

English



FUJITSU Software BS2000

SESAM/SQL-Server V9.0

SQL Reference Manual Part 2

User Guide

Edition October 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.

Content

1	Preface	7
1.1	Objectives and target groups of this manual	7
1.2	Summary of contents	8
1.3	Notational conventions	9
2	Basic information on working with utility statements	11
2.1	SESAM/SQL utility functions	12
2.2	Space states and permissible utility statements	13
2.3	Utility statements with partitioned tables	22
2.4	CHECK pragma	24
2.5	Data representation in the input and output files for LOAD and UNLOAD	26
2.5.1	Readable representation of the data in the input and output files	27
2.5.1.1	Readable representation of the input data for LOAD	27
2.5.1.2	Readable representation of the output data for UNLOAD	29
2.5.1.3	Readable representation of time values	32
2.5.2	Standard representation of the data in the input and output files	33
2.6	Tape backups with HSMS and ARCHIVE	36
2.6.1	Controlling an HSMS run with COPY and RECOVER	36
2.6.1.1	Setting up an HSMS archive for SESAM/SQL	37
2.6.1.2	Creating the HSMS parameter file	43
2.6.1.3	Deleting HSMS requests via the HSMS user interface	51
2.6.1.4	Changing the user ID, the pubset or the system	51
2.6.2	Controlling an ARCHIVE run with COPY and RECOVER	56
2.6.2.1	Creating an ARCHIVE parameter file	56
2.7	Processing input, output and error files	65
2.8	User IDs and link names	67

3	Utility statements	69
3.1	Overview of the utility statements (in alphabetical order)	70
3.2	Alphabetical reference section	71
	ALTER CATALOG - Modify the properties of a database	73
	ALTER DATA FOR TABLE - Shuffle column values, anonymize data	74
	ALTER MEDIA DESCRIPTION - Modify the media table	76
	ALTER PARTITIONING FOR TABLE - Modify the partitioning of a base table	80
	CHECK CONSTRAINTS - Check integrity constraints	91
	CHECK FORMAL - Check the format of tables and indexes	94
	COPY - Create a SESAM backup copy	97
	CREATE CATALOG - Create a database (catalog space)	105
	CREATE MEDIA DESCRIPTION - Define file attributes of database-specific files	115
	CREATE REPLICATION - Create replication	119
	DROP MEDIA DESCRIPTION - Delete all the media table entries for database-specific file type	127
	EXPORT TABLE - Export a table from a database to an export file	129
	IMPORT TABLE - Import a table from an export file into a database	133
	LOAD - Load user data into a base table	140
	MIGRATE - Migrate databases and tables	166
	MODIFY - Maintain metadata for SESAM backup copies	171
	RECOVER - Reset and repair a database, rebuild indexes	178
	Reset and repair of user spaces and the database	178
	Rebuild indexes	217
	REFRESH REPLICATION - Update a replication	219
	REFRESH SPACE - Extend a replication	223
	REORG - Reorganizing spaces and base tables	227
	REORG [CATALOG_]SPACE - Reorganize the catalog space and user spaces	228
	REORG ONLINE TABLE - Reorganize a base table	233
	UNLOAD - Unload user data from a table	234

- 4 Appendix 257**

- 4.1 **Syntax overview 257**
- 4.2 **SQL keywords 263**

- Related publications 273**

- Index 275**

1 Preface

The functions and architectural features of the SESAM/SQL-Server database system meet all the demands placed on a powerful database server in today's world. These characteristics are reflected in its name: SESAM/SQL-Server.

SESAM/SQL-Server is available in a standard edition for single-task operation and in an enterprise edition for multitask operation.

For the sake of simplicity, we shall use the name SESAM/SQL throughout this manual to refer to SESAM/SQL-Server.

The following introductory descriptions are contained centrally in the [“Core manual”](#):

- Brief product description
- Structure of the SESAM/SQL server documentation
- Demonstration database
- Readme file
- Changes since the last editions of the manuals

1.1 Objectives and target groups of this manual

This manual is intended for all SQL users who want to use the utility statements to create and maintain SESAM/SQL databases.

It is assumed that you are already familiar with the [“Core manual”](#), in particular with the SESAM/SQL objects and concepts and the utility concept upon which the utility statements are based.

If you plan on embedding utility statements in a program, you must be familiar with the programming language COBOL.

1.2 Summary of contents

This manual, the “SQL Reference Manual, Part 2: Utilities”, contains a complete description of the utility statements and the utility functions that can be implemented using these statements.

The chapter “Basic information on working with utility statements” contains information necessary for executing utility statements.



The chapter “Utility statements” contains an alphabetical reference section containing all the utility statements.

The section “Syntax overview” in the appendix is an alphabetical reference of all the syntax elements used in the manual. Several of the required syntax elements are described in the [“SQL Reference Manual Part 1: SQL Statements”](#).

The section “SQL keywords” in the appendix provides you with a list of keywords used in SQL and utility statements in SESAM/SQL.

1.3 Notational conventions

The following notational conventions are used in this manual:

<hr/> <hr/>	Syntax definitions. Continuation lines within syntax definitions are intended.
UPPERCASE	SQL keywords
<u>underscored</u>	Default values
bold	Used for emphasis in running text
<i>italics</i>	Variables in syntax definitions and running text
Schreibmaschinenschrift	Program text in syntax definitions and examples
::=	Definition character. The specification to the right of ::= defines the syntax of the element on the left.
[]	May be omitted The brackets are metacharacters and must not be entered in an SQL statement.
(multiple line) { }	Alternative specifications in syntax definitions. Each line contains one alternative. The braces are metacharacters and must not be entered in an SQL statement.
{ } (single line)	Encloses clauses in syntax definitions that can be repeated. The braces are metacharacters and must not be entered in an SQL statement.
,...	In syntax definitions, a comma followed by three dots means that you can repeat the preceding specification any number of times, separating each specification with a comma. If you do not repeat a specification, you must omit the comma.
...	In syntax definitions, an ellipsis means that you can repeat the preceding specification any number of times. In examples, the ellipsis means that the rest of the statement is of no significance to the example. The ellipsis is a metacharacter and must not be entered in an SQL statement.
	Indicates notes that are of particular importance.
	Indicates warnings.

2 Basic information on working with utility statements

This chapter contains the information you need to execute utility statements, i.e. information on

- [SESAM/SQL utility functions](#)
- [Space states and permissible utility statements](#)
- [CHECK pragma](#)
- [Data representation in the input and output files for LOAD and UNLOAD](#)
- [Tape backups with HSMS and ARCHIVE](#)
- [Processing input, output and error files](#)
- [User IDs and link names](#)

You will find an introduction to the utility concept in the [“Core manual”](#).

2.1 SESAM/SQL utility functions

SESAM/SQL provides utility functions by means of the utility statements.

The table below indicates which utility functions SESAM/SQL provides and which utility statements are used to implement them.

Utility function	Utility statement
Create a database (catalog space)	CREATE CATALOG
Modify properties of the database	ALTER CATALOG
Create a SESAM backup copy	COPY
Recover a database, catalog space, user space(s)	RECOVER [USING]
Create a replication	CREATE REPLICATION
Update a replication	REFRESH REPLICATION
Extend a replication	REFRESH SPACE
Reset database, catalog space, user space(s)	RECOVER TO
Rebuild indexes	RECOVER INDEX
Load user data into a base table	LOAD
Unload user data from a base table	UNLOAD
Shuffle column values, anonymize user data	ALTER DATA FOR TABLE
Export table from database to an export file	EXPORT TABLE
Import table from export file into database	IMPORT TABLE
Reorganize catalog space, user spaces or base tables	REORG
Changing the partitioning of a base table	ALTER PARTITIONING FOR TABLE
Check integrity constraints	CHECK CONSTRAINTS
Check format of tables and indexes	CHECK FORMAL
Edit media table	ALTER MEDIA DESCRIPTION CREATE MEDIA DESCRIPTION DROP MEDIA DESCRIPTION
Maintain information (metadata) on the SESAM backup copies	MODIFY
Migrate database and tables	MIGRATE

Table 1: SESAM/SQL utility functions

The individual utility statements are described in detail in [chapter “Utility statements” on page 69](#).

2.2 Space states and permissible utility statements

Except for the utility statements MODIFY and ALTER/CREATE/DROP MEDIA DESCRIPTION, SESAM/SQL initially starts executing a utility statement exclusively on one or more spaces (catalog space, user spaces). Depending on the utility statement involved, read or update accesses by other SESAM/SQL users on a space affected by the utility statement are either possible, possible to a certain extent or not possible at all during execution of the statement (see the [“Core manual”](#)). Locks set for user spaces, tables and indexes during the execution of a utility statement remain in place after execution of the utility statement in certain cases and result in a user space being placed in different states.

In SESAM/SQL, a user space can assume the following space states:

- | | |
|-------------------|--|
| “space o.k” | This state indicates that the user space can be processed without restriction and does not contain any locked tables or indexes. |
| “check pending” | <p>This state indicates that at least one base table in the user space is in the “check pending” state. A base table is in the “check pending” state if at least one integrity constraint for this base table has been violated, or if the integrity constraints for this table have not yet been checked after IMPORT TABLE, LOAD or RECOVER (if the current state could not be retrieved).</p> <p>Base tables in the “check pending” state can only be read with the CHECK OFF pragma (see page 24).</p> <p>They can be changed with DML statements.</p> |
| “copy pending” | This state indicates that you need to make a SESAM backup copy of the user space with the utility statement COPY before it can be updated again. |
| “load running” | <p>This state indicates that records are being loaded into a table in the user space with MIGRATE, LOAD OFFLINE or IMPORT TABLE.</p> <p>The space cannot be updated. Tables that are not affected by the load operation can be read.</p> <p>If the load operation is aborted with an error then the user space remains in this state and has to be retrieved using RECOVER.</p> |
| “recover pending” | This state indicates that SESAM/SQL, on repairing the user space with the utility statement RECOVER, was not able to apply the modifications logged in the DA-LOG files up to the current state. |
| “space defect” | This state indicates that the user space can only be repaired using media recovery procedures. |

“reorg pending” This state indicates that the maximum space size (64 Gbytes) has been reached. The space cannot be expanded.

The following SQL statements can still be executed in this state:

- DELETE to delete individual records
- Statements for querying data
- DROP TABLE DEFERRED
- DROP INDEX DEFERRED
- REORG SPACE

Reorganizing the user space (REORG SPACE) at best brings short-term relief.

For the long term, the load on the space should be relieved by deleting records which are not needed, relocating tables and indexes or partitioning tables.

The “index pending” state is no longer used as of SESAM/SQL V.3.1. Corrupt indexes are ignored on COPY and CREATE REPLICATION (see [section “Alphabetical reference section” on page 71](#)).

The following space states can occur simultaneously. They have to be processed in the following order:

1. “load running”
2. “check pending”
3. “copy pending”

As far as the user is concerned, only the states “space o.k.” and “space defect” are of interest for a catalog space. If a catalog space is in the state “space defect”, media recovery procedures must be performed (see the “[Core manual](#)”)

The [table 2](#) indicates which states a user space can have, which utility statements are permitted in the individual states, and which state the user space has after a permitted utility statement has been executed.

The state “space defect” can always occur and has therefore not been included in [table 2](#) as a resulting state.

If the resulting space state depends on whether logging has been activated for the user space, this is indicated in [table 2](#) by the information “with logging” or “without logging”.

