	LEV=0	LEV>0	MAX	AVG	REORGANIZE	
CST-ORD-PLACED					***** EMPTY SET *****						
OFFER	4	1	8	3	1	5	0	0	0,0	0	
SHORT-LIST	5	1	6	2	1	4	0	0	0,0	0	
P-ORD-SPEC	13	1	7	4	1	33	0	0	0,0	0	
MIN-STOCK-LEVEL	SINGULAR			0	1	0	0		0	0	
CONTAINING					***** EMPTY SET *****						
CONTAINED-IN					***** EMPTY SET *****						
SUPPLIERS	SINGULAR			1	1	2	0		0	0	
ARTICLES-AVAILABLE	1	63	63	63	1	80	0	0	0,0	0	
					2	10	0		0,0		
P-ORD-CONTENTS					***** EMPTY SET *****						
RESULT-SET					***** DYNAMIC SET *****						
SYS_INSTALLMENT	SINGULAR			0	1	1	0		0	0	
SYS_ARTICLE-TYPE	SINGULAR			4	**** CHAIN SET WITHOUT TABLES ****						0
SYS_ARTICLE-SELECTION	SINGULAR			5	1	5	0		0	0	
SYS_ARTICLE	SINGULAR			63	**** CHAIN SET WITHOUT TABLES ****						0
SYS_COLORS	SINGULAR			25	**** CHAIN SET WITHOUT TABLES ****						0
SYS_MATERIALS	SINGULAR			10	1	3	0		0	0	
					2	8	0		0		
IMPLICIT_RESULT_SET					***** DYNAMIC SET *****						

BSTATUS does not print statistics for dynamic sets or for sets for which there are no set occurrences. Such sets are indicated as an *EMPTY SET*.

SET IDENTIFICATION

Name of set in schema

OWNER OCCURRENCE

Number of occurrences of the set

SINGULAR: SYSTEM set

MEMBER OCCURRENCE

Number of stored member records in the set

MIN: Number of member records in the smallest set occurrence

MAX: Number of member records in the largest set occurrence

AVG: Average number of member records per set

TABLE STATISTICS

Information on the tables in the set

(pointer arrays, sort key tables or SEARCH key tables)

If the set does not have any tables, the following message is printed:

```
**** CHAIN SET WITHOUT TABLES ****
```

COL NR

Column numbers in the owner DBTT containing the address of the given table

FILLING RATIO

Table occupancy level, expressed as a percentage of all bytes reserved for the table;

LEV=0: Average table occupancy on index level 0 (main level)

0: Tables are set up, but no member records stored (empty set occurrence). For duplicates tables the occupancy level is always greater than 0, even in the case of empty set occurrences.

LEV>0: Average table occupancy on all index levels (main level not included)

LEVEL NR

Number of index levels in set (main level not included)

MAX: Highest index level in any table in a set occurrence

AVG: Average number of index levels in set

NR OCCURRENCES TO REORGANIZE

Number of set occurrences for which BREORG can be used to reduce the number of index levels; applies to any reorganization using the statement

```
REORGANIZE SET NAME IS set-name FILLING IS 100 PERCENT
```

Print owner statistics (DISPLAY TABLE FOR OWNER)

```

DISPLAY [IN CSV [csv-filename]] TABLE STATISTICS FOR OWNER IN SET
                                     { set-name-1[rsq-selection-1] ,... }
                                     { *ALL[ EXCEPT set-name-1,...] }

```

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

set-name

Name of a set for whose owner statistics are output

rsq-selection

rsq-selection = { ALL RSQS / RSQ {*rsq-1*[IO *rsq-2*]},... }

$1 \leq \text{RSQ} \leq 2^{24} - 2^{16} - 1$

For databases with a page length of 2048 bytes if the record type is an owner in a set.

$1 \leq \text{RSQ} \leq 231 - 1$

For databases with a page length of 4000/8096 bytes.

*ALL Statistics are to be printed for all owner record types of the subschema

*ALL EXCEPT *set-name-1*,...

Statistics are to be printed for all sets other than *set-name*.

The DISPLAY TABLE FOR OWNER statement prints out statistics on the storage space occupied by the tables of the owner records of a set.



This output may be very large since, unlike the other statements, its scope does not depend only on the metadata population but also on the user data population.

Example

1. DISPLAY TABLE STATISTICS FOR OWNER IN SET P-ORD-SPEC RSQ 1 TO 5

```

*-----*-----*-----*-----*
|                                     | TABLE STATISTICS | | | | |
|                                     |-----|-----|-----|-----|
| OWNER DBK | MEMBER | COL | FILLING RATIO | LEV | REORG |
|           | OCCUR- | NR  | LEV=0 | LEV>0 | NR  |
|           | RENCE  |     |       |       |     |
*-----*-----*-----*-----*
| 8:        | 1      | 7   | 1      | 47   | 0   | 0   | -   |
*-----*-----*-----*-----*
| 8:        | 2      | 6   | 1      | 41   | 0   | 0   | -   |
*-----*-----*-----*-----*
| 8:        | 3      | 2   | 1      | 14   | 0   | 0   | -   |
*-----*-----*-----*-----*
| 8:        | 4      | 2   | 1      | 14   | 0   | 0   | -   |
*-----*-----*-----*-----*
| 8:        | 5      | 2   | 1      | 14   | 0   | 0   | -   |
*-----*-----*-----*-----*

```

BSTATUS prints a separate table for each set.

OWNER DBK

Database key of the owner record in the format: (*ref:rsq*)

MEMBER OCCURRENCE

Number of members of the set occurrence specified by OWNER DBK

0: No member records stored; but table has been set up

TABLE STATISTICS

Information on all tables of the set occurrence

(pointer array, list or sort key table and SEARCH key tables)

COL NR

Column number in the owner DBTT containing the address of the corresponding table

FILLING RATIO

Table occupancy level, expressed as a percentage of all bytes reserved for the table;

LEV=0: Table occupancy level on index level 0 (main level):
 0: Tables have been set up, but no member records have been stored. For duplicates tables the occupancy level is always greater than 0, even in the case of empty set occurrences.

LEV>0: Table occupancy level on all index levels (excluding main level)

LEV NR

Number of index levels (excluding main level)

REORG

Indicates whether the number of index levels can be reduced by reorganization.

YES: the number of index stages can be reduced with the aid of the statement REORGANIZE SETNAME IS *set-name* FILLING IS 100 PERCENT in the BREORG utility

NO: it is not possible to reduce the number of index stages

-: the table has only one level (main level), so it is only possible to increase table occupancy

2. In case of a user specifying an owner RSQ or a range of RSQs, although these RSQs do not exist in the database, the user gets a message:

DISPLAY TABLE FOR OWNER IN SET P-ORD-SPEC RSQ 2

NO MATCHING RSQS: There is no record with RSQ 2 in the database.

```

*-----*-----*-----*-----*
|          |          |          |          |          |          |          |
|          | MEMBER  |          |          |          |          |          |
| OWNER DBK | OCCUR-  | COL | FILLING RATIO | LEV |          | REORG |
|          | RENCE   |     | *-----*-----* |     |          |
|          |          | NR  | LEV=0 | LEV>0 | NR  |          |
*-----*-----*-----*-----*
| ***** N O M A T C H I N G   R S Q S ***** |          |          |
*-----*-----*-----*-----*
    
```

Print record type statistics (DISPLAY RECORD)

```
DISPLAY [IN CSV [csv-filename]] RECORD STATISTICS FOR { record-name-1, ... }  
                                                                { ALL }
```

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

record-name

Name of a record type contained in the specified subschema for which record type statistics are to be printed out

ALL BSTATUS prints the record type statistics on all record types contained in the specified subschema

Example

DISPLAY RECORD STATISTICS FOR ALL

STATUS PER RECORD TYPE

R E C O R D S	D B T T						
	NR OF ENTRIES USED	HIGHEST RSQ USED	HIGHEST RSQ POSSIBLE	FILLING RATIO %	EXTENDIBLE	NR OF DBTT EXTENTS	HIGHEST USED EXTENT
CUSTOMER	0	0	331	0,0	SCAN	0	0
CST-ORDERS	0	0	497	0,0	SCAN	0	0
ORD-ITEM	0	0	1990	0,0	SCAN	0	0
INSTALMENT	0	0	995	0,0	SCAN	0	0
ART-TYPE	4	4	497	0,8	SCAN	0	0
ART-SELECTION	5	5	497	1,0	SCAN	0	0
ART-DESCR	13	13	497	2,6	SCAN	0	0
ARTICLE	63	63	995	6,3	SCAN	0	0
SUBSET	0	0	995	0,0	SCAN	0	0
COLORS	25	25	995	2,5	SCAN	0	0
MATERIALS	10	10	995	1,0	SCAN	0	0
SUPPLIER	1	1	662	0,1	SCAN	0	0
PURCHASE-ORDER	0	0	497	0,0	SCAN	0	0
P-ORD-ITEM	0	0	995	0,0	SCAN	0	0
T O T A L	121						

RECORDS

Names of record types

DBTT For each record type BSTATUS prints out the most important DBTT information

NR OF ENTRIES USED

The number of DBTT entries used is equivalent to the number of currently stored records of the record type

HIGHEST RSQ USED

Highest record sequence number assigned; if records have been erased, this number may be higher than the number of records currently stored

HIGHEST RSQ POSSIBLE

Highest record sequence number possible, i.e. greatest number of records of this record type which can be stored in the current DBTT

FILLING RATIO %

DBTT occupancy level, expressed as a percentage of the allocated DBTT entries

EXTENDIBLE:

Specifies whether or not the record type can be extended:

NO cannot be extended

SCAN can be extended with parameter SCAN=YES

NOSCAN can be extended with parameter SCAN=NO

NR OF DBTT EXTENTS:

Number of DBTT extents currently present

HIGHEST USED EXTENT:

Number of DBTT extents currently present minus the number of consecutive completely empty extents at the end of the DBTT. This information can be of use in the case of a DBTT reduction using BREORG.

TOTAL

Outputs the total number of DBTT entries used.

The DBTT entries which are locked in the case of the BMODTT statement KEEP are not counted in any specifications; only those for which records have been stored are regarded as "occupied".

Print CALC key statistics (DISPLAY CALC)

`DISPLAY` [`IN CSV` [*csv-filename*]] `CALC` KEY STATISTICS FOR

$$\left. \begin{array}{l} \text{RECORD} \left\{ \begin{array}{l} \textit{record-name-1}, \dots \\ \text{ALL} \end{array} \right\} \\ \text{SEARCHKEY} \left\{ \begin{array}{l} \textit{keyref-1}, \dots \\ \text{ALL} \end{array} \right\} \end{array} \right\} \text{IN REALM} \left\{ \begin{array}{l} \textit{realmname-1}, \dots \\ \text{ALL} \end{array} \right\}$$

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

RECORD

BSTATUS prints CALC key statistics for record types defined with LOCATION MODE IS CALC

record-name

Name of a record type for which BSTATUS is to print out the CALC key statistics

ALL

BSTATUS prints CALC key statistics for all record types which have been defined with LOCATION MODE IS CALC in the specified subschema

SEARCHKEY

BSTATUS prints the CALC key statistics for the hash areas of CALC SEARCH keys defined on record type or set level

keyref

Number of the CALC SEARCH key whose CALC key statistics are to be printed by BSTATUS.

The numbers can be taken from the SIA PRINT REPORT (see [page 158](#))

ALL

BSTATUS prints CALC key statistics for all hash areas of CALC SEARCH keys that are contained in the subschema

REALM

BSTATUS prints CALC key statistics for one or more realms

realmname

Name of a realm for which BSTATUS is to print out the CALC key statistics

ALL

BSTATUS prints CALC key statistics for all realms

Example

DISPLAY CALC SEARCHKEY 12,14,15,16,17 IN REALM ARTICLE-RLM.

STATUS OF CALC KEY BUCKETS – PER CALC KEY

R E A L M : ARTICLE-RLM

C A L C K E Y S		NR OF PRIMARY BUCKETS	RECORDS/ POINTERS THAT CAN BE ADDED	NR OF EMPTY BUCKETS	FILLING % PRIMARY BUCKETS	NR OF OVERFLOW BUCKETS	RECORDS/ POINTERS IN OVERFLOW	FILLING % OVERFLOW BUCKETS	DEPTH FACTOR FOR RECORD
KEY-REF:	12	1	109	0	4	0	0	-	2,00
KEY-REF:	14	3	597	0	10	0	0	-	2,00
KEY-REF:	15	11	806	3	7	0	0	-	2,00
KEY-REF:	16	1	107	0	19	0	0	-	2,00
KEY-REF:	17	1	305	0	8	0	0	-	2,00

BSTATUS prints separate CALC key statistics for each realm.

RECORDS

Only if DISPLAY CALC RECORD is specified:

Names of the record types for which BSTATUS is to print out the CALC key statistics

CALC KEYS

KEY REF: *keyref*

Only if DISPLAY CALC SEARCHKEY is specified:

Numbers of CALC SEARCH keys for which BSTATUS is to print out the CALC key statistics

NR OF PRIMARY BUCKETS

Number of primary pages reserved for the hash area

RECORDS/POINTERS THAT CAN BE ADDED

Number of records (for a direct hash area) or pointers (for an indirect hash area) that can still be stored in the primary pages of the hash area

NR OF EMPTY BUCKETS

Number of empty primary pages

FILLING % PRIMARY BUCKETS

Occupancy level of the primary pages

NR OF OVERFLOW BUCKETS

Current number of overflow pages

RECORDS/POINTERS IN OVERFLOW

Number of records (for a direct hash area) or number of pointers (for an indirect hash area) in the overflow pages

FILLING % OVERFLOW BUCKETS

Occupancy level i.e. occupancy level of the overflow pages

DEPTH FACTOR FOR RECORD

Depth factor for accessing a record, i.e. average number of accesses required for locating a record

BSTATUS calculates the depth factor according to the following formulas:

- Direct hash area

$$d = \sum_{i=1}^n \frac{(\text{no. of records in page}_i) \times \text{rank}_i}{\text{total no. of records}}$$

- Indirect hash area:

$$d = \sum_{i=1}^n \frac{(\text{no. of pointers in page}_i) \times \text{rank}_i}{\text{total no. of pointers}} + 1$$

n Number of all pages in the hash area (primary and overflow pages)

rank_i Ranking of the i -th CALC page in a chain of primary and overflow pages
 Primary page: rank 1
 1st overflow page: rank 2
 2nd overflow page: rank 3
 and so forth

d Depth factor

total no. of records

Number of all records stored in the primary and overflow pages of the hash area

total no. of pointers

Number of all pointers stored in the primary and overflow pages of the hash area

Print record number statistics (DISPLAY RECORDNUMBER)

DISPLAY [IN CSV [*csv-filename*]] RECORDNUMBER STATISTICS FOR

$$\left. \begin{array}{l} \left. \begin{array}{l} \text{REALM} \left\{ \begin{array}{l} \textit{realm-name-1}, \dots \\ \text{ALL} \end{array} \right\} \\ \text{RECORD} \left\{ \begin{array}{l} \textit{record-name-1}, \dots \\ \text{ALL} \end{array} \right\} \end{array} \right\} \end{array} \right\}$$

IN CSV

BSTATUS also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BSTATUS run (e.g. DISPLAY IN CSV 'BSTATUS.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

REALM

For each record type which can occur in the specified realm(s), BSTATUS prints the number of records stored therein

realm-name

Name of the realm for which BSTATUS is to print out the record number statistics

ALL

BSTATUS prints the record number statistics for all realms of the subschema

RECORD

For each specified record type, BSTATUS prints out the number of records stored in the realms in which the record type can occur

record-name

Name of the record type for which BSTATUS is to print out the record number statistics

ALL

BSTATUS prints the record number statistics for all record types of the subschema

Example

DISPLAY RECORDNUMBER STATISTICS FOR REALM ALL.

NUMBER OF RECORDS PER REALM

* RECORDS	* REALM REFS									
	3	4	5	6	7	8	9	10		
CUSTOMER	0	-	-	-	-	-	-	-	-	-
CST-ORDERS	0	-	-	-	-	-	-	-	-	-
ORD-ITEM	0	-	-	-	-	-	-	-	-	-
INSTALLMENT	0	-	-	-	-	-	-	-	-	-
ART-TYPE	-	-	1	0	0	3	0	0		
ART-SELECTION	-	-	2	0	0	3	0	0		
ART-DESCR	-	-	8	0	0	5	0	0		
ARTICLE	-	-	55	0	0	8	0	0		
SUBSET	-	-	-	0	0	-	-	-		
COLORS	-	-	-	-	-	-	-	-		
MATERIALS	-	-	-	-	-	-	-	-		
SUPPLIER	-	1	-	-	-	-	-	-		
PURCHASE-ORDER	-	0	-	-	-	-	-	-		
P-ORD-ITEM	-	0	-	-	-	-	-	-		

- AREA REF 3 = CUSTOMER-ORDER-RLM
- AREA REF 4 = PURCHASE-ORDER-RLM
- AREA REF 5 = CLOTHING
- AREA REF 6 = HOUSEHOLD-GOODS
- AREA REF 7 = SPORTS-ARTICLES
- AREA REF 8 = FOOD
- AREA REF 9 = LEISURE
- AREA REF 10 = STATIONERY

RECORDS

Names of record types

REALM REFS

Realm numbers; for each realm BSTATUS sets up a column containing the number of records stored in the realm

0: The record type can be contained in the realm, but no record is stored there

-: The record type cannot occur in the realm

AREA REF

Assignment of realm numbers to realm names

Terminate input (END)

END

6.4 Command sequence to start BSTATUS

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

```
01 /ADD-FILE-LINK LINK-NAME=DATABASE,  
    FILE-NAME=[ :catid: ][ $userid. ]dbname.DBDIR[ .copy-name]  
02 /SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,VERSION=version.  
03 /START-UDS-BSTATUS  
04 SUBSCHEMA IS subschema-name  
05 display-statements  
06 END
```

- 01 In this case specifying *:catid:* is permitted (see in the “[Database Operation](#)” manual).
- 02 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.
- 03 The UDS/SQL utility routine can also be started with the alias BSTATUS.

7 Printing out the contents of realms with BPRECORD

BPRECORD prints the contents of the user's database.

The following information can be obtained from the printout:

- from the realm information page (Act-Key-0 page) the date of the last realm update and system break information
- from the FPA entries the location and size of unused space in the realms (important for reorganizing a realm)
- from the DBTTs the available record sequence numbers (important for LOCATION MODE IS DIRECT or DIRECT-LONG)
- from the printout of the hash areas the distribution of records over the hash area (basis for improving hash routines)
- from the printout of data records and tables the connection of records to tables (useful for debugging DB applications)

With BPRECORD the DB administrator can print the contents of the user realms, of the database directory (DBDIR) or of the database compiler realm (DBCOM).

The DB administrator can select individual sections of a realm according to two aspects:

- Logical The administrator selects a particular record type, certain records of a record type, a table, etc.
- Physical The administrator selects the realm, certain page types certain page numbers, etc.

The two options may also be combined. When printing out CALC and data pages, the administrator may also specify whether BPRECORD is to include the page header (PAGE INFO), the page index entries (PAGE INDEX) or the set connection data (SCD) of the records in the BPRECORD printout (see [page 241](#)).

To prevent unauthorized access, BPRECORD can only be invoked under the DB administrator identification.

In addition to outputting the data to SYSLST, you can also output it to a file in CSV format. The use of CSV format facilitates the further processing of the data in other system environments (e.g. in spreadsheet applications). The output in CSV format is described in

the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.



When you run BPRECORD online, you must assume that the data output is not current since the DBH has not copied all the data from the buffer to the database yet. In order to obtain as much current data as possible, you should force a database update just before the BPRECORD run using the DAL command CHECKPOINT or NEW RLOG. However, the BPRECORD output may still differ from the actual contents of the database if an update is running parallel to this task.

7.1 System environment

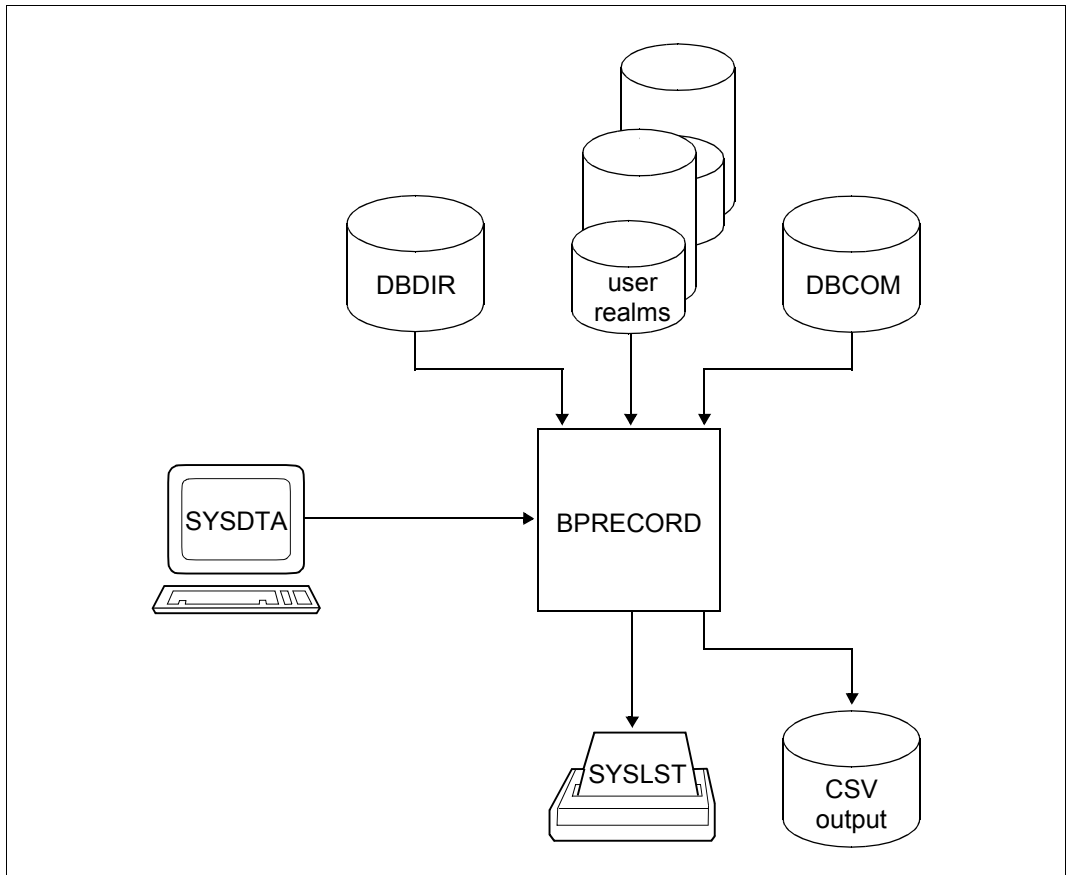


Figure 16: BPRECORD system environment

At startup BPRECORD takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

7.2 General description of the output of BPRECORD

BPRECORD is used to print out any combination of the five different page types making up a realm in the following sequence:

1.

***** ACTUAL-KEY-0 OF REALM < :SQL2:\$XXXXXXXX.SHIPPING.ARTICLE-RLM

PHYS REC LENGTH : 4000

DATASET INFO :

TOTAL NR PAGES: 2062
 FPA BASE BEGIN: AREA REF: 11
 BNR: 1
 NR PAGES IN FPA BASE: 1990

FPA EXTENTS

NR BNR
 1 47
 MAX NR PAGES IN EXTENT: 63680

HIGHEST PAGE NR FOR FORMATTING: 2061

CREATE DATA: DATE : 20150628 TIME: 095430

BACK UP DATA: DATE : 20150628 TIME: 095509

REALM VERSION NR: 3

SYSTEM BREAK:

OCCURRED: 0
 ADMIN USERID: \$XXXXXXXX
 CONFNAME: BREORG

FILE NAME: :SQL2:\$XXXXXXXX.SHIPPING.ARTICLE-RLM

REALM LAYOUT VERSION: 004.00

UDS VERSION: V2.8

INCR NR PAGES: 64

INCR MIN PAGES: 0

2.

***** F.P.A. ENTRIES OF REALM < :SQL2:\$XXXXXXXX.SHIPPING.ARTICLE-RLM >

	REALM	REF	11	BNR	1	ACT	KEY	X'0B000001'		
0.....29:	0	0	0	0	0	0	0	0	0	0
30.....39:	0	0	3960	3964	0	0	3956	0	0	0
40.....49:	0	0	0	0	0	0	2800	0	0	0
50.....69:	0	0	0	0	0	0	0	0	0	0
70.....79:	0	0	0	0	0	0	0	0	0	3980
80.....1989:	3980	3980	3980	3980	3980	3980	3980	3980	3980	3980
	REALM	REF	11	BNR	47	ACT	KEY	X'0B00002F'		
1990.....2059:	3980	3980	3980	3980	3980	3980	3980	3980	3980	3980
2060.....2061:	3980	0								

3.

```
***** D.B.T.T. ENTRIES OF REALM < :SQL2:$XXXXXXX.SHIPPING.ARTICLE-RLM >*****
===== RECORD REF      6, NAME: < ART-TYPE > =====
----- REALM REF  11  BNR      3  ACT KEY  X'0B000003' -----
RSQ      1/X'00000001': ( 5,      12/X'0500000C') ( 11,      37/X'0B000025')
RSQ      2/X'00000002': ( 8,      4/X'08000004') ( 11,      39/X'0B000027')
RSQ      3/X'00000003': ( 8,      5/X'08000005') ( 11,      41/X'0B000029')
RSQ      4/X'00000004': ( 8,      6/X'08000006') ( 11,      44/X'0B00002C')
RSQ      5/X'00000005'- 11/X'0000000B': ( 0,      0/X'00000000') ( 0,      0/X'00000000')
```

4.

```
***** CALC KEY BUCKETS OF REALM < :SQL2:$XXXXXXX.SHIPPING.ARTICLE-RLM >*****
===== CALC SEARCH KEY, KEY REF  14, SET NAME < SYS_ARTICLE > =====
----- REALM REF  11  BNR      16  ACT KEY  X'0B000010' -----
PAGE INFO:  TYPE 0  FREE SPACE  SIZE  10,  DISPL  30  NR OF PAGE INDICES  0  DISPL TO CALC TABLE HEADER  30
CALC KEY TABLE:  MAX ENTRIES  220  ACT ENTRIES  25  OVERFLOW BUCKET NEXT  0,  PRIOR  0
-1-
(  1) (00000000)  F2F3F0F1 F0F7F3F6 23010736
   RSQ      1  PPP  5,  13
-2-
(  1) (00000000)  F2F3F0F1 F0F7F4F2 23010742
   RSQ      4  PPP  5,  13
-3-
(  1) (00000000)  F2F3F0F1 F0F7F4F8 23010748
   RSQ      7  PPP  5,  13
.
.
.
```

5.

```
***** DATA / TABLE PAGES OF REALM < :SQL2:$XXXXXXX.SHIPPING.CLOTHING >*****
----- REALM REF  5  BNR      19  ACT KEY  X'05000013' -----
PAGE INFO:  TYPE 0  FREE SPACE  SIZE  2081,  DISPL  2209  NR OF PAGE INDICES  9  DISPL TO END OF PAGE  4000
LOGICAL RECORDS:
-2- PAGE INDEX:  DB_KEY  9,      48  COL-NR  0  LIST REC  DISPL  3767
(  1) (00000000)  00090000 00000030 05000013 00090000 00000030 05000013 00090000 00000000 00000030
(  33) (00000020)  05000013 00090000 00000030 05000013 00090000 00000030 05000013 00090000
(  65) (00000040)  00000030 05000013 00090000 00000030 05000013 00000000 000A0500 0012FF00
(  97) (00000060)  00000000 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
(  129) (00000080)  F8F3F5F9 F2F8F0F9 E360E2C8 C9D9E340 C4D9C5E2 E2404040 40404040 40404040 83592809T-SHIRT DRESS
(  161) (000000A0)  40404040 40404040 40404040 40404040 F2F3F7F1 F0F2F3F6 0020700C 0023900C 23710236
(  193) (000000C0)  00000500 000C050C 00000499 950C0000 00000000 050C00
```

The structure and function of the different page types are described in the “[Design and Definition](#)” manual.

SCHEMA NAME *schema-name*

Name of the schema, whose realms are to be printed by BPRECORD

ACTUAL-KEY-0 OF REALM *realm-name*

Realm information page of the realm *realm-name*; BPRECORD optionally prints the realm information page in normal text

F.P.A. ENTRIES OF REALM *realm-name*

FPA (free place administration) entries of the realm *realm-name*; BPRECORD optionally prints all FPA entries or only the FPA entries of certain pages

REALM REF *realm-ref* BNR *pnr*

BPRECORD prints the page address before the contents of each page

realm-ref: Number of the realm

pnr: Page number

D.B.T.T. ENTRIES OF REALM *realm-name*

DBTT entries of the realm *realm-name*; BPRECORD optionally prints:

- all DBTTs,
- only the DBTT of a certain record type, or
- only the DBTT entries of certain records of a record type

RECORD REF *recref*, NAME: *record-name*

is the header of the DBTT of a record type

recref: Number of the record type (record reference)

record-name: Name of the record type

CALC KEY BUCKETS OF REALM *realm-name*

Direct CALC pages or indirect CALC pages (primary and overflow pages) of the realm *realm-name*

CALC RECORD, REC REF *recref*, RECORD NAME *record-name*

Direct or indirect CALC pages of the record type *record-name*

CALC SEARCH KEY, KEY REF *keyref*, SET NAME *set-name*

CALC pages of CALC SEARCH key *keyref*

keyref: Number of the key

set-name: Name of the associated set

DATA/TABLE PAGES OF REALM *realm-name*

Data pages of the realm *realm-name*, which contain records (excluding CALC records) or tables and/or table entries;

BPRECORD only prints pages which are not empty. To decide whether or not a page is empty, BPRECORD first checks the FPA entries.

CALC pages and data pages are sorted according to ascending page numbers.

7.3 BPRECORD statements

Statement	Meaning
SCHEMA NAME	Name of schema which contains the realm to be printed (optional)
REALM NAME	Name of the realm to be printed (mandatory)
PRINT	Determine extent of output (optional)
DISPLAY PAGE	Print Act-Key-0 page (optional)
DISPLAY FPA	List FPA entries (optional)
DISPLAY DBTT	List DBTT entries (optional)
DISPLAY CALC	Print CALC pages (optional)
DISPLAY DATA	Print data pages (optional)
END	Terminate BPRECORD run (mandatory)

Table 17: BPRECORD statements

BPRECORD combines all DISPLAY statements for a certain realm and sorts them to avoid duplicate printouts.

The statements can extend over several lines. However, each line is limited to 72 characters. A continuation character is not required in multiple-line notation.



For purposes of clarity, PRINT and DISPLAY statements are best separated by a semicolon.

If a period (.) were used for separation, this would signify the end of statements entered for one realm. BPRECORD would then expect a new REALM statement or the END statement to be entered.

After SCHEMA-NAME/REALM-NAME, you must specify at least one DISPLAY statement.

Example for a statement sequence

```
SCHEMA NAME IS schema-name.
REALM NAME IS realm-name-1.
PRINT WITH SCD;
DISPLAY DATA PAGES ALL PAGES ALL TABLES;
DISPLAY DBTT OF ALL RECORDS.
REALM NAME IS realm-name-2.
PRINT WITH PAGEINDEX;
DISPLAY FPA OF ALL PAGES;
DISPLAY CALC PAGES ALL PAGES ALL RECORDS.
END
```

Physical selection (page selection)

The syntax elements *page-selection* and *rsq-selection* are used in a number of different statements:

$$page\text{-}selection \quad := \left\{ \begin{array}{l} \underline{ALL} \ \underline{PAGES} \\ \underline{PAGE} \ \{pno-1[\ \underline{TO} \ pno-2]\}, \dots \end{array} \right\}$$

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)

$$rsq\text{-}selection \quad := \left\{ \begin{array}{l} \underline{ALL} \ \underline{RSQS} \\ \underline{RSQ} \ \{rsq-1[\ \underline{TO} \ rsq-2]\}, \dots \end{array} \right\}$$

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.

Designate the schema (SCHEMA NAME)

SCHEMA NAME IS *schema-name*.

schema-name

Name of the schema containing the description of the realm or realms to be printed;

The following can be specified for *schema-name*:

- *user-schema-name* for a printout of a user realm
- COMPILER-SCHEMA for a printout of the DBCOM
- PRIVACY-AND-IQF-SCHEMA for a printout of the DBDIR

Default value:

User schema

The SCHEMA statement is optional. If it is specified, it must be the first BPRECORD statement, and may only be entered once.

BPRECORD accesses the SIA of the specified schema and obtains from it all information required to access the database.

Specify the realm to be printed (REALM NAME)

`REALM NAME IS realm-name.`

realm-name

Name of the realm to be printed; *realm-name* must be specified as follows:

- for user realms: the realm name defined in the AREA clause of the Schema DDL
- for the DBDIR: DATABASE-DIRECTORY
- for the DBCOM: DATABASE-COMPILER-REALM

The REALM statement must be entered at least once; it may be repeated any number of times. Note that all PRINT and DISPLAY statements referring to a particular realm must immediately follow the corresponding REALM statement. The first REALM statement must immediately follow the SCHEMA statement, or, if there is none, must be entered as the first BPRECORD statement (see [page 235](#)).