State of the user space	Permitted utility statements	Possible new states of the user space
“space o.k.”	ALTER DATA FOR TABLE	– “copy pending” (with logging)
	ALTER PARTITIONING FOR TABLE	– “space o.k.” – “copy pending” if records were moved or deleted (with logging)
	CHECK CONSTRAINTS	– “space o.k.” – “check pending”
	CHECK FORMAL, COPY, EXPORT TABLE, UNLOAD, REORG	– “space o.k.”
	LOAD OFFLINE	– “check pending” – “copy pending” (with logging) – “load running”, while records are being loaded or after LOAD OFFLINE is aborted due to an error. – “space o.k.” (without logging)
	LOAD ONLINE	– “check pending” – “space o.k.”
	IMPORT TABLE	– “copy pending”, if records are being imported (with logging). – “load running”, while records are being imported or after IMPORT TABLE is aborted due to an error. – “space o.k.”, if no records are being imported or if records are being imported with logging deactivated
MIGRATE	– “copy pending” (with logging) – “load running”, while records are being migrated or after MIGRATE is aborted due to an error. – “space o.k.” (without logging)	

Table 2: Space states and permissible utility statements

(part 1 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
"space o.k." (continued)	RECOVER	<ul style="list-style-type: none"> - "recover pending" - "check pending" - "copy pending" <li style="padding-left: 20px;">after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement. <li style="padding-left: 20px;">or <li style="padding-left: 20px;">after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. - "space o.k."
	RECOVER INDEX	<ul style="list-style-type: none"> - "copy pending" (with logging) - "space o.k." (without logging)
"check pending"	ALTER DATA FOR TABLE (The table itself must not have the state "check pending")	<ul style="list-style-type: none"> - "check pending"
	ALTER PARTITIONING FOR TABLE (The table itself must not have the state "check pending")	<ul style="list-style-type: none"> - "check pending"
	CHECK CONSTRAINTS ON TABLE CHECK CONSTRAINTS ON SPACE	<ul style="list-style-type: none"> - "space o.k."
	CHECK CONSTRAINTS (for individual integrity constraints), UNLOAD	<ul style="list-style-type: none"> - "check pending"
	EXPORT TABLE (The table itself must not have the state "check pending")	<ul style="list-style-type: none"> - "check pending"

Table 2: Space states and permissible utility statements

(part 2 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
„check pending“ (continued)	IMPORT TABLE	<ul style="list-style-type: none"> – “check pending” – “copy pending”, if records are being imported (with logging). – “load running”, while records are being imported or after IMPORT TABLE is aborted due to an error.
	LOAD OFFLINE	<ul style="list-style-type: none"> – “check pending” – “copy pending” (with logging) – “load running”, while records are being loaded or after LOAD OFFLINE is aborted due to an error. – “space o.k.” (without logging)
	LOAD ONLINE	<ul style="list-style-type: none"> – “check pending” – “space o.k.”
	MIGRATE	<ul style="list-style-type: none"> – “check pending” – “copy pending” (with logging) – “load running”, while records are being migrated or after MIGRATE is aborted due to an error.
	RECOVER	<ul style="list-style-type: none"> – “copy pending” after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement. or after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. – “check pending” – “space o.k.” – “recover pending”
	RECOVER INDEX	<ul style="list-style-type: none"> – “check pending” – “copy pending” (with logging)

Table 2: Space states and permissible utility statements

(part 3 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
"copy pending"	ALTER DATA FOR TABLE	<ul style="list-style-type: none"> – "copy pending" – "load running", while column values are being shuffled or after ALTER DATA FOR TABLE is aborted due to an error.
	ALTER PARTITIONING FOR TABLE	<ul style="list-style-type: none"> – "copy pending" – "load running", while records are being moved or after ALTER PARTITIONING FOR TABLE is aborted due to an error.
	CHECK CONSTRAINTS, CHECK FORMAL, UNLOAD, EXPORT TABLE	<ul style="list-style-type: none"> – "copy pending"
	COPY	<ul style="list-style-type: none"> – "space o.k."
	IMPORT TABLE	<ul style="list-style-type: none"> – "copy pending" – "load running", while records are being imported or after IMPORT TABLE is aborted due to an error.
	LOAD OFFLINE	<ul style="list-style-type: none"> – "check pending" – "copy pending" – "load running", while records are being loaded or after LOAD OFFLINE is aborted due to an error.
	MIGRATE	<ul style="list-style-type: none"> – "copy pending" – "load running", while records are being migrated or after MIGRATE is aborted due to an error.

Table 2: Space states and permissible utility statements

(part 4 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
"copy pending" (continued)	RECOVER	<ul style="list-style-type: none"> – "copy pending" after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement. or after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. – "check pending" – "space o.k." – "recover pending"
"space defect"	RECOVER	<ul style="list-style-type: none"> – "copy pending" after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement. or after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. – "check pending" – "space o.k." – "recover pending"

Table 2: Space states and permissible utility statements

(part 5 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
"recover pending"	RECOVER	<ul style="list-style-type: none"> - "copy pending" after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement, or after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. - "check pending" - "space o.k."
	UNLOAD OFFLINE	<ul style="list-style-type: none"> - "recover pending"
"load running"	RECOVER	<ul style="list-style-type: none"> - "copy pending" after RECOVER USING if the changes must be applied with CREATE INDEX or DROP INDEX for a partitioned table on this space, but not all partitions are contained in the RECOVER statement, or after RECOVER TO if the user space (with logging) had to be adjusted to definitions in the catalog space or after RECOVER ADJUST. - "check pending" - "space o.k."
	UNLOAD (The processed table must not itself have the state "load running") EXPORT (The processed table must not itself have the state "load running")	<ul style="list-style-type: none"> - "load running"

Table 2: Space states and permissible utility statements

(part 6 of 7)

State of the user space	Permitted utility statements	Possible new states of the user space
"reorg pending"	REORG SPACE	<ul style="list-style-type: none"> – "space o.k." – "reorg pending" Reorganizing the user space at best brings short-term relief.
	UNLOAD EXPORT TABLE CHECK FORMAL CHECK CONSTRAINTS	<ul style="list-style-type: none"> – "reorg pending"
	RECOVER	<ul style="list-style-type: none"> – "space o.k." – "reorg pending" – "check pending" – "copy pending" – "recover pending" Recovery of the user space at best brings short-term relief.

Table 2: Space states and permissible utility statements

(part 7 of 7)

2.3 Utility statements with partitioned tables

Many utility statements process the user spaces on which the data of a database is stored. The user spaces must be available if they are to be processed, i.e. the utility statement must be able to access them.

The data of a partitioned table is distributed to the so-called partitions on a number of user spaces. It can occur that one or more partitions or user spaces of the partitioned table are not available. Here a distinction is also made between logical and physical availability. Basic information on partitioned tables and the availability of partitions is provided in the “[Core manual](#)”.

The table below shows the requirements for the availability of the partitions for the various utility statements and how the response is in the event of an error, i.e. if at least one partition does not satisfy the availability requirement.

Statement	Requirement for the availability of the partitions	Response in event of error/ special measures if req.
ALTER DATA FOR TABLE	All partitions of the table, logical and physical	Rejection with SQLSTATE
ALTER PARTITIONING FOR TABLE	Partitions affected, logical and physical	Rejection with SQLSTATE
CHECK CONSTRAINTS ON SPACE	All partitions of the tables which have a partition on the space, logical and physical	Rejection with SQLSTATE
CHECK CONSTRAINTS ON TABLE	All partitions of the table, logical and physical	Rejection with SQLSTATE
CHECK FORMAL SPACE	All partitions of the space, physical	Rejection with SQLSTATE
CHECK FORMAL TABLE	All partitions of the table, physical	Message in error file and SQL warning
EXPORT TABLE ... ALL DATA	All partitions, physical	Rejection with SQLSTATE
EXPORT TABLE ... WHERE	Partitions affected, logical and physical	Rejection with SQLSTATE
IMPORT TABLE	All partitions, physical	Rejection with SQLSTATE

Table 3: Availability of partitions in the case of utility statements

(part 1 of 2)

Statement	Requirement for the availability of the partitions	Response in event of error/ special measures if req.
LOAD OFFLINE with counting field	All partitions, physical	Rejection with SQLSTATE
LOAD OFFLINE without counting field	Partitions affected, physical	Message in error file
... with GENERATE INDEX	All partitions, physical	SQL warning, indexes are set to "defective". RECOVER INDEX ON TABLE needed
... with CONSTRAINT CHECK	All partitions, logical and physical	Table in the "check pending" state. CHECK CONSTRAINTS needed
LOAD ONLINE with counting field	All partitions, logical and physical	Rejection with SQLSTATE
LOAD ONLINE without counting field	Partitions affected, logical and physical	Rejection with SQLSTATE
LOAD ONLINE with CHECK CONSTRAINTS	All partitions, logical and physical	Rejection with SQLSTATE
MIGRATE CALL DML TABLE	All partitions, physical	Rejection with SQLSTATE
RECOVER ... [USING] ... TO and NO INDEX not specified	All partitions, physical	Rejection with SQLSTATE
RECOVER ... [USING] ... TO and integrity constraint defined for the table	All partitions, physical	Rejection with SQLSTATE
RECOVER INDEX ON TABLE	All partitions, physical	Rejection with SQLSTATE
UNLOAD OFFLINE	All partitions, physical	Message in error file
UNLOAD ONLINE	Partitions affected, logical and physical	Rejection with SQLSTATE
Other UTILITY statements	None	---

Table 3: Availability of partitions in the case of utility statements

(part 2 of 2)

2.4 CHECK pragma

Syntactically, a pragma is like a normal SQL comment which SESAM/SQL interprets as a message to itself.

The CHECK pragma defines whether the SQL or utility statement containing the pragma can be used to process tables in the “check pending” state (see [page 13](#)). The effect of the CHECK pragma is limited to the statement containing the pragma. If the CHECK pragma occurs several times in a statement, the last pragma specified is valid.

Within a statement, the CHECK pragma is only effective if only separators (including pragmas and SQL comments) are located before the pragma in the statement involved. You will find a description of the separators in the “[SQL Reference Manual Part 1: SQL Statements](#)”.

The CHECK pragma can be used in conjunction with the following SQL statements (see the “[SQL Reference Manual Part 1: SQL Statements](#)”):

- CALL and in routines
- DECLARE CURSOR
- SELECT statement
- SELECT statement (dynamically compilable)
- DELETE ... WHERE CURRENT OF
- DELETE ... WHERE CURRENT OF (dynamically compilable)
- DELETE ... WHERE *search_condition*
- DELETE ... WHERE *search_condition* (dynamically compilable)
- INSERT
- MERGE
- UPDATE ... WHERE CURRENT OF
- UPDATE ... WHERE CURRENT OF (dynamically compilable)
- UPDATE ... WHERE *search_condition*
- UPDATE ... WHERE *search_condition* (dynamically compilable)

The CHECK pragma can be used in conjunction with the following utility statement:

- UNLOAD ONLINE (see [page 234](#))

Specifying the CHECK pragma in any other statement has no effect.

```
-- %PRAGMA CHECK { ON }  
                  { OFF }
```

ON

SESAM/SQL takes integrity constraints into account when the SQL or utility statement is executed. Because SESAM/SQL uses integrity constraints (e.g. unique constraints) to optimize statements, SESAM/SQL will reject a statement with SQLSTATE if you specify CHECK ON and one of the base tables referenced either directly or indirectly (via views or referential constraints) is in the state “check pending”.

OFF

SESAM/SQL ignores integrity constraints when the SQL or utility statement is executed. This allows you to process base tables that are in the state “check pending”, i.e. tables whose integrity constraints may not be satisfied. SESAM/SQL will reject an update SQL statement (INSERT, UPDATE, MERGE, DELETE) with SQLSTATE if you specify CHECK OFF and the table to be processed is not in the “check pending” state.

2.5 Data representation in the input and output files for LOAD and UNLOAD

The utility statement LOAD loads data from an input file to the database, and the utility statement UNLOAD writes data from the database to an output file.

There are a number of different options for representing data in an input file or output file:

- Very convenient options are the readable representation of data in delimiter format or in CSV format. Do this by specifying the clause DELIMITER_FORMAT or CSV_FORMAT in the LOAD or UNLOAD statement.

The values are contained in the input or output record, truncated to their significant length. The DELIMITER character indicates the end of the value.

The output file's coded character set (CCSN) is used to represent the values. In delimiter format you can also edit input and output files with the CCSN UTFE.

For files with the EBCDIC or UTFE character set this section of the manual uses the term "readable presentation" (readable with an editor). UTFE files, for example, can be read in BS2000 with EDT V17.0 or higher or in Windows with the openFT editor V10.0 or higher. In the case of files with the EBCDIC character set, the term "printable representation" is also traditionally used.

- Another way of representing the data in readable form is to specify the CHARACTER data type in the self-defined output format of the LOAD or UNLOAD statement.
- Another option is to represent the data in the format of the corresponding table column or in a format which is compatible with the format of the corresponding table column. To do this, describe the individual data formats explicitly in the LOAD or UNLOAD statement using the required data type. The representation of data in the table column format is referred to as the standard representation (see [section "Standard representation of the data in the input and output files" on page 33](#)).

2.5.1 Readable representation of the data in the input and output files

In many cases it is convenient to predefine the input data for LOAD and the output data of UNLOAD in the input/output file in a non-hexadecimal, encrypted form which can be read with an editor.

In addition to the data conversion provided in SQL, which is based on the compatibility between data types (see the “[SQL Reference Manual Part 1: SQL Statements](#)”), in the case of the LOAD and UNLOAD utility statements SESAM/SQL offers the conversion of data in a database to character strings.

2.5.1.1 Readable representation of the input data for LOAD

As far as possible SESAM/SQL converts the input file’s alphanumeric strings in accordance with the CCSN of the input file to the data type of the corresponding column in the table. This data type can be any of the SQL data types.

Apart from the exceptions described below, the input values must conform to the usual conventions for representing values of this data type in order to ensure correct conversion (see the “[SQL Reference Manual Part 1: SQL Statements](#)”).

Representation in delimiter format

Example

```
100;Siemens-AG;Otto-Hahn-Ring 6;81739;...
101;Login GmbH;Rosenheimer Str. 34;81667;...
```

Specify the DELIMITER_FORMAT clause with the DELIMITER character ';' for an EBCDIC file or 'N' for a UTFE file in the LOAD statement.

Rules for input files in delimiter format

- The values must be contained in the input file in their significant length.
- The values must be separated by DELIMITER characters.
- The delimiter must not be contained in any of the values in the input file, otherwise the value will not be interpreted correctly.
- The maximum length and size of a value depends on the target data type. If the maximum value is exceeded, conversion is not possible.
- A value may be longer than the defined length of the affected column of data type (NATIONAL) CHARACTER (VARYING); in the case of LOAD, however, the rest is truncated and a warning is issued.
- The input file may also have CCSN=UTFE.

Representation in CSV format

Example

```
100,"Siemens-AG",Otto-Hahn-Ring 6,81739,...
101,"Login,Logout GmbH",Rosenheimer!,Aiblinger Str.,81667,...
```

Specify the CSV_FORMAT clause with the DELIMITER character ',' (comma) for an EBCDIC file or N',' for a UTFE file in the LOAD statement. You also specify the QUOTE character "" and the ESCAPE character "!".

Rules for input files in CSV format

Fundamental information on the layout of CSV files is provided in the ["SQL Reference Manual Part 1: SQL Statements"](#).

- The values must be contained in the input file in their significant length.
- The values must be separated by DELIMITER characters.
- A record in a SAM file can contain multiple input lines separated by NEWLINE characters.
- When the SAM record ends with an ESCAPE character, the input line is regarded as not having been terminated and is continued with the following SAM record. An input line can thus extend over multiple SAM records. (Prerequisite: the ESCAPE character is specified in the LOAD statement.)
- Column values can contain the DELIMITER character or the NEWLINE character. The column values must then be enclosed in QUOTE characters. (Prerequisite: the QUOTE character is specified in the LOAD statement.)
The escape sequence *escape delimiter* in a column value is also interpreted as a DELIMITER character. (Prerequisite: the ESCAPE character is specified in the LOAD statement.)
- The input file may also have CCSN=UTFE.

Representation in the self-defined output format

Example

```
100Siemens-AG   Otto-Hahn-Ring 6       81739...
101Login GmbH  Rosenheimer Str. 34    81667...
```

Specify the data type CHARACTER in the load format for the LOAD statement for input values whose representation is formatted.

2.5.1.2 Readable representation of the output data for UNLOAD

In the case of an output with the data type CHARACTER, as far as possible SESAM/SQL converts the output values from a column in a table of any SQL data type to strings of the output file in accordance with the latter's CCSN. National values which cannot be converted in accordance with the database's CCSN result in an entry in the error file. Alternately, output can take place with the data type NCHAR.

Apart from the deviations described on [page 31](#), SESAM/SQL stores the output values in the output file in accordance with the SQL conventions for representing values of the corresponding data type.



The output file is always assigned the database's CCSN, even when all values are output in data type NCHAR, for example. You can read and edit an output file of this type with NCHAR data using EDT V17.0 or higher or the openFT editor as of V10.0.

Representation in delimiter format

Example

```
100;Siemens-AG;Otto-Hahn-Ring 6;81739;...
101;Login GmbH;Rosenheimer Str. 34;81667;...
```

Specify the DELIMITER_FORMAT clause with the DELIMITER character ';' for an EBCDIC file or N';' for a UTFE file in the UNLOAD statement.

Rules for output files in delimiter format

- Values are only written to the output file to their significant length.

Exceptions

- Values whose columns are defined as having the data type (NATIONAL) CHARACTER VARYING are output in the length stored and without a length field. Leading or trailing blanks are retained.
- Values whose columns are defined as having the data type (NATIONAL) CHARACTER are output with leading blanks, but without trailing blanks. If a value comprises blanks only, exactly one blank is written to the output file.
- The output file can also be created with CCSN=UTFE.

Representation in CSV format

Example

```

cust_num,company,street,zip,...
100,Siemens-AG,Otto-Hahn-Ring 6,81739,...
101;"Login,Logout GmbH";Rosenheimer Str. 34;81667;...

```

Specify the CSV_FORMAT clause with the DELIMITER character ',' for an EBCDIC file or N',' for a UTFE file in the UNLOAD statement. You also specify the QUOTE character "" and the ESCAPE character "!". The header line with the column names is also displayed when WITH HEADER is specified.

Rules for output files in CSV format

Fundamental information on the layout of CSV files is provided in the [“SQL Reference Manual Part 1: SQL Statements”](#).

- Values are only written to the output file to their significant length.
- The values must be separated by DELIMITER characters.
- When column values contain the DELIMITER character or the NEWLINE character, these column values are enclosed in QUOTE characters when a QUOTE character is specified.
When no QUOTE character is specified but an ESCAPE character is, the DELIMITER character is prefixed with an ESCAPE character.
A NEWLINE character can then not be output.
- The output file can also be created with CCSN=UTFE.

Representation in the self-defined output format

Example

```

100Siemens-AG   Otto-Hahn-Ring 6      81739...
101Login GmbH  Rosenheimer Str. 34   81667...

```

In the relevant output format of the UNLOAD statement, specify the data type CHARACTER for output values that are to be represented in formatted form. Make sure that the length specification for CHARACTER is such that every possible value in the assigned column of the table can be represented by the CHARACTER string. Formatted output of INTEGER values, for instance, requires up to 11 characters.

Deviations from the representation of values at the SQL interface

The representation of values at the SQL interface deviates from that described in the “[SQL Reference Manual Part 1: SQL Statements](#)” (e.g. under INSERT) as follows:

- strings, i.e. alphanumeric (EBCDIC) strings and Unicode strings, are represented without single quotes
- multiple values are represented without angle brackets
- the rules described below in the [section “Readable representation of time values”](#) apply to the representation of time values.

2.5.1.3 Readable representation of time values

You enter time values, i.e. values for the data types DATE, TIME(3) and TIMESTAMP(3), in the input file for LOAD in a different way to that laid down by the conventions at the SQL interface: the keywords DATE, TIME and TIMESTAMP are not part of a time value for LOAD. The same applies to the output of time values for UNLOAD.

Representation of values of the type DATE

Values of the type DATE are represented as follows:

year-month-day

<i>year</i>	Four-digit integer between 0001 and 9999 indicating the year.
<i>month</i>	Two-digit integer between 01 and 12 indicating the month.
<i>day</i>	Two-digit integer between 01 and 31 (depending on the month and year) indicating the day.

Example

2010-06-09 is the value for the date June 9, 2010.

Representation of values of the type TIME(3)

Values of the type TIME(3) are represented as follows:

hour:minute:sec.millisecond

<i>hour</i>	Two-digit integer between 00 and 23 indicating the hour.
<i>minute</i>	Two-digit integer between 00 and 59 indicating the minutes.
<i>sec</i>	Two-digit integer between 00 and 61 indicating the seconds.
<i>millisec</i>	Three-digit integer between 000 and 999 indicating the thousandths of a second.

Example

19:30:25.000 is the value for the time 19 hours, 30 minutes, 25 seconds.

Representation of values of the type TIMESTAMP(3)

Values of the type TIMESTAMP(3) are represented as follows:

year-month-day hour:minute:sec.millisecond

The explanations given under DATE and TIME(3) apply.

2.5.2 Standard representation of the data in the input and output files

This section is only of significance for you if you do not wish to work with readable input or output data. In this case, you describe the individual data formats explicitly using a data type which is compatible with the data type of the corresponding column in the table or you implicitly use the data type of the corresponding column in the table.

You should note that in the case of the standard representation, all values are represented in full according to the data type definition. This also applies to values which are not significant. For example, UNLOAD generates values in the output file for columns, which have the NULL value in the table, according to the rules described in WHEN NULL THEN (see [page 243](#)). The only exceptions are values of data type (NATIONAL) CHARACTER VARYING; these have always a significant length (the length field is decisive); accordingly, UNLOAD, for example, only sets NULL values that are to be represented as (NATIONAL) CHARACTER VARYING, to a length field with the content 0 in the absence of the WHEN NULL THEN clause.

It is particularly important to note the special nature of NULL values for the standard representation of the data when the output file of an UNLOAD is to serve as the input file of a subsequent LOAD. Given that NULL values of a certain length are also represented in the case of UNLOAD, it is important to prevent these values from being interpreted as significant in the subsequent LOAD. To do this, you must specify the appropriate WHEN ... THEN NULL constraints for LOAD, unless the table to be used for loading is a CALL-DML table and the affected value is the non-significant attribute value of the corresponding column.

In [table 4](#) you will find the data types available for the load formats (LOAD) and unload formats (UNLOAD) and information on how the corresponding values have to be represented in the input file and what representation SESAM/SQL uses to write these values to the output file for UNLOAD.

You specify the data type in the format descriptions for LOAD and UNLOAD just as you do at the SQL interface (see the “[SQL Reference Manual Part 1: SQL Statements](#)”) with the following exceptions:

- You cannot specify the data type FLOAT. You must use REAL or DOUBLE PRECISION instead of FLOAT.
- You cannot specify *dimension* (number of column elements for multiple columns).