Determine scope of output (PRINT)

```

PRINT [ { WITH } PAGEINFO ] [ { WITH } PAGEINDEX ]
      [ { WITHOUT } ] [ { WITHOUT } ]
      [ { WITH } SCD ] [ DBTT { DEC } ]
      [ { WITHOUT } ] [ { WITHOUT } ] [ { HEX } ]
      [ { WITHOUT } ] [ { WITHOUT } ] [ { BOTH } ] ];

```

PAGEINFO

Page header

PAGEINDEX

Page index entries

SCD Set connection data

PAGEINFO, PAGEINDEX and SCD apply to the output of CALC and data pages.

DEC Decimal

HEX Hexadecimal

BOTH Decimal and hexadecimal

DEC, HEX and BOTH apply to the output of DBTTs.

Default values:

WITHOUT and DEC

Use of the PRINT statement is optional. It affects all DISPLAY statements which the user enters between two REALM statements or between a REALM statement and the END statement.

If the PRINT statement appears more than once, only the last statement takes effect.

Example

PRINT WITH PAGEINFO WITH PAGEINDEX WITH SCD.

***** SCHEMA NAME < MAIL-ORDERS > *****
***** DATA / TABLE PAGES OF REALM < :SQL2:\$XXXXXXX.SHIPPING.CLOTHING > *****
REALM REF 5 BNR 18 ACT KEY X'05000012'

PAGE INFO: TYPE 0 — FREE SPACE SIZE 1963, DISPL 2103 — NR OF PAGE INDICES 10 — DISPL TO END OF PAGE 4000

LOGICAL RECORDS:

-2- PAGE INDEX: DB_KEY 9, 41 — COL-NR 0 — LIST REC — DISPL 3767
(1) (00000000) 00090000 00000029 05000012 00090000 00000029 05000012 00090000 00000029
(33) (00000020) 05000012 00090000 00000029 05000012 00090000 00000029 05000012 00090000
(65) (00000040) 00000029 05000012 00090000 00000029 05000012 00000000 00090500 0011FF00
(97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
(129) (00000080) F8F3F1F2 F1F3F1F1 C6D3D6E6 C9D5C740 D1C5D9E2 C5E840C4 D9C5E2E2 40E6C9E3 83121311FLOWING JERSEY DRESS WIT
(161) (000000A0) C840D1C1 C3D2C5E3 40404040 40404040 F2F3F6F1 F1F1F3F6 0020700C 0023900C H JACKET 23611136
(193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00

-3- PAGE INDEX: DB_KEY 9, 42 — COL-NR 0 — LIST REC — DISPL 3552
(1) (00000000) 00090000 0000002A 05000012 00090000 0000002A 05000012 00090000 0000002A
(33) (00000020) 05000012 00090000 0000002A 05000012 00090000 0000002A 05000012 00090000
(65) (00000040) 0000002A 05000012 00090000 0000002A 05000012 00000000 00090500 0011FF00
(97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
(129) (00000080) F8F3F1F2 F1F3F1F1 C6D3D6E6 C9D5C740 D1C5D9E2 C5E840C4 D9C5E2E2 40E6C9E3 83121311FLOWING JERSEY DRESS WIT
(161) (000000A0) C840D1C1 C3D2C5E3 40404040 40404040 F2F3F6F1 F1F1F3F8 0020700C 0023900C H JACKET 23611138
(193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00

-4- PAGE INDEX: DB_KEY 9, 43 — COL-NR 0 — LIST REC — DISPL 3337
(1) (00000000) 00090000 0000002B 05000012 00090000 0000002B 05000012 00090000 0000002B
(33) (00000020) 05000012 00090000 0000002B 05000012 00090000 0000002B 05000012 00090000
(65) (00000040) 0000002B 05000012 00090000 0000002B 05000012 00000000 00090500 0011FF00
(97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
(129) (00000080) F8F3F1F2 F1F3F1F1 C6D3D6E6 C9D5C740 D1C5D9E2 C5E840C4 D9C5E2E2 40E6C9E3 83121311FLOWING JERSEY DRESS WIT
(161) (000000A0) C840D1C1 C3D2C5E3 40404040 40404040 F2F3F6F1 F1F1F4F0 0020700C 0023900C H JACKET 23611140
(193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00

.
.
.

[page header

[page index entry

[Set Connection Data (SCD)

[data record

PAGE INFO (page header)

All pages, except for DBTT pages and FPA pages, contain the page header in the first 20 bytes. BPRECORD prints the following PAGE INFO:

TYPE Type of page

0: Data page or CALC page

1: Act-Key-0 page

FREE SPACE

Free storage space in the page:

SIZE

Length of the free storage space (in bytes)

DISPL

Displacement relative to the first unused byte

NO OF PAGE INDICES

Number of page index entries

DISPL TO END OF PAGE

Length of the page (in bytes)

PAGE INDEX (page index entry)

Page index entries are used to locate a record or table within the page. They occur in data pages and direct CALC pages. BPRECORD prints:

DB-KEY

Database key of a record in the form: *refref*, *rsq*

COL-NR

Column number in the DBTT

=0: Data record (LOGICAL RECORD or CALC KEY REC)

>0: Table record (TABLE REC)

DISPL Displacement relative to the data record

SCD (Set connection data)

The set connection data is explained in the [“Design and Definition”](#) manual.

Print act-key-0 page (DISPLAY PAGE)

```
DISPLAY [IN CSV [csv-filename]] PAGE ZERO;
```

The DISPLAY-PAGE-ZERO statement prints the first 108 bytes of the realm information page (Act-Key-0 page) in normal text. Use of this statement is optional; repeated input for the same realm is ignored by BPRECORD.

IN CSV

BPRECORD also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPRECORD run (e.g. DISPLAY IN CSV 'BPRECORD.CSV' ...).

For a detailed description of CSV format output, see the manual [“Database Operation”](#), section “Outputting database information in a neutral format”.

Example

```
DISPLAY PAGE ZERO
```

```
***** ACTUAL-KEY=0 OF REALM < :SQL2:$XXXXXXX.SHIPPING.ARTICLE-RLM > *****
PHYS REC LENGTH :      4000
DATASET INFO :
TOTAL NR PAGES:          2062
FPA BASE   BEGIN:      AREA REF:      11
              BNR:          1
              NR PAGES IN FPA BASE:    1990
FPA EXTENTS
              NR          BNR
              1          47
              MAX NR PAGES IN EXTENT:  63680
HIGHEST PAGE NR FOR FORMATTING:      2061
CREATE DATA:  DATE :      20150628   TIME:      095430
BACK UP DATA:  DATE :      20150628   TIME:      095509
REALM VERSION NR:      3
SYSTEM BREAK:
OCCURRED:      0
ADMIN USERID:  $XXXXXXX
CONFNAME:      BREORG
FILE NAME:      :SQL2:$XXXXXXX.SHIPPING.ARTICLE-RLM
REALM LAYOUT VERSION:  004.00
UDS VERSION:      V2.8
INCR NR PAGES:      64
INCR MIN PAGES:      0
```

PHYS REC LENGTH

Page length (in bytes)

DATASET INFO

General information on the realm:

FPA BASE

Information on the FPA base

AREA REF

Number of the realm that this free place administration applies to in the FPA base.

BNR

Page number of the first FPA page

TOTAL NR PAGES

Total number of pages administered in the realm

NR PAGES FPA BASE

Number of pages past the FPA base administered in the realm

FPA EXTENTS

Information on all FPA extents of the realm

NR

Number of FPA extent

BNR

Number of the first page of the FPA extent

MAX NR PAGES IN EXTENT

Maximum number of data pages administered in a FPA extent

HIGHEST PAGE NR FOR FORMATTING

Number of the database page up to which a format applies when extending a realm

nnnnnnnn

Number of the database page up to which a format applies when extending a realm.

If the value is the same as the value of TOTAL NR PAGES, then any new pages added to a realm during a realm extension are not formatted.

If the value is greater than the value of TOTAL NR PAGES, then all new pages up to the specified number that are added to a realm during a realm extension are formatted.

In the case of 2-Kbyte databases, further conditions must be fulfilled for reasons of safety thus making it unnecessary to format newly added pages.

UNKNOWN

All pages added to a realm during a realm extension are formatted.

CREATE DATA

Creation date for the realm:

DATE

Creation date in the format: yyyyymmdd

TIME

Creation time in the format: hhmmss

BACK UP DATA

Time last update occurred see CREATE DATA for format)

DATE

Date of last update

TIME

Time of last updates

REALM VERSION NR

Internal version number of the realm; only changed by utility routines intended for updates (e.g. BALTER, BCHANGE, BOUTLOAD, BREORG)

SYSTEM BREAK

Indicates whether or not the realm was closed correctly

OCCURRED

0: Last session terminated normally

1: Last session aborted, i.e. the realm may be inconsistent

ADMIN USERID

ID under which the database is maintained.

CONFNAME

Name of configuration or utility routine with which the database was last accessed

FILE NAME

Complete file name of the realm

DATABASE LAYOUT VERSION

Version number of present database layout version in the format nnn.nn
(output for the DBDIR)

REALM LAYOUT VERSION

Version number of the present realm layout structure in the format nnn.nn

UDS VERSION

UDS/SQL version number under which the database was last modified by DBH.
The field is used to ensure that warm starts are performed correctly.

INCR

Information on the online extensibility of the realm. This information is output for the DBDIR and the user realms.

The information is only output for realms for which the online extensibility was activated with the DAL command ACT INCR.

The information is also output when the online extensibility is deactivated.

NR PAGES

Number of pages added to the realm during an online extension.

MIN PAGES

Minimum number of free pages. If the number of free pages drops below this value, then an online extension is triggered by the DBH for the corresponding realm.

List FPA entries (DISPLAY FPA)

`DISPLAY [IN CSV [csv-filename]] FPA OF page-selection;`

IN CSV

BPRECORD also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPRECORD run (e.g. DISPLAY IN CSV 'BPRECORD.CSV' ...).

For a detailed description of CSV format output, see the manual "[Database Operation](#)", section "Outputting database information in a neutral format".

page-selection

See "[Physical selection \(page selection\)](#)" on page 236

The DISPLAY FPA statement prints the FPA entries of a realm as a whole or in part. Use of this statement is optional. If you specify several FPA statements, BPRECORD combines the statements internally to a single input.

Example

DISPLAY FPA OF ...

```
***** F.P.A. ENTRIES OF REALM < :SQL2:$XXXXXXXXX.SHIPPING.ARTICLE-RLM
-----
```

	REALM	REF	11	BNR	1	ACT	KEY	X'0B000001'		
0.....29:	0	0	0	0	0	0	0	0	0	0
30.....39:	0	0	3960	3964	0	0	3956	0	0	0
40.....49:	0	0	0	0	0	0	2800	0	0	0
50.....69:	0	0	0	0	0	0	0	0	0	0
70.....79:	0	0	0	0	0	0	0	0	0	3980
80.....1989:	3980	3980	3980	3980	3980	3980	3980	3980	3980	3980
			REALM	REF	11	BNR	47	ACT	KEY	X'0B00002F'
1990.....2059:	3980	3980	3980	3980	3980	3980	3980	3980	3980	3980
2060.....2061:	3980	0								



page numbers



free bytes per page
 0 page full
 1 ... 2027 page partly filled
 2028 page empty

Invalid FPA values are marked XX.

List DBTT entries (DISPLAY DBTT)

```

DISPLAY [IN CSV [csv-filename]] DBTT OF
      { ALL RECORDS
      { RECORD record-name FOR rsq-selection }
      };

```

IN CSV

BPRECORD also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPRECORD run (e.g. DISPLAY IN CSV 'BPRECORD.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

ALL-RECORDS

Prints the DBTTs of all record types contained in the specified realm.

record-name

Name of a record type whose DBTT (or DBTT entries) is/are to be printed.

rsq-selection

See “[Logical selection \(RSQ selection\)](#)” on page 236.

The DISPLAY DBTT statement is optional and can be repeated. Repeated input of the same DISPLAY DBTT statement is ignored by BPRECORD.



If multiple page numbers are specified in the DISPLAY DBTT statement, they must be specified in ascending order.

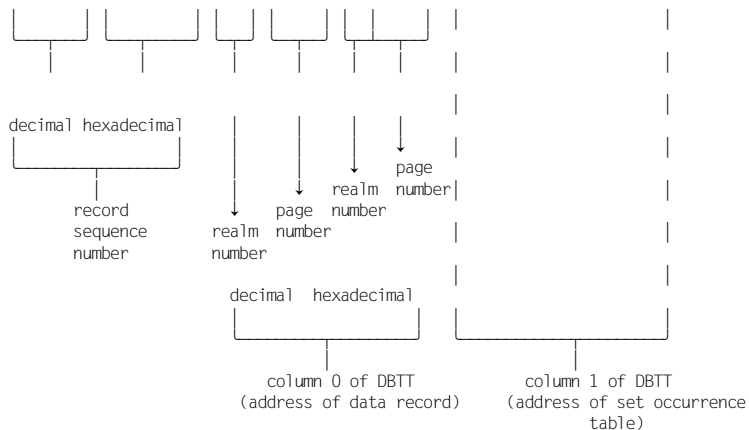
DBTT entries can be listed in decimal and/or hexadecimal format (see the PRINT statement).

Example

DISPLAY DBTT OF RECORD.ARTICLE-TYPE FOR RSQ 1 TO 11 (with PRINT DBTT BOTH)

```
***** D.B.T.T. ENTRIES OF REALM < :SQL2:$XXXXXXX.SHIPPING.ARTICLE-RLM
===== RECORD REF      6, NAME: < ART-TYPE                               > =====
----- REALM REF  11  BNR      3  ACT KEY  X'0B000003' -----

RSQ      1/X'00000001': (  5,      12/X'0500000C') ( 11,      37/X'0B000025')
RSQ      2/X'00000002': (  8,      4/X'08000004') ( 11,      39/X'0B000027')
RSQ      3/X'00000003': (  8,      5/X'08000005') ( 11,      41/X'0B000029')
RSQ      4/X'00000004': (  8,      6/X'08000006') ( 11,      44/X'0B00002C')
RSQ      5/X'00000005'- 11/X'0000000B': (  0,      0/X'00000000') (  0,      0/X'00000000')
```



Print CALC pages (DISPLAY CALC)

`DISPLAY [IN CSV [csv-filename]] CALC PAGES page-selection`

$$\left. \begin{array}{l} \text{ALL} \left[\begin{array}{l} \text{RECORDS} \\ \text{CALC SEARCHKEYS} \end{array} \right] \\ \text{ONLY} \left[\begin{array}{l} \text{RECORD } \textit{record-name} \\ \text{CALC SEARCHKEY } \textit{keyref} \end{array} \right] \textit{rsq-selection} \end{array} \right\} ;$$

IN CSV

BPRECORD also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPRECORD run (e.g. DISPLAY IN CSV 'BPRECORD.CSV' ...).

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

page-selection

See “[Physical selection \(page selection\)](#)” on page 236

Overflow pages lying outside of the page selection are also output with the primary CALC pages.

ALL Print all realm CALC pages defined by *page-selection*

ALL RECORDS

Print the CALC pages (defined by *page-selection*) of all record types defined with LOCATION CALC

ALL CALC SEARCHKEYS

Print the CALC pages (defined by *page-selection*) of all CALC SEARCH keys

ONLY RECORD *record-name*

Print all records or specific records (*rsq-selection*) from the CALC pages defined by *page-selection*

record-name

Name of a record type defined with LOCATION CALC

ONLY CALC SEARCH KEY *keyref*

Print all CALC index entries or part (*rsq-selection*) of the CALC index entries from the CALC pages (defined by *page-selection*) of the CALC SEARCH key

keyref

Key number of a CALC SEARCH key

(see "CALC-SEARCH-KEY INFORMATION" on page 158)

rsq-selection

See "Logical selection (RSQ selection)" on page 236.

Example 1

DISPLAY CALC PAGES PAGE 16 ALL CALC SEARCHKEYS

```
***** CALC KEY BUCKETS OF REALM < :SQL2:$XXXXXXXX.SHIPPING.ARTICLE-RLM
===== CALC SEARCH KEY, KEY REF 14, SET NAME < SYS_ARTICLE > =====
----- REALM REF 11 BNR 16 ----- ACT KEY X'0B000010' -----

PAGE INFO:  TYPE 0 — FREE SPACE  SIZE 10,  DISPL 30 — NR OF PAGE INDICES  0 — DISPL TO CALC TABLE HEADER  30

CALC KEY TABLE:  MAX ENTRIES  220 — ACT ENTRIES  25 — OVERFLOW BUCKET NEXT  0,  PRIOR  0

-1-
(  1) (00000000) F2F3F0F1 F0F7F3F6                23010736
      RSQ      1 — PPP  5,      13

-2-
(  1) (00000000) F2F3F0F1 F0F7F4F2                23010742
      RSQ      4 — PPP  5,      13

-3-
(  1) (00000000) F2F3F0F1 F0F7F4F8                23010748
      RSQ      7 — PPP  5,      13

-4-
(  1) (00000000) F2F3F2F1 F0F7F3F8                23210738
      RSQ      8 — PPP  5,      14

-5-
(  1) (00000000) F2F3F2F1 F0F7F4F0                23210740
      RSQ      9 — PPP  5,      14

-6-
(  1) (00000000) F2F3F2F1 F0F7F4F6                23210746
      RSQ     12 — PPP  5,      14

-7-
(  1) (00000000) F2F3F3F1 F0F8F3F8                23310838
      RSQ     21 — PPP  5,      15
```

Example 2

DISPLAY CALC PAGES ALL PAGES ONLY RECORD ART-DESCR ALL RSQS
 (record type defined with LOCATION CALC)

```
***** SCHEMA NAME < MAIL-ORDERS > *****
***** CALC KEY BUCKETS OF REALM < :SQL2:$XXXXXXX.SHIPPING.CLOTHING > *****
===== CALC RECORD, REC REF 8, RECORD NAME < ART-DESCR > =====
----- REALM REF 5 BNR 2 ----- ACT KEY X'05000002' -----
```

PAGE INFO: TYPE 0 — FREE SPACE SIZE 20, DISPL 40 — NR OF PAGE INDICES 0 — DISPL TO CALC TABLE HEADER 40

CALC KEY TABLE: MAX ENTRIES 79 — ACT ENTRIES 2 — OVERFLOW BUCKET NEXT 0, PRIOR 0

-1-

```
( 1) (00000000) C6D3D6E6 C9D5C740 D1C5D9E2 C5E840C4 D9C5E2E2 40404040 40404040 40404040 FLOWING JERSEY DRESS
( 33) (00000020) 40404040 40404040
      RSQ      8 — PPP 5, 16
```

-2-

```
( 1) (00000000) D7D6D3D6 40C4D9C5 E2E24040 40404040 40404040 40404040 40404040 POLO DRESS
( 33) (00000020) 40404040 40404040
      RSQ     11 — PPP 5, 19
```

----- REALM REF 5 BNR 3 ----- ACT KEY X'05000003' -----

PAGE INFO: TYPE 0 — FREE SPACE SIZE 20, DISPL 40 — NR OF PAGE INDICES 0 — DISPL TO CALC TABLE HEADER 40

CALC KEY TABLE: MAX ENTRIES 79 — ACT ENTRIES 4 — OVERFLOW BUCKET NEXT 0, PRIOR 0

-1-

```
( 1) (00000000) D1C5D9E2 C5E840C3 D9C5D7C5 40C4D9C5 E2E24040 40404040 40404040 40404040 JERSEY CREPE DRESS
( 33) (00000020) 40404040 40404040
      RSQ      7 — PPP 5, 15
```

-2-

```
( 1) (00000000) D7D3C5C1 E3C5C440 C4D9C5E2 E240E6C9 E3C840D1 C1C3D2C5 E3404040 40404040 PLEATED DRESS WITH JACKET
( 33) (00000020) 40404040 40404040
      RSQ      6 — PPP 5, 14
```

----- REALM REF 5 BNR 4 ----- ACT KEY X'05000004' -----

PAGE INFO: TYPE 0 — FREE SPACE SIZE 20, DISPL 40 — NR OF PAGE INDICES 0 — DISPL TO CALC TABLE HEADER 40

CALC KEY TABLE: MAX ENTRIES 79 — ACT ENTRIES 2 — OVERFLOW BUCKET NEXT 0, PRIOR 0

-1-

```
( 1) (00000000) C6D3D6E6 C9D5C740 D1C5D9E2 C5E840C4 D9C5E2E2 40E6C9E3 C840D1C1 C3D2C5E3 FLOWING JERSEY DRESS WITH JACKET
( 33) (00000020) 40404040 40404040
      RSQ      9 — PPP 5, 17
```

-2-

```
( 1) (00000000) E360E2C8 C9D9E340 C4D9C5E2 E2404040 40404040 40404040 40404040 T-SHIRT DRESS
( 33) (00000020) 40404040 40404040
      RSQ     10 — PPP 5, 18
```

CALC RECORD

Hash area of a record type defined with LOCATION CALC

REC REF

Number of the record type

RECORD NAME

Name of the record type

CALC SEARCH KEY

Hash area of a SEARCH key defined with USING CALC

KEY REF

Number of the key

SET NAME

Name of the set to which this key belongs

PAGE INFO

Page header (see PRINT statement);
in CALC pages, DISPL TO END OF PAGE is replaced by:

DISPL TO CALC TABLE HEADER

Displacement to the CALC key table header

LOGICAL RECORDS

Print out of records with the following options:

PAGE INDEX

Page index entry (see PRINT statement)

SCD

Set connection data (see PRINT statement)

CALC KEY TABLE

BPRECORD prints the following information from the CALC key table header:

MAX ENTRIES

Maximum number of entries possible

ACT ENTRIES

Actual number of current entries

OVERFLOW BUCKET

Linkage to overflow pages:

NEXT n

Page number of the next overflow page

0: No overflow page exists

PRIOR m

Page number of the preceding page

0: Primary page

Print data pages (DISPLAY DATA)

`DISPLAY` [`IN CSV` [`csv-filename`]] `DATA` `PAGES` `page-selection`

$$\left. \begin{array}{l} \text{ALL} \left[\begin{array}{l} \text{RECORDS} \\ \text{TABLES} \end{array} \right] \\ \\ \text{ONLY} \left\{ \begin{array}{l} \text{RECORD } \textit{record-name} \\ \text{TABLES OF } \left\{ \begin{array}{l} \text{OWNER } \textit{record-name} \\ \text{SET } \textit{set-name} \\ \text{KEY } \textit{keyref} \end{array} \right\} \textit{rsq-selection} \end{array} \right\} \end{array} \right\};$$

IN CSV

BPRECORD also outputs the data in CSV format.

csv-filename

Name of the file to which the data is to be output in CSV format. The specification of *csv-filename* is mandatory in the first IN CSV statement of a BPRECORD run (e.g. DISPLAY IN CSV 'BPRECORD.CSV' ...).



When national items exist, the output contains data in UTF-16 format.

For a detailed description of CSV format output, see the manual “[Database Operation](#)”, section “Outputting database information in a neutral format”.

page-selection

See “[Physical selection \(page selection\)](#)” on page 236.

Overflow pages lying outside of the page selection are also output with the primary table pages.

ALL Print all realm data pages defined by *page-selection*

ALL RECORDS

Print all record types from the data pages defined by *page-selection*

ALL TABLES

Print all tables from the data pages defined by *page-selection*

ONLY RECORD *record-name*

From the data pages defined by *page-selection*, print the records or certain record (*rsq-selection*) of the record type *record-name*

record-name

Name of a record type which has not been defined with LOCATION MODE CALC

ONLY TABLES OF

From the data pages defined by *page-selection*, print out the tables or certain table entries (*rsq-selection*)

OWNER *record-name*

Of the owner record type *record-name*

SET *set-name*

Of the set *set-name*

KEY *keyref*

Of the key with the number *keyref* (see [page 155](#)).

rsq-selection

See "Logical selection (RSQ selection)" on [page 236](#). An RSQ selection is of no use for SYSTEM sets and is ignored if present.

Example 1

DISPLAY DATA PAGES PAGE 19 ALL RECORDS
(only data records)

```
***** DATA / TABLE PAGES OF REALM < :SQL2:$XXXXXXXX.SHIPPING.CLOTHING > *****
----- REALM REF 5 BNR 19 ----- ACT KEY X'05000013' -----
```

PAGE INFO: TYPE 0 — FREE SPACE SIZE 2081, DISPL 2209 — NR OF PAGE INDICES 9 — DISPL TO END OF PAGE 4000

LOGICAL RECORDS:

```
-2- PAGE INDEX: DB_KEY 9, 48 — COL-NR 0 — LIST REC — DISPL 3767
( 1) (00000000) 00090000 00000030 05000013 00090000 00000030 05000013 00090000 00000030
( 33) (00000020) 05000013 00090000 00000030 05000013 00090000 00000030 05000013 00090000
( 65) (00000040) 00000030 05000013 00090000 00000030 05000013 00000000 000A0500 0012FF00
( 97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
( 129) (00000080) F8F3F5F9 F2F8F0F9 E360E2C8 C9D9E340 C4D9C5E2 E2404040 40404040 40404040 83592809T-SHIRT DRESS
( 161) (000000A0) 40404040 40404040 40404040 40404040 F2F3F7F1 F0F2F3F6 0020700C 0023900C 23710236
( 193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00
```

```
-3- PAGE INDEX: DB_KEY 9, 49 — COL-NR 0 — LIST REC — DISPL 3552
( 1) (00000000) 00090000 00000031 05000013 00090000 00000031 05000013 00090000 00000031
( 33) (00000020) 05000013 00090000 00000031 05000013 00090000 00000031 05000013 00090000
( 65) (00000040) 00000031 05000013 00090000 00000031 05000013 00000000 000A0500 0012FF00
( 97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
( 129) (00000080) F8F3F5F9 F2F8F0F9 E360E2C8 C9D9E340 C4D9C5E2 E2404040 40404040 40404040 83592809T-SHIRT DRESS
( 161) (000000A0) 40404040 40404040 40404040 40404040 F2F3F7F1 F0F2F3F8 0020700C 0023900C 23710238
( 193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00
```

```
-4- PAGE INDEX: DB_KEY 9, 50 — COL-NR 0 — LIST REC — DISPL 3337
( 1) (00000000) 00090000 00000032 05000013 00090000 00000032 05000013 00090000 00000032
( 33) (00000020) 05000013 00090000 00000032 05000013 00090000 00000032 05000013 00090000
( 65) (00000040) 00000032 05000013 00090000 00000032 05000013 00000000 000A0500 0012FF00
( 97) (00000060) 00000000 00000000 00000000 00000000 00000000 00000000 00000001 0400000E
( 129) (00000080) F8F3F5F9 F2F8F0F9 E360E2C8 C9D9E340 C4D9C5E2 E2404040 40404040 40404040 83592809T-SHIRT DRESS
( 161) (000000A0) 40404040 40404040 40404040 40404040 F2F3F7F1 F0F2F4F0 0020700C 0023900C 23710240
( 193) (000000C0) 00000500 000C050C 00000499 950C0000 00000000 050C00
```

For descriptions of PAGE INFO, PAGE INDEX and SCD, see [page 241](#).

Example 2

DISPLAY DATA PAGES ALL PAGES ALL TABLES
(only tables)

```
***** DATA / TABLE PAGES OF REALM < :SQL2:$XXXXXXXXX.SHIPPING.ARTICLE-RLM > *****
----- REALM REF 11 BNR 31 ----- ACT KEY X'0B00001F' -----

PAGE INFO: TYPE 0 --- FREE SPACE SIZE 0, DISPL 32 --- NR OF PAGE INDICES 1 --- DISPL TO END OF PAGE 4000

LOGICAL RECORDS:
-1- PAGE INDEX: DB_KEY 0, 5 --- COL-NR 1 --- TABLE REC --- DISPL 32
TABLE: MAX ENTRIES 112 --- TABLE NEXT 0 --- TABLE DESCR X'40' --- NEXT HIGHER LEVEL 0
ACT ENTRIES 5 --- PRIOR 0 --- LEVEL NR 0 --- LAST ENTRY 31
-1-
( 1) (00000000) RSQ 2 --- PPP 8, 4
C2C1E5C1 D9C9C1D5 40C2C5C5 D9404040 40404040 40404040 40 BAVARIAN BEER
-2-
( 1) (00000000) RSQ 4 --- PPP 5, 18
C3D6D5E3 C5D4D7D6 D9C1D9E8 40C3D3D6 E3C8C9D5 C7404040 40 CONTEMPORARY CLOTHING
-3-
( 1) (00000000) RSQ 1 --- PPP 5, 12
C5D3C5C7 C1D5E340 C3D3D6E3 C8C9D5C7 40404040 40404040 40 ELEGANT CLOTHING
-4-
( 1) (00000000) RSQ 3 --- PPP 8, 5
D3C5D4D6 D5C1C4C5 40404040 40404040 40404040 40404040 40 LEMONADE
-5-
( 1) (00000000) RSQ 5 --- PPP 8, 6
E8D6C7C8 E4D9E340 40404040 40404040 40404040 40404040 40 YOGHURT
```

[Table header

[Associated table entries

TABLE

From the header of the table, BPRECORD prints the following information:

MAX ENTRIES

Self-explanatory

ACT ENTRIES

Number of current entries

TABLE

Linkage of table pages

NEXT n

Page number of the next table page

PRIOR m

Page number of the preceding table page;

0: No next or prior table page exists

TABLE DESCR

Description of the table

Bit 2^7 = 1: List

Bit 2^6 = 1: Multi-level table

Bit 2^5 = 1: Table ATTACHED TO OWNER

Bit 2^4 = 1: Duplicates table

Bit 2^3 = 1: Table in ACTKEY format

Bit 2^2 = 1: Table in ACTKEY format with chaining to the last page

LEVEL NR

Level of the table

NEXT HIGHER LEVEL

Page number of the page of the next higher level

LAST ENTRY

Page number of the last page of the basic level



In the case of table headers in ACTKEY format the chainings NEXT, PRIOR, NEXT HIGHER LEVEL and LAST ENTRY are displayed in eight-digit hexadecimal format. Non-existent chainings (e.g. LAST ENTRY in pages of the main level and PRIOR in pages of the highest level) remain free in the display. In the case of table headers in BNR format the chainings specify the block numbers concerned in the same realm in decimal format (cf. structure of the tables in the “Design and Definition” manual).

Terminate BPRECORD (END)

END

7.4 Command sequence to start BPRECORD

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

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

- 01 In this case specifying *:catid:* is permitted (see the “[Database Operation](#)” manual).
- 02 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.
- 03 The UDS/SQL utility routine can also be started with the alias BPRECORD.

8 Executing online services with the UDS online utility

The UDS online utility offers the option of online execution (i.e. while the DBH is operating) of some functions for compressing databases, some of which are otherwise implemented by utility routines with comparable functions.

You can compress databases offline as follows:

- using BPGSIZE (see the [“Creation and Restructuring”](#) manual) or
- by unloading and reloading the data using BOUTLOAD/BINILOAD (see the [“Creation and Restructuring”](#) manual) or
- by creating tables with BREORG (see the [chapter “Database reorganization with BREORG” on page 307](#)).

The UDS online utility enables you to relocate and compress records and tables specifically in various realms and during ongoing operation. Furthermore, whole data pages which contain the records of a distributable list can be relocated from one realm to another during ongoing operation.

When data is compressed with BPGSIZE or BOUTLOAD/BINILOAD, locks of DBTT entries (cf. the KEEP statement of the utility routine BMODTT) are canceled; these locks are retained when compression takes place with the UDS online utility.

With the UDS online utility you can change the free place search in the realm and thus also call the comparable functions SET and RESET of the BMODTT utility routine (see [page 349](#)) and make them effective during live operation.

For distributable lists the UDS online utility enables you to change the preferred realm during ongoing operation, i.e. the realm in which the DBH searches for pages which are currently free for storage purposes.

The UDS online utility enables you to update probable position pointers (PPP) to the most recent values during ongoing operation.

8.1 Functions of the UDS online utility

The UDS online utility enables you to make changes to the physical properties of the data contained in the databases during normal DBH operation. In this case the contents of the user data can be neither seen nor changed in the UDS online utility.

All requested changes are performed by the DBH in transactions which are comparable to normal user transactions and are controlled solely by the UDS online utility. In this respect the UDS online utility behaves like a totally normal user program.

The UDS online utility offers the following functions in the form of new DMLs which modify the database itself or how it is processed:

- With the online DML RELOCATE you relocate records and tables in a single realm. Here the occupancy ratio in the front part of the realm is generally increased. As a result, contiguous free data pages are created in the back part of the realm. Furthermore, you can relieve the load on a realm by relocating pages of a distributable list to another realm. A description of the DML RELOCATE is provided in the [section “DML RELOCATE - Relocating data pages” on page 262](#).
- With the online DML FPASCAN you change the settings for the free place search in the realm during live operation. A description of the DML FPASCAN is provided in the [section “DML FPASCAN - Defining the search mode for the free place search” on page 265](#).
- With the online DML PREFRLM you change the setting of the preferred realm for a distributable list. A description of the DML PREFRLM is provided in the [section “DML PREFRLM - Defining the preferred realm for distributable lists” on page 265](#).
- With the online DML REORGPPP you can update PPPs to the most recent values during live operation in a single realm. A description of the DML REORGPPP is provided in the [section “DML REORGPPP – Reorganizing probable position pointers \(PPPs\)” on page 266](#).

In a UDS online utility procedure the RELOCATE, FPASCAN and PREFRLM functions always refer to a database which is identified unambiguously by the subschema and to a source realm. If multiple functions (RELOCATE, FPASCAN and PREFRLM) are to be executed in a UDS online utility procedure, the subschema name and realm name must be identical in each case in all these functions.

In order to make the requested changes in the transactions of the UDS online utility, resources must be used in the DBH in contention with the other application programs which are running in parallel (locks, I/O buffer, etc.). In order to avoid any negative repercussions on these applications, you should use the UDS online utility at times when the normal load and changes in the realm being processed by the UDS online utility tend to be low.