Data type	Length of the value in the file	Standard representation in the input and output files
CHARACTER	max. 32 000 bytes (256 characters)	Any alphanumeric (EBCDIC) string
CHARACTER VARYING	max. 32 000 bytes (32000 characters) preceded by: 2 bytes (as length field)	Length field (contains the length of the subsequent string in binary representation); any alphanumeric (EBCDIC) string
NATIONAL CHARACTER	max. 256 bytes (128 code units)	Any Unicode string
NATIONAL CHARACTER VARYING	max. 32000 bytes (16000 code units) preceded by: 2 bytes (as length field)	Length field (contains the length, in bytes , of the subsequent string in binary representation); any Unicode string
NUMERIC	max. 31 bytes (number of digits)	Unpacked representation, representation of sign in the first half-byte of the last byte. Example Representation of 364 with 3 digits: X'F3 F6 F4' Representation of -364 with 3 digits: X'F3 F6 D4' Representation of 364 with 4 digits: X'F0 F3 F6 F4'
DECIMAL	max. 16 bytes ([no. of digits/2]) [. . .] means "next-highest integer"	Packed representation, representation of sign in the second half-byte of the last byte. Example Representation of 364 with 3 digits: X'36 4F' Representation of -364 with 3 digits: X'F3 F6 4D' Representation of 364 with 4 digits: X'00 36 4F'
INTEGER	4 bytes	Binary representation, representation of the sign in the first bit (representation of negative numbers as a twos complement). Example Representation of 364 as X'00 00 01 6C' Representation of -364 as X'FF FF FE 94'
SMALLINT	2 bytes	Binary representation, representation of the sign in the first bit (representation of negative numbers as a twos complement). Example Representation of 364 as X'01 6C' Representation of -364 as X'FE 94'

Table 4: Representation of the data in the input file for LOAD and in the output file for UNLOAD (part 1 of 2)

Data type	Length of the value in the file	Standard representation in the input and output files
REAL	4 bytes	Representation according to BS2000 conventions, i.e.: binary representation of the exponent in the first byte with representation of the exponent's sign in the first bit (representation of negative numbers as a twos complement). Binary representation of the mantissa in bytes 2 to 4. Representation of the mantissa's sign in the first bit of byte 2 (representation of negative numbers as a twos complement).
DOUBLE PRECISION	8 bytes	Representation according to BS2000 conventions, i.e.: binary representation of the exponent in the first byte with representation of the exponent's sign in the first bit (representation of negative numbers as a twos complement). Binary representation of the mantissa in bytes 2 to 8. Representation of the mantissa's sign in the first bit of byte 2 (representation of negative numbers as a twos complement).
DATE	6 bytes	A pair of bytes is used to represent the year, the month, the day and is to be interpreted as SMALLINT. Example DATE'1994-06-08' is represented as X'07CA 0006 0008'
TIME(3)	8 bytes	A pair of bytes is used to represent the hours, the minutes, the seconds, the milliseconds and is to be interpreted as SMALLINT. Example TIME'13:57:19.210' is represented as X'000D 0039 0013 00D2'
TIMESTAMP(3)	14 bytes	The first 6 bytes represent the date, the other 8 bytes represent the time.

Table 4: Representation of the data in the input file for LOAD and in the output file for UNLOAD (part 2 of 2)

2.6 Tape backups with HSMS and ARCHIVE

You can create SESAM backup copies on magnetic tape cartridge by means of the utility statement COPY ... USING DIRECTORY via the HSMS or ARCHIVE software products.

If an HSMS parameter file is present then the name specified for COPY ... USING DIRECTORY is interpreted as the HSMS archive. Otherwise the backup is performed using ARCHIVE.

When you retrieve the backup using RECOVER or CREATE REPLICATION, the correct procedure is determined by the metadata that COPY writes in the CAT-REC file or the catalog space.

You can find basic information about creating SESAM backup copies in the [“Core manual”](#).

2.6.1 Controlling an HSMS run with COPY and RECOVER

You must perform the following steps before you can use HSMS for SESAM/SQL:

- Set up an HSMS archive
The properties of the archive are determined via the HSMS archive attributes.
- Create a SESAM specific HSMS parameter file
The HSMS parameter file is used to pass parameters which cannot be addressed via the HSMS archive attributes.
If you simply want to use default values, you must create an empty HSMS parameter file to ensure that HSMS is selected as the backup procedure.

HSMS supports the SECOS software products co-ownership system which allows you to administer access rights to foreign user IDs (see the [“Security Control System - Access Control”](#) manual).

2.6.1.1 Setting up an HSMS archive for SESAM/SQL

If an HSMS archive is to be used with SESAM/SQL then it must satisfy the following requirements:

- The HSMS archive must be used exclusively as a backup archive.
- The execution of the HSMS statements BACKUP-FILES and RESTORE-FILES must not be restricted to specific tape processing times.
- The HSMS archive's directory file must not be processed using the DIRCONV conversion program as otherwise CATIDs in the HSMS archive may be modified. An exception to this rule is the migration from an SF pubset to an SM pubset. In this case it is necessary to use DIRCONV (see [section "Changing the user ID, the pubset or the system" on page 51](#)).

If these conditions are not met, it is possible that backups can no longer be found in the HSMS archive.

You are recommended to set up a private HSMS archive for use by SESAM/SQL. This makes it possible to customize the properties of the HSMS archive for the use of SESAM/SQL precisely and independently of any other requirements.

The HSMS system settings must not contradict the requirements of SESAM/SQL. Tape accesses relating to the HSMS archive that has been set up for SESAM backups must be permitted at all times. To ensure this, the HSMS administrator must use the HSMS statement MODIFY-TAPE-CONTROL to make the following settings:

```
MODIFY-TAPE-CONTROL ARCHIVE-NAME = <hsms_archive_name>  
    ,WRITE-CONTROL = *PROCESS-REQUESTS,READ-CONTROL = *PROCESS-REQUESTS
```

The name *hsms_archive_name* is a maximum of 22 characters long. If tape processing is restricted to a specific period and if a space backup is initiated outside of this period then the space remains locked until the backup has been terminated. This lock affects DML, DDL and utility accesses.

Creating an HSMS archive with CREATE-ARCHIVE

You use the HSMS statement CREATE-ARCHIVE to create an HSMS archive. The HSMS archive must be created on the DBH user ID.

The list below contains all the CREATE-ARCHIVE attributes that are of relevance for a backup archive suitable for use with SESAM/SQL.

Wherever parameters or specific parameter values are mandatory, your attention is drawn to this. For a full description of the operands, refer to the “[HSMS \(BS2000\)](#)” manual.

CREATE-ARCHIVE

```

ARCHIVE-NAME = <filename 1..22 without cat-id>
,ENVIRONMENT = *STD
,OWNER-FIELD = *NONE / <c-string 1..54>
,ALLOWED-USAGE = *BACKUP(...)
    *BACKUP(...)
        |   SAVE-FILE-STRUCTURE = *SEVERAL-SVID
,USER-ACCESS = *ALL-USERS(...)
    *ALL-USERS(...)
        |   ACCESS = *READ / *WRITE
,DIRECTORY-NAME = <filename 1..54>(...)
    <filename 1..54>(...)
        |   NEW-DIRECTORY = *NO / *YES
,RETENTION-PERIOD = <integer 0..32767 days>
,COMPRESS-FILES = *NO / *YES
,S2-DEVICE-TYPE = *STD / <c-string 1..8>
,LOCATION = *ANY / <alphanum-name 1..8>
,OPERATION-CONTROL = *PARAMETERS(...)
    *PARAMETERS(...)
        |   PARALLEL-RUNS = 1 / <integer 1..16>
        |   ,WRITE-CHECKPOINTS = *YES / *NO
        |   ,OPERATOR-INTERACTION = *NOT-ALLOWED / *ALLOWED
,TAPE-CONTROL = *PARAMETERS(...)
    *PARAMETERS(...)
        |   NEW-STD-SAVE-FILE = *IN-PERIODS(...)
        |       *IN-PERIODS(...)
        |           |   CONTINUATION-PERIOD = <integer 1..31 days>
        |       ,BLOCKING-FACTOR = *STD / <integer 1..15 2Kbyte> / *MAX
        |       ,STREAM-MODE = *NO / *YES
        |       ,UNLOAD-TAPE = *NO / *YES

```

Operands

ARCHIVE-NAME = <filename 1..22 without cat-id>

Name of the HSMS archive. This is specified with COPY ... USING DIRECTORY. Excluding the user ID, the length of the name is restricted to 12 characters.

The user ID is preset to the user ID on which the HSMS archive was set up. This must be the DBH user ID. A different ID can only be specified by an HSMS administrator.

ENVIRONMENT = *STD

Designates the HSMS environment in which the archive is to be defined.

This attribute must be set to *STD. *STD is the default environment in which the user is entered in the user catalog.

OWNER-FIELD = *NONE / <c-string 1 .. 54>

The user can enter a comment relating to the HSMS archive that is to be set up here.

The comment can be output using the HSMS statement SHOW-ARCHIVE-ATTRIBUTES specified with INFORMATION=*FULL.

ALLOWED-USAGE = *BACKUP (...)

The HSMS archive is used exclusively as a backup archive for BS2000 files.

SAVE-FILE-STRUCTURE = *SEVERAL-SVID

HSMS creates save files in a SEVERAL-SVID structure.

A save file in a SEVERAL-SVID structure can accommodate a number of different save versions. Data is appended to the current save file until a change of save file is initiated.

The save version identification (SVID) identifies a save version and is specified internally by HSMS.

The save file structure must implement SEVERAL-SVID.

USER-ACCESS = *ALL-USERS (...)

Allows backups to be retrieved to other user IDs that are not owners of the archive.

ACCESS = *READ

Permits read access to backups stored on a user ID other than that of the HSMS archive. The user ID must be present in the environment that was declared for the HSMS archive using the ENVIRONMENT operand.

ACCESS = *WRITE

Permits read and write access to backups stored on a user ID other than that of the HSMS archive. The user ID must be present in the environment that was declared for the HSMS archive using the ENVIRONMENT operand.

DIRECTORY-NAME = <filename 1..54> (...)

This is where information concerning the backup copies that are administered in the HSMS archive is stored. The name may be extended by the caller's user ID and the default catalog ID. A different ID can only be specified by an HSMS administrator.

NEW-DIRECTORY = *YES

HSMS creates a new archive directory. The file for the new archive directory must be empty. However, it can be catalogued.

NEW-DIRECTORY = *NO

An existing directory becomes subject to HSMS administration. The archive directory must previously have been made available with the HSMS function IMPORT-FILES. This option is used when a backup has to be retrieved at another computer or an archive directory has to be taken over into the HSMS archive.

RETENTION-PERIOD = <0..32767 days>

Number of days to be used by default as the physical retention period for the save files.

COMPRESS-FILES = *NO / *YES

Specifies whether or not the data should be compressed before being written to the output medium. The value *YES is recommended for operation with SESAM/SQL.

S2-DEVICE-TYPE = *STD / <c-string 1..8>

Specifies the device type to be used at storage level S2 for backups to this HSMS archive.

LOCATION = *ANY / <alphanum-name 1..8>

Location of the storage medium for the backup. The location must be known to MAREN. If MAREN is not in use then you must specify *ANY. In this case, no location is used for data medium selection.

OPERATION-CONTROL = *PARAMETERS(...)

Parameters that are relevant for the processing sequence.

PARALLEL-RUNS = 1 / <integer 1..16>

Number of backup tasks that can run simultaneously.

During the backup, there must be a tape device available for each task.

WRITE-CHECKPOINTS = *YES / *NO

Specifies whether restart points should be written to the ARCHIVE checkpoint file during processing. These points permit a restart following an abort.

This function can be used for backing up database files which are located on additional mirror units. In this case you should specify *YES (default setting). If this is not the case, specify *NO.

OPERATOR-INTERACTION = *NOT-ALLOWED / *ALLOWED

Specifies whether messages which require a response from the operator should be output at the console.

If *NOT-ALLOWED is specified, HSMS performs default processing.

TAPE-CONTROL = *PARAMETERS(...)

Parameters that apply to the writing of magnetic tape cartridges

NEW-STD-SAVE-FILE = *IN-PERIODS (...)**CONTINUATION-PERIOD = <integer 1..31 days>**

Continuation period after which HSMS automatically creates a new save file. If you specify SAVE-FILES = *STD in the HSMS parameter file then HSMS automatically changes save file when the period specified here expires.