The statements of the UDS online utility enable you to control the scope of the changes which are to be performed in a transaction, thus permitting you to guarantee that application programs which execute in parallel are hindered very little. You can also use specific wait statuses to control the order of the various transactions of the UDS online utility to ensure that hindrances for applications which execute in parallel are reduced still further.

Overall it is not an aim to make all the changes requested with the UDS online utility in as short a time as possible. Rather, the aim is to deal with these in the smallest possible units and with as few repercussions as possible on applications which execute in parallel.

The various requests of the UDS online utility to the DBH supply information on the changes which have been made. You can use this information in turn for further control of the UDS online utility.

Multiple instances of the UDS online utility which make changes in different realms can be active at any given time.

Please note that the actions of the UDS online utility may result in the volume of the ALOG data becoming considerably greater, and take this into account when you plan the use of the UDS online utility.

8.2 DML RELOCATE - Relocating data pages

You use the online DML RELOCATE to relocate data records and tables specifically within a realm during live operation. As far as possible, relocation takes place to free space at the beginning of the realm. Free space on pages which are not completely full is also utilized. As a result of relocation free space occurs in the back area of the realm. A RELOCATE DML always refers to a user realm which, like the corresponding DBTT realms, too, must be attached in update mode in the DBH.

Depending on the variant of RELOCATE, records and tables are relocated which contain pages which are not completely filled or whole data pages containing parts of multipage tables.

The following are not relocated:

- Completely filled pages if they contain FPA entries or DBTT entries
- DBTT anchor pages
- CALC pages
- Anchor records of SYSTEM sets (anchor records contain information which is comparable to DBTT pages).

Relocation procedure

Relocation is implemented as follows:

A suitable data page is searched for from the end of the realm. For the records and tables of such a page or for the whole page free space into which the records and tables can be relocated individually is searched for from the beginning of the realm.

Clustered data which is flagged with corresponding SSL declarations is, as far as possible, also stored again on one and the same page (retention of clusters).

If pages of a distributable list are to be relocated to another realm, the free pages required are searched for in the target realm (TARGET-REALM).

You specify the number of pages which are as far as possible to be emptied by relocation using a RELOCATE DML by means of the statements of the UDS online utility (PAGES-PER-DML parameter of the SDF statement SET-RELOCATE-PARAMETERS, see [section "Defining the properties of a RELOCATE DML \(SET-RELOCATE-PARAMETERS\)" on page 278](#)).

If conflicts with other transactions occur when the relocation takes place, you must reckon on the RELOCATE DML aborting prematurely in order to reduce the hindrance for parallel application to a minimum. Relocations which have already been executed in the transaction are generally retained in the new status.

Relocation progress data

In each session section the DBH maintains internal progress data concerning the pages at the end of the source realm which have already been emptied by means of relocation (source level) and concerning the pages which have already been filled at the beginning of the target realm (target level), thus enabling relocation to be continued in multiple consecutive online utility runs without any great effort.

The source and target levels are provided as variables (variable RELOC_ORIGIN for source level and variable RELOC_DESTINATION for target level) in order to inform the UDS online utility and contain the block numbers of the relevant database pages at which there will be continued in the event of further relocations.

Initializing the source and target levels

In general the first relocation transaction of a sequence of relocation transactions results in the initialization of the source level on the last page which is not empty at the end of the realm and of the target level on the first page at the beginning of the realm - in other words initially ACTKEY0. After a successful RELOCATE DML has been processed, the position is set to the pages on which a subsequent RELOCATE DML continues its work. These values are provided as the source and target levels in the UDS online utility.

The progress data for a relocation is not stored in the database for later session sections. It is lost when the realm concerned is detached. After the realm has been attached, progress data from previous session sections is therefore no longer available. The next relocation transaction can only be performed if the source and target levels are re-initialized.

After the UDS online utility has been restarted, it is possible to continue a relocation specifically in a session section which has not been interrupted. In this case you use the INITIALIZE parameter of the SDF statement SET-RELOCATE-PARAMETERS to control initialization of the source and target levels:

- INITIALIZE=* ANY
As the UDS online utility may not have recognized that the progress data has been lost, INITIALIZE=* ANY enables the relocation transaction to be executed in such a manner that any progress data which still exists can still be used, otherwise initialization takes place.
- INITIALIZE=*NO
INITIALIZE=*NO enables initialization to be explicitly prevented if the realm was detached and then re-attached after the last relocation transaction. It cannot be excluded that changes which mean that it no longer makes sense to continue relocation were made to the realm between the times when it was detached and then re-attached.

– INITIALIZE=*YES

It is, however, also possible to use INITIALIZE=*YES to start initialization of the relocation again. In this case the levels are initialized as described above. This initialization becomes effective in the first corresponding transaction. All subsequent transactions of a UDS online utility are executed with INITIALIZE=*ANY. Whether these transactions are executed in one or more consecutive procedure sequences (SDF statement REPEAT-PROCEDURE) is irrelevant.

Initialization of source and target levels also implicitly leads to the DML FPASCAN being executed with *REUSE. But it does no harm to call FPASCAN explicitly. The DML FPASCAN places the page from which the records and tables which are to be stored will be searched for at the beginning of the realm or on the target level so that parallel storage operations caused by the application program through the DMLs MODIFY and STORE are not stored behind the source level of the relocation. A description of FPASCAN is provided in the [section "DML FPASCAN - Defining the search mode for the free place search" on page 265](#)).

End of relocation

A sequence of relocation transactions where the source realm is identical to the target realm is ended internally when the source level is lower than the target level. This is reported to the UDS online utility and results in no further relocation transactions being executed in this realm as these have nothing more to do and repeatedly report the same information on the source and target levels.

Filling level to which pages are to be relocated

When relocation takes place, whether the data contained on the page can be relocated can generally only be decided after the data page has been read. The SKIP-ABOVE-FILLING parameter of the SDF statement SET-RELOCATE-PARAMETERS enables you to exclude pages which exceed a particular percentage filling level from relocation (see [section "Defining the properties of a RELOCATE DML \(SET-RELOCATE-PARAMETERS\)" on page 278](#)). The default value 100 includes all relocatable pages. A lower value allows you, for instance, to restrict relocation to pages which are not very full. The DBH also uses a specification of under 100 to exclude pages from relocation on the basis of the free place administration data. This optimization consequently enables you to prevent unnecessary reading of data pages.

Behavior in the event of locked source pages

The CLASH-HANDLING parameter of the SDF statement SET-RELOCATE-PARAMETERS permits you to control how the DBH should behave when it encounters a locked source page (see [section “Defining the properties of a RELOCATE DML \(SET-RELOCATE-PARAMETERS\)” on page 278](#)). With the default value BREAK-DML the DML is aborted and the relocations which have already taken place are retained. The controller in the UDS online utility can react to this situation and repeat the next relocation request after a particular wait time. However, SKIP-PAGE also provides the option of skipping a locked page. No option to wait while the individual RELOCATE DML is processed is offered because this could result in the risk of deadlocks which could have a negative influence on parallel applications.

8.3 DML FPASCAN - Defining the search mode for the free place search

You use the online DML FPASCAN and the SEARCH-MODE parameter of the SDF statement SET-FPA-SCAN-PARAMETERS to define the search mode for the free place search, i.e. the page in a realm from which space is searched for to store new records and tables which are caused by application programs through the DMLs MODIFY and STORE:

- SEARCH-MODE=*REUSE specifies the start page for the free place search at the beginning of the realm (“Second Scan”). This offers an option of using free space (created, for example, by deleting records or tables) again immediately for new storage operations.
- In the case of SEARCH-MODE=*NOREUSE the DBH switches to “First Scan” when the next checkpoint is reached. In this search mode space for the new data which is to be stored is searched for at the end of the realm, i.e. from the first page of the contiguous range of free pages at the end of the realm.

8.4 DML PREFRLM - Defining the preferred realm for distributable lists

You use the online DML PREFRLM to define the preferred realm for a distributable list, i.e. the realm in which the DBH is currently searching for free pages for the purpose of storing. You define the SET-NAME parameter (which must name a distributable list) and PREFERRED-REALM-NAME parameter required for this purpose up front using the SDF statement SET-PREF-REALM-PARAMETERS.

8.5 DML REORGPPP – Reorganizing probable position pointers (PPPs)

You use the online DML REORGPPP to reorganize PPPs in data records and tables specifically within a realm during live operation.

A REORGPPP DML always refers to a user realm which, like the corresponding DBTT realms, too, must be attached in update mode in the DBH.

The following pages are skipped during PPP reorganization:

- FPA pages and DBTT pages
- DBTT anchor pages
- Free pages

PPP reorganization procedure

Each page in the specified user realm (except skipped pages, see above) is read sequentially and all PPPs are updated in this page.

To update PPPs, specify the number of pages that have to be processed as far as possible using a REORGPPP DML by means of the statements of the UDS online utility (PAGES-PER-DML parameter of the SDF statement SET-REORGANIZE-PPP-PARAMETERS, see [section “Defining the properties of a REORGPPP DML \(SET-REORGANIZE-PPP-PARAMETERS\)” on page 283](#).



If conflicts with other transactions occur during PPP reorganization, the online DML REORGPPP aborts prematurely in order to reduce the hindrance for parallel application to a minimum. PPP that have already been reorganized in the transaction are retained updated.

PPP reorganization progress data

In each session section the DBH maintains internal progress data concerning the page number in the specified realm where PPPs have been already reorganized, thus enabling reorganization to be continued in multiple consecutive online utility runs without any great effort.

The current page number is provided as the REORG-PPP-CURRENT variable in order to inform the UDS online utility and contains the block number of the relevant database page where further PPP reorganization is continued if applicable.

Initializing the PPP reorganization process

The first REORGPPP DML starts with page number 0, i.e. at the beginning of a realm. After a successful REORGPPP DML has been processed, the position is set to the page number where a subsequent REORGPPP DML continues its work. This value is provided as the current page number in the UDS online utility.

The progress data for reorganization is not stored in the database for later session sections. It is lost when the realm concerned is detached. After the realm has been attached, progress data from previous session sections is therefore no longer available. The next PPP reorganization transaction can only be performed if the current page number is re-initialized.

After the UDS online utility has been restarted, it is possible to continue reorganization specifically in a session section which has not been interrupted. In this case, use the INITIALIZE parameter of the SDF statement SET-REORGANIZE-PPP-PARAMETERS to control initialization of the current page number:

- INITIALIZE=* ANY
As the UDS online utility may not have recognized that the progress data has been lost, INITIALIZE=* ANY enables the REORGPPP transaction to be executed using any progress data that still exists. If no progress data is existing, initialization takes place.
- INITIALIZE=*NO
INITIALIZE=*NO enables initialization to be explicitly prevented if the realm was detached and then re-attached after the last REORGPPP transaction.
- INITIALIZE=*YES
It is, however, also possible to use INITIALIZE=*YES to start initialization of the reorganization process again. In this case the initial page number is initialized. This initialization becomes effective in the first corresponding transaction. All subsequent transactions of a UDS online utility are executed with INITIALIZE=*ANY. Whether these transactions are executed in one or more consecutive procedure sequences (SDF statement REPEAT-PROCEDURE) is not relevant.

End of PPP reorganization

A sequence of REORGPPP transactions is ended internally when the current page number reaches the last page number in the realm. This is reported to the UDS online utility. As a result, no further reorganization transactions are executed in this realm.

Behavior in the event of locked pages

The CLASH-HANDLING parameter of the SDF statement SET-REORGANIZE-PPP-PARAMETERS permits you to control how the DBH should behave when it encounters a locked page (see [section “Defining the properties of a REORGPPP DML \(SET-](#)

[REORGANIZE-PPP-PARAMETERS\)](#)” on page 283”).

- With the default value BREAK-DML, the DML is aborted and the PPP reorganizations that have already taken place are retained. The controller in the UDS online utility can react to this situation and repeat the next reorganization request after a particular wait time.
- SKIP-PAGE provides the option of skipping a locked page.

An option to wait while the individual REORGPPP DML is processed is not offered, because this could result in the risk of deadlocks which could have a negative influence on parallel applications.

8.6 Command sequence for starting the UDS online utility

The UDS online utility is started as follows:

```
/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.

8.7 Statements of the UDS online utility

The input interface of the UDS online utility is split into SDF statements and procedure statements.

The SDF statements are executed once and immediately. They control the behavior of the UDS online utility itself and are used to prepare transaction processing.

The procedure statements are started under the user's control. They are used for detailed definition of transaction processing.

8.8 SDF statements of the UDS online utility

The following SDF statements are provided for the UDS online utility:

Statement	Meaning
DECLARE-PROCEDURE	Open procedure declaration
DECLARE-VARIABLE	Define variable
DELETE-PROCEDURE	Delete procedure
DELETE-VARIABLE	Delete variable
END	Terminate UDS online utility
REPEAT-PROCEDURE	Execute procedure
SET-FPA-SCAN-PARAMETERS	Define search mode for free place search
SET-ONLINE-UTILITY-PARAMETERS	Define online utility parameters
SET-PREFERRED-REALM-PARAMETERS	Define preferred realm for a distributable list
SET-RELOCATE-PARAMETERS	Define properties of a RELOCATE DML
SET-REORGANIZE-PPP-PARAMETERS	Define properties of a REORGPPP DML
SHOW-FPA-SCAN-PARAMETERS	Show parameters which are currently valid for DML FPASCAN
SHOW-PREF-REALM-PARAMETERS	Show parameters currently valid for DML PREFRLM
SHOW-PROCEDURE	Show procedure
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

Table 18: SDF statements for the UDS online utility

The statements of the UDS online utility are described below in alphabetical order.

Defining a procedure (DECLARE-PROCEDURE)

The DECLARE-PROCEDURE statement defines a procedure.

The UDS online utility stores a procedure declaration internally for later processing. Faulty procedure statements are only recognized when the procedure executes.

DECLARE-PROCEDURE
PROCEDURE-NAME = <structured-name 1..20> ,CODE = <c-string 1..1800> / <filename> / *SYSDTA

PROCEDURE-NAME = <structured-name 1..20>

Name of the procedure which is to be stored. The procedure name may be up to 20 characters long and must be unique. The names of predefined standard procedures may not be used.

CODE = <c-string 1..1800>

Complete code of the procedure.

CODE = <filename>

Name of the file containing the procedure statements. This file can be either a SAM file or an ISAM file with default parameters (RECFORM=V, KEYLEN=8, KEY- POS=5).

CODE = *SYSDTA

The procedure statements should be read in from SYSDTA.

Defining a variable (DECLARE-VARIABLE)

The DECLARE-VARIABLE statement defines a variable which can be used in procedure statements and conditions.

Name of predefined variables (see [section "Predefined variables" on page 298](#)) may not be used in the DECLARE-VARIABLE statement. Some of the predefined variables are of the type C (e.g. RCODE).

DECLARE-VARIABLE

```
VARIABLE-NAME = <structured-name 1..20>
,TYPE = *STRING(...) / *INTEGER
    *STRING
      | LENGTH = <integer 1..20>
,INITIAL-VALUE = <integer 1..16777215> / <c-string> / *STD / *NONE
```

VARIABLE-NAME = <structured-name 1..20>

Name of the variable. Names of predefined variables (see [section "Predefined variables" on page 298](#)) may not be used.

TYPE =

Type of variable.

TYPE = *STRING(...)

The variable is a text variable.

LENGTH = <integer 1..20>

Length of the text variable. The default value is 5.

TYPE = *INTEGER

The variable is an integer variable.

INITIAL-VALUE =

Value which is to be predefined for the variable.

INITIAL-VALUE = <integer 1..16777215>

The variable should be initialized with an integer.

INITIAL-VALUE = <c-string>

The variable should be initialized with a string.

INITIAL-VALUE = *STD

INITIAL-VALUE *STD assigns blanks to a variable of the *STRING type and 0 to a variable of the INTEGER type.

INITIAL-VALUE = *NONE

In the case of INITIAL-VALUE *NONE the initialization value is undefined.

Deleting a procedure (DELETE-PROCEDURE)

The DELETE-PROCEDURE statement deletes a previously declared and stored procedure. A predefined standard procedure (see [section “Predefined standard procedures” on page 299](#)) cannot be deleted.

DELETE-PROCEDURE
PROCEDURE-NAME = <structured-name 1..20>

PROCEDURE-NAME= <structured-name 1..20>

Name of the procedure which is to be deleted.

Deleting a variable (DELETE-VARIABLE)

The DELETE-VARIABLE statement deletes a previously declared variable. A predefined variable (see [section “Predefined variables” on page 298](#)) cannot be deleted.

DELETE-VARIABLE
VARIABLE-NAME = <structured-name 1..20>

VARIABLE-NAME = <structured-name 1..20>

Name of the variable which is to be deleted.

Terminating the UDS online utility (END)

The END statement terminates the UDS online utility.

END

Executing a procedure (REPEAT-PROCEDURE)

The REPEAT-PROCEDURE statement executes a previously declared and stored procedure or a predefined standard procedure.

In accordance with the DML FPASCAN or RELOCATE which is contained in the procedure, the SET-FPA-SCAN-PARAMETERS or SET-RELOCATE-PARAMETERS statement must have been executed beforehand. The values of the last associated statement are used.

REPEAT-PROCEDURE
<pre>PROCEDURE-NAME = *STDRELOC / *STDFPASCAN / *STDREPPP / <structured-name 1..20> ,CYCLE-LIMIT = <integer 1..16777215> / *MAX</pre>

PROCEDURE-NAME =

Name of the procedure which is to be executed.

PROCEDURE-NAME = *STDRELOC

The standard procedure *STDRELOC (see [section "Predefined standard procedures" on page 299](#)) is to be executed.

PROCEDURE-NAME = *STDFPASCAN

The standard procedure *STDFPASCAN (see [section "Predefined standard procedures" on page 299](#)) is to be executed.

PROCEDURE-NAME = *STDREPPP

The standard procedure *STDREPPP (see [section "Predefined standard procedures" on page 299](#)) is to be executed.

PROCEDURE-NAME = <structured-name 1..20>

Name of the previously declared and stored procedure which is to be executed.

CYCLE-LIMIT =

Specifies the maximum number of procedure cycles.

CYCLE-LIMIT = <integer 1..16777215>

Maximum number of procedure cycles. The default value is 1. If an illegal value (≤ 0) is specified, the default value is used. In the case of a 100% FPASCAN transaction, deviating from the default value generally makes no sense, but the UDS online utility does not check on this.

CYCLE-LIMIT = *MAX

The highest possible value (2^{24-1}) is used for the procedure cycles. This more or less corresponds to an unlimited number of procedure cycles, which means that the possible relocations of a realm can be executed in an online utility run without any interruption.

Defining the search mode for the free place search (SET-FPA-SCAN-PARAMETERS)

The SET-FPA-SCAN-PARAMETERS statement defines which search mode is used for the free place search in a realm (First Scan or Second Scan).

SET-FPA-SCAN-PARAMETERS
SUBSCHEMA-NAME = <structured-name 1..30> ,REALM-NAME = <structured-name 1..30> ,SEARCH-MODE = <u>*REUSE</u> / *NOREUSE

SUBSCHEMA-NAME = <structured-name 1..30>

Name of the subschema in which the search mode is to be defined. Only names of user subschemas are permitted.

REALM-NAME = <structured-name 1..30>

Name of the realm in which the search mode is to be defined. The realm name must be a valid name in the defined subschema. Only names of user realms are permitted. The corresponding check is performed when the procedure statement FPASCAN is executed (see [section "Defining the start page for the free place search \(FPASCAN\)" on page 292](#)).

SEARCH-MODE =

Determines the search mode, i.e. the page in a realm from which space is searched for to store the new records and tables which are created by application programs through the DMLs MODIFY and STORE.

SEARCH-MODE = *REUSE

Space for new records and tables which are to be stored is searched for from the beginning of the realm (Second Scan). A change of the search for space becomes effective immediately with *REUSE.

SEARCH-MODE = *NOREUSE

The search for space for the new records and tables which are to be stored takes place at the end of the realm after the first page of the contiguous area of free pages (First Scan). A change of the search for space is initially registered in *NOREUSE and becomes effective when the next checkpoint is reached.

Defining online utility parameters (SET-ONLINE-UTILITY-PARAMETERS)

SET-ONLINE-UTILITY-PARAMETERS defines whether a connection is to be established to the independent DBH or to the linked-in DBH.

This SDF statement must be the first statement when the UDS online utility is used. It cannot be used again after this.

SET-ONLINE-UTILITY-PARAMETERS
DBH = <u>*INDEPENDENT</u> / *LINKED-IN ,CONFIGURATION-NAME = <structured-name 1..17>

DBH =

Specifies whether a connection is to be established to the independent DBH or to the linked-in DBH.

DBH = *INDEPENDENT

A connection is established to the independent DBH.

DBH = *LINKED-IN

A connection is established to the linked-in DBH. The option of using the linked-in DBH initially contradicts the availability objectives which are actually pursued with the UDS online utility. However, it is certainly conceivable that in concrete applications preference will be given to the high-speed and compact handling of UDS online utility functions, for example because use is to be made of an interruption of the independent session which is caused for other reasons.

CONFIGURATION-NAME = <structured-name 1..17>

Name of the database configuration which is to be used.

The syntax of the configuration name is checked. The connection modules are subsequently loaded dynamically and a connection is set up to the specified session. If this connection fails, the UDS online utility terminates and issues a message to this effect.

Defining the preferred realm (SET-PREF-REALM-PARAMETERS)

SET-PREF-REALM-PARAMETERS sets the parameters for the DML PREFRLM. The preferred realm for a distributable list is set or changed.

SET-PREF-REALM-PARAMETERS
SUBSCHEMA-NAME= <structured-name 1..30> ,SET-NAME= <structured-name 1..30> ,PREFERRED-REALM-NAME= <structured-name 1..30>

SUBSCHEMA-NAME = <structured-name 1..30>

Subschema name. Only names of user subschemas are permitted.

SET-NAME= <structured-name 1..30>

Set whose preferred realm is to be changed. The set must be a valid name in the defined subschema and a distributable list.

PREFERRED-REALM-NAME= <structured-name 1..30>

Name of the new realm which is to be the preferred realm for the SET-NAME set. The realm name must be a valid name in the defined subschema and be specified in the WITHIN clause of the DDL in the declaration of the member set type.

The corresponding checks are performed only when the DMLs of the UDS online utility are executed.

Defining the properties of a RELOCATE DML (SET-RELOCATE-PARAMETERS)

SET-RELOCATE-PARAMETERS sets the parameters which are needed to enable a RELOCATE-DML to be executed.

Relocation takes place in a sequence of relocation actions which are executed in one or more consecutive online utility runs. Here the DBH maintains the cross-transaction information on the progress of the relocation (e.g. source and target levels) internally. This information is lost when the session or session section ends or when the database of the realm concerned is detached.

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>

```

SUBSCHEMA-NAME = <structured-name 1..30>

Subschema name. Only names of user subschemas are permitted.

REALM-NAME = <structured-name 1..30>

Realm name. The realm name must be a valid name in the defined subschema. Only names of user realms are permitted. Names of temporary realms are not permitted. The subschema must also include all realms which contain the DBTTs whose entries refer to records and tables in the realm which are to be relocated. The corresponding check takes place only when the DMLs are executed. Record types and sets of the realm, on the other hand, need not be fully contained in the specified subschema. To this extent the relocation of records and tables takes place independently of the specifications in the subschema.

RELOCATE-TYPE =

Relocate type. Defines the relocation variant.

RELOCATE-TYPE = *RECORD-PAGES(...)

Relocates records and small tables within a realm.

INITIALIZE =

Defines the behavior of the RELOCATE DML when relocation in a realm is distributed over multiple online utility runs.

Whenever initialization of source and target levels takes place when INITIALIZE = *ANY or *YES, the free space search is implicitly positioned to the beginning of the realm; this corresponds to calling FPASCAN with SEARCH-MODE=*REUSE. In particular, after a complete relocation the source and target levels are not automatically reinitialized. Further execution of relocation actions using INITIALIZE = *NO or INITIALIZE = *ANY will consequently always result in the return information that there is nothing more to do.

INITIALIZE = *ANY

The source and target levels are reinitialized in accordance with certain conditions:

- If a relocation had already been started in a previous utility routine run, this relocation is continued. The source and target levels are not reinitialized.
- If no relocation had been started in a previous online utility run in this session section or in the period since the realm concerned was last attached, the first relocation action implicitly leads to the source and target levels being initialized.

INITIALIZE = *YES

The source and target levels are always reinitialized.

When the first relocation action takes place, the source and target levels are always initialized. The relocation consequently begins again.

You can use this setting to relocate these pages retroactively above all when a previous CLASH-HANDLING=*SKIP-PAGE means that access conflicts with other user transactions have resulted in relocatable pages being skipped.

INITIALIZE = *NO

The source and target levels are not reinitialized.

A relocation which had already been started in the previous online utility run is continued. If no relocation had been started, the source and target levels are not initialized. If you want to execute relocation in a realm in multiple online utility runs, you can use INITIALIZE=*NO to prevent an intended continuation resulting in a restart. It could be that between two online utility runs offline activities are performed in the realm which mean that there is no sense in continuing relocation (now reinitialized).

PAGES-PER-DML = <integer>

Specifies the maximum number of source pages which are to be emptied with a RELOCATE DML (default value: 1).

SKIP-ABOVE-FILLING = <1..100>

Defines a percentage value which specifies the maximum level to which the source page may be filled to permit it to be emptied. The default value of 100 includes all relocatable pages. A lower value enables you to restrict relocation to pages which are not very full. The DBH also uses a specification of under 100 to exclude pages from relocation on the basis of the free place administration data. This optimization consequently also enables you to prevent unnecessary reading of data pages.

CLASH-HANDLING =

Defines how the RELOCATE DML behaves when a source page which is locked by another transaction is to be processed next.

In CLASH-HANDLING no option is offered of waiting for the end of the conflict while the RELOCATE DML is processed because in unfavorable situations this can result in deadlocks and this could impair applications which execute parallel to the UDS online utility.

CLASH-HANDLING = *BREAK-DML

The RELOCATE DML is aborted, but any relocations already performed in this DML or in previous DMLs of the same transaction are retained. With the next RELOCATE DML - as a rule in a new transaction -, possibly after a delay in the UDS online utility, another attempt is made to relocate this page.

When CLASH-HANDLING=*BREAK-DML, all required pages are relocated if the source and target levels match. However, if conflicts occur it is possible that additional RELOCATE DMLs are required for this purpose.

CLASH-HANDLING = *SKIP-PAGE

The page is skipped. No attempt is made in a subsequent DML to repeat relocation for this page. However, if too many source or target pages were skipped in a RELOCATE DML, the DML is always aborted.

When CLASH-HANDLING=*SKIP-PAGE, the required number of pages (PAGES-PER-DML parameter) is generally relocated with a RELOCATE DML. However, it can happen that not all required pages are relocated within a RELOCATE cycle. In this case a follow-up action (INITIALIZE=*YES) can be used to complete the relocation later when, for example, corresponding conflicts are less likely.

RELOCATE-TYPE = *BASE-LEVEL-TABLE-PAGES(...)

Relocates level 0 pages of multi-level tables within a realm, in other words moves whole level 0 pages of a table to the free beginning of a realm.

INITIALIZE =

See ["INITIALIZE =" on page 279](#).

PAGES-PER-DML= <integer 1..16777215>

Specifies the maximum number of source pages which are to be emptied with a RELOCATE-DML (default value: 1).

CLASH-HANDLING =

See ["CLASH-HANDLING =" on page 280](#).

RELOCATE-TYPE = *INDEX-LEVEL-TABLE-PAGES(...)

Relocates level N pages of multi-level tables within a realm, in other words moves level N pages of a table to the free beginning of a realm.



Relocation (RELOCATE) is always performed within a transaction. The current transaction of the UDS online utility is opened with the READY UPDATE statement to the DBH. This applies for all relocate types except *INDEX-LEVEL-TABLE-PAGES, for which a READY EXCLUSIVE UPDATE is mandatory to permit relocation and necessary update of references without a problem.

INITIALIZE =

See ["INITIALIZE =" on page 279](#).

PAGES-PER-DML= <integer 1..16777215>

Specifies the maximum number of source pages which are to be emptied with a RELOCATE-DML (default value: 1).

RELOCATE-TYPE = *DISTRIBUTABLE-TABLE-PAGES(...)

Relocates level 0 pages of a distributable list from one realm to another realm.

INITIALIZE =

See ["INITIALIZE =" on page 279](#).

PAGES-PER-DML= <integer 1..16777215>

Specifies the maximum number of source pages which are to be emptied with a RELOCATE-DML (default value: 1).

CLASH-HANDLING =

See ["CLASH-HANDLING =" on page 280](#).

SET-NAME= <structured-name 1..30>

Specifies the distributable list for which relocation is to take place.

The set specified must be a distributable list, otherwise no relocation will take place. In this case an error message is issued and the procedure is aborted.

The corresponding checks are performed only when the DMLs of the UDS online utility are executed.

TARGET-REALM-NAME= <structured-name 1..30>

Defines the target realm, in other words the realm to which relocation is to take place.

This realm must be different from the source realm (REALM-NAME parameter) and must be known in the subschema selected (SUBSCHEMA-NAME parameter) and belong to the distributable list (WITHIN clause of the member record type's DDL-RECORD declaration).

The corresponding checks are performed only when the DMLs of the UDS online utility are executed.

Defining the properties of a REORGPPP DML (SET-REORGANIZE-PPP-PARAMETERS)

SET-REORGANIZE-PPP-PARAMETERS sets the parameters which are needed to enable a REORGPPP DML to be executed. PPP reorganization takes place in a sequence of reorganization actions that are executed in one or more consecutive online utility runs. Here the DBH internally maintains the cross-transaction information on the progress of the PPP reorganization, e.g. current page number. This information is lost when the session or session section ends or when the database of the realm concerned is detached.

SET-REORGANIZE-PPP-PARAMETERS	
SUBSCHEMA-NAME	= <structured-name 1..30>
,REALM-NAME	= <structured-name 1..30>
,INITIALIZE	= <u>*ANY</u> / *YES / *NO
,PAGES-PER-DML	= <integer 1..16777215>
,CLASH-HANDLING	= <u>*BREAK-DML</u> / *SKIP-PAGE

SUBSCHEMA-NAME = <structured-name 1..30>

Name of the subschema. Only names of user subschemas are permitted.

REALM-NAME = <structured-name 1..30>

Name of the realm. The realm name must be a valid name in the defined subschema. Only names of user realms are permitted. Names of temporary realms are not permitted. The subschema must also include all realms which contain the DBTTs whose entries refer to records and tables in the realm where probable position pointers are to be updated. The corresponding check takes only place when the DMLs are executed. Record types and sets of the realm, on the other hand, need not be fully contained in the specified subschema. To this extent the reorganization of pointers in records and tables takes place independently of the specifications in the subschema.

INITIALIZE =

Defines the behaviour of the REORGPPP DML when PPP reorganization in a realm is distributed over multiple online utility runs.

Initialization of the current page takes place when INITIALIZE = *ANY or *YES. In particular, after a complete PPP reorganization the current page number is not automatically re-initialized. Further execution of PPP reorganization actions using INITIALIZE = *NO or INITIALIZE = *ANY results in the return information that there is nothing more to do.

INITIALIZE = *ANY

The current page number is re-initialized in accordance with the following conditions:

- If PPP reorganization has already been started in a previous utility routine run, this reorganization is continued. The current page number is not re-initialized.
- If no PPP reorganization has been started in a previous online utility run in this session section or in the period since the realm concerned was last attached, the first PPP reorganization action implicitly leads to the current page number being initialized.

INITIALIZE = *YES

The current page number is always re-initialized.

The PPP reorganization consequently begins again.

You can use this setting to reorganize PPP in these pages retroactively above all when a previous CLASH-HANDLING=*SKIP-PAGE means that access conflicts with other user transactions have resulted in some pages being skipped.

INITIALIZE = *NO

The current page number is not re-initialized.

A PPP reorganization which has already been started in the previous online utility run is continued. If no PPP reorganization has been started, the current page number is not initialized.

PAGES-PER-DML = <integer 1..16777215>

Specifies the maximum number of pages where PPPs are to be updated with a REORGPPP DML. Default value: 1.

CLASH-HANDLING =

Defines how the behaviour of the REORGPPP DML when a page that is locked by another transaction is to be processed next.

CLASH-HANDLING does not offer any option for waiting until the conflict ends while the REORGPPP DML is processed because in unfavourable situations this can result in deadlocks and impair applications which execute parallel to the UDS online utility.

CLASH-HANDLING = *BREAK-DML

The REORGPPP DML is aborted, but any PPP reorganizations already performed in this DML or in previous DMLs of the same transaction are retained. With the next REORGPPP DML - as a rule in a new transaction -, possibly after a delay in the UDS online utility, it is attempted again to perform PPP reorganization in this page.

When CLASH-HANDLING=*BREAK-DML, PPP in all required pages are updated if the last page in realm is reached. However, if conflicts occur it is possible that additional REORGPPP DMLs are required for this purpose.

CLASH-HANDLING = *SKIP-PAGE

The page is skipped. It is not attempted in a subsequent DML to repeat PPP reorganization for this page. However, if too many pages are skipped in a REORGPPP DML, the DML is aborted.

When CLASH-HANDLING=*SKIP-PAGE, the required number of pages (PAGES-PER-DML parameter) is generally handled with a REORGPPP DML. However, it can happen that not all required pages are handled within a REORGPPP cycle. In this case a follow-up action (INITIALIZE=*YES) can be used to complete the PPP reorganization later when, for example, corresponding conflicts are less likely.

Showing the parameters which are currently valid for DML FPASCAN (SHOW-FPA-SCAN-PARAMETERS)

SHOW-FPA-SCAN-PARAMETERS shows the parameters which are currently valid for the DML FPASCAN.