BLOCKING-FACTOR = *STD / <integer 1..15 2Kbyte> / *MAX

Blocking factor used to write the save file to magnetic tape cartridge. You specify the number of 2KB blocks (PAM pages) that are written on magnetic tape cartridge I/O operations. The value depends on the device type.

*MAX selects the maximum blocking factor that is possible in the current BS2000 version. Currently this value is 128 (block size 256 KBytes). Backup files written with the maximum value of a BS2000 version cannot be read in lower BS2000 versions which do not support this blocking factor. *MAX is recommended for use with SESAM/SQL.

STREAM-MODE = *NO / *YES

Specifies whether the magnetic tape cartridge is to run in streaming mode if this is supported by the tape device in question.

The value *YES is recommended for operation with SESAM/SQL.

UNLOAD-TAPE = *NO / *YES

Specifies whether or not a magnetic tape cartridge should be unloaded at the end of processing. This setting only affects the RESTORE-FILES statement since SESAM/SQL sets the UNLOAD-TAPE parameter in the BACKUP-FILES statement. The value *YES is recommended for operation with SESAM/SQL.

Model procedure for setting up an HSMS archive

You can use the model procedure SESAM.P.CREATE.HSMS-ARCHIVE to set up an HSMS archive suitable for use with SESAM/SQL. This procedure is present in the library SIPANY.SESAM-SQL.090.TOOLS.

You can use parameters to specify those archive properties that are user-definable in the light of the SESAM/SQL-specific presettings. When doing this, you must take account of the restrictions described in [section “Setting up an HSMS archive for SESAM/SQL” on page 37](#).

Modifying the attributes of an HSMS archive with MODIFY-ARCHIVE-ATTRIBUTES

You use the statement MODIFY-ARCHIVE-ATTRIBUTES to modify the attributes of an HSMS archive. The following restrictions apply:

- The archive type cannot be changed.
- If you want to use an archive directory for a different HSMS basic function, you must delete the old archive definition and take over the archive directory into a new HSMS archive.

The attributes of the HSMS archive can only be modified in so far as this is stated in the description of the individual operands as otherwise it may no longer be possible to use the HSMS archive for SESAM/SQL.

Modifying an HSMS archive directory with MODIFY-ARCHIVE

You use the statement MODIFY-ARCHIVE to modify an HSMS archive directory. With MODIFY-ARCHIVE, you can

- Modify the additional information for a save version
- Delete save files
- Administer volume pools

Only entire save files can be deleted. When you do this, all the save versions in the save file are also deleted.

2.6.1.2 Creating the HSMS parameter file

The HSMS parameter file is a SESAM-specific SAM file with variable record format for parameterizing the HSMS call. It must be created on the DBH's BS2000 user ID. The HSMS archive itself must also be located on the DBH user ID.

The parameter file may have a write password. A read password is not allowed since the parameter file must be read by the service task.

The HSMS parameter file is opened and read for each COPY, RECOVER, CREATE REPLICATION statement and then closed before HSMS is called. This means that you can modify the file between two different COPY and RECOVER statements.

The name of the HSMS parameter file must be derived either from the path name of the HSMS archive or from the name of the DBH session:

hsms_archive_name.HSMS-PAR

or

SESAM*cn*.HSMS-PAR

where

hsms_archive_name

Name of the HSMS archive specified in the USING DIRECTORY of the COPY statement.

The format corresponds to that of the file name without the catalog ID but the length is limited to 22 characters (or 12 characters without the user ID).

c Configuration name

n DBH name

First of all, SESAM/SQL searches for the file *hsms_archive_name*.HSMS-PAR, and then, across different sessions, the file SESAM*cn*.HSMS-PAR. If neither of the two files are available, a tape backup is performed with ARCHIVE.

If you simply want to use default values, you must create an empty HSMS parameter file to ensure that HSMS is selected as the backup procedure.

Overview of permitted HSMS parameters

In the HSMS parameter file, you can specify parameters that control the behavior of BACKUP-FILES (for COPY statements) and RESTORE-FILES (for RECOVER or CREATE REPLICATION statements).

You can specify the following HSMS parameters in the HSMS parameter file:

Parameter	Meaning
SAVE-FILE	Define subdivision of save files
SAVE-CATREC-COPY	Back up CAT-REC copy
SAVE-ACL	Save ACL entries
SAVE-DIRECTORY	Back up archive directory
HSMS-ARCHIVE-USERID	Current user ID of the HSMS archive
REQUEST-NAME	Name for addressing the request
PARALLEL-RUNS	Number of backup tasks that can run simultaneously
REORGANIZE-SPACE	Reorganize volumes
MAIL	Send report by email

Table 5: Permitted HSMS parameters

In the case of BACKUP-FILES statements, only the specifications SAVE-FILE, SAVE-CATREC-COPY, SAVE-ACL, SAVE-DIRECTORY, PARALLEL-RUNS, REQUEST-NAME, and MAIL are evaluated.

In the case of RESTORE-FILES statements, only the specifications HSMS-ARCHIVE-USERID, PARALLEL-RUNS, REORGANIZE-SPACE, REQUEST-NAME, and MAIL are evaluated.

The default value is assigned to all the parameters that are not specified in the file. If the entire parameter file is empty then the default values apply.

The value that is preset in the HSMS archive definition is used for the PARALLEL-RUNS parameter. The parameter HSMS-ARCHIV-USER-ID does not have a default value. It is only specified if it has changed since the time the backup was created.

Description of the HSMS parameters

The HSMS parameter file can contain entries for an HSMS run involving COPY, RECOVER or CREATE REPLICATION. The specifications are evaluated by SESAM/SQL, but no consistency check is carried out.

The HSMS parameter file may contain the following parameters and parameter values:

```
SAVE-FILE = STD / NEW / CONTINUE
SAVE-CATREC-COPY = YES / NO
SAVE-ACL = YES / NO
SAVE-DIRECTORY = YES / NO
HSMS-ARCHIVE-USERID = <name 1..8>
REQUEST-NAME = <name 1..8>
PARALLEL-RUNS = <integer 1..16>
REORGANIZE-SPACE = YES / NO
MAIL = YES / NO
```

SAVE-FILE

The SAVE-FILE parameter in the parameter file is used to control the SAVE-FILE operand in the BACKUP-FILES statement (see the “[HSMS \(BS2000\)](#)” manual).

After every request, HSMS performs a spool operation that cannot be suppressed by the user. The SAVE-FILE parameter allows you to choose whether this operation should optimize tape utilization or performance.

If the default value is used, spool operations result in wait times. These spool operations occur when a save file is continued with the CAT-REC file after the spaces have been backed up. In this way, tape capacities can be used efficiently.

To avoid wait times, you can create a new save file for every backup job issued to HSMS. In the case of a backup including the CAT-REC copy, at least two tapes are written. In this case, tape capacity is not used efficiently.

SAVE-FILE = STD

SESAM backups are written to the default save file. New backups are appended to the default save file until HSMS initiates a periodic change.

SESAM/SQL calls BACKUP-FILES to perform a backup which includes the CAT-REC copy with the following settings for the SAVE-FILE operand:

<i>SAVE-FILE operand</i>	<i>To back up the</i>
SAVE-FILE = *STD	Spaces
SAVE-FILE = *STD	CAT-REC copy

SAVE-FILE = NEW

Two save files are used for a SESAM backup request.

The spaces are written to one of the save files and the CAT-REC copy to the other.

SESAM/SQL calls BACKUP-FILES to perform a backup which includes the CAT-REC copy with the following settings for the SAVE-FILE operand:

<i>SAVE-FILE operand</i>	<i>To back up the</i>
SAVE-FILE = *NEW	Spaces
SAVE-FILE = *NEW	CAT-REC copy

SAVE-FILE = CONTINUE

A SESAM backup request is called for the spaces and the CAT-REC copy individually and saved in a save file.

SESAM/SQL calls BACKUP-FILES to perform a backup which includes the CAT-REC copy with the following settings for the SAVE-FILE operand:

<i>SAVE-FILE operand</i>	<i>To back up the</i>
SAVE-FILE = *NEW	Spaces
SAVE-FILE = *CONTINUE (SAVE-FILE-ID = *LATEST)	CAT-REC copy

SAVE-CATREC-COPY

This parameter is used to control the backup of the CAT-REC copy.

SAVE-CATREC-COPY = YES

The CAT-REC copy is also backed up. A CAT-REC copy is created only on COPY CATALOG or COPY CATALOG_SPACE.

SAVE-CATREC-COPY = NO

The CAT-REC copy is not backed up.

SAVE-ACL

This parameter is used to control the backup of the ACL entries (up to SECOS V3.0). The ACL entries must be backed up if the backup is to be restored to a different computer.

SAVE-ACL = YES

The ACL entries are backed up.

SAVE-ACL = NO

The ACL entries are not backed up.

SAVE-DIRECTORY

This parameter controls the backup of the archive directory. The archive directory must be backed up if the backup is to be restored to a different computer.

SAVE-DIRECTORY = YES

The archive directory is backed up.

SAVE-DIRECTORY = NO

The archive directory is not backed up.

HSMS-ARCHIVE-USERID = <name 1..8>

This parameter is used to administer the current user ID of the HSMS archive. It must be specified if a backup is to be restored for which the HSMS archive was located on a different user ID at the time of the backup.

The following characters can be used for the user ID of the HSMS archive: A ... Z, 0 ... 9, \$, #, @. You can choose any name provided that it does not start with a digit.

REQUEST-NAME = <name 1..8>

This parameter defines the request name with which the request can be addressed in the HSMS statements for request management.

The value of the REQUEST-NAME parameter in the HSMS parameter file is taken over for the REQUEST-NAME operand of the BACKUP-FILES statement.

If the REQUEST-NAME parameter is not specified in the HSMS parameter file, REQUEST-NAME=*STD is set.

If the volume is assigned using MAREN, the user can use an optional rep to specify the MAREN location from which the volume must come. When this optional HSMS rep is installed, REQUEST-NAME is regarded as a MAREN location and is transferred to MAREN. See also the section "Volume assignment using MAREN" in the "[HSMS \(BS2000\)](#)" manual.

PARALLEL-RUNS = <integer 1..16>

This parameter defines the number of save tasks that can run in parallel. A tape device must be available for each task.

The value of the PARALLEL-RUNS parameter in the HSMS parameter file is transferred to the PARALLEL-RUNS operand of the BACKUP-FILES or RESTORE-FILES statement.

If the PARALLEL-RUNS parameter is not specified in the HSMS parameter file then the corresponding value from the HSMS archive definition applies.

REORGANIZE-SPACE = YES / NO

This parameter determines whether the files are to be deleted before being restored or whether they are to remain present on the disk.

The value of the REORGANIZE-SPACE parameter in the HSMS parameter file is transferred to the REORGANIZE-SPACE operand of the RESTORE-FILES statement.

If the REORGANIZE-SPACE parameter is not specified in the HSMS parameter file then the value *NO is entered for the REORGANIZE-SPACE operand in the RESTORE-FILES statement.

MAIL = YES / NO

This parameter determines whether the report of a BACKUP-FILES or RESTORE-FILES statement is written to a file (MAIL=NO) or whether it is sent by email as an attachment (MAIL=YES). The address of the email recipient must be entered in the caller's user entry.

Syntax rules

- Each line of the parameter file may contain only one parameter in the form *parameter=value*.
- Keywords (parameter names) must not be abbreviated.
- Interspersed blanks are permitted
- Lines which start with an asterisk (*) in column 1 are treated as comments.
- No parameter may be specified more than once

General rules

- The parameter specifications are evaluated each time backups to tape are affected by a COPY, RECOVER or CREATE REPLICATION statement.