SHOW-FPA-SCAN-PARAMETERS

Showing a procedure (SHOW-PROCEDURE)

SHOW-PROCEDURE shows a procedure once. Standard procedures can also be shown.

SHOW-PROCEDURE
PROCEDURE-NAME = <structured-name 1..20> / *STDRELOC / *STDFPASCAN / *STDREPPP

PROCEDURE-NAME =

Name of the procedure which is to be shown.

PROCEDURE-NAME = <structured-name 1..20>

Name of the user-defined procedure which is to be shown.

PROCEDURE-NAME = *STDRELOC

The standard procedure *STDRELOC (see [section "Predefined standard procedures" on page 299](#)) is to be shown.

PROCEDURE-NAME = *STDFPASCAN

The standard procedure *STDFPASCAN (see [section "Predefined standard procedures" on page 299](#)) is to be shown.

PROCEDURE-NAME = *STDREPPP

The standard procedure *STDREPPP (see [section "Predefined standard procedures" on page 299](#)) is to be shown.

Showing the parameters which are currently valid for DML PREFRLM (SHOW-PREF-REALM-PARAMETERS)

SHOW-PREF-REALM-PARAMETERS shows the parameters which are currently valid for the DML PREFRLM.

SHOW-PREF-REALM-PARAMETERS

Showing the parameters which are currently valid for DML RELOCATE (SHOW-RELOCATE-PARAMETERS)

SHOW-RELOCATE-PARAMETERS shows all parameters of the DML RELOCATE. All parameters which are not relevant for the RELOCATE-TYPE currently set have the value "UNDEFINED".

SHOW-RELOCATE-PARAMETERS

Showing the parameters which are currently valid for DML REORGPPP (SHOW-REORGANIZE-PPP-PARAMETERS)

SHOW-REORGANIZE-PPP-PARAMETERS shows the parameters which are currently valid for the DML REORGPPP.

SHOW-REORGANIZE-PPP-PARAMETERS

Showing the current value of a variable (SHOW-VARIABLE)

SHOW-VARIABLE shows the current value of a variable once.

When the statement is executed, an S variable of the same name is assigned the value of the variable shown in accordance with the type. If the S variable does not yet exist, it is created with the default parameters (e.g. visible at procedure level).

SHOW-VARIABLE
VARIABLE-NAME = <structured-name 1..20> / RCODE / STATUSCODE / RELOC-DESTINATION / RELOC-FREED-PAGES / RELOC-LOCKED-PAGES / RELOC-ORIGIN / RELOC-ORIGIN-LOCKS / RELOC-DEST-LOCKS / REORG-PPP-CURRENT / REORG-PPP-PAGES / REORG-PPP-LOCKED

VARIABLE-NAME= <structured-name 1..20>

Name of the variable whose value is to be shown.

VARIABLE-NAME = RCODE / STATUSCODE / RELOC-DESTINATION / RELOC-FREED-PAGES / RELOC-LOCKED-PAGES / RELOC-ORIGIN / RELOC-ORIGIN-LOCKS / RELOC-DEST-LOCKS / REORG-PPP-CURRENT / REORG-PPP-PAGES / REORG-PPP-LOCKED

Name of a predefined variable (see [section "Predefined variables" on page 298](#)).

8.9 Procedure statements of the UDS online utility

The procedure statements which the UDS online utility processes as DML and procedure requests and which are transferred in text form as a result of the CODE parameter in the DECLARE-PROCEDURE statement are listed below (see [section “Defining a procedure \(DECLARE-PROCEDURE\)” on page 271](#)).

Statement	Meaning
<u>ADD</u>	Add value to a variable
<u>BREAK</u>	Terminate procedure sequence immediately
<u>END</u>	Terminate input of procedure statements
<u>EXIT</u>	Terminate procedure sequence after current cycle
<u>FINISH</u>	Terminate current transaction
<u>FPASCAN</u>	Define start page for free place search
<u>MOVE</u>	Define value of a variable
<u>PREFRLM</u>	Set or change preferred realm for a distributable list
<u>READY</u> <u>[EXCLUSIVE]</u> <u>UPDATE</u>	Start current transaction of the UDS online utility
<u>RELOCATE</u>	Perform relocation
<u>REMARK</u> or <u>*</u>	Insert remark
<u>REORGPPP</u>	Performing a PPP reorganization
<u>SHOW</u>	Show value of a variable
<u>WAIT</u>	Define wait time

Table 19: Procedure statements of the UDS online utility

Adding a value to a variable (ADD)

$$\text{ADD } name, \left\{ \begin{array}{l} value \\ variable \end{array} \right\} [, condition]$$

name

Name of the variable to which a value is to be added. The *name* variable must be numeric.

value

Value which is to be added to the *name* variable.

variable

Variable whose value is to be added to the *name* variable. The *variable* variable must be numeric.

condition

Condition under which the ADD statement is to be executed. A description of the syntax of conditions is provided in [section “Condition syntax” on page 300](#).

With the ADD statement the *value* value or the value of the *variable* variable is added to the *name* variable which is defined beforehand. The variables must be numeric. When a condition is specified, it must have the value *true*. When no condition is specified, the values are always added.

Terminating a procedure sequence immediately (BREAK)

BREAK *condition*

condition

Condition under which the BREAK statement is to be executed. A description of the syntax of conditions is provided in [section “Condition syntax” on page 300](#).

The BREAK statement immediately terminates the procedure sequence which was started with REPEAT-PROCEDURE if the *condition* condition has the value *true*. No further procedure statements are executed in what is then the last cycle of the procedure. BREAK can be used in particular to terminate procedure processing if READY is not successful.

Terminating input of procedure statements (END)

END

The END statement terminates an input sequence of procedure statements which was started with DECLARE-PROCEDURE using CODE=*SYSDTA. When input via CODE=*c-string* or *<filename>*, the end of the procedure is recognized on account of the end of the c-string or the end of file.

Terminating the procedure sequence after the current cycle (EXIT)

EXIT *condition*

condition

Condition under which the EXIT statement is to be executed. A description of the syntax of conditions is provided in [section “Condition syntax” on page 300](#).

The EXIT statement terminates the procedure sequence which was started with REPEAT-PROCEDURE after the current cycle has been completed if the *condition* condition has the value *true*.

Terminating the current transaction (FINISH)

FINISH [WITH CANCEL]

The FINISH statement sends a FINISH DML to the DBH which terminates the current transaction of the UDS online utility. The DML returns a status code which is available in the predefined variable RCODE (see [section “Predefined variables” on page 298](#)).

After a DML FPASCAN, specification of WITH CANCEL has no effect (see [section “Defining the start page for the free place search \(FPASCAN\)” on page 292](#)).

Defining the start page for the free place search (FPASCAN)

FPASCAN

The FPASCAN statement sends an FPASCAN DML to the DBH which defines the start page for the free place search. The start page is specified in the SDF statement SET-FPA-SCAN-PARAMETERS (see [section “Defining the search mode for the free place search \(SET-FPA-SCAN-PARAMETERS\)” on page 275](#)).

If SEARCH- MODE=*REUSE is specified in SET-FPA-SCAN-PARAMETERS, free space is searched for from the beginning of the realm (search mode “Second Scan”).

When SEARCH-MODE=*NOREUSE, required storage space is searched for in the contiguous free area at the end of the realm and PLACEMENT OPTIMIZATION is taken into account (search mode “First Scan”). The change in the search for space becomes effective immediately with *REUSE and is initially noted in the case of *NOREUSE and becomes effective when the next checkpoint is reached. These actions are not canceled by a subsequent explicit and internal FINISH WITH CANCEL. WITH CANCEL is then ignored.

The DML returns a status code which is available in the predefined RCODE variable (see [section “Predefined variables” on page 298](#)).

Defining the value of a variable (MOVE)

$$\underline{\text{MOVE}} \text{ } name, \left\{ \begin{array}{l} value \\ variable \end{array} \right\} [, condition]$$

name

Name of the variable whose value is to be defined.

value

Value which is to be entered in the *name* variable.

variable

Variable whose value is to be entered in the *name* variable.

condition

Condition under which the MOVE statement is to be executed. A description of the syntax of conditions is provided in [section "Condition syntax" on page 300](#).

The MOVE statement is used to enter the *value* value in the *name* variable which was declared beforehand or to take over the value of the predefined or declared *variable* variable. When the *condition* condition is specified, it must have the value *true* to permit the procedure statement to be executed. If no condition is specified, the corresponding value is always entered. When a value is taken over from another variable, the variable types must be compatible.

Setting or changing the preferred realm for a distributable list (PREFRLM)

PREFRLM

When a database is set up, a preferred realm in which the DBH stores new level 0 pages is assigned for each distributable list.

The PREFRLM statement is used to send a DML which sets a new preferred realm or changes the preferred realm for a distributable list to the DBH.

The parameters for the PREFRLM statement are defined using the SDF statement SET-PREF-REALM-PARAMETERS.

Starting current transactions of the UDS online utility (READY UPDATE)

READY [EXCLUSIVE] UPDATE

The READY UPDATE statement sends a READY DML to the DBH which starts the current transaction of the UDS online utility.

The name of the realm which is to be processed must have been specified before execution with the SET-RELOCATE-PARAMETERS statement (see [section “Defining the properties of a RELOCATE DML \(SET-RELOCATE-PARAMETERS\)” on page 278](#)). Initialization of a string of relocation transactions is based on the INITIALIZE parameter.

The EXCLUSIVE entry is mandatory for RELOCATE-TYPE=*INDEX-LEVEL-TABLE-PAGES in a relocation transaction. The EXCLUSIVE entry is not permitted for the other RELOCATE types and is rejected with the following error message:
USAGE MODE EXCLUSIVE NOT ALLOWED .

Specification of the UPDATE parameter is mandatory in the case of a relocation transaction. The DML returns a status code which is available in the predefined variable RCODE (see [section “Predefined variables” on page 298](#)).

Performing a relocation (RELOCATE)

RELOCATE

The RELOCATE statement sends a RELOCATE DML to the DBH which executes the relocation of records and tables.

The number of relocations to be performed with a DML is determined by the PAGES-PER-DML parameter in the SET-RELOCATE-PARAMETERS statement (see [section “Defining the properties of a RELOCATE DML \(SET-RELOCATE-PARAMETERS\)” on page 278](#)).

The DML returns a status code which is available in the predefined variables RCODE and STATUSCODE (see [section “Predefined variables” on page 298](#)). In addition, when the DML terminates normally the predefined variables RELOCDESTINATION, RELOC-ORIGIN, RELOC-FREED-PAGES and RELOC-LOCKED-PAGES are supplied with values and can be used to control - among other things - the progress of the relocation.

Inserting a remark (REMARK)

```
{REMARK  
*  
}
```

A statement beginning REMARK or '*' is interpreted as a remark when the statement is executed and is ignored when a procedure is executed.

An individual remark statement may contain a maximum of 75 characters. The character ';' is permitted in the remark only if it is enclosed in quotes, otherwise it indicates the end of the command.

Performing a PPP reorganization (REORGPPP)

REORGPPP

The REORGPPP statement sends a REORGPPP DML to the DBH that executes the PPP reorganization.

The number of pages where PPP reorganization is to be performed with a DML is determined by the PAGES-PER-DML parameter in the SET-REORGANIZE-PPP-PARAMETERS statement.

The DML returns a status code which is available in the predefined variables RCODE and STATUSCODE (see [section "Predefined variables" on page 298](#)). In addition, when the DML terminates normally, the predefined variables REORG-PPP-CURRENT, REORG-PPP-PAGES, and REORG-PPP-LOCKED supply values that can be used to control, for example, the progress of the PPP reorganization.

If all user realms in the database are handled in this way and PPPs are updated successfully in all processed pages, the new pages need not to be formatted during an online realm extension. For this, all user realms must be contained in the subschema.

This considerably improves performance.

Showing the value of a variable (SHOW)

`SHOW name[,condition]`

name

Name of the variable whose value is to be shown.

condition

Condition under which the SHOW statement is to be executed. A description of the syntax of conditions is provided in [section “Condition syntax” on page 300](#).

The SHOW statement enables the current value of a predefined or explicitly defined variable to be output, possibly depending on a fulfilled condition. When a condition is specified, it must have the *true* value to permit the procedure statement to be executed. If no condition is specified, the corresponding value is always output.

When the statement is executed, an S variable of the same name is assigned the value of the variable shown in accordance with the type. If the S variable does not yet exist, it is created with the default parameters (e.g. visible at procedure level).

Defining the wait time (WAIT)

`WAIT n[,condition]`

n

Number of seconds which are to be waited.

condition

Condition under which the WAIT statement is to be executed. A description of the syntax of conditions is provided in [section “Condition syntax” on page 300](#).

With the WAIT statement *n* seconds are waited, possibly depending on a fulfilled condition. When a condition is specified, it must have the *true* value to permit the procedure statement to be executed. If no condition is specified, the system always waits.

8.10 Error handling of the UDS online utility

The UDS online utility evaluates the status codes of the procedure commands which are sent to the DBH and in some cases responds to these.

1. If it can be recognized from the status code that a transaction has been terminated, subsequent DBH commands are not executed until the next READY within the same procedure cycle.
2. The predefined variables RCODE and STATUSCODE (see [section “Predefined variables” on page 298](#)) are initialized immediately before a DBH command is executed.
3. The following status codes result in the procedure aborting:

020, 022, 042, 092, 093, 099, 103, 113, 123, 131, 132, 134, 136, 137, 142, 144, 145, 146, 151, 152, 154, 155, 161, 163, 164, 165, 166, 782, 783, 784, 785, 786, 789, 805, 901, 950, 954.

In all these cases a fault exists which is expected to be encountered again if a procedure cycle is executed again without additional measures being taken. The procedure is generally aborted only at the end of the current procedure cycle.

DML requests up to the end of the current procedure cycle are suppressed if it is clear that no further transaction is open. Other procedure requests (e.g. MOVE, SHOW, WAIT) are still executed until the procedure cycle ends.

In addition, status code 162 may occur during the execution of the DML FPASCAN, which also results in the procedure aborting.

A detailed description of the status codes is provided in the [“Application Programming”](#) and [“Messages”](#) manuals.

8.11 Predefined variables

Some variables of the UDS online utility are predefined and can be used in statements and conditions without a declaration.

Variable	Meaning
RCODE	Statement code and status code (5 bytes, printable) of a DML READY, RELOCATE, FPASCAN or FINISH
STATUSCODE	Status code (3 bytes, printable) of a DML READY, RELOCATE, FPASCAN or FINISH
RELOC-DESTINATION	Block number of the current target level of the relocation
RELOC-DEST-LOCKS	Number of target pages locked by other transactions when the last normal RELOCATE-DML was executed
RELOC-FREED-PAGES	Number of database pages from which records or tables were relocated with the last normally executed RELOCATE DML. The value is also retained if the transaction is reset later
RELOC-LOCKED-PAGES	Number of pages locked by other transactions when the last normal RELOCATE-DML was executed; sum of RELOC-DEST-LOCKS and RELOC-ORIGIN-LOCKS
RELOC-ORIGIN	Block number of the current source level for relocation
RELOC-ORIGIN-LOCKS	Number of source pages locked by other transactions when the last normal RELOCATE-DML was executed
REORG-PPP-CURRENT	Page number for next PPPs reorganization
REORG-PPP-LOCKED	Number of pages locked by other transactions when the last normal REORGPPP DML was executed
REORG-PPP-PAGES	Number of pages where PPPs were reorganized with the last normally executed REORGPPP DML

Table 20: Predefined variables of the UDS online utility

8.12 Predefined standard procedures

A standard procedure is provided for each task of the UDS online utility.

The standard procedure ***STDRELOC** permits a simple sequence of relocation transactions which is automatically terminated when the first READY UPDATE statement fails or the source and target levels are identical. It is consequently equivalent to

```
READY UPDATE;RELOCATE;  
EXIT COND=STATUSCODE EQ 010;FINISH
```

The standard procedure ***STDFPASCAN** executes an FPASCAN transaction, but this is not executed if the READY UPDATE statement fails. It is consequently equivalent to

```
READY UPDATE;FPASCAN;FINISH
```

The standard procedure ***STDREPPP** is equivalent to

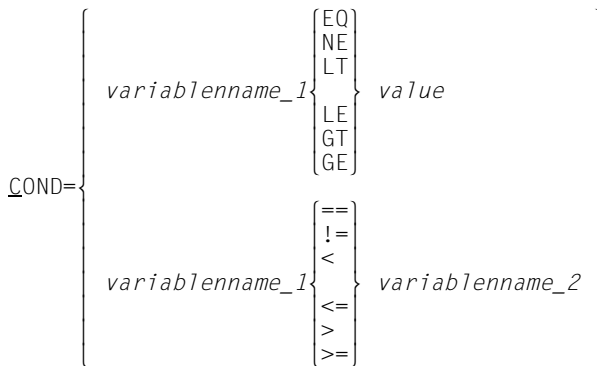
```
READY UPDATE; REORGPPP; EXIT COND = STATUSCODE EQ 010; FINISH;
```

8.13 Status codes

A description of the status codes of the UDS online utility is provided in the [“Application Programming”](#) and [“Messages”](#) manuals.

8.14 Condition syntax

You can specify conditions in the procedure statements as follows:



A condition is initiated at procedure level by means of COND= or C=. The condition itself consists of three strings, each separated by a blank:

- The first string is the name of a predefined or declared variable.
- The second string is the relational operator (EQ, NE, LT, LE, GT, GE, ==, !=, <, <=, >, >=).
- The third string is a value or the name of a variable. The type of a value must fit the type of variable specified in the first string.

With the relational operators ==, !=, <, <=, > and >= the third string is interpreted as the name of a variable, which must fit the type of variable in the first string.

In the case of conditions with the predefined variables RCODE and STATUSCODE it must be borne in mind that strings are concerned, not numbers. Consequently leading zeros must also be specified explicitly.

The conditions are evaluated when the corresponding procedure statement is executed.

8.15 Examples

A few examples of how to use the UDS online utility are provided below.

Example 1: Relocation of a fixed number of pages using the standard procedure

You want to relocate around 2,000 pages toward the beginning of the realm. You use the standard procedure *STDRELOC.

```
//SET-ONLINE-UTILITY-PARAMETERS -  
// DBH=*INDEPENDENT, -  
// CONFIGURATION-NAME=SESSION1  
//SET-RELOCATE-PARAMETERS -  
// SUBSCHEMA-NAME=SUB1, -  
// REALM-NAME=AREA1, -  
// PAGES-PER-DML=2  
//REPEAT-PROCEDURE -  
// PROCEDURE-NAME=*STDRELOC, -  
// CYCLE-LIMIT=1000  
//SHOW-VARIABLE RELOC-ORIGIN  
//SHOW-VARIABLE RELOC-DESTINATION
```

Example 2: Relocation of all possible pages using the standard procedure

You want to relocate as much as possible toward the beginning of the realm. You use the standard procedure *STDRELOC. The realm has 10,000 pages.

```
//SET-ONLINE-UTILITY-PARAMETERS -  
// DBH=*INDEPENDENT, -  
// CONFIGURATION-NAME=SESSION2  
//SET-RELOCATE-PARAMETERS -  
// SUBSCHEMA-NAME=SUB2, -  
// REALM-NAME=AREA2, -  
// PAGES-PER-DML=1  
//REPEAT-PROCEDURE -  
// PROCEDURE-NAME=*STDRELOC -  
// CYCLE-LIMIT=10000  
//SHOW-VARIABLE RELOC-ORIGIN
```

Example 3: Preventing conflicts

You want to move as much as possible toward the beginning of the realm. However, at some time the database will be detached. The UDS online utility will then be terminated immediately. The realm has 10,000 pages. In the event of information about conflicts (here: more than 5) the UDS online utility should also terminate. Normally three pages are emptied in a transaction.

```
//SET-ONLINE-UTILITY-PARAMETERS -
// DBH=*INDEPENDENT, -
// CONFIGURATION-NAME=SESSION3
//SET-RELOCATE-PARAMETERS -
// SUBSCHEMA-NAME=SUB3, -
// REALM-NAME=AREA3, -
// PAGES-PER-DML=3
//DECLARE-VARIABLE -
// VARIABLE-NAME=NRCLASHS, -
// TYPE=*INTEGER, -
// INITIAL-VALUE=0
//DECLARE-PROCEDURE -
// PROCEDURE-NAME=TA3, -
// CODE=*SYSDTA
/SEND-DATA -
/ RECORD='READY UPDATE;RELOCATE;'
/SEND-DATA -
/ RECORD='EXIT COND=STATUSCODE EQ 010'
/SEND-DATA -
/ RECORD='ADD NRCLASHS,1,COND=RELOC-FREED-PAGES LT 3;FINISH'
/SEND-DATA -
/ RECORD='EXIT COND=NRCLASHS GE 5'
/SEND-DATA RECORD=*EOF
//REPEAT-PROCEDURE -
// PROCEDURE-NAME=TA3, -
// CYCLE-LIMIT=10000
//SHOW-VARIABLE RELOC-ORIGIN
//SHOW-VARIABLE RELOC-DESTINATION
```

Alternatively, you can also enter the DMLs of the procedure TA3 directly via SYSDTA. It is essential here that the procedure input is terminated with END.

```
//...
//DECLARE-PROCEDURE -
// PROCEDURE-NAME=TA3, -
// CODE=*SYSDTA
READY UPDATE;RELOCATE;
EXIT COND=STATUSCODE EQ 010
ADD NRCLASHS,1,COND=RELOC-FREED-PAGES LT 3;FINISH
```

```
EXIT COND=NRCLASHS GE 5;END
//REPEAT-PROCEDURE -
// PROCEDURE-NAME=TA3, -
// CYCLE-LIMIT=10000
//...
```

Example 4: The prevention of conflicts is based on the locks reported

You want to move as much as possible toward the beginning of the realm. However, at some time the database will be detached. The UDS online utility will then be terminated immediately. The realm has 10,000 pages. In the event of information about conflicts (here: more than 10) the UDS online utility should also terminate. But it is not clear how many pages are normally emptied in a transaction. The lock conflicts which have occurred are used as a criterion.

```
//SET-ONLINE-UTILITY-PARAMETERS -
// DBH=*INDEPENDENT, -
// CONFIGURATION-NAME=SESSION4
//SET-RELOCATE-PARAMETERS -
// SUBSCHEMA-NAME=SUB4, -
// REALM-NAME=AREA4, -
// PAGES-PER-DML=2
//DECLARE-VARIABLE -
// VARIABLE-NAME=NRCLASHS, -
// TYPE=*INTEGER, -
// INITIAL-VALUE=0
//DECLARE-PROCEDURE -
// PROCEDURE-NAME=TA4, -
// CODE='READY UPDATE;RELOCATE;-
//EXIT COND=STATUSCODE EQ 010;-
//ADD NRCLASHS,1,COND=RELOC-LOCKED-PAGES GT 0;FINISH;-
//EXIT COND=NRCLASHS GE 10'
//REPEAT-PROCEDURE -
// PROCEDURE-NAME=TA4, -
// CYCLE-LIMIT=10000
//SHOW-VARIABLE RELOC-ORIGIN
//SHOW-VARIABLE RELOC-DESTINATION
```

Example 5: Preventing conflicts with waiting

You want to move as much as possible toward the beginning of the realm. However, at some time the database will be detached. The UDS online utility will then be terminated immediately. The realm has 10,000 pages. If there are indications of conflicts, the UDS online utility should wait 10 seconds because it is to be expected that the applications which are executing in parallel will soon become active in other areas. The lock conflicts which have occurred are used as a criterion. Following procedure execution, how many pages were released in total is shown in the FREED variable. The FREED variable is only supplied with a value after a successful FINISH so that any asynchronous abortions of relocation transactions do not distort the value.

```
//SET-ONLINE-UTILITY-PARAMETERS -
// DBH=*INDEPENDENT, -
// CONFIGURATION-NAME=SESSION5
//SET-RELOCATE-PARAMETERS -
// SUBSCHEMA-NAME=SUB5, -
// REALM-NAME=AREA5, -
// PAGES-PER-DML=2
//DECLARE-VARIABLE -
// VARIABLE-NAME=FREED, -
// TYPE=*INTEGER, -
// INITIAL-VALUE=0
//DECLARE-PROCEDURE -
// PROCEDURE-NAME=TA5, -
// CODE = 'READY UPDATE;RELOCATE;-
// EXIT COND=STATUSCODE EQ 010;-
// ADD FREED,RELOC-FREED-PAGES;-
// FINISH;-
// WAIT 10,COND=RELOC-LOCKED-PAGES GT 0'
//REPEAT-PROCEDURE -
// PROCEDURE-NAME=TA5 -
// CYCLE-LIMIT=10000
//SHOW-VARIABLE RELOC-ORIGIN
//SHOW-VARIABLE FREED
```


Example 6: Information on the status code

You want to move as much as possible toward the beginning of the realm. If a special situation arises, it should be possible to distinguish the status codes.

```
//SET-ONLINE-UTILITY-PARAMETERS -
// DBH=*INDEPENDENT, -
// CONFIGURATION-NAME=SESSION6
//SET-RELOCATE-PARAMETERS -
// SUBSCHEMA-NAME=SUB6, -
// REALM-NAME=AREA6, -
// PAGES-PER-DML=1
//DECLARE-VARIABLE -
// VARIABLE-NAME=READY-CODE, -
// TYPE=*STRING,
// INITIAL-VALUE=*STD
//DECLARE-VARIABLE -
// VARIABLE-NAME=RELOC-CODE, -
// TYPE=*STRING,
// INITIAL-VALUE=*STD
//DECLARE-PROCEDURE -
// PROCEDURE-NAME=TA6, -
// CODE='READY UPDATE;
//MOVE READY-CODE,RCODE,COND=STATUSCODE NE 000;
//RELOCATE;-
//MOVE RELOC-CODE,RCODE,COND=STATUSCODE NE 000;
//EXIT COND=STATUSCODE EQ 010;-
//FINISH'
//REPEAT-PROCEDURE -
// PROCEDURE-NAME=TA6, -
// CYCLE-LIMIT=10000
//SHOW-VARIABLE READY-CODE
//SHOW-VARIABLE RELOC-CODE
//SHOW-VARIABLE RELOC-ORIGIN
//SHOW-VARIABLE RELOC-DESTINATION
```

Example 7: Reducing the size of realms

You can reduce the size of a realm by initially relocating records and small tables in the realm toward the beginning of the realm online in a RELOCATE run (see [“Example 2: Relocation of all possible pages using the standard procedure” on page 301](#)).

You can then reorganize the DBTT sections, CALC areas and set occurrences located at the back of the realm which cover several pages offline using the BREORG utility routine.

Finally you can also use BREORG to release the pages which have become free at the back of the realm and thus reduce the size of the realm.

Example 8: Reorganizing PPPs

You can set the parameters which are needed to enable a REORGPPP DML to be executed using the SET-REORGANIZE-PPP-PARAMETERS statement. PPP reorganization takes place in a sequence of reorganization actions that are executed in one or more consecutive online utility runs.

When the REORGPPP DML terminates normally, the predefined variables REORG-PPP-CURRENT, REORG-PPP-PAGES, and REORG-PPP-LOCKED supply values that can be used to control the progress of the PPP reorganization.

```
//SET-ONLINE-UTILITY-PARAMETERS CONFIGURATION-NAME=DBDMLUTI
//SET-REORGANIZE-PPP-PARAMETERS SUBSCHEMA-NAME = SUB, -
//                                REALM-NAME= AREA-3 ,-
//                                INITIALIZE=ANY, PAGES-PER-DML=1000
//DECLARE-VARIABLE VARIABLE-NAME=READY-CODE, -
//                                TYPE=*STRING, INITIAL-VALUE=*STD
//DECLARE-VARIABLE VARIABLE-NAME=REOPPP-CODE, -
//                                TYPE=*STRING, INITIAL-VALUE=*STD
//DECLARE-PROCEDURE PROCEDURE-NAME=TA1, -
// CODE=' READY UPDATE; -
//MOVE READY-CODE,RCODE,COND=STATUSCODE NE 000; REORGPPP;-
//MOVE REOPPP-CODE,RCODE,COND=STATUSCODE NE 000;-
//EXIT COND=STATUSCODE EQ 010; FINISH'
//REPEAT-PROCEDURE PROCEDURE-NAME=TA1, CYCLE-LIMIT=1
//SHOW-VARIABLE READY-CODE
//SHOW-VARIABLE REOPPP-CODE
//SHOW-VARIABLE REORG-PPP-CURRENT
//SHOW-VARIABLE REORG-PPP-PAGES
//SHOW-VARIABLE REORG-PPP-LOCKED
```

9 Database reorganization with BREORG

Reorganization is an important part of database maintenance. It helps save not only storage space, but also time.

Storage space can be saved by reducing the size of realms, reducing the size of the Database Key Translation Table (DBTT) of a record type and thereby reducing the maximum permissible number of records, and creating new set tables with a higher occupancy level.

Time can also be saved when accessing the database by reducing the number of overflow pages for hash areas, updating physical pointers which are in the form of probable position pointers (PPP), and creating new set tables with a different occupancy level.

Reorganization may become necessary when a realm has become too small, or more records of a record type are stored than originally planned.

Reorganization should be performed on a regular basis if record types which are specified with LOCATION MODE IS CALC and frequently updated are stored in the database as sort key tables, pointer arrays, chains, or lists. In such cases, updating may render the probable position pointers (PPP) in the tables incorrect. This would adversely affect access times to the records via the tables (see the [“Database Operation”](#) manual).

9.1 Functions

The BREORG utility routine provides the following functions:

- Define buffer size
- Modify realm size
- Modify record population
- Reorganize CALC areas
- Reorganize tables and set constructs
- Reorganize probable position pointers (PPP)

BREORG functions can be applied on the user realms of the database as well as the PRIVACY-AND-IQF database in the DBDIR and the compiler database in the DBCOM.

When required, BREORG automatically extends the realms of the processed database. For details, please refer to the “[Database Operation](#)” manual, Automatic realm extension.

At startup BREORG takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

9.2 System environment

To run BREORG, information about the areas, sets, record types and tables to be reorganized is required. BREORG obtains this information from the schema information area (SIA) of the associated schema.

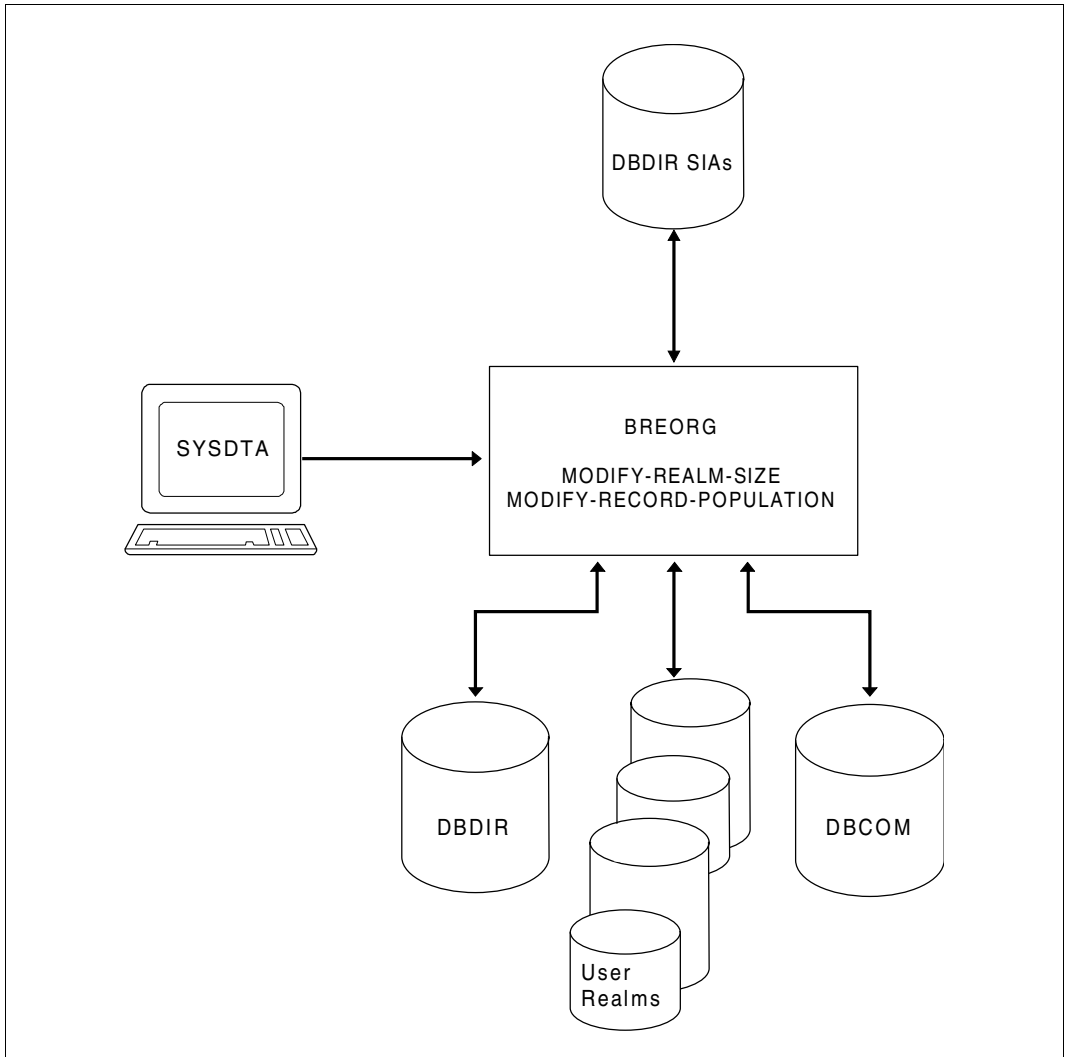


Figure 17: System environment for the MODIFY-REALM-SIZE and MODIFY-RECORD-POPULATION functions

For its REORGANIZE functions, BREORG also requires subschema information (only for the recreation of multi-level LIST sets). This information is obtained from the Subschema Information Area (SSIA).

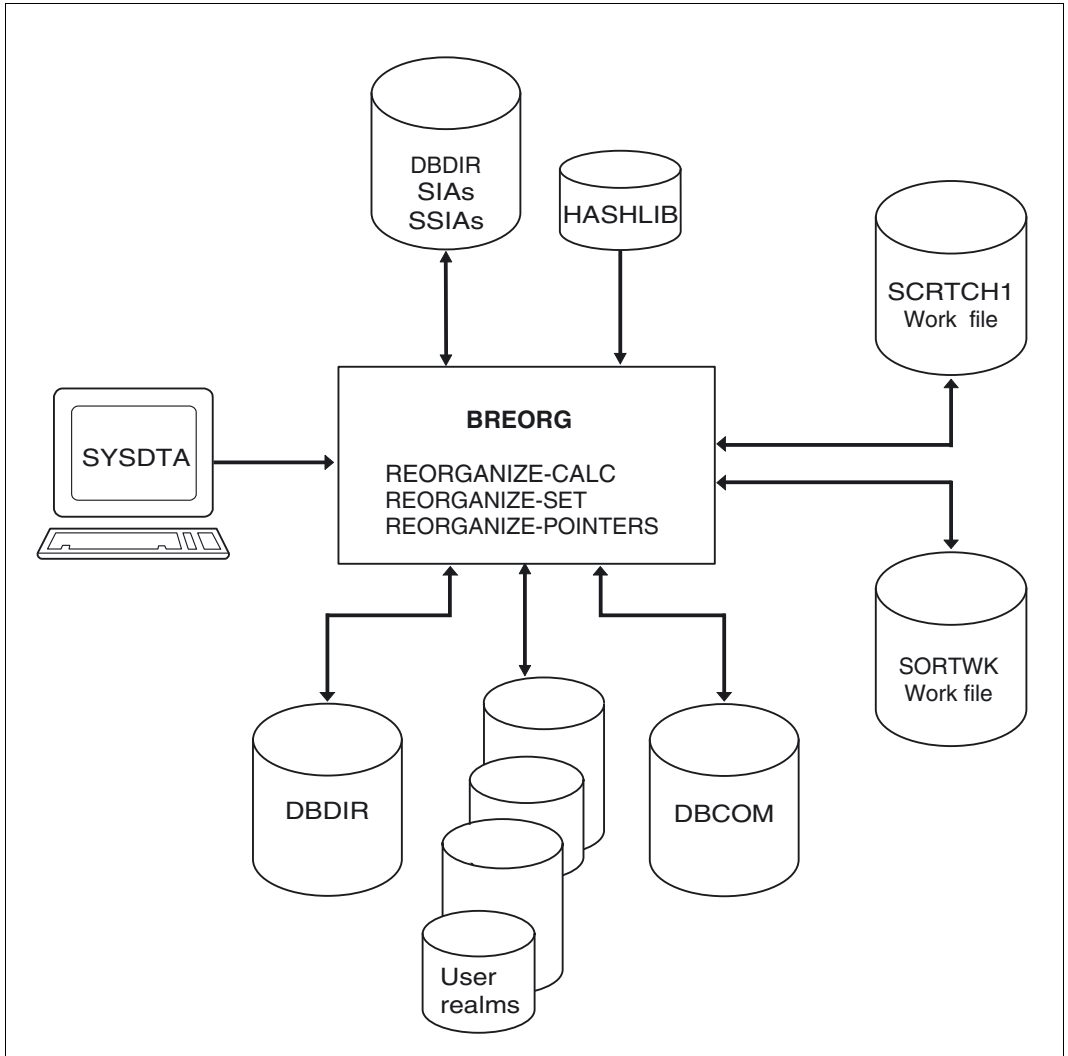


Figure 18: System environment for the REORGANIZE-CALC, REORGANIZE-SET and REORGANIZE POINTERS functions

Work files

For the REORGANIZE functions, BREORG requires different work files on disk. These files are automatically created and stored on public disk under the appropriate user ID and erased again after normal completion of the run.

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

The statements REORGANIZE-CALC and REORGANIZE-SET require two work files using the standard file link names SCRTCH1 and SORTWK:

SCRTCH1

BREORG requires this file in order to reorganize direct and indirect hash areas or multi-level tables.

It contains an intermediate version of the CALC entries or table lines to be processed.

SORTWK

Requires the SORT used by BREORG for sorting internal evaluation records (see manual "[SORT \(BS2000\)](#)").

If these work files are to be explicitly created, they must be assigned the following attributes:

Work-file-1

File link name SCRTCH1

Access method = PAM

Primary allocation, calculated as follows:

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

- for the reorganization of indirect hash areas:

$$(12 + \textit{key length}) * \textit{number of entries Bytes}$$

- for the reorganization of direct hash areas:

$$8 * \textit{number of entries Bytes}$$

- for the reorganization of multi-level tables:

$$12 * \textit{number of entries Bytes}$$

key length

Length of CALC key

number of entries

Number of CALC index entries or occupied table lines

Work-file-2

SORT needs Work-file-2 if there is not enough virtual memory for pre-sorting. The primary allocation should be based on the data population that is to be sorted while taking account of the safety factor recommended by SORT (see the discussion of work files in the manual “[SORT \(BS2000\)](#)”). There should always be an appropriate secondary allocation in case it is necessary to extend the storage space.

File link name SORTWK

Access method = PAM

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

- for the reorganization of indirect hash areas:

$(12 + \textit{key length}) * \textit{number of entries}$ Bytes

- for the reorganization of direct hash areas:

$(\textit{record length} + \textit{key length} + 7) * \textit{number of entries}$ Bytes

- for the reorganization of multi-level tables:

$12 * \textit{number of entries}$ Bytes

key length

Length of CALC key

number of entries

Number of CALC index entries or occupied table lines

record length

Length of records or table lines. Secondary allocation in case the storage space must be enlarged; *secondary* should be not less than 120, and not zero.

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

UTI . <i>tsn</i> . SCRTCH1	(360,360) for REORGANIZE-SET and for REORGANIZE-CALC
UTI . <i>tsn</i> . SORTWK	(120,120)

tsn stands for the task sequence number of the current task.

Work files for the REORGANIZE-POINTERS statement

With the REORGANIZE-POINTERS statement, BREORG uses work files for the record types and an additional work file for sorting.

You can also create the work files for the record types yourself via the file name, and the work file for sorting via the file link name.

Work files for the record types

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

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

Access method: SAM

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

$$\text{number of ppps} * 11 \text{ Bytes}$$

If you do not set up the files yourself, then BREORG uses the size of the DBTT for the record type *yyyyy* together with the size of the realm *xxx* to calculate the expected data population.

The minimum size is based on 5000 objects for buffering.

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

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

Access method: SAM

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 size of required individual files UTI.BREORG.*dbname*.*xxx*.*yyyyy* (*yyyyy*=0 bzw. *yyyyy*>1).

The work files are deleted once the probable position pointer (PPP) update has been completed.

Work file for sorting

SORT needs this work file if there is not enough virtual memory for pre-sorting. The primary allocation should be based on the data population that is to be sorted while taking account of the safety factor recommended by SORT (see the discussion of work files in the manual "[SORT \(BS2000\)](#)"). There should always be an appropriate secondary allocation in case it is necessary to extend the storage space.

File link name: SRT1WK

Access method: PAM

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

If the work file was created explicitly for sorting, it must also be deleted again explicitly if need be.

9.3 Database saving

If you have set up ALOG files and enabled AFIM logging, BREORG records after-images for all functions.

If ALOG files are missing when BREORG starts (even though AFIM logging was enabled), BREORG terminates with an error message before processing begins.

If there is an error involving an ALOG file while BREORG is running, further logging is suppressed, and BREORG terminates after execution of the current statement. This generally results in the creation of a logging gap.

Console message on AFIM logging

BREORG examines job switch 29. If this job switch is set, the console message LOGGING STOPPED FOR DATABASE *dbname* is issued when AFIM logging is aborted.

9.4 BREORG statements

Rules for input of statements

Incorrectly specified statements can be corrected interactively via SDF.

If the input is not made via the SDF mask, it is not always legal to enter any statement at any time, so some statements (e.g. OPEN-DATABASE) may be rejected.

The following statements are executed in the order given by the user, but only after the END statement is specified:

1. MODIFY-REALM-SIZE
2. MODIFY-RECORD-POPULATION,
3. REORGANIZE-CALC
4. REORGANIZE-POINTERS
5. REORGANIZE-SET.

If a statement containing an error is encountered when processing statements in batch mode, all statements until the first incorrect statement will be executed.

Statement	Meaning
ALLOCATE-BUFFERPOOL	Define buffer size (in Mbytes)
END	Terminate input of statements
MODIFY-REALM-SIZE	Modify realm size
MODIFY-RECORD-POPULATION	Modify record population
OPEN-DATABASE	Open database
REORGANIZE-CALC	Reorganize CALC areas
REORGANIZE-POINTERS	Reorganize all probable position pointers (PPP) contained in one realm
REORGANIZE-SET	Reorganize tables and set constructs
SPECIFY-SCHEMA	Specify schema
SPECIFY-SUBSCHEMA	Specify subschema
UNDO	Undo statement

Table 21: BREORG statements

The individual statements of BREORG are described below in alphabetical order.

Define buffer size (ALLOCATE-BUFFERPOOL)

The ALLOCATE-BUFFERPOOL statement defines the size of the used buffer pool in Mbytes.

If default values are not to be used for buffer initialization, this statement must be the first statement specified.

After the initial allocation, the ALLOCATE-BUFFERPOOL statement is no longer offered in the SDF mask.

This statement cannot be canceled with the UNDO statement.

ALLOCATE-BUFFERPOOL
BUFFER-SIZE = <u>*STD</u> / <integer 1..2000>

BUFFER-SIZE = *STD

The default size of the buffer pool is defined as 1 or 2 Mbytes, depending on the system involved.

BUFFER-SIZE = <integer 1..2000>

The size of the buffer pool must lie within the given limits. The maximum value depends on the system configuration and the version of the operating system.

Terminate input of statements (END)

The END statement is used to terminate the input of statements. All entered statements are executed after this statement.

The END statement cannot be canceled with the UNDO statement.

END

This statement has no operands.

Modify realm size (MODIFY-REALM-SIZE)

The MODIFY-REALM-SIZE statement can be used to change the size of a realm of the database.

```
MODIFY-REALM-SIZE
```

```
REALM-NAME = <realm-name>
,REALM-SIZE = <integer 1..16777216> / *RELATIVE(...)/ *MINIMUM
  *RELATIVE(...)
    | DIFFERENCE = <integer -16777216..16777216>
```

REALM-NAME = <realm-name>

Name of the realm which is to be modified.

REALM-SIZE = <integer 1..16777216>

The new size is equivalent to the specified value in database pages.

REALM-SIZE = *RELATIVE (...)

The new size is calculated from the old size and the specified difference (which may be a positive or negative value), but cannot be less than the size attained by specifying MINIMUM.

DIFFERENCE = <integer -16777216..16777216>

Difference with respect to the old realm size in database pages.

REALM-SIZE = *MINIMUM

The realm is reduced by the number of empty pages at the end.

Enlarging a realm

The physical enlargement of the file *realm-name* is requested by BREORG from the BS2000 DMS. The new pages are collected by the free place administration of the realm. If there is not enough free space (as determined by the free place administration, FPA) for the realm, then new free place administration tables (FPA-Extents) are created as needed.

When the DBDIR or DBCOM is expanded, the new, empty pages are always formatted. Whether or not new, empty pages are formatted when expanding a user realm depends on the state of the probable position pointers (PPP):

- If the probable position pointers were updated in all user realms with the REORGANIZE-POINTERS statement, then the new, empty pages are not formatted.
- If a probable position pointers reference in the new pages cannot be ruled out, then the new, empty pages are formatted.



If you are using private disks for storage, you should ensure that the disk provides sufficient space for enlarging the realms.

If the disk on which the realm ends does not have enough space for a realm enlargement, a continuation disk must be assigned for the file before the BREORG run.

Reducing a realm

BREORG reduces the size of the realm in the following manner: First, the achievable realm size is determined from the results of the MODIFY-REALM-SIZE command and the data in the realm. Unneeded FPA sections are released. FPA sections still needed are generally moved to the beginning of the realm (if this is possible). BREORG then requests the physical reduction of the realm via the DMS of BS2000, so no MODIFY-FILE-ATTRIBUTES statement is required to reduce the realm file.

Reorganization of occupied pages in the realm is not connected with this statement; only free pages are removed from the end of the realm and the free place administration is adjusted.



The realm DBDIR may only be modified under the PRIVACY-AND-IQF-SCHEMA.

The name DBDIR is internally converted to the realm name DATABASE-DIRECTORY, the name DBCOM to the realm name DATABASE-COMPILER-REALM.

Execution messages

On executing a MODIFY-REALM-SIZE statement, the results for the new free place administration (FPA) of a realm are output as follows:

```
***** RESULTS OF FPA-REORGANISATION OF AREA area name
NEW FPA FIRST PAGE      : area ref - page no
NEW FPA LAST PAGE      : area ref - page no
NEW NR OF EXTENTS      : number extents
NEW FPA SIZE            : number pages
NEW NR OF PAGES        : number pages

NR OF DATABASE ACCESSES : number physical io
```

area-name

Name of the enlarged realm or reduced realm

area ref - page no

For FPA FIRST PAGE: Smallest act-key of all act-keys of FPA pages (not necessarily identical to the beginning of the FPA)

For FPA LAST PAGE: Largest act-key of all act-keys of FPA pages (not necessarily identical to the end of the FPA)

number extents

Number of FPA extents

number pages

For NEW FPA SIZE: Number of pages in the new FPA area

For NEW NR OF PAGES: New page count for the realm

number physical io

Number of physical input and output operations

Modify record population (MODIFY-RECORD-POPULATION)

The MODIFY-RECORD-POPULATION statement can be used to change the maximum record population (DBTT) of a record type.

```
MODIFY-RECORD-POPULATION
```

```
RECORD-NAME = <record-name>
```

```
,RECORD-POPULATION = <integer 1..2147483647> / *RELATIVE(...) / *MINIMUM
```

```
  *RELATIVE(...)
```

```
    | DIFFERENCE = <integer -2147483647..2147483647>
```

RECORD-NAME = <record-name>

Name of the record type for which the record population is to be changed.

RECORD-POPULATION = <integer 1..2147483647>

The number of permissible DBTT entries for the record type is equivalent to the specified value.

RECORD-POPULATION = *RELATIVE (...)

The new size of the DBTT is calculated from the old size and the specified difference (which may be a positive or negative value), but cannot be less than the size attained by specifying MINIMUM.

DIFFERENCE = <integer -2147483647..2147483647>

Difference with respect to the old DBTT size; as a number of DBTT entries.

RECORD-POPULATION = *MINIMUM

The DBTT of the record type is reduced to the smallest possible value.



Entries that are invalid for databases with a 2-kbyte format cannot be detected via SDF if the entered values lie within the ranges specified above.

Since these databases are subject to additional checks, the old limits are still applicable to them.

Since the DBTT always occupies all pages that are reserved for it entirely, it is conceivable that after a BREORG run in which an increase is requested, more records can be stored than were actually specified in the statement. In the reverse case, if you use the statement to deallocate fewer DBTT entries than are actually contained in a DBTT page of the specified record type, then the maximum permissible number of records in this record type stays the same.



Since entire pages are always reserved for the DBTT, the number of desired entries is rounded up to full pages.

However, if the maximum RSQ is exceeded in the process, the number is rounded down to full DBTT pages.

The SSIA-RECORD may only be modified under the PRIVACY-AND-IQF-SCHEMA.

When a database is reorganized, DBTT extents may be created independently of the activation of online DBTT extension. An increase in the size of the DBTT due to BREORG is implemented in DBCOM and DBDIR by enlarging the existing DBTT. In the user realm, the DBTT is enlarged by BREORG by means of DBTT extents if the target DBTT has a total size greater than 128 PAM pages. If the target DBTT is smaller than or equal to 128 PAM pages then BREORG always implements it as a DBTT base, i.e. as a single unit. The corresponding messages inform you of the results of increases or reductions in the size of DBTTs.



The last DBTT extent is also always present in full. The number of required entries is not just rounded up to full pages, but to full DBTT extents.

Execution messages

On executing a MODIFY-RECORD-POPULATION statement, the results of the DBTT reorganization of the record type are output as follows:

```
***** BEGIN OF DBTT-SIZE-MODIFICATION AT hh:mm:ss
***** RESULTS OF DBTT-REORGANIZATION OF RECORD record-name
      NEW DBTT FIRST PAGE      : area ref - page no
      NEW DBTT LAST PAGE       : area ref - page no
      NEW NR OF EXTENTS        : number extents
      NEW DBTT SIZE            : number pages
      NEW NR OF DBTT ENTRIES   : number entries
***** END OF DBTT-SIZE-MODIFICATION AT hh:mm:ss
```

record-name

Name of the modified record type

area ref - page no

Act-key of the first or last DBTT page

number extents

New number of DBTT extents

number pages

New number of DBTT pages

number entries

New number of records

Open database (OPEN-DATABASE)

The OPEN-DATABASE statement defines the database to be processed by subsequent statements of BREORG.

This statement is not offered if the database is assigned using LINK=DATABASE.

OPEN-DATABASE
DATABASE-NAME = <dbname> ,SCHEMA-NAME = <u>*STD</u> / <schema-name> ,USER-IDENTIFICATION = <u>*OWN</u> / <userid>

DATABASE-NAME = <dbname>

Name of the database. You can only process a database that is cataloged under your own user ID. A database under a foreign user ID can only be processed from the system administrator ID TSOS.

SCHEMA-NAME = *STD

The name of the user schema that was defined for the database is used.

SCHEMA-NAME = <schema-name>

schema-name specifies the database schema for whose objects the BREORG statements are to be executed.

Possible values:

PRIVACY-AND-IQF-SCHEMA

COMPILER-SCHEMA

Name of the user schema

USER-IDENTIFICATION = *OWN

The database is located under the user's own user ID.

USER-IDENTIFICATION = <userid>

The specification of a foreign user ID is only permitted under the system administrator ID TSOS.

In order to process a database, BREORG requires information on the realms, record types, and set relations in the database. The schema name allows BREORG to access the SIA in which this information is contained and subsequently modified as required.

Reorganize CALC areas (REORGANIZE-CALC)

The REORGANIZE-CALC statement is used to reorganize CALC areas that belong to a particular record type. These are:

- areas created by means of LOCATION MODE CALC
- SEARCH KEY USING CALC on record type level
- SEARCH KEY USING CALC in singular sets in which the record type is a member

REORGANIZE-CALC

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>

RECORD-NAME = <record-name>

Name of the record type whose CALC areas are to be reorganized.

CALC-RECORD = NONE

LOCATION MODE CALC areas are not reorganized.

CALC-RECORD = list-poss(6): *WITHIN-POPULATION (...)

The LOCATION MODE CALC areas located in the specified REALM or REALMs are reorganized.

REALM = *ALL

All CALC areas are reorganized.

REALM = <realm-name>

Only the CALC area that is located in the specified REALM is reorganized.

POPULATION = *UNCHANGED

Only the probable position pointers (PPP) are updated.

A PPP update is useful for indirect LOCATION MODE CALC areas.

POPULATION = <integer 1..2147483647>

The affected CALC area is recreated. The number of specified entries is converted into a number of pages, and the number of pages is then rounded up to the next primary number (= size of the hash area in database pages).

In indirect CALC areas, the probable position pointers (PPP) are updated as well.

CALC-SEARCHKEY = NONE

The SEARCH KEY USING CALC areas that belong to the record type are not reorganized.

CALC-SEARCHKEY = list-poss(30): *KEY-POPULATION (...)

The CALC areas of the specified KEYs are reorganized.

KEY-REF = *ALL

All CALC SEARCH KEY areas are reorganized.

KEY-REF = <integer 1..65535>

Only the CALC SEARCH KEY area with the specified KEY REF is reorganized. The KEY REF can be obtained from the BPSIA log.

POPULATION = *STD

The affected CALC area is recreated. If a LOCATION MODE CALC area is present, the new size of the SEARCH KEY USING CALC area is calculated from the LOCATION MODE CALC area, or the sum of the LOCATION MODE CALC areas if distributed CALC areas are involved. If no LOCATION MODE CALC area exists, the size is based on the DBTT size of the record type.

POPULATION = *UNCHANGED

Only a probable position pointer (PPP) update is performed.

POPULATION = <integer 1..2147483647>

The affected CALC area is recreated with the specified size. The calculated number of pages is rounded up to the next primary number. In addition, the probable position pointers (PPP) are also updated.



If the same REALM or the same KEY REF is entered more than once in a list, the last specification applies.

If *ALL is entered for REALM or KEY REF in a list, the *ALL specification is assumed.

Reorganizing LOCATION MODE CALC areas

Records of a record type which is defined with LOCATION MODE IS CALC are usually stored in the database in a hash area. Their address in this hash area can be calculated by the Database Handler from the respective CALC key and the size of the area.

Only table entries consisting of the CALC key, the RSQ and the probable position pointers (PPP) are stored in the “indirect” hash area. Such indirect hash areas are generated for a LOCATION MODE CALC specification if the record type is a member in a set specified using MODE IS LIST or if PLACEMENT OPTIMIZATION or COMPRESSION FOR ALL was specified for it in the SSL.

For reorganization, BREORG calculates the size of the new hash area based on the POPULATION specification and reserves the appropriate number of pages. For each record or table entry, it then determines the address in the newly allocated hash area and relocates it there. After reorganization, the pages of the previous hash area are then available for other purposes.

This makes it possible to change the distribution of the entries in the hash area and so avoid the creation of overflow pages. You should print out an overview of the number and occupancy level of the primary pages and the overflow pages in the newly created hash area after the reorganization.

The number of entries in the DBTT of the record type in question is not altered by BREORG. However, in the case of a direct hash area, it does enter the new physical address of the respective record in column 0 of the DBTT which contains the physical addresses of all records.

Reorganizing CALC SEARCH KEY areas

A CALC SEARCH KEY area does not contain the records themselves, but the table entries. Each entry contains the CALC key, the RSQ (record sequence number) and the probable position pointer (PPP) of the corresponding record.

Three cases must be distinguished when reorganizing CALC SEARCH KEYS:

1. POPULATION = STD has been specified:
In this case, BREORG determines the new size of the CALC SEARCH KEY area itself. If LOCATION MODE IS CALC has been specified for the record type, this size - or the sum of all areas for a distributed record type is used to determine the population. Otherwise, the size of the DBTT (number of entries) is used as the value for POPULATION.

2. POPULATION = UNCHANGED has been specified:
BREORG updates the probable position pointers (PPP) of the table entries. The distribution of the table entries in the primary area and overflow pages remains the same.

Updating of the probable position pointers may, for example, become necessary if the positions of the records as members of a LIST set have been altered during database processing.

3. A new value has been specified with POPULATION = INTEGER ...:
BREORG uses the POPULATION specification to calculate the size of the new hash area and reserves an appropriate number of contiguous pages. It then relocates the table entries to the newly-assigned pages and updates their probable position pointers (PPP). Since BREORG recreates the table entries in each case, the relocation results in a new distribution over the primary area and overflow pages.

The CALC pages which were originally reserved are deallocated by BREORG during reorganization.

Since the records themselves are not relocated when CALC SEARCH KEY areas are reorganized, the information in the corresponding DBTT remains unchanged.

Determining the size of a new hash area

The number of CALC pages which BREORG newly allocates as a result of the POPULATION specification can be calculated using the following formulas:

- For an indirect hash area

- for 2 Kbytes

$$\frac{2018}{(\text{calc-key-length} + 7)} = \text{entries-per-page}^1$$

- for 4 Kbytes

$$\frac{3970}{(\text{calc-key-length} + 10)} = \text{entries-per-page}^1$$

- for 8 Kbytes

$$\frac{8066}{(\text{calc-key-length} + 10)} = \text{entries-per-page}^1$$

and

$$\frac{\text{integer} - 1}{\text{entries-per-page}} + 1 = \text{no-of-pages}^2$$

- For an direct hash area

- for 2 Kbytes

$$\frac{2018}{(\text{record-length} + \text{calc-key-length} + 15)} = \text{entries-per-page}^1$$

- for 4 Kbytes

$$\frac{3970}{(\text{record-length} + \text{calc-key-length} + 22)} = \text{entries-per-page}^1$$

¹ Round down the result

² If the result is not a primary number, it is rounded up to the next higher primary number

- for 8 Kbytes

$$\frac{8066}{(\text{record-length} + \text{calc-key-length} + 22)} = \text{entries-per-page}^1$$

and

$$\frac{\text{integer} - 1}{\text{entries-per-page}} + 1 = \text{no-of-pages}^2$$

no-of-pages

Number of pages in the hash area

calc-key-length

Length of CALC key (see [page 149](#))

integer

New quantity of data records as per POPULATION specifications

record-length

Length of the record type (user section and system section) (see [page 145ff](#))

entries-per-page

Number of entries (records or CALC table entries) per page



If the realm which is affected by the reorganization is configured with a secondary allocation = 0, e.g. because you do not want automatic realm extension, you must ensure that sufficient contiguous empty pages (at least *no-of-pages* pages) are available in this realm!

Since the old hash area can be re-used, it may be viewed as a free area.

If the realm is configured with a secondary allocation > 0, BREORG automatically extends the realm concerned when required.

¹ Round down the result

² If the result is not a primary number, it is rounded up to the next higher primary number

Execution messages

On executing a REORGANIZE-CALC statement, the results of the CALC reorganization of the record type and/or of the CALC SEARCH KEYS in the set are output as follows:

```
***** RESULTS OF CALC-REORGANIZATION OF {RECORD record-name
                                           }
                                           {SEARCH-KEY IN SET set-name }

NEW CALC BEGIN           : area ref - page no
NEW NR OF PRIMARY BUCKETS : number pages
NEW NR OF OVERFLOW BUCKETS: number pages
NR OF DATABASE ACCESSES  : number physical io
```

record-name

Name of the reorganized record type

set-name

Name of the reorganized set

area ref - page no

Act-key of the first CALC page

number pages

For NEW NR OF PRIMARY BUCKETS: New number of CALC BUCKETS

For NEW NR OF OVERFLOW BUCKETS: New number of overflow pages

number physical io

Number of physical input and output operations

Reorganize all probable position pointers (PPP) in a realm (REORGANIZE-POINTERS)

The REORGANIZE-POINTERS statement can be used to update all the probable position pointers (PPP) contained in a user realm.

REORGANIZE-POINTERS

REALM-NAME = <realname>

REALM-NAME = <realname>

Name of the realm whose probable position pointers (PPP) are to be updated. The statement can be specified several times, one after the other, for different realms in the same BREORG run.

REORGANIZE-POINTERS is significantly faster than reorganizing the probable position pointers (PPP) using the other REORGANIZE functions. If all the realms in the database are handled in this way, there is no need to format the new pages when subsequently extending the realms with MODIFY-REALM-SIZE or online via the DBH. This considerably improves performance. There is no performance gain when reducing the size of a realm.

With the REORGANIZE-POINTERS statement, BREORG uses work files for the record types and an additional work file for sorting (see [“Work files for the REORGANIZE-POINTERS statement” on page 313](#)).

Reorganize tables and set constructs (REORGANIZE-SET)

The REORGANIZE-SET statement can be used to reorganize set constructs (LIST, CHAIN, POINTER ARRAY) and any SEARCH KEY USING INDEX on set level or record type level. The reorganization of a SEARCH KEY USING INDEX on record type level occurs via the associated implicit set.

You can reorganize

- all set occurrences
- individual set occurrences which you preset via the RSQ of the owner
- areas of set occurrences which you define via an area specification of the owner RSQ

You can reorganize the following tables with the REORGANIZE-SET function to set levels:

- non-indexed and indexed pointer arrays
- non-indexed and indexed lists
- indexed sort key tables
- chains
- indexed SEARCH key tables (also duplicates tables)

You can reorganize indexed SEARCH key tables (also duplicates tables) with the REORGANIZE-SET function to record levels. The name of the implicit set must be specified.

BREORG updates all address references in the specified tables that were stored as probable position pointers. It can also optionally recreate the tables with a different occupancy level. BREORG also updates all probable position pointers in chains.

If you want to reorganize several tables in a BREORG session, you should first carry out the functions that relocate the data records. You should then update the probable position pointers which still refer to the old position of the data records.

REORGANIZE-SET

```

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>

```

SET-NAME = <set-name>

Name of the set or implicit set to be reorganized. The name of the implicit set is constructed by combining SYS_ and *record-name*.



The underscore (_) must be entered as a hyphen (-) sign!

OWNER-SELECTION = *ALL

All set occurrences are reorganized.

OWNER-SELECTION = list-poss(30): <integer 1..2147483647> / *RANGE(...)

Owner and set occurrences are reorganized.

<integer 1..2147483647>

The owners with the specified RSQs are reorganized.

***RANGE (...)**

All set occurrences whose owner RSQs lie within the specified ranges are reorganized.

FROM-RSQ = <integer 1..2147483647>

RSQ of the first owner whose SET and/or TABLE OCCURRENCE/S is/are to be reorganized.

TO-RSQ = <integer 1..2147483647>

RSQ of the last owner whose SET and/or TABLE OCCURRENCE/S is/are to be reorganized.

KEY-SELECTION = *ALL

Every SEARCH KEY USING INDEX and the set construct are reorganized.

KEY-SELECTION = list-poss(30): <integer 0..32767>

Every SEARCH KEY USING INDEX for which a KEY REF is specified is reorganized. The KEY REF can be determined from the BPSIA log.

If the set constructs CHAIN, LIST and POINTER ARRAY were not defined with a SORTED INDEXED BY specification, no KEY REF will have been entered in the BPSIA log. If this is the case, the value 0 must be specified for the KEY REF (only permitted in this situation). The set is then reorganized.

FILLING = *UNCHANGED

Only the probable position pointers (PPP) are updated in the tables or set constructs.

FILLING = <integer 1..100>

The tables are reorganized with the specified filling ratio.

The REORGANIZE-SET function reorganizes tables (ASC/DESC KEY, SEARCH KEY) and chains. In this case, reorganization means that BREORG updates the probable position pointers (PPP)

- in the sort key table entries and the SEARCH key table entries,
- in the SCD (set connection data) of records in chains (forwards/backwards chaining),
and
- in the SCD of data pages with owner links (PHYSICALLY LINKED TO OWNER)
- in the SCD of owner records with table links (WITH PHYSICAL LINK)

or sets up new tables.

The linkage of a table to the owner in ... WITH PHYSICAL LINK ... is an Act-key.

If FILLING = *integer*... has been specified, BREORG reorganizes all tables and fills the new table pages with the available updated entries up to the specified percentage. In multi-level tables, the specified percentage applies only to the main level (level 0). On level 1, 95% of the table is filled; on every other level, one table entry is left free. In addition, BREORG updates the entries in the DBTT of the owner record type (column number >0).

In the case of an ASC or DESC key table of a set with MODE IS LIST, the records themselves are in the table pages, i.e. the records are relocated when such a table is created. In this case BREORG also updates the DBTT entries in column 0 of the DBTT of the record type concerned.

The following overview shows which probable position pointers (PPP) and tables can be reorganized using the REORGANIZE SET function (see 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 22: 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 22: Overview of options in the REORGANIZE SET function

(part 2 of 2)

¹ These PPPs are standard with MODE IS CHAIN.



If an entire table is contained in one page, BREORG does not set up a new table, even if FILLING... has been specified.

When setting up a new table with multiple table pages on level 0, BREORG inserts at least two entries in each table page.

When a database is reorganized, DBTT extents may be created independently of the activation of online DBTT extension. An increase in the size of the DBTT due to BREORG is implemented in DBCOM and DBDIR by enlarging the existing DBTT. In the user realm, the DBTT is enlarged by BREORG by means of DBTT extents if the target DBTT has a total size greater than 128 PAM pages. If the target DBTT is smaller than or equal to 128 PAM pages then BREORG always implements it as a DBTT base, i.e. as a single unit. The corresponding messages inform you of the results of increases or reductions in the size of DBTTs.

Execution messages

On executing a REORGANIZE-SET statement, the results of the reorganization of a set or a table are output as follows:

```
***** RESULTS OF SET-REORGANISATION OF SET set-name
      NR OF PROCESSED TABLES      : number table occurrences
      NR OF PPP UPDATES           : number of actualized ppps
      NR OF DATABASE ACCESSES     : number physical io
```

set-name

Name of the set

number table occurrences

Number of tables processed in the set occurrences

number of actualized ppps

Number of updated probable position pointers (PPP)

number physical io

Number of physical input and output operations

Specify schema (SPECIFY-SCHEMA)

The SPECIFY-SCHEMA statement defines the schema that contains the objects for which the database is to be processed.

This statement is only offered if the database is assigned with LINK=DATABASE; otherwise, the schema name is specified in the OPEN-DATABASE statement.

If the SPECIFY-SCHEMA statement is not entered as the first statement after ALLOCATE-BUFFERPOOL, the user schema is assumed.

After the initial schema specification, the SPECIFY-SCHEMA statement is no longer offered in the SDF mask.

SPECIFY-SCHEMA
SCHEMA-NAME = * <u>STD</u> / <schema-name>

SCHEMA-NAME = *STD

The name of the user schema that was defined for the database is used (default value).