In the following situations, the COPY, RECOVER or CREATE REPLICATION statement is aborted and an error message is issued:

- if the HSMS parameter file is not a SAM file.
- if the HSMS parameter file cannot be opened, closed or read because of a DMS error.
- If parameters are specified which cannot be recognized (e.g. if keywords are written incorrectly).
The message text contains the number of the line in which the incorrect parameter specification is located and the name of the HSMS parameter file.
- if incorrect values are specified.
The message text contains the number of the line in which the incorrect parameter specification is located and the name of the HSMS parameter file.

Example of an HSMS parameter file

```
* HSMS PARAMETER FILE SESAM/SQL
* permitted specifications:
*
* SAVE-FILE           = STD / NEW / CONTINUE
* SAVE-CATREC-COPY    = YES / NO
* SAVE-ACL            = YES / NO
* SAVE-DIRECTORY      = YES / NO
* HSMS-ARCHIVE-USERID = <userid hsms archive>
* REQUEST-NAME        = <request name>
* PARALLEL-RUNS       = <integer 1 ... 16>
* REORGANIZE-SPACE    = YES / NO
* MAIL                = YES / NO
*
SAVE-FILE=STD
SAVE-CATREC-COPY=NO
SAVE-ACL=NO
SAVE-DIRECTORY=NO
MAIL=YES
*
```

Use of the HSMS statements BACKUP-FILES and RESTORE-FILES by SESAM/SQL

The HSMS statements BACKUP-FILES and RESTORE-FILES are important when an HSMS archive is used for SESAM backup copies (see the “[HSMS \(BS2000\)](#)” manual).

The BACKUP-FILES statement controls the writing of the spaces and the CAT-REC copy to the HSMS archive when the SESAM/SQL statement COPY is run.

It can take a maximum of two HSMS calls to write a backup to an HSMS archive. In this case, one call backs up the catalog space and the user spaces and the other call backs up the CAT-REC copy.

The RESTORE-FILES statement controls the reading of the catalog space and the user spaces from an HSMS archive.

A number of calls are required to read a backup copy from an HSMS archive. The actual number of calls that are needed depends on the composition of the spaces that are to be restored:

- Restoration of the catalog space
- Restoration of the user spaces
(if the spaces are distributed over multiple pubsets then several calls will be required.)

The name of the log file for both statements is `<dbh_id>.<hsms_archive_name>.SYSLST`. New data is appended to the log file.

HSMS only creates the log file if the statements formulated by SESAM/SQL result in HSMS actions.

The error messages can be found in the service task logs if HSMS has not created a log file because it has rejected statements. HSMS rejects a statement, for example, if a non-existent archive is specified.

2.6.1.3 Deleting HSMS requests via the HSMS user interface

As long as an HSMS request is running, the service task will have the state TPR if

- a SESAM backup copy is being generated with HSMS (COPY statement),
- a SESAM backup copy is being restored with HSMS (RECOVER statement).

This type of request can be deleted from the request file from within the HSMS user interface by means of the DELETE-REQUESTS statement.

However, deletion does not take effect until the HSMS request has terminated and the service task again has the state TU.

As far as SESAM/SQL is concerned, the request has been aborted.

2.6.1.4 Changing the user ID, the pubset or the system

This section describes how you proceed when handling the following special cases of RECOVER and CREATE REPLICATION:

- migrating an HSMS archive to another user ID
- restoring backups to a different pubset or another user ID
- restoring backups to another system

For a description of the relevant parameters, refer to [section “Setting up an HSMS archive for SESAM/SQL” on page 37](#) and [section “Creating the HSMS parameter file” on page 43](#).

To restore a SESAM backup to another user ID or to a different system, you can use a CAT-REC copy that was backed up at the same time.

For this to be possible, the CAT-REC copy must be made available on the DB user ID with the HSMS statement RESTORE-FILES and renamed to the name of the CAT-REC file “<catalog.name>.CAT-REC”. You can then use RECOVER to restore the backup copy from the HSMS archive.

The restored CAT-REC file already contains a CAT-LOG entry that was created after the backup. If you do not want to apply the modifications that are logged here, use the statement RECOVER CATALOG TO to restore the backup.

Alternatively, you can use the utility monitor to restore the entry (see the “[Utility Monitor](#)” manual).

You can use the HSMS statement SHOW-ARCHIVE to list the save versions and backed up files present in the HSMS archive. The required CAT-REC copy is selected on the basis of its creation date.

Co-ownership

When you use HSMS together with SECOS you can specify access rights to files in great detail by means of a co-ownership definition. A distinction is made between read, write and execute rights. To use the co-ownership facility, you must be working with SECOS and HSMS (see the manuals “[Security Control System - Access Control](#)” and “[HSMS \(BS2000\)](#)”).

If you administer SESAM backups with HSMS then setting up co-ownership allows you to store the database on a user ID other than the DBH user ID.

For more information about “Database files and job variables on foreign user IDs“ see the section of the same name in the “[Core manual](#)”.

migrating an HSMS archive to another user ID

When a backup is created with COPY, the name of the HSMS archive and the user ID are stored in the RECOVERY_UNITS. When the backup is restored with RECOVER or CREATE REPLICATION then the HSMS archive is searched for under this user ID.

If, in the meantime, the HSMS archive and thus also the DBH have been moved to a different user ID, this ID must be specified in the HSMS parameter file.

restoring backups to a different pubset or another user ID

The backup copies are stored in the HSMS archive with fully qualified file names, i.e. with catalog ID and user ID.

If REORGANIZE-SPACE=NO is specified then the backups are restored to the user ID that is entered for the database in the database catalog. This is done on the basis of the pubset on which the database file is located at restore time.

In the case of REORGANIZE-SPACE=YES or if the database is not available at restore time, the database is restored to the user ID that is entered for the database in the database catalog. This is done on the basis of the pubset that is stored for this file in the HSMS archive.

To restore a backup to a different user ID, the DBH user ID requires the co-ownership right for the following IDs:

- the user ID on which the database files were located at backup time,
- the user ID to which the backup is to be restored.

It may be advisable to assign partially qualified co-ownership. This ensures that co-ownership is defined for all the affected files.

In the case of CREATE REPLICATION, HSMS uses the original name together with the suffix “.REPL” when restoring the files.

In the second step, SESAM/SQL renames the files to the name of the replication. The new user ID must allow the DBH user ID co-ownership of all the file names used in this process.

If the database is not available at restore time and if a backup is to be restored to a pubset other than the one used for the backup then the files must be created prior to backup using the CREATE FILE command.

If the individual spaces of a database are to be distributed over multiple pubsets when restored and these pubsets are not the ones used for the backup then the files must be created prior to backup using the CREATE FILE command.

Migrating an HSMS archive from an SF pubset to an SM pubset

If the HSMS archive is to be migrated from an SF pubset to an SM pubset then the following steps must be performed before restoring the backup:

- The HSMS archive directory must be provided on the SM pubset.
- In the HSMS archive directory, the catalog IDs must be renamed to the CATIDs of the SM pubset using the DIRCONV statement RENAME-CATID.
- The HSMS archive must be set up on the SM pubset with the same attributes and the newly modified HSMS archive directory.

The backup can then be restored as described on [page 52](#). SESAM/SQL recognizes that the DBH is now running in an SM pubset environment and takes account of this when restoring the backup.

Database files on an SM pubset

You can only back up files located on the same SM pubset using a single HSMS statement BACKUP-FILES. You can therefore only back up database files on the same SM pubset using a SESAM statement COPY.

Restoring backups to another system

You can restore a backup to a different system if the archive directory, the CAT-REC copy and the ACL entries were also backed up. The name of the ID must remain the same if a backup is to be restored to a different system.

The HSMS archive on the target system must be made available in two steps if it is to be possible to access a backup of the archive:

1. The HSMS archive directory is restored to the target system using the HSMS statement **IMPORT-FILES**:

```
IMPORT-FILES
  FILE-NAMES=:<catid>:<bs2000_user_id>.<hsms_directory_name>,
  NEW-FILE-NAMES = *BY-RULE(
    NEW-CATALOG-ID = <new_cat_id>),
  SAVE-FILE = *BY-VOLUME(
    SAVE-FILE-ID = <save_version>,
    VOLUMES = <vs1 1>, ...)
```

You can use the operand `FILE-NAMES=*DIRECTORY` in the above statement.

If the pubset changes then the original pubset must be specified in `FILE-NAMES` and in the current pubset in `NEW-CATALOG-ID`. The save version that is to be restored must be specified in the format `S.yymmdd.hhmmss` in `<save_version>`. The archive numbers of the volumes that contain the original HSMS archive are specified in `<vs1>`,

2. The HSMS archive is set up on the target system with the same attributes as on the original system and with the archive directory that has just been restored.

```
CREATE-ARCHIVE
  ARCHIVE-NAME = <hsms_archive_name>,
  ALLOWED-USAGE = *BACKUP(
    SAVE-FILE-STRUCTURE = *SEVERAL-SVID),
  DIRECTORY-NAME = <hsms_directory_name>(
    NEW-DIRECTORY = *NO),
  TAPE-CONTROL = *PARAMETERS(
    NEW-STD-SAVE-FILE = *IN-PERIODS(
      CONTINUATION-PERIOD = <number_of_days>))
```

The name of the original HSMS archive must be specified for `<hsms_archive_name>`. `<hsms_directory_name>` is the name of the HSMS archive directory that has just been restored.

The CAT-REC.COPY is then restored from the created HSMS archive using the HSMS statement RESTORE-FILES.

```
RESTORE-FILES
  FILE-NAMES =
    :<catid>:<bs2000_user_id>.<catalog_name>.CAT-REC.COPY,
  NEW-FILE.NAMES = *BY-RULE(
    NEW-CATALOG-ID = <new catid>),
  ARCHIVE-NAME = <hsms_archive_name>,
  SELECT-SAVE-VERSIONS = *BY-ATTRIBUTES(
    SAVE-VERSION-DATE = <date> (TIME = <time>))
```

The restored <catalog_name>.CAT-REC.COPY must be renamed to <catalog_name>.CAT-REC.

The environment of the target system must then be checked in terms of pubsets and private disks:

- If the target system has the same environment as the original system, no changes are required.
- If the pubset is different or if there are no private disks present, then the files must be created before the restore using the CREATE-FILE command.

Before the DBH is started, the database can be retrieved via the utility monitor by means of the statement RECOVER CATALOG <catalog_name>. If you do not want to apply the modifications that are logged here, use RECOVER CATALOG TO to restore the backup. Alternatively, you can use the utility monitor to delete the CAT-LOG entry in the CAT-REC file.

2.6.2 Controlling an ARCHIVE run with COPY and RECOVER

If you want to perform a backup using ARCHIVE, specify the name of an ARCHIVE-DIRECTORY in USING DIRECTORY in the COPY statement. There must not be any HSMS parameter file for the archive you want to back up.

The ARCHIVE run is controlled via a SESAM specific ARCHIVE parameter file. The following section describes how you can create this parameter file.

2.6.2.1 Creating an ARCHIVE parameter file

The ARCHIVE parameter file must be a SAM file located on the DBH user ID. The file may have a write password, but must not have a read password, as it needs to be read by the service task.

The ARCHIVE parameter file is opened and read for each COPY or RECOVER statement and then closed before ARCHIVE is called. This means that you can modify the file between two different COPY and RECOVER statements.

The name of the ARCHIVE parameter file must be derived either from the path name of the ARCHIVE directory file or from the name of the DBH session:

archive_directory_name.ARC-PAR

or

SESAM*cn*.ARC-PAR

archive_directory_name

Name of the ARCHIVE directory file you specified in the USING DIRECTORY clause of the COPY statement.

The ARCHIVE directory path name (i.e. *:catid:\$userid.filename*) may be up to 46 characters long, to ensure that the pathname of the ARCHIVE parameter file does not exceed the maximum length of the BS2000 pathname.

c Configuration name

n DBH name

SESAM/SQL first searches for the file *archive-directory-file*.ARC-PAR and then searches for the session-independent file SESAM*cn*.ARC-PAR. If neither of the two files are available, the COPY or RECOVER statement is executed using default values for the ARCHIVE run (see [page 61](#)). If the ARCHIVE parameter file is empty, the ARCHIVE run is aborted with an error message.

Summary of the permitted ARCHIVE parameters

In the ARCHIVE parameter file, you can specify parameters for the ARCHIVE statements PARAM, SAVE and RESTORE (see the “[ARCHIVE \(BS2000\)](#)” manual).

The parameters for the PARAM statement set parameter values for the entire ARCHIVE run started with the COPY and RECOVER utility statements.

The parameters for the SAVE statement set parameters for the backup run started with the COPY utility statement.

The parameters for the RESTORE statement set parameters for the backup run started with the RECOVER and CREATE REPLICATION utility statements.

The ARCHIVE parameter file may also contain the BS2000 command SECURE-RESOURCE-ALLOCATION. You can use this command to reserve resources for the ARCHIVE run and thus prevent the ARCHIVE run failing due to a resource bottleneck (see the “[Commands](#)” manual).

You can specify the following ARCHIVE parameters in the ARCHIVE parameter file:

Parameter	Meaning
<i>For the PARAM statement:</i>	
OPERATOR	Output messages to the console or execute standard processing
STREAM	Control streaming mode
<i>For the SAVE statement:</i>	
DIRSAVE	Save ARCHIVE directory file
DEVICE	Specify device type of volume
SAVE-ACL	Save ACL entries
DRIVES	Specify number of parallel runs
COMPRESS	Compress backup using software
RETPD	Specify retention period in days
<i>For the RESTORE statement:</i>	
SPACE	Reorganize volume or write backup to the space on the disk occupied by the original version
DRIVES	Specify number of parallel runs
ENVIRONMENT-ATTRIBUTES	Specify whether the attributes for access protection are to be taken from the target file or from the backup file

Table 6: Permitted ARCHIVE parameters

Description of the ARCHIVE parameter file

The ARCHIVE parameter file has the same structure as the global ARCHIVE parameter file (see the “[ARCHIVE \(BS2000\)](#)” manual), however, it is extended by the specification of the BS2000 command SECURE-RESOURCE-ALLOCATION.

The ARCHIVE parameter file may contain the following entries for an ARCHIVE run for COPY, RECOVER and CREATE REPLICATION. The specifications are evaluated by SESAM/SQL, but no consistency check is carried out. Make sure, therefore, that the specifications for the ARCHIVE operands, such as the device type, do not contradict the relevant specifications made with the SECURE-RESOURCE-ALLOCATION command.

You will find a full description of the specifications in the ARCHIVE parameter file in the “[ARCHIVE \(BS2000\)](#)” manual along with the corresponding statements. This section merely gives a brief overview of their meaning and points out specific features in connection with COPY, RECOVER and CREATE REPLICATION:

```
SECURE-RESOURCE-ALLOCATION ...  
OPERATOR = NO / YES  
STREAM = NO / YES  
DIRSAVE = NO / YES  
DEVICE = TAPE-C4 / device-type  
SAVE-ACL = NO / YES  
DRIVES = 1 / integer  
COMPRESS = NO / YES  
RETPD = 0 / days  
SPACE = REORG / KEEP  
ENVIRONMENT-ATTRIBUTES = FROM-TARGET / FROM-ORIGIN
```

SECURE-RESOURCE-ALLOCATION ...

This BS2000 command allows you to reserve resources for the ARCHIVE run in advance. The resources are always reserved for each COPY, RECOVER or CREATE REPLICATION statement and then released after execution. This means that it is not possible to reserve the resources before the first of a sequence of COPY, RECOVER or CREATE REPLICATION statements and only release the resources after the last access.

You should only use the operands DEVICE and WAIT, in order to avoid contradictions, since the other operands are generally not required in conjunction with COPY, RECOVER and CREATE REPLICATION.

DEVICE=*PARAMETERS(TYPE=...,NUMBER=...) specifies how many devices of a specific type are to be reserved. Clearly, the specification must match the relevant ARCHIVE parameters.

With WAIT, you specify for how long the system may wait for a job to be completed.

Please refer to the “[Commands](#)” manual for details on the SECURE-RESOURCE-ALLOCATION command.

OPERATOR = NO / YES

This parameter determines whether ARCHIVE processes ARCHIVE messages requiring an operator response using standard processing settings, without outputting the message to the operator terminal or whether ARCHIVE outputs these messages and awaits an operator response.

If you do not specify this parameter, the default value YES for SESAM/SQ applies.

STREAM = NO / YES

This parameter determines whether input and output operations on tape devices are performed in streaming mode.

If you do not specify this parameter, the default value YES for SESAM/SQ applies.

DIRSAVE = NO / YES

This parameter defines whether the current directory file should be saved.

If you do not specify this parameter, the default value for BS2000 applies, namely NO (see the BS2000 default values on [page 61](#)).

DEVICE = TAPE-C4 / *device-type*

This parameter determines the device type (MTC or magnetic tape) for all the VSNs. The DEVICE specification for the SAVE statement must correspond to the specification in the SECURE-RESOURCE-ALLOCATION command to ensure that the reserved devices can be accessed by ARCHIVE. SESAM/SQL does not check this specification.

If you do not specify this parameter, the default value for BS2000 applies, namely TAPE-C4 (see the BS2000 default values on [page 61](#)).

SAVE-ACL = NO / YES

This parameter determines whether the ACL entries for the files to be backed up are stored.

If you do not specify this parameter, the default value for BS2000 applies, namely YES (see the BS2000 default values on [page 61](#)).

DRIVES = 1 / integer

This parameter specifies the number of parallel runs (up to 16). Several devices are operated in parallel. The value for *integer* must be less than or equal to the number of devices available.

If you specify DRIVES=1, SESAM/SQL does not package the data. If you specify DRIVES>1, one packet is formed for each space. ARCHIVE then distributes these packets across the devices. The specification for DRIVES should correspond to the specification for NUMBER in the SECURE-RESOURCE-ALLOCATION command, to ensure that all the devices required are reserved.

In addition, the specification for DRIVES in a RECOVER statement should match the specification in the corresponding COPY statement to ensure that ARCHIVE can distribute the packets across the devices in the same way as the RESTORE statement. If the specifications do not match, ARCHIVE does not make use of parallelism, i.e all the spaces are read in serially.

If you do not specify this parameter, the default value for ARCHIVE statements applies, namely 1.

COMPRESS = NO / YES

This parameter determines whether the files are to be written to the save file using software compression.

If you do not specify this parameter, the default value for BS2000 applies, namely NO (see the BS2000 default values on [page 61](#)).

RETPD = 0 / days

This parameter defines a retention period for the backup version in days. You can enter a value between 0 and 32767 for *days*.

If you do not specify this parameter, the default value for BS2000 applies, namely 0 (see the BS2000 default values on [page 61](#)).

SPACE = REORG / KEEP

This parameter determines whether the files to be replaced by files with the same name from the save run are to be deleted.

If you do not specify this parameter, the default value for BS2000 applies, namely REORG (see the BS2000 default values on [page 61](#))

ENVIRONMENT-ATTRIBUTES = FROM-TARGET / FROM-ORIGIN

This parameter determines whether the attributes for access protection are to be taken from the backup version or the original version.

If you do not specify this parameter, the default value of the ARCHIVE statement RESTORE applies, namely FROM-TARGET.

Syntax rules

- Each line may contain only one parameter in the form *parameter=value*. *Exception*: the SECURE-RESOURCE-ALLOCATION command.
- The SECURE-RESOURCE-ALLOCATION command must be entered as a complete BS2000 command.
If the command extends over a number of lines, a hyphen “-” must be specified as a continuation character before column 72.
The parameter specifications for the SECURE-RESOURCE-ALLOCATION command must not exceed 1736 characters.
- Keywords must not be abbreviated.
- Lines which start with an asterisk (*) in column 1 are treated as comments.
- Each parameter and the SECURE-RESOURCE-ALLOCATION command may only be specified once.

General rules

- The default BS2000 values are set in the global parameter file SYSPAR.ARCHIVE.*version*. The system administrator can modify this file and set other default values which apply for all ARCHIVE runs unless you explicitly specify other values in the ARCHIVE parameter file.
- The parameter specifications are evaluated each time backups to tape are affected by a COPY, RECOVER or CREATE REPLICATION statement.
- The OPERATOR, STREAM and DRIVES parameters and the SECURE-RESOURCE-ALLOCATION command are evaluated for a COPY, for a RECOVER or for a CREATE REPLICATION statement.
- The DIRSAVE, DEVICE, SAVE-ACL, COMPRESS and RETPD parameters are only evaluated for a COPY statement.
- The SPACE and ENVIRONMENT-ATTRIBUTES parameters are only evaluated for RECOVER or for CREATE REPLICATION statements.
- Parameters specified in the ARCHIVE parameter file are assigned the value you specify.
Parameters which are not specified in the ARCHIVE parameter file are assigned the values specified in the parameter description.
- If no ARCHIVE parameter file is available, the OPERATOR and STREAM parameters are assigned the SESAM/SQL default value YES. SESAM/SQL does not assign values to any of the other parameters, i.e. ARCHIVE default values are used (see parameter description). No SECURE-RESOURCE-ALLOCATION command is executed.

- In the RESTORE statement, the value YES is always assigned to the REPLACE parameter.
- In addition to the parameters you can specify, SESAM/SQL always assigns the following values to the following parameters of the PARAM statement:

```
CATID=YES  
OLS=YES  
UNLOAD=YES
```

For more details refer to the PARAM statement in the “[ARCHIVE \(BS2000\)](#)” manual.

- After the ARCHIVE run has terminated, SESAM/SQL releases the reserved resources with a further SECURE-RESOURCE-ALLOCATION command without parameters.
- The ARCHIVE statements are logged in a file named *archive_directory_name*.SYSLST. If several RECOVER SPACE statements are processed in parallel for the same database, the other statements are logged in files named *archive_directory_name*.SYSLST.*tsn*.

In the following situations, the COPY, RECOVER or CREATE REPLICATION statement is aborted and an error message is issued:

- If the ARCHIVE parameter file is empty.
- If the ARCHIVE parameter file is not a SAM file.
- If the ARCHIVE parameter file cannot be opened, closed or read as a result of a DMS error.
- If parameters are specified which cannot be recognized (e.g. if keywords are written incorrectly); the message text contains the number of the line in which the incorrect parameter specification is located and the name of the ARCHIVE parameter file.
- If incorrect values are specified; the message text contains the number of the line in which the incorrect parameter specification is located and the name of the ARCHIVE parameter file.

Exceptions

- SESAM/SQL does not check the device type specification DEVICE= ; this check is done by ARCHIVE.
- The parameter specifications for the SECURE-RESOURCE-ALLOCATION command are checked by BS2000.

In the following situations the COPY, RECOVER or CREATE REPLICATION statement is aborted and the corresponding error message from ARCHIVE or from BS2000 is written to the file *archive_directory_name*.SYSLST:

- if the ARCHIVE run is aborted or errored (includes incorrect device type specification DEVICE= (see [page 59](#)))
- if BS2000 detects errors in the SECURE-RESOURCE-ALLOCATION command (includes syntax errors in the parameters specified for the SECURE-RESOURCE-ALLOCATION command (see [page 58](#))).