SCHEMA-NAME = <schema-name>

schema-name specifies the database schema for whose objects the BREORG statements are to be executed.

Possible values:

PRIVACY-AND-IQF-SCHEMA

COMPILER-SCHEMA

Name of the user schema

In order to process a database, BREORG requires information on the size of the realms, the record types, and the set relations in the database. The schema name allows BREORG to access the SIA in which this information is contained and subsequently modified as required.

Specify subschema (SPECIFY-SUBSCHEMA)

The SPECIFY-SUBSCHEMA statement is used to define the subschema that is required for the creation of new multi-level LIST sets with user-defined keys (i.e. SORTED INDEXED BY DEFINED KEYS).

SPECIFY-SUBSCHEMA
SUBSCHEMA-NAME = <subschema-name>

SUBSCHEMA-NAME = <subschema-name>

Name of a subschema that is included in the database and contains the key description of the LIST set.

If BREORG's REORGANIZE-SET function is to be used in reorganizing a multi-level list, the description and key of the member record type which is to be reorganized must be obtained from the associated SSIA. BREORG accesses the SSIA on the basis of the subschema name specified in the SPECIFY-SUBSCHEMA statement.



This statement may be specified more than once. It remains in effect for all following statements until the next correctly entered SPECIFY-SUBSCHEMA statement is encountered.

Undo statement (UNDO)

The UNDO statement cancels the last correctly entered statement.
Exceptions: ALLOCATE-BUFFERPOOL, END, and UNDO statements.

Each subsequent UNDO statement cancels the preceding statement in the chain.

The UNDO statement itself cannot be reversed with another UNDO statement.

UNDO

This statement has no operands.

9.5 Command sequence to start BREORG

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

```
01 [/ADD-FILE-LINK LINK-NAME=DATABASE,
      FILE-NAME=[ :catid:][ $userid.]dbname.DBDIR]]
02 [/CREATE-FILE FILE-NAME=work-file-1[,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=SCRTCH1,FILE-NAME=work-file-1,
      ACCESS-METHOD=*UPAM]
03 [/CREATE-FILE FILE-NAME=work-file-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 statements of BREORG
11 //END
```

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

02, 03 These CREATE-FILE commands can be used to create work files for the REORGANIZE-SET or REORGANIZE-CALC function (see [page 311](#)). By analogy, you can optionally create the work files for the REORGANIZE-POINTERS statement (see [page 313](#))

04 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.

- 05 BREORG can be called from any user ID. The UDS/SQL utility routine can also be started with the alias BREORG or START-UDS-REORGANIZATION.
- 08 The SPECIFY-SCHEMA statement is only offered if a command /ADD-FILE-LINK..., LINK-NAME=DATABASE was issued earlier.
- 09 Only required for the REORGANIZE-SET function when creating new multi-level LIST sets with user-defined keys (i.e. SORTED INDEXED BY DEFINED KEYS).

9.6 Examples

Example 1

The realm CLOTHING in the SHIPPING database is to be reduced by 12 database pages.

```

/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,   VERSION=02.8A00
/START-UDS-BREORG
***** START      BREORG      (UDS/SQL V2.8 0000 )   2015-06-28  11:40:58
//OPEN-DATABASE DATABASE-NAME=SHIPPING,SCHEMA-NAME=MAIL-ORDERS
//MODIFY-REALM-SIZE REALM-NAME=CLOTHING,REALM-SIZE=*RELATIVE(DIFFERENCE=-12)
//END
***** BEGIN OF REALM-SIZE-MODIFICATION      AT 11:40:58
***** RESULTS OF FPA-REORGANIZATION OF AREA CLOTHING
      NEW FPA FIRST PAGE      : NOT CHANGED
      NEW FPA LAST  PAGE      : NOT CHANGED
      NEW FPA SIZE             : NOT CHANGED
      NEW NR OF PAGES         :              42
***** END    OF REALM-SIZE-MODIFICATION      AT 11:40:58

***** DIAGNOSTIC SUMMARY OF BREORG

      NO WARNINGS
      NO ERRORS
      NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :              149
***** NORMAL END  BREORG      (UDS/SQL V2.8 0000 )   2015-06-28  11:40:58

```

Example 2

The realm ARTICLE-RLM is increased so much that the FPA base is not large enough any more. Exactly 1 FPA extent will be the result.

```
/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,   VERSION=02.8A00
/START-UDS-BREORG
***** START      BREORG      (UDS/SQL V2.8 0000 )    2015-06-28  11:40:58
//OPEN-DATABASE DATABASE-NAME=SHIPPING,SCHEMA-NAME=MAIL-ORDERS
//MODIFY-REALM-SIZE REALM-NAME=ARTICLE-RLM, REALM-SIZE=*RELATIVE(DIFFERENCE=2000)
//END
***** BEGIN OF REALM-SIZE-MODIFICATION      AT 11:40:59
***** RESULTS OF FPA-REORGANIZATION OF AREA ARTICLE-RLM
      NEW FPA FIRST PAGE      : NOT CHANGED
      NEW FPA LAST  PAGE      :           11-           78
      NEW NR OF EXTENTS      :           1
      NEW FPA SIZE           :           33
      NEW NR OF PAGES        :           2062
***** END    OF REALM-SIZE-MODIFICATION      AT 11:40:59

***** DIAGNOSTIC SUMMARY OF BREORG

      NO WARNINGS
      NO ERRORS
      NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY
***** NR OF DATABASE ACCESSES :           108
***** NORMAL END  BREORG      (UDS/SQL V2.8 0000 )    2015-06-28  11:40:59
```

Example 3

The CALC areas of the ARTICLE record type in the SHIPPING database are to be reorganized.

The SEARCH KEY USING CALC areas that belong to the ARTICLE record type are not reorganized.

```

/SELECT-PRODUCT-VERSION PRODUCT-NAME=UDS-SQL,    VERSION=02.8A00
/START-UDS-BREORG
***** START      BREORG      (UDS/SQL V2.8 0000 )    2015-06-28  11:40:59
//OPEN-DATABASE DATABASE-NAME=SHIPPING,SCHEMA-NAME=MAIL-ORDERS
//REORGANIZE-CALC RECORD-NAME=ARTICLE,CALC-RECORD=*WITHIN-POPULATION(POPULATION=200), -
//  CALC-SEARCHKEY=NONE
//END
***** BEGIN OF CALC-REORGANIZATION      AT 11:41:00
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :      5-      5
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :      6-      7
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :      7-      5
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :      8-      9
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :      9-      5
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** RESULTS OF CALC-REORGANIZATION OF RECORD ARTICLE
NEW CALC BEGIN      :     10-      4
NEW NR OF PRIMARY BUCKETS :      2
NEW NR OF OVERFLOW BUCKETS:      0
***** END    OF CALC-REORGANIZATION      AT 11:41:00

***** DIAGNOSTIC SUMMARY OF BREORG

                NO WARNINGS
                NO ERRORS
                NO SYSTEM-ERRORS

***** END OF DIAGNOSTIC SUMMARY

```

```
***** NR OF DATABASE ACCESSES :          149
***** NORMAL END   BREORG   (UDS/SQL V2.8 0000 )   2015-06-28  11:41:00
```

10 Controlling the reuse of database keys and the free place search with BMODTT

The BMODTT utility routine is used to control the free place search and the reuse of database keys.

Reusing database keys

The contents of the DBTT entries of deleted records differ from those for which a record has never been stored. By default or in the case of the BMODTT statement REUSE, DBH and BINLOAD make no distinction between the two types of DBTT entries when records are stored. Only after a BMODTT statement KEEP the database keys of deleted records are not used. The BMODTT statement REMOVE enables you to equate (once) all database keys of deleted records to those of records which have never been used, as a result of which the memory of all the records which have been deleted in the past is lost; the existing option concerning their reusability by the BMODTT or BOUTLOAD utility routine (REUSE, KEEP) remains unchanged.

The information on deleted records is not shown by BOUTLOAD and is also lost when the BOUTLOAD statement EXPORT-RECORD or REMOVE-RECORD with RECORD-NAME=*ALL is used as the database is reformatted in this case. The information on the locked DBTT entries is not transferred to the new database when the database is converted using BPGSIZE, either.

The 'NO REUSE' column in the BPSIA log provides you with information on the fundamental reusability of database keys.

Free place search

The SET and RESET statements can be used to control how the free place search is performed, i.e. if the primary task is to optimally utilize the space (SET) or if the 'ATTACHED' and 'PLACEMENT OPTIMIZATION' location specifications set via SSL are to be fulfilled (RESET, default setting).

10.1 System environment

The database administrator is not allowed to invoke BMODTT in the course of an updating session.

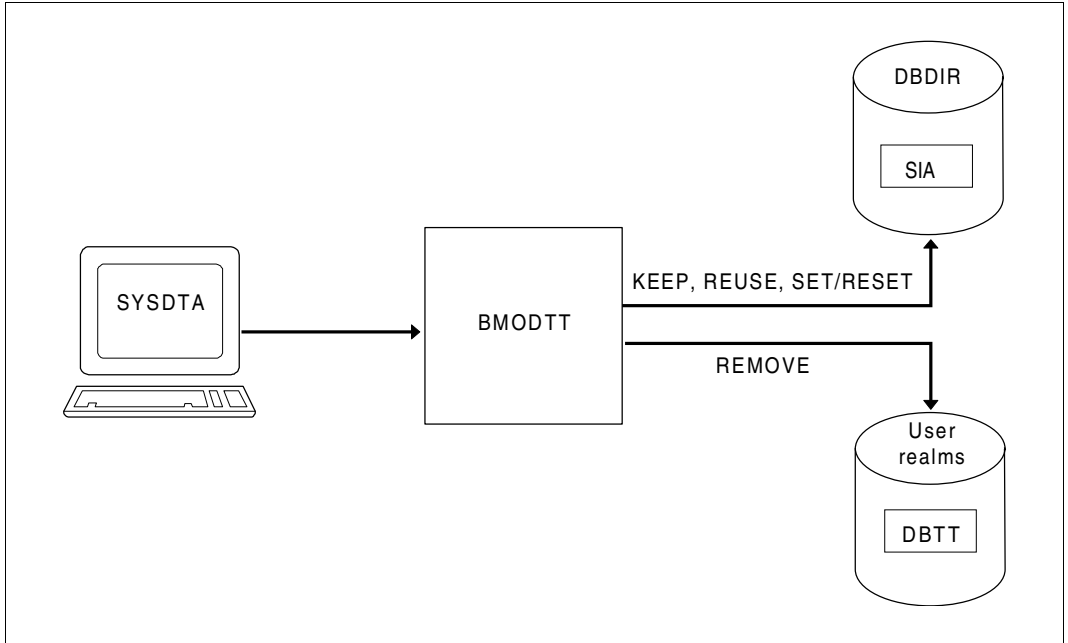


Figure 19: BMODTT system environment

At startup BMODTT takes into account any assigned UDS/SQL pubset declaration (see the “[Database Operation](#)” manual, Pubset declaration job variable). Faulty assignment leads to the program aborting.

10.2 BMODTT statements

Statement	Meaning
KEEP	Lock deallocated database keys
REMOVE	The locked database keys are released for one-time reuse
RESET	Free place search from the end of the occupied parts of the realm to the beginning
REUSE	Deallocate database keys for reuse
SET	Free place search from the beginning of the realm

Table 23: BMODTT statements

These statements are explained below in combined formats that reflect their structure.

$\left\{ \begin{array}{l} \text{KEEP} \\ \text{REMOVE} \\ \text{REUSE} \end{array} \right\}$	$\left\{ \begin{array}{l} \text{DBKEY OF RECORD} \\ \text{OF RECORD} \end{array} \right\}$	$\left\{ \begin{array}{l} \text{rec-name-1[, rec-name-2]...} \\ \text{*ALL[EXCEPT rec-name-1[, rec-name-2]...]} \end{array} \right\}$
--	--	--

KEEP Deallocated database keys are locked and cannot be reused.

REMOVE

The locked database keys are released for one-time reuse with the option of their reusability retained. The highest database key used for the specified record types is determined again and stored in the appropriate SIA.

Furthermore, the level at which a STORE statement or BINILOAD assigns database keys for the record types concerned is set ahead of the first free entry of the DBTT or on the entry with RSQ 1 if the latter is free. It is not possible to set the level separately without releasing the locked database keys.

REUSE

Deallocated database keys can always be reused. This is the default setting before using BMODTT.

rec-name-1[,rec-name-2]...

List of affected record types.

*ALL Affects all record types.

*ALL EXCEPT *rec-name-1[,rec-name-2]...*

Affects all record types in the database, except for those listed after EXCEPT.

$\left. \begin{array}{l} \text{SET} \\ \text{RESET} \end{array} \right\}$	REUSE-FREE-SPACE OF REALM	$\left\{ \begin{array}{l} \text{realm-1[,realm-2]...} \\ \text{*ALL[EXCEPT realm-1[,realm-2]...]} \end{array} \right\}$
---	---------------------------	--

SET In the free place search, 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.

This is the default setting before using the BMODTT utility routine.

realm-1[,realm-2],...

List of affected realms.

***ALL** Affects all realms of the database.

***ALL EXCEPT** *realm-1[,realm-2],...*

Affects all realms of the database, except for those listed after EXCEPT.

10.3 Command sequence to start BMODTT

The command sequence described here is based on the assumption that UDS/SQL was installed with IMON (see the section “START commands of the UDS/SQL programs” in chapter 2 of the “[Creation and Restructuring](#)” manual).

```
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
```

- 01 In this case specifying *:catid:* is permitted (see the “[Database Operation](#)” manual).
- 02 The version of the utility routine is selected.
Specification of the version is generally recommended, since several UDS/SQL versions can be installed in parallel.
- 03 BMODTT may only be invoked under the database administrator’s user ID. The UDS/SQL utility routine can also be started with the alias BMODTT.
- 04 The default value (REUSE) is reset whenever BCHANGE or BALTER is used for restructuring.

The BMODTT utility routine is restartable.

Glossary

This Glossary contains the definitions of some of the important terms and concepts used in the UDS/SQL manuals. Terms that appear in *italics* within a particular definition have also been defined in this Glossary. In cases where two or more terms are used synonymously, a “See” reference points to the more commonly used term in these manuals.

A

access, contending

See *contending access*.

access, direct

See *direct access*.

access, sequential

See *sequential access*.

access authorization

The rights of a specified user group with regard to access to the *database*. Access rights are defined during live database operation using ONLINE-PRIVACY utility routine or, in offline mode, using the BPRIVACY utility routine.

access path

Means of finding a certain subset of all *records* qualified by a search query, without having to carry out a sequential search of the whole *database*.

access rights

Right of access to a *database* as defined in the BPRIVACY utility routine.

access type

Type of access, e.g. read, update etc.

act-key

(actual key) Actual address of a *page*, consisting of *realm number* and *page number*.

act-key-0 page

First *page* of a *realm*; contains general information on the realm such as

- when the realm was created,
- when the realm was last updated,
- *internal version number* of the realm,
- *system break information*
- if applicable, *warm start* information.

act-key-N page

Characteristic page of a *realm*, with the highest *page number*.

Copy of the *act-key-0 page*.

address, physical

See *act-key* or *probable position pointer (PPP)*.

administrator task

Task of the *independent DBH*; The *database administrator* can control execution of the *independent DBH* via this task.

AFIM

See *after-image*.

after-image

Modified portion of a *page* **after** its content has been updated.

The *DBH* writes after-images to the *RLOG file* as well as the *ALOG file*.

after-image, ALOG file

The after-images are written to the *ALOG file* when the *ALOG buffer* is full. The purpose of the after-images in the *ALOG file* is to secure the data contained in the database and thus they must be maintained for a long period of time. They are used to reconstruct an original database or update a *shadow database*.

after-image, RLOG file

After-images are logged in the *RLOG file* **before** the updates are applied to the *database*. The after-images held in the *RLOG file* are required for *warm start* only. They are thus periodically overwritten.

ALOG file

File for securing the data contained in the database in the long term; see *after-image*.

ALOG sequence number

See *sequence number*.

anchor record

Record automatically created by UDS/SQL as *owner record* for *SYSTEM sets*. It cannot contain any *items* defined with the *schema DDL* and cannot be accessed.

application

Realization of a job in one or several *user programs* working with UDS/SQL *databases*.

application program (AP)

E.g. *COBOL DML* program or *IQS*.

area

See *realm*.

ascending key (ASC key)

Primary key of a *set*. Defines the sequence of *member records* in the *set occurrences* by ascending key values.

authorization

Identification used for user groups.

authorized users

Specified user groups who are authorized to access the *database*.

automatic DBTT extension

Some utility routines automatically extend the number of records possible for a record type if too few are available; no separate administration is required to do this.

See also *online DBTT extension*.

automatic realm extension

Some utility routines automatically extend realms when insufficient free space is available; no separate administration is required to do this.

See also *online realm extension*.

B

backup database

See *shadow database*.

base interface block (BIB)

(Base Interface Block) Standard interface between UDS/SQL and each individual user; it contains, among other things, the *RECORD AREA* (user records as defined in the *subschema*).

before-image

Copy of a *page* taken before its contents are updated.

The *DBH* writes before-images to the *RLOG files* during database operation before the updates are applied to the *database*. A prerequisite is that the RLOG files exist.

BFIM

See *before-image*.

BIB

See *base interface block*.

buffer pool

See *system buffer pools* and *exclusive buffer pool*.

C

CALC key

Key whose value is converted into a relative *page number* by means of a *hash routine*.

CALC page

Page of a *hash area*.

CALC SEARCH key

Secondary key. Used as *access path* for *direct access* via *hash routine*.

CALC table

Table in the direct/indirect *CALC* page whose entries point to the stored records.

Each line contains:

- the *CALC* key,
- the *record sequence number*
- the displacement to the related *page index entry* (direct *CALC* page) or the *probable position pointer* (indirect *CALC* page).

CALL DML

DML that is called by various programming languages (Assembler, COBOL, FORTRAN, PASCAL, PL/1) via the *CALL* interface.

catalog identifier

Name of the public volume set (PVS) under which the BS2000 UDS/SQL files are stored. The catalog identifier is part of the database or file name and must be enclosed in colons: “:catid:”.

chain

Storage mode for a *set occurrence* in which every *record* contains a pointer to the subsequent record.

Character Separated Values (CSV)

Output format in which the values are separated by a predefined character.

checkpoint

Consistency point, at which the *ALOG* file was changed and to which it is possible to return at any time using *BMEND* utility routine

check records

Elements which provide information for checking the database. They vary in length from 20 to 271 bytes.

CHECK-TABLE

Check table produced by the *DDL* compiler during *Subschema DDL* compilation, and used by the COBOL compiler and *CALL DML* to check whether the *DML* statements specified in the *application program* are permitted. It is part of the *COSSD* or *SSITAB* module.

clone pair, clone pubset, clone session, clone unit

A clone unit is the copy of an (original) unit (logical disk in BS2000) at a particular time ("Point-in-Time copy"). The TimeFinder/Clone component creates this copy optionally as a complete copy or as a "snapshot".

After they have been activated, the unit and clone unit are split; applications can access both.

The unit and clone unit together form a clone pair. TimeFinder/Clone manages this pair in what is known as a clone session.

If clone units exist for all units of a pubset, these clone units together form the clone pubset.

Details of this are provided in the manual "[Introduction to System Administration](#)".

COBOL DML

DML integrated in the COBOL language.

COBOL runtime system

Runtime system; sharable routines selected by the COBOL compiler (COBOL2000 or COBOL85) for the execution of complex statements.

COBOL Subschema Directory (COSSD)

Provides the COBOL compiler with subschema information for compilation of the DB *application programs*.

common memory

Shareable memory area used by several different tasks. In UDS/SQL, it always consists of the *common pool* and the *communication pool* and, depending on the application, the *SSITAB pool* (see *SSITAB module*) if *CALL DML* is used.

If UDS-D is used, it also consists of the *distribution pool* and the *transfer pool*.

common pool

Communication area of the *independent DBH*. Enables *DBH* modules to communicate with each other. Contains, among other things, an input/output buffer for *pages (buffer pools)*.

communication partners

Tasks or data display terminals.

communication pool

Communication area of the *independent DBH* for *application programs*. One of its functions is to store base interface blocks (*BIB*).

compatible database interface (KDBS)

see *KDBS*

compiler database

The *realms* and files of the *database* which are required by the UDS/SQL compiler. They are

- *DBDIR* (*Database Directory*)
- *DBCOM* (*Database Compiler Realm*)
- *COSSD* (*COBOL Subschema Directory*).

COMPILER-SCHEMA

UDS/SQL-internal *schema* of the *compiler database*.

COMPILER-SUBSCHEMA

UDS/SQL-internal *subschema* of the *compiler database*.

compound key

Key consisting of several *key items*.

compression

Only the filled *items* of a *record* are stored (see *SSL* clause *COMPRESSION*).

configuration

See *DB configuration*.

configuration user ID

User ID in which the *database administrator* starts the *DBH*.

configuration name

Freely selectable name of the *database configuration* for a particular *session*. The *DBH* uses it to form:

- the name of the *Session Log File*,
- the names of the *DB status file* and its backup copy,
- the names of the *RLOG files*,
- the names of the temporary *realms*,
- the names of session job variables,
- the *event names* of *PI eventing*,
- the *DCAM application* name for the administration,
- the names of the *common pools*
- the names of the dump files.

connection module

Module that must be linked into every UDS/SQL *application program* and which establishes the connection with the *DBH*.

consistency

State of the database without conflicts in the data stored in it.

consistency, logical

State of the database in which the stored data has no internal conflicts and reflects the real-world situation.

consistency, physical

State of the database in which the stored data is consistent with regard to correct physical storage, *access paths* and description information.

consistency, storage

See *physical consistency*.

consistency error

A violation of the *physical consistency* of the stored data.

consistency point

Point (in time) at which the *database* is consistent, i.e. all modifying transaction have been terminated and their modifications have been executed in the database.

consistency record

Administration record with consistency time and date stamps in the *DBDIR*. For an update in a *realm* the *DBH* enters the date and time in the consistency record and in the updated realm. When realms or *databases* are attached for a *session*, the *DBH* uses this time stamp to check the consistency of the realms within each database.

contending access

Different *transactions* attempting to access a *page* simultaneously.

conversation

SQL-specific administration data is retained across transaction boundaries in an *SQL* application. This kind of data administration unit is called a conversation. In openUTM such an administrative unit is also called a service.

copy

See *database copy*.

COSSD

See *COBOL Subschema Directory*.

CRA

(Current Record of Area) *Record* which is marked in the *currency table* as the current record of a particular *realm* (area).

CRR

(Current Record of Record) *Record* which is marked in the *currency table* as the current record of a particular *record type* (Record).

CRS

(Current Record of Set) *Record* which is marked in the *currency table* as the current record of a particular *set*.

CRU

(Current Record of Rununit) *Record* which is marked in the *currency table* as the current record of the *processing chain*.

CSV

see *Character Separated Values*

currency table

The currency table contains:

- CURRENT OF AREA table (table of CRAs),
- CURRENT OF RECORD table (table of CRRs) and
- CURRENT OF SET table (table of CRSs).

CURRENT OF AREA table

See *currency table*.

CURRENT OF RECORD table

See *currency table*.

CURRENT OF SET table

See *currency table*.

D**DAL**

(Database Administrator Language) Comprises the commands which monitor and control a *session*.

data backup

Protection against loss of data as a result of hardware or software failure.

data deadlock

See *deadlock*.

data protection (privacy)

Protection against unauthorized access to data. Implemented in UDS/SQL by means of the schema/subschema concept and access authorization. *Access rights* are granted by means of the BPRIVACY utility routine.

database (DB)

Related data resources that are evaluated, processed and administered with the help of a *database system*.

A database is identified by the database name.

An UDS/SQL database consists of the *user database* and the *compiler database*. To prevent the loss of data, a *shadow database* may be operated together with (i.e. parallel to) the original database.

database administrator

Person who manages and controls *database* operation. The DB administrator is responsible for the utility routines and the Database Administrator Language (*DAL*).

database copy

Copy of a consistent *database*; may be taken at a freely selectable point in time.

database compiler realm (DBCOM)

Stores information on the *realms*, *records* and *sets* defined by the user in the *Schema DDL* and *Subschema DDL*.

database copy update

Updating of a *database copy* to the status of a *checkpoint* by applying the appropriate *after-images*.

database directory (DBDIR)

Contains, among other things, the *SIA*, all the *SSIAs* and information on *access rights*.

database job variable

Job variable in which UDS/SQL stores information on the status of a *database*.

database key (DB key)

Key whose value represents a unique identifier of a *record* in the *database*. It consists of the *record reference number* and the *record sequence number*. The database key values are either defined by the database programmer or automatically assigned by UDS/SQL.

database key item

Item of type DATABASE-KEY or DATABASE-KEY-LONG that is used to accommodate *database key* values.

Items of type DATABASE-KEY and DATABASE-KEY-LONG differ in terms of the item length (4 bytes / 8 bytes) and value range.

DATABASE-KEY item

See *database key item*.

DATABASE-KEY-LONG item

See *database key item*.

database page

See *page*.

DATABASE-STATUS

Five-byte item indicating the database status and consisting of the *statement code* and the *status code*.

database system

Software system that supports all tasks in connection with managing and controlling large data resources. The database system provides mechanisms for stable and expandable data organization without redundancies. They allow many users to access *databases* concurrently and guarantee a consistent data repository.

DB status file

(database status file) Contains information on the most recently reset *transactions*.

openUTM-S or, in the case of distributed processing, UDS-D/openUTM-D needs this information for a *session restart*.

DB configuration

(database configuration) The *databases* attached to a *DBH* at any one point during *session* runtime. As the result of *DAL* commands or *DBH* error handling, the database configuration can change in the course of a session.

At the *session start*, the DB configuration may be empty. Databases can be attached with *DAL* commands after the start of the session. They can also be detached during the session with *DAL* commands.

DBC.COM

See *database compiler realm*.

DBDIR

See *database directory*.

DBH

Database Handler: program (or group of programs) which controls access to the *database(s)* of a *session* and assumes all the attendant administrative functions.

DBH end

End of the *DBH* program run. DBH end can be either a *session end* or a *session abort*.

DBH, independent

See *independent DBH*.

DB key

See *database key*.

DBH, linked-in

See *linked-in DBH*.

DBH load parameters

See *load parameters (DBH)*.

DBH start

Start of the *DBH* program run. DBH start can be either a *session start* or a *session restart*.

DBTT

(Database Key Translation Table) Table from which UDS/SQL can obtain the *page address (act-key)* of a *record* and associated tables by means of the database key value.

The DBTT for the SSIA-RECORD consists only of the DBTT base. For all other record types, the DBTT consists of a base table (DBTT base) and possibly of one or more extension tables (DBTT extents) resulting from an online DBTT extension or created by BREORG.

DBTT anchor page

Page lying within the realm of the associated DBTT in which the DBTT base and DBTT extents are administered. Depending on the number of DBTT extents multiple chained DBTT anchor pages may be required for their administration.

DBTT base

see *DBTT*

DBTT extent

see *DBTT*

DBTT page

Page containing the *DBTT* or part of the *DBTT* for a particular *record type*.

DCAM

Component of the TRANSDATA data communication program.

DCAM application

Communication application using the *DCAM* communication method. A DCAM application enables communication between

- a DCAM application and terminals.
- different DCAM applications within the same or different hosts, and with *remote configurations*.
- a DCAM and a openUTM application.

DDL

(Data Description Language) Formalized language for defining the logical data structure.

deadlock

Mutual blocking of *transactions*.

A deadlock can occur in the following situations:

- Data deadlock: This occurs when *transactions* block each other with *contending access*.
- Task deadlock: This occurs when a *transaction* that is holding a lock cannot release it, since no openUTM task is free. This deadlock situation can only occur with UDS/SQL-openUTM interoperation.

descending key (DESC key)

Primary key of a set. Determines the sequence of *member records* in the *set occurrences* to reflect descending key values.

direct access

Access to a *record* via an item content. UDS/SQL supports direct access via the *database key*, *hash routines* and *multi-level tables*.

direct hash area

See *hash area*.

distributed database

A logically connected set of data resources that is distributed over more than one UDS/SQL configuration.

distributed transaction

Transaction that addresses at least one *remote configuration*. A transaction can be distributed over:

- UDS-D,
- openUTM-D,
- UDS-D and openUTM-D.

distribution pool

Area in the *independent DBH* used for communication between *UDSCT*, *server tasks*, *user tasks* and the *master task* with regard to UDS-D-specific data. The distribution pool contains the *distribution table* and the UDS-D-specific system tables.

distribution table

Table created by UDS-D using the input file assigned in the *distribution pool*. With the aid of the distribution table, the distribution component in the *user task* decides whether a *processing chain* should be processed locally or remotely. Assigned in the distribution table are:
subschema - database
database - configuration
configuration - host computer.

DML

Data Manipulation Language: language for accessing a UDS/SQL *database*.

dummy subtransaction

A primary *subtransaction* is created by UDS-D when the first *READY* statement in a *transaction* addresses a *remote database*.

A dummy subtransaction is used to inform the *local configuration* of the transaction so that the *database* can be recovered following an error.

duplicates header

Contains general information on a *duplicates table* or a *page* of a *duplicates table*, i.e.

- chaining reference to the next and previous *overflow page*
- the number of free bytes in the page of the *duplicates table*.

duplicates table

Special *SEARCH-KEY table* in which a key value which occurs more than once is stored only once.

For each key value, the duplicates table contains:

- a table index entry with the key value and a pointer to the associated table entry
- a table entry (DB key list), which can extend over several pages, containing the *record sequence numbers* of the *records* which contain this key value.

duplicates table, main level

Main level, Level 0. Contains a table index entry and the beginning of the associated table entry (DB key list).

dynamic set

Set which exists only for the life of a *transaction* and which stores *member records* retrieved as result of search queries.

E

ESTIMATE-REPORT

Report produced after BGSIA run. Used to estimate the size of the *user realms*.

event name

Identification used in eventing.

exclusive buffer pool

Buffer which, in addition to the *system buffer pools*, is used exclusively for buffering *pages* of the specified *database*.

F

foreign key

Record element whose value matches the primary key values of another table (UDS/SQL *record type*). Foreign keys in the sense of UDS/SQL are qualified as "REFERENCES owner record type" in the member record type of a set relationship in the BPSQLSIA protocol.

FPA

See *free place administration*.

FPA base

See *free place administration*.

FPA extent

See *free place administration*.

FPA page

Free place administration page.

free place administration (FPA)

Free space is managed both at realm level (*FPA pages*) and at page and table level. Free place administration of the pages is carried out in a base table (FPA base) and possibly in one or more extension tables (FPA extents) created by means of an online realm extension or BREORG.

function code

Coding of a *DML* statement; included in information output by means of the *DAL* command DISPLAY or by UDSMON.

G**group item**

Nameable grouping of *record elements*.

H**hash area**

Storage area in which UDS/SQL stores data and from which it retrieves data on the basis of key values which are converted into relative *page numbers*. A hash area may contain the *record* addresses as well as the records themselves. A *direct hash area* contains the records themselves; an *indirect hash area*, by contrast, contains the addresses of records stored at some other location.

hash routine

Module which performs *hashing*.

hashing

Method of converting a key value into a *page address*.

HASHLIB

Module library for the storage of *hash routines* for one *database*.

I

identifier

Name allocated by the database designer to an *item* that UDS/SQL creates automatically. UDS/SQL adapts item type and length to the specified item usage.

implicit set

SYSTEM set created by UDS/SQL when a *SEARCH key* is defined at record type level.

inconsistency

State of the database in which the data values contained in it are inconsistent.

independent DBH

Independent program system enabling more than one user to access a single *database (mono-DB operation)* or several databases (*multi-DB operation*) simultaneously. The independent DBH is designed as a task family, consisting of

- a *master task (UDSSQL)*
- one or more *server tasks (UDSSUB)*
- an *administrator task (UDSADM)*

index level

Hierarchy level of an *index page*.

index page

Page in which the highest (lowest) key values of the next-lower level of an indexed table are stored.

INDEX search key

Secondary key. Used as *access path* for *direct access* via a *multi-level table*.

indirect hash area

See *hash area*.

integrity

State of the database in which the data contained in it is complete and free of errors.

- entity integrity
- *referential integrity*
- user integrity

interconfiguration

Concerning at least one *remote configuration*.

interconfiguration consistency

A *distributed transaction* that has caused updates in at least one *remote configuration* is terminated in such a way that the updates are either executed on the *databases* in each participating *DB configuration* or on none at all.

Interconfiguration consistency is assured by the *two-phase commit protocol*.

interconfiguration deadlock

Situation where *distributed transactions* are mutually locked due to *contending accesses*.

interface

In software: memory area used by several different programs for the transfer of data.

internal version number

Each *realm* of the *database*, including *DBDIR* and *DBCOM*, has an internal version number which the utility routines (e.g. BREORG, BALTER) increment by one whenever a realm is updated. This internal version number is kept in the *act-key-0 page* of the realm itself and also in the PHYS VERSION RECORD in the *DBDIR*.

item

Smallest nameable unit of data within a *record type*. It is defined by item type and item length.

K**KDBS**

Compatible database interface. Enables programs to be applied to applications of *DB systems* by different manufacturers.

key

Item used by the database programmer for *direct access* to records; an optimized *access path* is provided for the key by UDS/SQL in accordance with the *schema* definition.

key, compound

Key consisting of several *key items*.

key item

Item defined as a *key* in the *schema*.

key reference number

Keys are numbered consecutively in ascending order, beginning at 1.

L

linked-in control system

UDS/SQL component for *linked-in DBH*, responsible for control functions (corresponds to the *subcontrol system* of the *independent DBH*).

linked-in DBH

Module linked in to or dynamically loaded for the current DB *application program* and controlling access to a single *database (mono-DB operation)* or several databases simultaneously (*multi-DB operation*).

list

Table containing the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

In a distributable list the data pages which contain the member records (level 0 pages) can be distributed over more than one realm. The pages containing the higher-ranking table levels all reside in one realm (table realm of a distributable list).

load parameters (DBH)

Parameters requested by the *DBH* at the beginning of the *session*. They define the basic characteristics of a session.

local application program

An *application program* is local with regard to a *configuration* if it was linked to the configuration using `/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name`

local configuration

The *configuration* assigned to an *application program* before it is called using /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name.

The application program communicates with the local configuration via the *communication pool*. The local configuration is in the same host as the application program.

local database

Database in a *local configuration*.

local distribution table

A *distribution table* is considered local to a *DBH* if it is held in the *DBH's distribution pool*.

local host

Host computer containing the *application program*.

local transaction

Transaction that only addresses the *local configuration*.

logging

Recording of all updates in the *database*.

logical connection

Assignment of two *communication partners* that enables them to exchange data. *DCAM applications* communicate via logical connections.

M

main reference

In the *DBH* the main reference is used to manage the resources required for processing a transaction's requests, including those for transferring the requests from the application program to the *DBH* and back.

mainref number

Number assigned to the *transaction* at *READY*. This number is unique only at a given time; at the end of the transaction, it is assigned to another transaction.

master task

Task of the *independent DBH* in which the *UDSQL* module executes. Controls the start and end of a *session* and communicates with the *database administrator* directly or via the *administrator task*.

member

See *member record* or *member record type*.

member, AUTOMATIC

Record is inserted at storage time.

member, MANDATORY

Record cannot be removed.

member, MANUAL

Record is not inserted automatically at storage time.

member, OPTIONAL

Record can be removed.

member record

Lower-ranking *record* in a *set occurrence*.

member record type

Lower-ranking *record type* in a *set*.

mono-DB configuration

Type of configuration where only one *database* takes part in a *session*.

mono-DB operation

Mode of *database* operation where the *DBH* uses only one *database* of a *configuration*.

multi-DB configuration

Type of configuration where several *databases* take part in a *session*.

multi-DB operation

Mode of *database* operation where the *DBH* uses several *databases* of a *configuration*.

multi-DB program

Application program that addresses more than one *database*. The *databases* may be part of one or more *mono-DB* or *multi-DB configurations*.

multi-level table

SEARCH KEY table which contains a line for each *record* of the associated *record type* or each *member record* of the *set occurrence*, as appropriate. Each line comprises the key value of the record and the record pointer. It is also referred to as an indexed table.

multithreading

A mechanism that enables the *DBH* to fully exploit the CPU.

Multithreading means that the DBH processes several jobs concurrently by using so-called threads. Each thread has information on the current status of a particular job stored in it. When a job needs to wait for the completion of an I/O operation, DBH uses the CPU to process some other job.

N

network

All computers linked via TRANSDATA.

O

OLTP

(Online Transaction Processing) In an OLTP application, a very large number of users access the same programs and data. This usually occurs under the control of a transaction monitor (TP monitor).

online backup

If AFIM logging is active, the *database* can be saved during a session. The ability to save a database online is determined with the BMEND utility routine.

online DBTT extension

Extension during ongoing database operation of the number of possible records of a record type. The DAL commands ACT DBTT-INCR, DEACT DBTT-INCR, DISPLAY DBTT-INCR and EXTEND DBTT can be used to administer the online extension of DBTTs.

See also *automatic DBTT extension*.

online realm extension

Extension of *user realms* and *DBDIR* in ongoing database operation. The DAL commands ACT INCR, DEACT INCR, DISPLAY INCR, EXTEND REALM and REACT INCR are provided for administering the online extensibility of realms.

See also *automatic realm extension*.

open transaction

Transaction which has not been closed with FINISH or FINISH WITH CANCEL, or with COMMIT or ROLLBACK.

openUTM

(universal transaction monitor) Facilitates the creation and operation of transaction-oriented applications.

operator task (OT)

See *master task*

original database

The term “original database” refers solely to the naming of the database files (*dbname.dbfile*), not to the status of the database content (see also *shadow database*).

overflow page

Page in hash areas and duplicates tables for storing data that does not fit in the primary page. Their structure is the same as that of the pages of the hash area or duplicates table in question.

owner

See *owner record* or *owner record type*.

owner record

Higher-ranking *record* in a *set occurrence*.

owner record type

Higher-ranking *record type* in a *set*.

P**page**

Physical subunit of a *realm*. UDS/SQL identifies pages by means of unique keys (*act-key*).

The length of a page may be optionally 2048, 4000 or 8096 bytes. All pages within a database must have the same length. Pages with a length of 4000 or 8096 bytes are embedded in a *page container*.

page address

In a page address, a distinction is made between the current address of a *page*, i.e. the *act-key*, and the probable address of a page, the *probable position pointer (PPP)*.

page container

Pages with a length of 4000 or 8096 bytes are embedded in a so-called page container, which consists of a 64-byte header that precedes the page and a 32-byte trailer at the end of the page.

page header (page info)

The first 20 bytes of a database *page* (except for the *FPA* and *DBTT pages* with a length of 2048 bytes). They contain:

- the *act-key* of the *page* itself,
- the number of *page index entries*
- the length and displacement of the bytes which are still vacant in this page.
- the page type (*ACT-Key-0 page*, *FPA page*, *DBTT page*, *DBTT anchor page*, normal data page or *CALC page*)

page index entry

Indicates the position of a *record* within a *page*.

page number

In each *realm* the *pages* are numbered consecutively in ascending order starting starting from 0. The page number is part of the *page address*.

Page number = PAM page number -1 for databases with a page length of 2048 bytes

Page number = (PAM page number-1) / 2 for databases with a page length of 4000 bytes

Page number = (PAM page number-1) / 4 for databases with a page length of 8096 bytes.

password for UDS/SQL files

Password serving to protect the files created by UDS/SQL (default: C'UDS.'). The *DB administrator* can define other passwords with PP CATPASS or MODIFY-FILE-ATTRIBUTES.

pattern

Symbolic representation of all possible *item* contents, used at item definition.

pattern string

String defining a *pattern*.

PETA

Preliminary end of transaction: UDS-D or openUTM-D statement that causes a preliminary transaction end.

The PETA statement belongs to the first phase of the *two-phase commit protocol* which terminates a *distributed transaction*.

The PETA statement stores the following information failproof in the *RLOG file* of the local *DBH*:

- each updated *page*
- rollback and locking information
- the names of all participating *configurations*.

This information is required for any future *warm start*.

pointer array

Table of pointers to the *member records* of a *set occurrence*. Used for *sequential* and *direct access* to member records.

PPP

See *probable position pointer (PPP)*.

prepared to commit (PTC)

Part of the *two-phase commit protocol*:

State of a *subtransaction* after execution of a *PETA* statement and before receipt of the message that the complete *transaction* is to be terminated with FINISH or FINISH WITH CANCEL.

primary key

Distinguished from *secondary keys* for reasons of efficiency. Usually a unique identifier for a *record*.

primary key (DDL)

The *key* of a *record type* which is defined by means of "LOCATION MODE IS CALC" or the *key* of an order-determining *key* of a set occurrence which is defined by means of "ORDER IS SORTED [INDEXED]". Also used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

primary key (SQL)

In the broader sense (SQL), a *record element* uniquely identifying a record.

In UDS-SQL, the database key of an owner record output as the "PRIMARY KEY" in the BPSQLSIA log (see also *foreign key*).

A *record element* which uniquely identifies a record is flagged as "UNIQUE" in the BPSQLSIA log unless it is the aforementioned "PRIMARY KEY".

primary subtransaction

Subtransaction that runs in the local configuration.

The primary subtransaction is opened by the first *READY* statement in a *transaction* on a *local database*.

If the first *READY* statement addresses a *remote database*, UDS-D generates a *dummy subtransaction* as the primary subtransaction.

PRIVACY-AND-IQF SCHEMA

UDS/SQL-internal *schema* for protection against unauthorized access.

PRIVACY-AND-IQF SUBSCHEMA

UDS/SQL-internal *subschema* for protection against unauthorized access.

probable position pointer (PPP)

Probable address of a *page*, comprising *realm number* and *page number*.

UDS/SQL does not always update probable position pointers (PPP) when the storage location of data is changed.

processing chain

Sequence of DML statements applied to a *database* within a *transaction*.

PTC state

See *prepared to commit*.

pubset declaration

See *UDS/SQL pubset declaration*

pubset declaration job variable

Job variable in which a *UDS/SQL pubset declaration* is specified.

P1 eventing

Manner in which tasks communicate with each other.

R

READY

Start of a *transaction* or a *processing chain* in *COBOL DML* programs.

READYC

Start of a *transaction* or a *processing chain* in *CALL DML* programs.

realm

Nameable physical subunit of the *database*. Equivalent to a file. Apart from the *user realms* for user data there are also the realms *DBDIR* and *DBCOM*, which are required by UDS/SQL.

realm configuration

Comprises all the database *realms* taking part in a *session*.

realm copy

See *database copy*.

realm reference number

Realms are numbered consecutively in ascending order, starting with 1. The realm reference number (area reference) is part of the *page address*.

reconfiguration

Regrouping of databases in a *DB configuration* after a *session abort*. A prerequisite for reconfiguration is that the *SLF* has been deleted or that its contents have been marked as invalid.

record

Single occurrence of a *record type*; consists of one item content for each of the *items* defined for the record type and is the smallest unit of data managed by UDS/SQL via a unique identifier, the *database key*.
The reserved word **RECORD** is used in DDL and SSL syntax to declare a record type.

record address

Address of the page containing the *record*. See *page address*.

RECORD AREA

Area in the *USER WORK AREA (UWA)* which can be referenced by the user. The record area contains the *record types* and the implicitly defined items (IMPLICITLY-DEFINED-DATA-NAMES) of the database such as the AREA-ID items of the WITHIN clauses of the schema. The length of the record area is essentially defined by the record types contained in it.

record element

Item, vector or group item.

record hierarchy

Owner/member relationship between *record types*: the *owner record type* is the higher-ranking part of the relationship; the *member record type* is the lower-ranking part.

REC-REF

See *record reference number*.

record reference number

Record types are numbered consecutively in ascending order, starting at 1. The record reference number is part of the *database key*.

record SEARCH KEY table

SEARCH KEY table for selection of a *record* from a *record type*.

record sequence number (RSQ)

The record sequence number can be assigned by the database programmer; if not, UDS/SQL numbers the *records* of a *record type* contiguously in ascending order, in the sequence in which they are stored; numbering starts at 1. The record sequence number is part of the *database key*.

record type

Nameable grouping of *record elements*.

record type, linear

Record type that is neither the *owner* nor the *member* of a set (corresponds to record types of a conventional file).

referential integrity

Integrity of the relationships between tables (UDS/SQL *record types*).

remote application program

Application program that is not local with regard to a particular *configuration*.

remote configuration

DB-configurations that are not assigned to the *application program* via /SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=conf-name but via the *distribution table* once the application program is running. The *connection module* of the application program communicates with the remote configurations via *DCAM applications*.

Remote configurations can be situated on *local* or *remote* hosts.

remote database

Database in a *remote configuration*.

remote host

Host computer that is not local.

repeating group

Group item with repetition factor. The repetition factor, which must be greater than 1, specifies the number of duplicates of the group item to be incorporated in the repeating group.

request

The functions of the *DAL* commands ADD DB, ADD RN, DROP DB, DROP RN, NEW RLOG and CHECKPOINT are held in the *DBH* as "requests" and are not executed until the *DAL* command PERFORM is entered.

restart of BMEND

Resumption of an aborted BMEND run.

restart of a session

See *session restart*.

restructuring

Modification of the *Schema DDL* or *SSL* for *databases* already containing data.

return code

Internal code which the called program sends to the calling program; Return code $\neq 0$ means an error has occurred.

RLOG file

Backup file used by the *DBH* during a session to store *before-images* (BFIMs) and *after-images* (AFIMs) of data which is updated. With the aid of the *RLOG file*, the *DBH* can cancel updates effected by incomplete *transactions*. There is one RLOG file per *configuration*. An RLOG file consists of two physical files.

rollback

Canceling of all updates effected within a *transaction*.

RSQ

See *record sequence number*.

RUNUNIT-ID

See *transaction identification*.

S**schema**

Formalized description of all data structures permitted in the *database*. A UDS/SQL schema is defined by means of the *Schema DDL*.

Schema DDL

Formalized language for defining a *schema*.

Schema Information Area (SIA)

The SIA contains the complete database definition. The *DBH* loads the SIA into main memory at the start of DB processing.

SEARCH KEY

Secondary key; *access paths* using secondary keys are created by UDS/SQL by means of *hash routines* and *multi-level tables*.

SEARCH KEY table

Multi-level table used by UDS/SQL as an *access path* via a *secondary key*.

secondary key

Any *key* which is not a *primary key*. Used for *direct access* to a *record* or a set of records with the same key values or within a search interval.

secondary subtransactions

Subtransactions that address *remote configurations*.

sequence number

Identifier in the name of the *ALOG files* (000000001 - 999999999). The first ALOG file of a *database* is always numbered 000000001.

sequential access

Accessing a *record* on the basis of its position within a predefined record sequence.

server task

Task of the *independent DBH* in which the *UDSSUB* module executes; processes the requests of the *DB application programs*.

session

Period between starting and normal termination of the *DBH (independent/linked-in)* in which it is possible to work with the *databases* of the *configuration*. Normally, a session consists of a sequence of *session sections* and *session interrupts*.

session abort

Occurs when the *DBH* is terminated abnormally after a successful *session start*. A session abort can be caused by: power failure, computer failure, BS2000 problems, *DBH* problems, %TERM.

session end

Is the result of:

- *DAL* when using *independent DBH*,
- TERM in the *DML application program* when using *linked-in DBH*,
- *DBH* error handling.

During a *session interrupt*, the user can also effect session end by invalidating the *SLF* contents. Inconsistent *databases* can be made consistent again by a *warm start*, even without an *SLF*.

session interrupt

The period between a *session abort* and the related *session restart*.

session job variable

Job variable in which *UDS/SQL* stores information about a session.

Session Log File (SLF)

File which is permanently assigned to a *session* and which is required by the *DBH* in the event of a *session restart*. It contains information on the current *DB configuration*, the number of current file identifiers and the current values of the *DBH load parameters*.

session restart

Starting of the *DBH*, under the same *configuration name* and *configuration user ID*, after a *session abort*. With the aid of the *SLF*, the *DBH load parameters* and the current file identifiers which existed when the session aborted are re-established, and the *databases* of the previous *configuration* are reattached, if necessary by means of a *warm start*.

session section

Period from the start of the *DBH*, either at the *session start* or a *restart*, to the normal *session end* or to a *session abort*.

session section number

Number which identifies a session section unambiguously.

session start

State of a session in which the *DBH* is started under a configuration name for which there is no *Session Log File (SLF)* with valid contents.

set

Nameable relationship between two *record types*.

set, dynamic

See *dynamic set*.

set, implicit

See *implicit set*.

set, singular

See *SYSTEM set*.

set, standard

See *standard set*.

Set Connection Data (SCD)

Linkage information for the *records* of a *set occurrence*.

set occurrence

Single instance of a *set*. Comprises exactly one *owner record* and any number of subordinate *member records*.

set reference number

Sets are numbered contiguously in ascending order, beginning at 1.

set SEARCH KEY table

SEARCH KEY table for selecting a *member record* from a *set occurrence*.

SF pubset

See *single feature pubset*

shadow database

Backup of all the files of a database, each saved under the name "*dbname.dbfile.copyname*".

A shadow database can be created at any time and processed parallel to the original database in RETRIEVAL mode.

In addition BMEND can be used to apply *ALOG files* that have already been closed to the database parallel to the UDS/SQL *session*.

Shared user buffer pool

Shared buffer of several databases which is used in addition to the *System Buffer Pool*, solely for buffering *pages* of the *databases* that have been assigned to it.

SIA

See *Schema Information Area*.

SIB

See *SQL Interface Block*.

single feature pubset

A single feature pubset (SF pubset) consists of one or more homogeneous disks which must have the same major properties (disk format, allocation unit).

SLF

See *session log file*.

SM pubset

See *system managed pubset*

snap pair, snap pubset, snap session, snap unit

A snap unit is the copy of an (original) unit (logical disk in BS2000) at a particular time (“Point-in-Time copy”). The TimeFinder/Snap component creates this copy as a “snapshot” in accordance with the “Copy-On-First-Write strategy”: Only if data is modified is the original data concerned written beforehand into a central save pool of the Symmetrix system. The snap unit contains the references (track pointers) to the original data. In the case of unmodified data the references point to the unit, in the case of modified data to the save pool.

After they have been activated, the unit and snap unit are split; applications can access both.

The unit and snap unit together form a snap pair. TimeFinder/Snap manages this pair in what is known as a snap session.

If snap units exist for all units of a pubset, these snap units together form the snap pubset.

Details of this are provided in the manual "[Introduction to System Administration](#)".

sort key table

Table pointing to the *member records* of a *set occurrence*.

source program

Program written in a programming language and not yet translated into machine language.

spanned record

Record exceeding the length of a *page*. **Only UDS/SQL-internal records** can be spanned records;

User record types must not exceed

- 2020 bytes for a page length of 2048 bytes
- 3968 bytes for a page length of 4000 bytes
- 8064 bytes for a page length of 8096 bytes.

SQL

SQL is a relational database language which has been standardized by ISO (International Organization for Standardization).

SQL conversation

See *conversation*.

SQL DML

SQL Data Manipulation Language for querying and updating data.

SQL Interface Block (SIB)

Interface between UDS/SQL and SQL application program(s); contains the SQL statement, any existing parameters and the statement results.

SQL transaction

Related sequence of *SQL* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.

SSIA

See *Subschema Information Area*.

SSIA-RECORD

UDS/SQL-internal *record type*, located in the *DBDIR*. *Records* belonging to this type are, for example, the Schema Information Area (*SIA*) and the Subschema Information Areas (*SSIAs*).

SSITAB module

Module generated by the BCALLSI utility routine; makes available the subschema information required by *CALL DML* programs.

SSL

See *Storage Structure Language*.

standard set

A *set* other than a *dynamic*, *implicit* or *SYSTEM set*.

statement code

Number stored in the first part of the *DATABASE-STATUS* item. Its function is to indicate which *DML* statement resulted in an exception condition.

status code

Number stored in the second part of the *DATABASE-STATUS* item. It indicates which exception condition has occurred.

Storage Structure Language (SSL)

Formalized language for describing the storage structure.

string

A series of consecutive alphanumeric characters.

subcontrol system

Component for the *independent DBH*. Responsible for control functions.

subschema

Section of a *schema* required for a particular *application*; it can be restructured, within limits, for the intended application; a subschema is defined by means of the *Subschema DDL*.

Subschema DDL

Formalized language for defining a *subschema*.

Subschema Information Area (SSIA)

The SSIA contains all subschema information required by the *DBH* to carry out, on behalf of the user, the *database* accesses permitted within the specified *subschema*. The *DBH* loads the SSIA into main memory when it is referenced in a *READY* command.

subschema module

Module resulting from *subschema* compilation when a *COBOL DML* program is compiled. It must be linked in to the *application program* and includes the *USER WORK AREA (UWA)* as well as the *RECORD AREA*, which is also part of the *base interface block (BIB)*. The name of the subschema module is the first 8 bytes of the subschema name.

subschema record

Record defined in the *Subschema DDL*.

SUB-SCHEMA SECTION

In COBOL programs with *DML* statements: section of the DATA DIVISION used for specifying the schema name and the subschema name.

subtransaction

In a distributed *transaction*, all the *processing chains* that address the databases in **one** *configuration* form a subtransaction.

system area

Realm required only by UDS/SQL. The system areas of a database include:

- the *Database Directory (DBDIR)*,
- the *Database Compiler Realm (DBCOM)*,
- the *COBOL Subschema Directory (COSSD)*

system break information

Indicates whether the *database* is consistent or inconsistent.

system buffer pools

Input/output buffer for database pages (see *page*). The buffer is part of the *common pool (independent DBH)* or the *DBH work area (linked-in DBH)*. Its size is determined by the *DBH load parameters* 2KB-BUFFER-SIZE, 4KB-BUFFER-SIZE or 8KB-BUFFER-SIZE.

system managed pubset

A system managed pubset consists of one or more volume sets which, as with an *SF pubset*, comprise a collection of multiple homogeneous disks; here, too, homogeneity relates to particular physical properties such as disk format and allocation unit.

SYSTEM record

See *anchor record*.

SYSTEM set

Set whose *owner record type* is the symbolic *record type* SYSTEM.

T

table, multi-level

See *multi-level table*.

table (SQL)

A table in the context of *SQL* corresponds to a UDS/SQL *record type*.

table header

Contains general information on a table or *table page*:

- the table type and the level number of the table page,
- the number of reserved and current entries in this table page,
- the chaining reference to other table pages on the same level,
- the pointer to the associated table page on the next higher level,
- the pointer to the page containing the last table on the main level (for the highest-level table only).

table page

Page containing a table or part of a table. If a *table* which does not extend over several pages or the highest level of a multi-level *table* is concerned, "table page" only refers to the object involved, not the entire *page*.

TANGRAM

(Task and Group Affinity Management) Subsystem of BS2000 that plans the allocation of processors for task groups which access large quantities of shared data in multi-task applications.

task attribute TP

There are 4 task attributes in BS2000: SYS, TP, DIALOG and BATCH. Special runtime parameters that are significant for task scheduling are assigned to each of these task attributes.

In contrast to the other task attributes, the TP attribute is characterized by optimized main memory management that is specially tailored to transaction processing requirements.

task communication

Communication between the *DBH modules*. See also *common pool*.

task deadlock

See *deadlock*.

task priority

In BS2000, it is possible to define a priority for a task. This priority is taken into account when initiating and activating the task.

Priorities may be fixed or variable. Variable priorities are adapted dynamically; fixed priorities do not change.

Note that UDS/SQL server tasks should be started with a fixed priority in order to ensure consistent performance.

TCUA

See *Transaction Currency Area*.

time acknowledgment

Message sent by the *UDS-D task* to the remote *application program* to indicate that there is still a *DML* statement being processed.

transaction (TA)

Related sequence of *DML* statements which is processed by UDS/SQL either as a whole or not at all. This method ensures that the *database(s)* is/are always in a consistent state.

For UDS-D:

The total set of *subtransactions* active at a given time.

transaction, committing a

Terminating a *transaction* with FINISH, i.e. all updates performed within the transaction are committed to the *database*.

transaction, rolling back a

Terminating a *transaction* with FINISH WITH CANCEL, i.e. all updates performed on the *database* within the transaction are rolled back.

Transaction Currency Area (TCUA)

Contains currency information.

transaction identification (TA-ID)

Assigned by the *DBH* to identify a particular *transaction*. Can be requested with the *DAL* command DISPLAY.

transfer pool

UDS-D-specific storage area in which the *UDSCT* receives the *BIBs* from *remote application programs*.

two-phase commit protocol

Procedure by which a *distributed transaction* that has made changes in at least one *remote configuration* is terminated in such a way as to safeguard *inter-configuration consistency* or UDS/SQL openUTM-D consistency. The two-phase commit is controlled

- by the distribution component in the *user task* if the *transaction* is distributed via UDS-D.
- by openUTM-D if the transaction is distributed via openUTM-D or via openUTM-D and UDS-D.

U

UDSADM

Module of the *independent DBH*; executes in the *administrator task*.

UDSHASH

Module generated by the BGSIA utility routine. It contains the names of all the *hash routines* defined in the *Schema DDL*.

UDSNET

Distribution component in the *user task*.

UDSSQL

Start module of the *independent DBH*; executes in the *master task*.

UDSSUB

Start module of the *independent DBH*; executes in the *server task*.

UDS-D task UDSCT

Task started for each *configuration* by UDS/SQL so that it can participate in distributed processing with UDS-D.

UDS/SQL / openUTM-D consistency

A *transaction* that has updated both openUTM data and UDS/SQL *databases* is terminated in such a way that the openUTM data and the UDS/SQL databases are either updated together or not at all.

UDS/SQL pubset declaration

Declaration in a *pubset declaration job variable* for restricting the UDS/SQL pubset environment. This reduces or prevents the risk of file names being ambiguous.

unique throughout the network

Unique in all the computers that are included in the *network*.

user database

The *realms* and files of the *database* required by the user in order to be able to store data in, and to retrieve data from a database are:

- the *Database Directory (DBDIR)*,
- the *user realms*
- the module library for *hash routines (HASHLIB)*.

user realm

A *realm* defined in the realm entry of the *Schema DDL*. It contains, among other things, the user records.

user task

Execution of an *application* program or openUTM program, including the parts linked by the system.

USER-WORK-AREA (UWA)

Transfer area for communication between the *application program* and the *DBH*.

UTM

See openUTM.

UWA

See *USER-WORK-AREA (UWA)*.

V

vector

Item with repetition factor. The repetition factor must be greater than 1. It specifies how many duplicates of the item are combined in the vector.

version number, internal

See *internal version number*.

W

warm start

A warm start is performed by UDS/SQL if an inconsistent *database* is attached to a *session*. For UDS/SQL this involves applying all updates of completed *transactions* to the database which have not yet been applied, *rolling back* all database transactions that are open, and making the database consistent. The related *RLOG file* and the *DB status file* are required for a warm start.

Abbreviations

ACS	Alias Catalog Service
Act-Key	ACTual KEY
AFIM	AFter-IMage
AP	Application Program
ASC	ASCending
BIB	Base Interface Block
BFIM	BeFore-IMage
COBOL	COmmon Business Oriented Language
CODASYL	COnference on DAta SYstem Languages
CRA	CuRrent of Area
CRR	CuRrent of Record
CRS	CuRrent of Set
CRU	Current of RunUnit
COSSD	COBOL SubSchema Directory
DAL	Database Administration Language
DB	DataBase
DBCOR	DataBase COmpiler Realm
DBDIR	DataBase DIRectory
DBH	DataBase Handler
DB-Key	DataBase Key
DBTT	DataBase key Translation Table
DDL	Data Description Language
DESC	DESCending
DML	Data Manipulation Language
DRV	Dual Recording by Volume
DSA	Database System Access
DSSM	Dynamic SubSystem Management

Abbreviations

FC	Function Code
FPA	Free Place Administration
GS	Global Storage
HSMS	Hierarchic Storage Management System
ID	IDentification
IQL	Interactive Query Language
IQS	Interactive Query System
KDBS	Kompatible Datenbank-Schnittstelle (= compatible database interface)
KDCS	Kompatible Datenkommunikationsschnittstelle (= compatible data communications interface)
LM	Lock Manager
LMS	Library Maintenance System
MPVS	Multiple Public Volume Set
MR-NR	MainRef NumberR
MT	Master task
OLTP	OnLine transaction processing
openUTM	Universal Transaction Monitor
OT	Operator Task
PETA	Preliminary End of TrAnsaction
PPP	Probable Position Pointer
PTC	Prepared To Commit
PTT	Primäre Teiltransaktion (= primary subtransaction)
PVS	Public Volume Set
REC-REF	RECOrd REFErence number
RSQ	Record Sequence Number
SC	SubControl
SCD	Set Connection Data
SCI	Software Configuration Inventory
SECOLTP	SECure OnLine Transaction Processing
SECOS	SECurity COntrol System
SET-REF	SET-REFerence
SIA	Schema Information Area
SIB	SQL Interface Block

SLF	Session Log File
SQL	Structured Query Language
SSD	Solid State Disk
SSIA	SubSchema Information Area
SSITAB	SubSchema Information TABLE
SSL	Storage Structure Language
ST	ServerTask
STT	Sekundäre Teiltransaktion (= secondary subtransaction)
TA	TrAnsaction
TA-ID	TrAnsaction IDentification
TANGRAM	TAsk aNd GRoup Affinity Management
TCUA	Transaction CUurrency Area
UDS/SQL	Universal Database System/Structured Query Language
UWA	User Work Area

Related publications

You will find the manuals on the internet at <http://manuals.ts.fujitsu.com>. You can order printed copies of those manuals which are displayed with an order number.

UDS/SQL (BS2000)
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)
Ready Reference

UDS (BS2000)
Interactive Query System IQS
User's Guide

UDS-KDBS (BS2000)
Compatible Database Interface
User Guide

SQL for UDS/SQL

Language Reference Manual

BS2000 OSD/BC

Commands

User Guide

BS2000 OSD/BC

Introduction to System Administration

User Guide

BS2000 OSD/BC

Executive Macros

User Guide

BS2000 OSD/BC

Introductory Guide to DMS

User Guide

SDF (BS2000)

SDF Dialog Interface

User Guide

SORT (BS2000)

User Guide

SPACEOPT (BS2000)

Disk Optimization and Reorganization

User Guide

LMS (BS2000)

SDF Format

User Guide

DSSM/SSCM

Subsystem Management in BS2000

User Guide

ARCHIVE (BS2000)

User Guide

DRV (BS2000)

Dual Recording by Volume

User Guide

HSMS / HSMS-SV (BS2000)
Hierarchical Storage Management System
Volume 1: Functions, Management and Installation
User Guide

SECOS (BS2000)
Security Control System
User Guide

openNet Server (BS2000)
BCAM
Reference Manual

DCAM (BS2000)
Program Interfaces
Reference Manual

DCAM (BS2000)
Macros
User Guide

OMNIS/OMNIS-MENU (BS2000)
Functions and Commands
User Guide

OMNIS/OMNIS-MENU (BS2000)
Administration and Programming
User Guide

openUTM
Concepts and Functions
User Guide

openUTM
Programming Applications with KDCS for COBOL, C and C++
User Guide

openUTM
Generating Applications
User Guide

openUTM
Administering Applications
User Guide

openUTM
Using openUTM Applications under BS2000
User Guide

openUTM
Messages, Debugging and Diagnostics in BS2000
User Guide

COBOL2000 (BS2000)
COBOL Compiler
Reference Manual

COBOL2000 (BS2000)
COBOL Compiler
User's Guide

COBOL85 (BS2000)
COBOL Compiler
Reference Manual

COBOL85 (BS2000)
COBOL Compiler
User's Guide

CRTE (BS2000)
Common Runtime Environment
User Guide

DRIVE/WINDOWS (BS2000)
Programming System
User Guide

DRIVE/WINDOWS (BS2000)
Programming Language
Reference Guide

DRIVE/WINDOWS (BS2000)
System Directory of DRIVE Statements
Reference Manual

DRIVE/WINDOWS (BS2000/SINIX)
Directory of DRIVE SQL Statements for UDS
Reference Manual

DAB (BS2000)
Disk Access Buffer
User Guide

Unicode in BS2000
Introduction

XHCS (BS2000)
8-Bit Code and Unicode Processing in BS2000
User Guide

BS2000 OSD/BC
Softbooks English
DVD

openSM2 (BS2000)
Software Monitor
User Guide

SNMP Management (BS2000)
User Guide

Index

A

access

- contending 355
- direct 355
- hash area 222
- relational 177
- sequential 355

access authorization 355

access path 355

access rights 355

access type 355

act-key 356

ACTKEY format

see table header

act-key-0 page 356

check 68

print 242

act-key-N page 356

ADD statement

UDS online utility 290

additional link

member records 154

ADD-REALM statement

BMEND 34

address

hash area 150, 159

physical 356

table 157

administrator task 356

AFIM 356

apply 57

AFIM logging 315

after-image 356

ALOG file 356

RLOG file 356

after-image file 315

ALIAS item

length 174

allocate buffer size 35

ALLOCATE-BUFFER-POOL statement

BMEND 35

BREORG 317

ALOG file 356

ALOG sequence number 357

alog-seq-no 23

alphanumeric data type (SQL) 180

anchor record 154, 357

database key 154

appl 23

application 357

application program (AP) 357

apply AFIMs 57

area 357

AREA INFORMATION 142, 163

ascending key (ASC key) 357

attach realm 34

authorization 357

authorized users 357

automatic DBTT extension 357

automatic realm extension 357

B

backup database 358

Base Interface Block (BIB) 358

base tables

SQL accesses 192

BCHECK 67

CHECK statement 81

command sequence 92

consistency criterion 82

- BCHECK (cont.)
 - consistency error messages 106
 - error messages 100
 - examples 99
 - execution messages 100
 - interactive input of statements 79
 - job switch 132
 - KEY statement 90
 - key value check 86, 87
 - messages 99
 - REALM statement 85
 - RECORD statement 86
 - SCHEMA statement 84
 - SCRTH1 file 76
 - SET statement 88
 - SORTCORE statement 80
 - SORTWK file 76
 - system environment 71, 72
 - TYPE statement 82
 - warnings 99
 - work files 76
- bcheck inconsistencies 67
- before-image 358
- BFIM 358
- BIB (Base Interface Block) 358
- BMEND
 - ADD-REALM statement 34
 - ALLOCATE-BUFFER-POOL statement 35
 - command sequence 60
 - DISABLE-ONLINE-COPY statement 36
 - ENABLE-ONLINE-COPY statement 37
 - END statement 38
 - functions 30
 - job variables 61
 - KILL-LOG statement 39
 - OPEN-DATABASE statement 40
 - REMOVE-REALM statement 41
 - SHOW-LOG-INFORMATION statement 42
 - START-LOG statement 48
 - STOP-LOG statement 55
 - UNDO statement 56
 - UPDATE-DATABASE statement 57
- BMODTT 349
 - command sequence 353
 - KEEP statement 351
 - RESET statement 352
 - REUSE statement 351
 - SET statement 352
 - statements 351
 - system environment 350
- BNR format
 - see table header
- BPRECORD 229
 - command sequence 258
 - DISPLAY CALC statement 249
 - DISPLAY DATA statement 254
 - DISPLAY DBTT statement 247
 - DISPLAY PAGE ZERO statement 242
 - DISPLAY-FPA statement 246
 - PRINT statement 239
 - REALM statement 238
 - SCHEMA statement 237
 - statement sequence 235
 - statements 235
 - system environment 232
- BPSIA 133
 - command sequence 137
 - DISPLAY-SCHEMA statement 135
 - DISPLAY-SUBSCHEMA statement 135, 136, 226, 257
 - statements 135
 - system environment 134
- BPSQLSIA
 - command sequence 188
 - conversion rules 190
 - END statement 185
 - OPEN-DATABASE statement 186
 - output 189
 - PRINT-RELATIONAL-SCHEMAINFO statement 187
 - system environment 178
 - utility routine 177
- BREAK statement
 - UDS online utility 291

- BREORG 308
 - ALLOCATE-BUFFER-POOL statement 317
 - command sequence 343
 - END statement 318
 - examples 345
 - functions 308
 - job switch 315
 - MODIFY-REALM-SIZE statement 319
 - MODIFY-RECORD-POPULATION statement 322
 - OPEN-DATABASE statement 325
 - REORGANIZE-CALC statement 326
 - REORGANIZE-POINTERS statement 333
 - REORGANIZE-SET statement 334
 - SPECIFY-SCHEMA statement 340
 - SPECIFY-SUBSCHEMA statement 341
 - system environment 309
 - UNDO statement 342
 - work files 311
- BSTATUS 200
 - command sequence 227
 - DISPLAY CALC statement 220
 - DISPLAY RECORD statement 217
 - DISPLAY RECORDNUMBER statement 224
 - DISPLAY TABLE FOR OWNER statement 214
 - DISPLAY TABLE FOR SET statement 211
 - DISPLAY-REALM statement 208
 - SUBSCHEMA statement 207
 - system environment 203
 - work files 204
- buffer pools
 - see exclusive buffer pool
 - see system buffer pools
- buffer size
 - allocate 35
 - define 317
- C**
- CALC areas
 - reorganize 326, 347
- CALC INFORMATION 149
- CALC key 149, 158, 166, 171, 174, 220, 221, 234, 358
 - length 149, 158, 166, 174
 - statistics 202, 220
 - table 158, 253
- CALC KEY INFORMATION 166
- CALC key statistics
 - print 220
- CALC page 252, 358
 - check 68
 - print 249
- CALC SEARCH key 249, 358
- CALC table 359
- CALC table line 68
- CALC-SEARCH-KEY-INFORMATION 158
- calculate size
 - SORTWK file 205
- CALL DML 359
- catalog identifier 359
- catid 23
- chain 359
- Character Separated Values (CSV) 359
- check
 - act-key-0 page 68
 - CALC pages 68
 - coherence 70, 71
 - compiler database 67
 - consistency of database 67
 - consistency, physical 67
 - database, original 70
 - for consistency 67
 - original database 70
 - page header 68
 - page structures of realms 68
 - physical structures 67
 - PRIVACY-AND-IQF database 67
 - record displacement 68
 - record sequence numbers 68
 - shadow database 70
 - sort sequence of keys 68
 - system data 67
 - table pages 68
 - user realm 67

- check records [359](#)
 - generate [68](#)
- check run
 - database files [72](#)
 - offline [71](#)
 - online [71](#)
 - parallel [71](#)
- CHECK statement
 - BCHECK [81](#)
- check system data [67](#)
- checking
 - incremental [70](#)
 - overall [70](#)
- checking mode [68](#)
 - select [81](#)
 - set [68](#)
- checking procedures [69](#)
- checkpoint [359](#)
- CHECK-TABLE [359](#)
- clone [360](#)
- COBOL DML [360](#)
- COBOL runtime system [360](#)
- COBOL Subschema Directory (COSSD) [360](#)
- CODASYL subschema [177](#)
- coherence check [70, 71](#)
- column number
 - DBTT [157, 213, 215](#)
- command sequence
 - BCHECK [92](#)
 - BMEND [60](#)
 - BMODTT [353](#)
 - BPRECORD [258](#)
 - BPSIA [137](#)
 - BPSQLSIA [188](#)
 - BREORG [343](#)
 - BSTATUS [227](#)
- common memory [360](#)
- common pool [360](#)
- communication partners [360](#)
- communication pool [360](#)
- compatible database interface [360, 372](#)
- compiler database [361](#)
 - check [67](#)
- COMPILER-SCHEMA [361](#)
- COMPILER-SUBSCHEMA [361](#)
- compound key [361](#)
- compression [361](#)
- conditions
 - UDS online utility [300](#)
- configuration [361](#)
- configuration identification [361](#)
- configuration name [361](#)
- connection module [361](#)
- consistency [361](#)
 - check for [67](#)
 - logical [362](#)
 - of database, check [67](#)
 - physical [362](#)
 - physical, checking [67](#)
 - storage [362](#)
- consistency criteria
 - selecting [82](#)
- consistency criterion
 - BCHECK [82](#)
 - select [82](#)
- consistency error [362](#)
- consistency error messages
 - BCHECK [106](#)
- consistency point [362](#)
- consistency record [362](#)
- contending access [362](#)
- conversation [362](#)
- conversion rules
 - BPSQLSIA [190](#)
- copy [362](#)
- copyname [23](#)
- COSSD [362](#)
- counter procedure [69](#)
- CRA [362](#)
- creation date
 - realm [244](#)
- criteria
 - global consistency check [82](#)
- CRR [363](#)
- c-string [24](#)
- CSV [363](#)

- CSV format
 - BPRECORD 242, 246, 247, 249, 254
 - BPSIA 135, 136
 - BSTATUS 208, 211, 214, 217, 220, 224
 - output 135, 136, 208, 211, 214, 217, 220, 224, 242, 246, 247, 249, 254
- csv-dateiname 24
- currency table 163, 363
 - length 161
- CURRENT
 - OF AREA table 363
 - OF RECORD table 363
 - OF SET table 363
- Current
 - Record of Rununit 363
 - Record of Set (CRS) 363
- D**
- dal-cmd 24
- data backup 363
- data deadlock 363
- Data Manipulation Language (DML) 368
- data pages 254
 - print 254
- data protection (privacy) 364
- data type
 - structured-name 25
- data types 23
 - SQL 180
- database
 - name 160
 - open 40, 186, 325
- database (DB) 364
- database administrator 364
- Database Administrator Language (DAL) 363
- database compiler realm (DBCOM) 364
- database copy 364
- database copy update 364
- database directory (DBDIR) 364
- database job variable 364
- database key 364
 - anchor record 154
 - item 165, 365
- database operation
 - SHARED-RETRIEVAL 71
- database page 365
- database pages 142
 - length 140, 243
 - occupancy 209
- database system 365
- database, original
 - check 70
- DATABASE-KEY item 365
- DATABASE-KEY-LONG item 365
- DATABASE-STATUS 365
- date 24
- DB configuration 365
- DB key 164, 366
- DB status file 365
- DBCOM 365
- DBDIR 366
- DBH 366
 - end 366
 - independent 366
 - linked-in 366
 - start 366
- DBH load parameters 366
- dbname 24
- DBTT 366
 - column number 157, 213, 215
 - filling ratio 219
 - list entries 247
 - page 367
 - size 219
- DBTT anchor page 366
- DBTT anchor table 148
- DBTT base 366
- DBTT entries
 - description 247
 - length 148
 - used 219
- DBTT extension
 - automatic 357
 - online 376
- DBTT extent 367
- DBTT extents
 - number 148, 219

- DBTT INFORMATION [147](#)
- DCAM [367](#)
- DCAM application [367](#)
- DDL [367](#)
- deactivate logging [55](#)
- deadlock [367](#)
- DECLARE-PROCEDURE statement
 - UDS online utility [271](#)
- DECLARE-VARIABLE statement
 - UDS online utility [272](#)
- define
 - buffer size [317](#)
 - output scope [239](#)
- define size
 - sort buffer [80](#)
- defining a procedure [271](#)
- defining a variable [272](#)
- defining FPA level in a realm [275](#)
- defining ONLINE-UTILITY parameters [276](#)
- DELETE-PROCEDURE statement
 - UDS online utility [273](#)
- DELETE-VARIABLE statement
 - UDS online utility [273](#)
- deleting a procedure [273](#)
- deleting a variable [273](#)
- descending key (DESC key) [367](#)
- description
 - DBTT entries [247](#)
 - realm statistics [209](#)
 - table [257](#)
- designate
 - schema [237](#)
 - subschema [207](#)
- determine size
 - hash area [330](#)
- device [24](#)
- direct access [367](#)
- direct hash area [367](#)
- disable
 - online backup capability [36](#)
- DISABLE-ONLINE-COPY statement
 - BMEND [36](#)
- DISPLAY CALC statement
 - BPRECORD [249](#)
 - BSTATUS [220](#)
- DISPLAY DATA statement
 - BPRECORD [254](#)
- DISPLAY DBTT statement
 - BPRECORD [247](#)
- DISPLAY PAGE ZERO statement
 - BPRECORD [242](#)
- DISPLAY RECORD statement
 - BSTATUS [217](#)
- DISPLAY RECORDNUMBER statement
 - BSTATUS [224](#)
- DISPLAY statement
 - output in CSV format [135](#), [136](#), [208](#), [211](#), [214](#), [217](#), [220](#), [224](#), [242](#), [246](#), [247](#), [249](#), [254](#)
- DISPLAY TABLE FOR OWNER statement [214](#)
 - BSTATUS [214](#)
- DISPLAY TABLE FOR SET statement
 - BSTATUS [211](#)
- DISPLAY-FPA statement
 - BPRECORD [246](#)
- DISPLAY-REALM statement
 - BSTATUS [208](#)
- DISPLAY-SCHEMA statement
 - BPSIA [135](#)
- DISPLAY-SUBSCHEMA statement
 - BPSIA [135](#), [136](#)
- distributable list [87](#), [89](#), [152](#), [153](#), [259](#), [260](#), [262](#), [265](#), [270](#), [277](#), [281](#), [282](#), [289](#), [293](#), [373](#)
 - preferred realm [153](#), [259](#), [260](#), [265](#), [270](#), [277](#), [289](#), [293](#)
 - table part [152](#)
 - table realm [152](#), [373](#)
- distributed database [367](#)
- distributed transaction [368](#)
- distribution pool [368](#)
- distribution table [368](#)
- DML
 - FPASCAN [265](#)
 - PREFRLM [265](#)
 - RELOCATE [262](#)
 - REORGLPPP [266](#)

- DML FPASCAN
 - showing currently valid parameters 286
- DML RELOCATE
 - showing currently valid parameters 287
- DML REORGPDP
 - showing currently valid parameters 287
- dummy subtransaction 368
- duplicates header 368
- duplicates table 257, 369
 - main level 369
- dynamic set 152, 154, 369
- E**
- enable online backup capability 37
- ENABLE-ONLINE-COPY statement
 - BMEND 37
- END statement
 - BMEND 38
 - BPSIA 136, 257
 - BPSQLSIA 185
 - BREORG 318
 - BSTATUS 226
 - UDS online utility 273, 291
- entries
 - table 256
- error handling
 - UDS online utility 297
- error messages
 - BCHECK 100
- ESTIMATE-REPORT 369
- event name 369
- examples
 - BCHECK 99
 - BREORG 345
 - UDS online utility 301
- exclusive buffer pool 369
- executing a procedure 274
- execution messages
 - BCHECK 100
- EXIT statement
 - UDS online utility 291
- EXTENSIBILITY 219
- extensibility
 - extensibility 219
 - extensibility, record type 219
- F**
- filling percentage
 - hash area 222
 - pages 209
- filling ratio
 - DBTT 219
 - set tables 213, 216
 - table 213, 216
- FINISH statement
 - UDS online utility 292
- First Scan
 - free place search 143, 265, 275, 292
- foreign key 369
- format subschema 166, 168
- FPA 369
- FPA base 243, 370
- FPA entries
 - list 246
- FPA extent 142, 243, 320, 370
- FPA page 370
- FPASCAN 265
- FPASCAN statement
 - UDS online utility 292
- free pages 209
- free place administration 140, 142, 247, 320, 370
- free place search
 - First Scan 143, 265, 275, 292
 - search mode 265
 - Second Scan 143, 265, 275, 292
- free space
 - hash area 222
- function code 370
- functions
 - BREORG 308
 - UDS online utility 260
- G**
- generate check records 68
- global consistency check
 - criteria 82
- group item 370

H

hash area 221, 370
 access 222
 address 150, 159
 determine size 330
 filling percentage 222
 free space 222
 number of overflow pages 222
 number of primary pages 159, 221
 occupancy level 222
 print 220, 249, 252
hash routine 370
hashing 370
HASHLIB 371
header table 256
host 24

I

identifier 371
identify schema 84
iist
 distributable, see distributable list
implicit set 146, 152, 371
inconsistency 371
 BCHECK 67
incremental checking 70, 71
independent DBH 371
index level 371
index page 371
INDEX search key 371
indirect hash area 371
input
 terminate 38, 185
 termination 318
integer 24
integrity 372
interactive input of statements
 BCHECK 79
interconfiguration 372
 consistency 372
 deadlock 372
interface 372
internal realm version 244
internal version number 372

item 372
 database key 165
 length 176
 variable 169
item string 168
 length 169
ITEM STRING LIST 168
item type 166, 174, 176

J

job switch
 BCHECK 132
 BREORG 315
job variable
 BMEND 61
 structure 62
 update 65

K

KDBS 360, 372
KEEP statement
 BMODTT 351
key 373
 compound 373
 length 139, 149, 158, 171, 174
 number 139, 156, 158, 162, 171, 176, 252
KEY INFORMATION 155, 175
key item 373
 length 156, 171, 176
 number 176
 type 176
KEY ITEM LIST 170
key reference number 373
KEY statement
 BCHECK 90
key value check
 BCHECK 86, 87
keys
 number per record type 164
 number per set 173
 subschemata 176
keyword 20, 21
KILL-LOG statement
 BMEND 39

kset 24

L

length

- ALIAS item 174
- CALC key 158, 166, 174
- CALC keys 149
- currency table 161
- database pages 140, 243
- DBTT entries 148
- item 176
- item string 169
- key 139, 149, 158, 171, 174
- key item 156, 171, 176
- record type 139, 145
- set connection data 154
- SSIA 161

linked-in control system 373

linked-in DBH 373

list 373

- distributable, see distributable list
- FPA entries 246

list entries

DBTT 247

load parameters DBH 373

local application program 373

local configuration 374

local database 374

local distribution table 374

local host 374

local transaction 374

LOCATION MODE 145, 173

logging 374

- deactivate 55
- start 48
- stop 39

logging information

output 42

logical connection 374

M

main reference 374

mainref number 374

master task 374

member 375

- AUTOMATIC 375
- MANDATORY 375
- MANUAL 375
- OPTIONAL 375

member record 375

additional link 154

member record type 154, 375

membership
set 152

messages

BCHECK 99

Messages BCHECK 99

modify

- realm size 319
- record population 322

MODIFY-REALM-SIZE statement

BREORG 319

MODIFY-RECORD-POPULATION statement

BREORG 322

mono-DB configuration 375

mono-DB operation 375

MOVE statement

UDS online utility 293

multi-DB configuration 375

multi-DB operation 375

multi-DB program 375

multi-level table 375

multithreading 376

N

name 24

- database 160
- realm 209, 238
- record type 145, 226, 252
- schema 137, 138, 160, 237
- set 152, 252
- subschemata 137, 161

national data type (SQL) 180

national items 254

network 376

notational conventions 20, 21

SDF statements 22

number

- DBTT extents [148, 219](#)
 - key [139, 156, 158, 162, 171, 176, 252](#)
 - key items [176](#)
 - of DBTT entries [148](#)
 - of records stored [226](#)
 - overflow pages [222](#)
 - primary pages [150](#)
 - realm [139, 141, 142, 148, 157, 161, 226, 234, 243](#)
 - record type [139, 144, 145, 148, 252](#)
 - set [139, 152, 156, 158, 176](#)
 - stored records [213, 219](#)
- number of index levels
- set tables [213](#)
 - tables [213, 216](#)
- number of overflow pages
- hash area [222](#)
- number of primary pages
- hash area [159, 221](#)
- number per realm
- record type [144](#)
- number per record type
- keys [164](#)
- number per set
- keys [173](#)
- number per subschema
- realm [161](#)
- numeric data type (SQL) [181](#)

O

- occupancy
 - database pages [209](#)
- occupancy level
 - hash area [222](#)
 - overflow pages [222](#)
- offline check run [71](#)
- OLTP [376](#)
- online backup [376](#)
- online backup capability
 - disable [36](#)
 - enable [37](#)
- online check run [71](#)
- online DBTT extension [376](#)

- online realm extension [376](#)
- open database [40, 186, 325](#)
- open transaction [376](#)
- OPEN-DATABASE statement
 - BMEND [40](#)
 - BPSQLSIA [186](#)
 - BREORG [325](#)
- openUTM [377](#)
- operator task (OT) [377](#)
- optional word [20, 21](#)
- original database [377](#)
 - check [70](#)
- original realms
 - overall checking [72](#)
- output
 - BPSQLSIA [189](#)
 - in CSV format [135, 136, 208, 211, 214, 217, 220, 224, 242, 246, 247, 249, 254](#)
 - logging information [42](#)
- output scope
 - define [239](#)
- overall checking [70, 71](#)
 - original realms [72](#)
 - shadow database [73](#)
- overflow pages [377](#)
 - number [222](#)
 - occupancy level [222](#)
 - remove [328, 329](#)
- owner [377](#)
 - statistics [201, 215](#)
- owner record [377](#)
- owner record type [154, 377](#)
- owner statistics
 - print [214](#)

P

- P1 eventing [380](#)
- page [377](#)
- page address [377](#)
- page container [378](#)
- page header [240, 241, 252](#)
 - check [68](#)
- page header (page info) [378](#)
- PAGE INDEX [241, 252, 256](#)

- page index entry 239, 240, 241, 252, 378
- PAGE INFO 241, 252, 256
- page number 142, 148, 154, 236, 247, 257, 378
- page numbers 247
- page selection 236
- page structures of realms
 - check 68
- pages
 - filling percentage 209
 - free 209
- parallel check runs 71
- password for UDS/SQL files 378
- pattern 378
- pattern string 378
- PETA 379
- physical consistency
 - checking 67
- physical structures
 - check 67
- pointer array 379
- pointers
 - reorganize 328, 329
- PPP (probable position pointer) 150, 157, 158, 307, 308, 313, 320, 326, 328, 329, 333, 379, 380
- PPP reorganization
 - end 267
 - initialize 267
 - locked pages 267
 - procedure 266
 - progress data 266
- predefined variable
 - UDS online utility 298
- preferred realm
 - defining 277, 287
 - distributable list 153, 259, 260, 265, 270, 277, 289, 293
- PREFRLM 265
- PREFRLM statement
 - UDS online utility 293
- prepared to commit (PTC) 379
- primary key 379
- primary key (DDL) 379
- primary key (SQL) 379
- primary pages
 - number 150
- primary subtransaction 380
- print
 - act-key-0 page 242
 - CALC key statistics 220
 - CALC pages 249
 - data pages 254
 - hash area 220, 249, 252
 - owner statistics 214
 - realm statistics 208
 - record number statistics 224
 - record type-statistics 217
 - Schema Information Area 135, 138
 - set statistics 211
 - Subschema Information Area 135, 138
- print realm 238
- PRINT statement
 - BPRECORD 239
- print table 256
- PRINT-RELATIONAL-SCHEMAINFO statement
 - BPSQLSIA 187
- PRIVACY-AND-IQF database
 - check 67
- PRIVACY-AND-IQF SCHEMA 380
- PRIVACY-AND-IQF SUBSCHEMA 380
- probable position pointer
 - reorganize 266
- probable position pointer (PPP) 150, 157, 158, 307, 308, 313, 320, 326, 328, 329, 333, 379, 380
 - reorganize 333
- probable position pointers
 - update 260
- procedure statements
 - UDS online utility 289
- processing chain 380
- PTC state 380
- pubset declaration 30, 71, 134, 178, 203, 231, 308, 350, 380
- pubset declaration job variable 380

R

- Readme file 17
- READY 381
- READY UPDATE statement
 - UDS online utility 294
- READYC 381
- realm 381
 - creation date 244
 - number 148, 157
 - number per subschema 161
 - remove 41
 - statistics 208
- realm attach 34
- realm configuration 381
- realm copy 381
- realm extension
 - automatic 357
 - online 376
- realm name 142, 209, 238
- realm number 139, 141, 142, 150, 161, 226, 234, 243
- realm reduce 345
- realm reference number 381
- realm size 210
 - modify 319
- REALM statement
 - BCHECK 85
 - BPRECORD 238
- realm statistics 200
 - description 209
 - print 208
- realm time of last update 244
- realm to be output
 - specify 238
- realm version
 - internal 244
- realm-name 25
- realmref 25
- realms to be checked
 - specify 85
- reconfiguration 381
- record 168, 241, 254, 256, 381
- record address 381
- RECORD AREA 165, 173, 382
 - record displacement
 - check 68
 - record element 382
 - record hierarchy 382
 - RECORD INFORMATION 145, 164
 - record number statistics 202, 224
 - print 224
 - record population
 - modify 322
 - record reference number 382
 - record SEARCH KEY table 382
 - record sequence number 148, 219, 236, 382
 - check 68
 - RECORD statement
 - BCHECK 86
 - record type 141, 382
 - length 139, 145
 - linear 382
 - name 145, 148, 226, 252
 - number 139, 144, 145, 148, 252
 - number per realm 144
 - schema format 166, 173, 176
 - specify for checking 86
 - statistics 201, 217
 - subschema format 166, 173, 176
 - record type-statistics
 - print 217
 - RECORD WITHIN LIST 144
 - record-name 25
 - recordref 25
 - records stored
 - number of 226
 - REC-REF 382
 - reduce realm 345
 - REFERENCE NUMBERS 141, 162
 - referential integrity 382
 - relational
 - access 177
 - relational access 177
 - relational schema information 177
 - RELOCATE 262
 - RELOCATE statement
 - UDS online utility 294

- relocating
 - data pages 262
- relocating data pages 262
- REMARK statement
 - UDS online utility 295
- remote application program 382
- remote configuration 383
- remote database 383
- remote host 383
- remove
 - overflow pages 328, 329
 - realms 41
- REMOVE-REALM statement
 - BMEND 41
- reorganize
 - CALC areas 326, 347
 - pointers 328, 329
 - probable position pointer 266
 - probable position pointer (PPP) 333
 - set constructs 334
 - set tables 216
 - tables 213, 216, 334, 336
- REORGANIZE-CALC statement
 - BREORG 326
- REORGANIZE-POINTERS statement
 - BREORG 333
- REORGANIZE-SET statement
 - BREORG 334
- REORGPPP 260, 266
- REORGPPP DML
 - defining properties 283
- REORGPPP statement
 - UDS online utility 295
- repeating group 383
- REPEAT-PROCEDURE statement
 - UDS online utility 274
- request 383
- RESET statement
 - BMODTT 352
- restart
 - of a session 383
 - of BMEND 383
- restructuring 383
- return code 383
- REUSE statement
 - BMODTT 351
- RLOG file 383
- rollback 384
- RSQ 384
- RUNUNIT-ID 384
- S**
- S variable
 - SHOW-VARIABLE statement 288, 296
- saving 315
- schema 384
 - designate 237
 - identify 84
 - name 137, 138, 160, 237
 - specify 340
- schema DDL 384
- schema format 166, 168
 - record type 166, 173, 176
- schema information
 - relational 177
- Schema Information Area 133, 138
 - print 135, 138
- Schema Information Area (SIA) 384
- SCHEMA statement
 - BCHECK 84
 - BPRECORD 237
- schema-name 25
- SCRATCH1 file 204, 311
 - BCHECK 76
- SDF 178
- SDF statements
 - UDS online utility 270
- SDF statements notational conventions 22
- SEARCH KEY 384
- SEARCH KEY table 384
- SEARCH keys to be checked
 - specify 90
- search mode
 - for free place search 265
- Second Scan
 - free place search 143, 265, 275, 292
- secondary key 384
- secondary subtransaction 384

- select
 - checking mode 81
 - consistency criterion 82
 - subschemas 187
- selecting
 - consistency criteria 82
- sequence number 384
- sequential access 385
- server task 385
- session 385
 - abort 385
 - end 385
 - interrupt 385
 - start 386
- session job variable 385
- Session Log File (SLF) 385
- session restart 386
- session section 386
- session section number 386
- set 386
 - checking mode 68
 - dynamic 152, 154, 386
 - implicit 146, 152, 386
 - membership 152
 - name 152, 252
 - number 139, 152, 156, 158, 176
 - singular 152, 386
 - sort sequence 152
 - specify for checking 88
 - standard 386
 - statistics 201, 211
 - type 152
- set connection data 145, 154, 157, 176, 239, 240, 241
 - length 154
- Set Connection Data (SCD) 386
- set constructs
 - reorganize 334
- SET INFORMATION 151, 172
- set mode 152
- set occurrence 386
- SET OCCURRENCE SELECTION 173
- set occurrence selection 152
- set reference number 386
- set SEARCH KEY table 387
- SET statement
 - BCHECK 88
 - BMODTT 352
- set statistics
 - print 211
- set tables
 - filling ratio 213, 216
 - number of index levels 213
 - reorganize 216
- SET-FPA-SCAN-PARAMETERS statement
 - UDS online utility 275
- set-name 25
- SET-ONLINE-UTILITY-PARAMETERS statement
 - UDS online utility 276
- SET-PREF-REALM-PARAMETERS statement
 - UDS online utility 277
- SET-RELOCATE-PARAMETERS statement
 - UDS online utility 278
- SET-REORGANIZE-PPP-PARAMETERS statement
 - UDS online utility 283
- SF pubset 387
- shadow database 387
 - check 70
 - overall check 73
- Shared User buffer pool 387
- SHARED-RETRIEVAL
 - database operation 71
- SHOW statement
 - UDS online utility 296
- SHOW-FPA-SCAN-PARAMETERS statement
 - UDS online utility 286
- showing a procedure
 - UDS online utility 286
- SHOW-LOG-INFORMATION statement
 - BMEND 42
- SHOW-PREF-REALM-PARAMETERS statement
 - UDS online utility 287
- SHOW-PROCEDURE statement
 - UDS online utility 286
- SHOW-RELOCATE-PARAMETERS statement
 - UDS online utility 287

- SHOW-REORGANIZE-PPP-PARAMETERS
 - statement
 - UDS online utility 287
- SHOW-VARIABLE statement
 - S variable 288, 296
 - UDS online utility 288
- SIA 387
- SIA PRINT REPORT 138
- SIA report 138, 141
- SIA VALIDATION DATE 161
- SIB 387
- single feature pubset 387
- singular set 152
- size
 - DBTT 219
 - realm 210
- size calculation
 - SORTWK file 205
- SLF 387
- SM pubset 387
- snap 388
- sort buffer
 - define size 80
- sort check 68, 69
- sort key table 388
- sort sequence
 - set 152
- sort sequence of keys
 - check 68
- SORTCORE statement
 - BCHECK 80
- sorting procedure 69
- SORTWK file 205, 312
 - BCHECK 76
 - calculate file 205
 - calculating size 205
- source level 263
- source program 388
- spanned record 84, 388
- specify
 - realm to be output 238
 - realms to be checked 85
 - record types to be checked 86
 - schema 340
 - SEARCH keys to be checked 90
 - sets to be checked 88
 - subschemata 341
- SPECIFY-SCHEMA statement
 - BREORG 340
- SPECIFY-SUBSCHEMA statement
 - BREORG 341
- SQL 388
- SQL accesses
 - base tables 192
- SQL conversation 388
- SQL data types 180
- SQL DML 388
- SQL interface 177
- SQL Interface Block (SIB) 389
- SQL transaction 389
- SSIA 389
- SSIA (Subschema Information Area)
 - length 161
- SSIA PRINT REPORT 160
- SSIA report 160
- SSIA-RECORD 389
- SSITAB module 389
- SSL 389
- standard procedures
 - UDS online utility 299
- standard set 389
- start logging 48
- starting
 - UDS online utility 269
- START-LOG statement
 - BMEND 48
- statement
 - undo 56, 342
- statement code 389
- statement sequence
 - BPRECORD 235
- statements
 - BMODTT 351
 - BPRECORD 235
 - BPSIA 135

statistics

- CALC key 202, 220
- owner 201, 215
- realm 200, 208
- record number 202, 224
- record type 201, 217
- set 201, 211

status code 389

Status-Codes

- UDS-Online-Utility 299

stop logging 39

STOP-LOG statement

- BMEND 55

Storage Structure Language (SSL) 389

stored records

- number 213, 219

string 169, 389

structure

- job variable 62

structured data type (SQL) 182

structured-name (data type) 25

subcontrol system 389

subschemata 390

- designate 207
- format 166, 168
- keys 176
- name 137, 161
- specify 341

Subschema DDL 390

subschemata format

- record type 166, 173, 176

Subschema Information Area 133, 160

- print 135, 138

Subschema Information Area (SSIA) 390

subschemata module 390

subschemata record 390

SUB-SCHEMATA SECTION 390

SUBSCHEMATA statement 207

- BSTATUS 207

subschemata-name 25

subschemata

- select 187

subtransaction 390

- summing checks 68, 69
 - using internal results 78
- syntax description 22
- system area 390
- system break information 244, 390
- system buffer pools 391
- system environment
 - BCHECK 71, 72
 - BMODTT 350
 - BPRECORD 232
 - BPSIA 134
 - BPSQLSIA 178
 - BREORG 309
 - BSTATUS 203
- system managed pubset 391
- SYSTEM record 391
- SYSTEM set 152, 391

T

table

- address 157
- CALC key 158, 253
- description 257
- entries 256
- filling ratio 213, 216
- header 256
- multi-level 391
- number of index levels 213, 216
- print 256
- reorganize 213, 216, 334, 336

table (SQL) 391

table header 391

- ACTKEY format 257
- BNR format 257

table pages 391

- check 68

table part

- distributable list 152

table realm

- distributable list 152, 373

TANGRAM 392

target level 263

task attribute TP 392

task communication 392

- Task Currency Area [160, 163, 164, 173](#)
 - task deadlock [392](#)
 - task priority [392](#)
 - TCUA [392](#)
 - terminate input [38, 185, 318](#)
 - terminating the UDS online utility [273](#)
 - time [26](#)
 - time acknowledgment [392](#)
 - time of last update
 - realm [244](#)
 - transaction [392](#)
 - committing a [392](#)
 - roll back [393](#)
 - Transaction Currency Area [139, 393](#)
 - transaction identification (TA-ID) [393](#)
 - transfer pool [393](#)
 - two-phase commit protocol [393](#)
 - type
 - key item [176](#)
 - set [152](#)
 - TYPE statement
 - BCHECK [82](#)
- U**
- UDS [393](#)
 - UDS online utility
 - conditions [300](#)
 - DECLARE-PROCEDURE statement [271](#)
 - DECLARE-VARIABLE statement [272](#)
 - DELETE-PROCEDURE statement [273](#)
 - DELETE-VARIABLE statement [273](#)
 - END statement [273](#)
 - error handling [297](#)
 - examples [301](#)
 - functions [260](#)
 - predefined variable [298](#)
 - procedure statements [289](#)
 - REPEAT-PROCEDURE statement [274](#)
 - SDF statements [270](#)
 - SET_FPA-SCAN-PARAMETERS [275](#)
 - SET-ONLINE-UTILITY-PARAMETERS [276](#)
 - SET-PREF-REALM-PARAMETERS [277](#)
 - SET-RELOCATE-PARAMETERS [278](#)
 - SET-REORGANIZE-PPP-PARAMETERS [283](#)
 - SHOW-FPA-SCAN-PARAMETERS [286](#)
 - showing a procedure [286](#)
 - showing value of a variable [288](#)
 - SHOW-PROCEDURE [286](#)
 - SHOW-RELOCATE-PARAMETERS [287](#)
 - SHOW-REORGANIZE-PPP-PARAMETERS [287](#)
 - SHOW-VARIABLE [288](#)
 - standard procedures [299](#)
 - starting [269](#)
 - UDS/SQL [394](#)
 - UDS/SQL / openUTM-D consistency [394](#)
 - UDS/SQL pubset declaration [30, 71, 134, 178, 203, 231, 308, 350, 394](#)
 - UDSADM [393](#)
 - UDS-D task UDSCCT [394](#)
 - UDSHASH [393](#)
 - UDSNET [393](#)
 - UDS-online utility
 - SHOW-PREF-REALM-PARAMETERS [287](#)
 - UDS-Online-Utility
 - Status-Codes [299](#)
 - UDSSUB [393](#)
 - UNDO statement
 - BMEND [56](#)
 - BREORG [342](#)
 - undo statement [56, 342](#)
 - Unicode [254](#)
 - unique throughout the network [394](#)
 - update job variable [65](#)
 - UPDATE-DATABASE statement
 - BMEND [57](#)
 - used DBTT entries [219](#)
 - user database [394](#)
 - user realm [394](#)
 - check [67](#)
 - user task [394](#)
 - userid [26](#)
 - USER-WORK-AREA (UWA) [164, 394](#)
 - using internal results
 - summing checks [78](#)

Index

utility routine
 BPSQLSIA 177
UWA 394

V

variable 20, 21
 item 169
 showing value 288
vector 395
version number
 internal 395
volume 26

W

WAIT statement
 UDS online utility 296
warm start 395
warnings
 BCHECK 99
work files
 BCHECK 76
 BREORG 311
 BSTATUS 204

X

x-string 26