Sample ARCHIVE parameter file

```
SECURE-RESOURCE-ALLOCATION DEVICE=(TYPE=TAPE-C4,NUMBER=1),  
WAIT=(TIME=TASK-STD)  
*   ***reserve resources***  
*  
OPERATOR=YES  
*   ***output messages to console***  
*  
STREAM=YES  
*   ***perform input and output in streaming mode***  
*  
DIRSAVE=YES  
*   ***store directory file***  
*  
DEVICE=TAPE-C4  
*   ***specify device type***  
*  
SAVE-ACL=YES  
*   ***store ACL entries***  
*  
DRIVES=1  
*   ***no parallel runs, no data packaging***  
*  
COMPRESS=YES  
*   ***write data to backup file in compressed form***  
*  
RETPD=365  
*   ***retention period for backup version is 365 days***  
*  
SPACE=REORG  
*   ***reorganize disk***  
*  
ENVIRONMENT-ATTRIBUTES=FROM-TARGET  
*   ***take access protection attributes from backup version***  
*
```


2.7 Processing input, output and error files

In the case of several of the utility statements, you can or must specify file names for input, output and error files (see [chapter “Utility statements” on page 69](#)). You can specify the file names with the catalog ID and the BS2000 user ID. A file name must be unique without taking the catalog ID into consideration, i.e. two file names cannot be distinguished from each other merely by their catalog IDs.

The following points must be observed for input files:

- If you specify the file name without a BS2000 user ID, SESAM/SQL expects to find the file on the BS2000 user ID under which the DBH is running.
- If the BS2000 user ID you specify is not the DBH user ID, the BS2000 user ID of the DBH must have the appropriate access authorizations.

The following points must be observed for output and error files:

- If you specify the file name without a BS2000 user ID, SESAM/SQL creates the file on the BS2000 user ID under which the DBH is running, provided that the file does not already exist.
If you do not specify a catalog ID, SESAM/SQL uses the standard pubset of the BS2000 user ID under which the DBH is running.
- If the BS2000 user ID you specify is not the DBH user ID, the database administrator must make preparations for creating the file in this BS2000 user ID (see the “Core Manual”, section “Database files and job variables on foreign user IDs”). In addition, the BS2000 user ID of the DBH must have the appropriate access authorization.
- If no error file is specified, a standard file name will be used and the file will be created on the DB user ID or, if this is not possible, on the DBH user ID.
- If the database is in a DB user ID, SESAM/SQL will try to create the error file in the DB user ID as well. If this is not possible, the error file will be created in the DBH user ID.

If, when it creates a file, SESAM/SQL discovers that a file with the same name already exists on a different catalog ID, the utility statement is aborted with SQLSTATE. The same thing happens if, when opening a file, SESAM/SQL finds that there are several files with the same name under different catalog IDs. You can work round this problem using the CATID list; for more details, see the “[Core manual](#)”.

SESAM/SQL rejects any attempt to create a file on a BS2000 user ID (DB user ID) other than that of the DBH user ID during execution of a utility statement in the event of insufficient access authorization with SQLSTATE.

Reading error files

You can read an error file created by SESAM/SQL on execution of the utility statements LOAD, UNLOAD or CHECK FORMAL using the file editor EDT (see the “EDT (BS2000)” manual) or output it using the SHOW-FILE command.

The names of the error files for the utility statements LOAD, UNLOAD and CHECK FORMAL are as follows:

LOAD	<i>catalog.space</i> .EXC.L
UNLOAD OFFLINE	<i>catalog.space</i> .EXC.U
UNLOAD ONLINE	<i>file_name</i> .EXC.U
CHECK FORMAL	<i>catalog.space</i> .EXC.C

Where *catalog* and *space* are the names of the database and user space, respectively, that are affected by the execution of the utility statement.

For a partitioned table, the space name of the first partition is always used in the case of LOAD and UNLOAD. The error file contains the error information for all partitions in the table.

In the case of CHECK FORMAL, an error file with the space name of the partition is created for each errored partition.

2.8 User IDs and link names

BS2000 user ID for the SESAM/SQL database

All the files of a SESAM/SQL database must be cataloged on the same BS2000 user ID. This may be

- the user ID under which the DBH is running, i.e. the **DBH user ID** or
- a user ID on which the database was created, i.e. the **DB user ID**.

The DBH enters the BS2000 user ID in the SQL database catalog list (see the “[Database Operation](#)” manual).

The utility statements execute under the control of the DBH under the DBH user ID.

If a database is in a DB user ID, the DBH and the utilities always start by trying to access files on this user ID.

In this case the database administrator must make preparations for creating files via DBH and utilities (e.g. SESAM backup copies and log files) in this BS2000 user ID; for more details, see the “Core Manual”, section “Database files and job variables on foreign User IDs”.

If you are working with SESAM backup copies on magnetic tape cartridge, then different conditions apply for the HSMS and ARCHIVE software products:

- With HSMS, you can also create and restore backup copies of SESAM/SQL databases that are not present on the DBH user ID
- With ARCHIVE you can only create and restore backup copies of SESAM/SQL databases that are present on the DBH user ID.

Using link names

You cannot assign BS2000 link names to backup and work files for the execution of utility statements.

3 Utility statements

This chapter describes the utility statements as follows:

- Overview of the utility statements
- alphabetical reference section

3.1 Overview of the utility statements (in alphabetical order)

The table below provides you with an overview of the utility statements

UTILITY statement	Utility function	Page
ALTER CATALOG	Modify properties of the database	73
ALTER DATA FOR TABLE	Shuffle column values, anonymize user data	74
ALTER MEDIA DESCRIPTION	Modify the media table	76
ALTER PARTITIONING FOR TABLE	Changing the partitioning of a base table	80
CHECK CONSTRAINTS	Check integrity constraints	91
CHECK FORMAL	Check the format of base tables and indexes	94
COPY	Create a SESAM backup copy	97
CREATE CATALOG	Create a database (catalog space)	105
CREATE MEDIA DESCRIPTION	Enter the first media record for a database-specific file type in the media table	115
CREATE REPLICATION	Create a replication	119
DROP MEDIA DESCRIPTION	Delete all the media records for a database-specific file type from the media table	127
EXPORT TABLE	Export a table from a database to an export file	129
IMPORT TABLE	Import a table from an export file into a data base	133
LOAD	Load user data into a base table	140
MIGRATE	Migrate databases and tables	166
MODIFY	Maintain information (metadata) on the SESAM backup copies	171
RECOVER	Reset and repair a database, rebuild indexes	178
REFRESH REPLICATION	Update a replication	219
REFRESH SPACE	Extend a replication	223
REORG	Reorganize the catalog space, user spaces, and base tables	227
UNLOAD	Unload user data from a base table	234

Table 7: Overview of the utility statements

3.2 Alphabetical reference section

In this section, the utility statements are described in a uniform manner. The statements are described in alphabetical order, and there is an entry for each statement with the name of the statement as the header.

The description of the utility statements is based on information found in the “[Core manual](#)” and the introductory section of the “[SQL Reference Manual Part 1: SQL Statements](#)”.

All the syntax elements needed for representing the utility statements are listed in alphabetical order in [section “Syntax overview” on page 257](#). Syntax elements described in the “[SQL Reference Manual Part 1: SQL Statements](#)” are also included but are not explained in greater detail in this manual.

Structure of an entry

Each entry consists of several parts.

Certain parts may be missing from an entry if they have no meaning for that particular utility statement. An entry may also include additional information after the syntax diagram that describes special features or characteristics of the utility statement involved. The most important parts of each entry are described below.

Statement name - Brief description

A brief description of the function of the statement follows the heading.

This section also describes the prerequisites for successfully executing the statement. In particular, the required access permissions are mentioned.

STATEMENT_NAME CLAUSE *parameter* . . .

parameter

Explanation of the parameters.

The clauses and parameters are described in the order in which they appear in the syntax diagram.

Examples

This section includes one or more examples illustrating how the statement is used. Most of the examples are based on the sample database ORDERCUST described in the “[Core manual](#)”.



If an example in the manual is accompanied by the symbol on the left, this means that it is present as a component in an instruction file or an ESQL-COBOL program in the demonstration database.

See also

Related statements

ALTER CATALOG - Modify the properties of a database

ALTER CATALOG modifies the properties of a database.

The coded character set (CCS; synonym: code table) for the database (see CODE-TABLE clause on [page 108](#)) can be modified.

The current authorization identifier must have the special privilege UTILITY.

```
ALTER CATALOG catalog
      ALTER CODE_TABLE ccs_name
```

catalog

Name of the SESAM/SQL database whose properties will be modified.

ALTER_CODE_TABLE *ccs_name*

Name of a coded character set (CCS) defined in BS2000 (system component XHCS), up to 8 characters in length.

ccs_name must be specified as an alphanumeric literal consisting of uppercase letters, digits, @, # or \$. The first character may not be a digit.

The CCS names of EBCDIC character sets or self-defined character sets based on EBCDIC character sets can be specified for *ccs_name*.

The value `_NONE_` can also be specified for *ccs_name*. No coded character set is then used for the database. This corresponds to the behavior of SESAM/SQL before V5.0.

The *ccs_name* (see also: CREATE CATALOG) is entered and used the database's metadata, e.g. in the utility statement LOAD for checking against the input file's CCSN.

Introductory information on using coded character sets in SESAM/SQL is provided in the section "Unicode concept in SESAM/SQL" in the "[Core manual](#)".

Example

This example changes the coded character set in the ORDERCUST database:



```
ALTER CATALOG ordercust ALTER CODE_TABLE 'EDF041'
```

See also

CREATE CATALOG

ALTER DATA FOR TABLE - Shuffle column values, anonymize data

ALTER DATA FOR TABLE shuffles column values of a base table in such a manner that no conclusions can be drawn regarding the original content. The original values of a column and its frequency distribution are retained. The shuffling differs from column to column and from case to case.

Important (e.g. person-related) user data is thus anonymized. You will find introductory information on anonymization in SESAM/SQL in the “[Core manual](#)”.

The current routing code must have the special privilege UTILITY or be the owner of the schema to which the base table belongs.

```
ALTER DATA FOR TABLE table
    SHUFFLE VALUES FOR COLUMN {column
                               (column, ...)} , ...
```

table

Name of a base table.

It may not be a CALL-DML table and may not be in the “check pending” state.

{*column*
(*column*, ...)}

Names of the columns of any data type whose values are to be shuffled.

Columns whose names are specified in parentheses are regarded as a unit and are shuffled contiguously. The logical relationship between these columns is retained.

When multiple columns are specified, all versions of the multiple column as a whole are shuffled. The definition length of all defined versions may not exceed 512 bytes.

The columns of the primary key may not be specified.

CHECK, uniqueness and reference constraints may only be defined for individual columns which are to be shuffled. They may not be defined for more than one column if at least one of these columns is to be shuffled.

The user spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

When indexes are defined for the table, they are reconstructed. The index spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

Example

This example shuffles the person-related data of the contacts table. The *fname*, *lname* and *title* columns are regarded as a unit and shuffled together.



```
ALTER DATA FOR TABLE contacts.anonymous SHUFFLE VALUES FOR COLUMN  
ac_no,c_no,(fname,lname,title),  
contact_tel,position,dept,contact_info
```

ALTER MEDIA DESCRIPTION - Modify the media table

You use ALTER MEDIA DESCRIPTION to modify the description of the file attributes of a certain database-specific or space-specific file type in the media table. You can also use this utility statement to create new entries (media records) in the media table or delete existing entries. The media table is described in the [“Core manual”](#).

SESAM/SQL recognizes the following database-specific file types:

- CAT-REC file
- CAT-LOG files
- DA-LOG files
- PBI files
(media records also apply to the work file in the case of online backup with HSMS)

SESAM/SQL recognizes the following space-specific file type:

- DDL-TA-LOG files

Any modifications you make with ALTER MEDIA DESCRIPTION only affect new files of the specified file type. This statement has no effect on existing files.

The current authorization identifier must have the special privilege UTILITY.

FOR {
 DALOG
 CATLOG
 CATREC
 PBI
 DDLTA }

Indicates the file type for which modifications are to be entered in the media table.

DALOG The file type DA-LOG file is to be modified.

CATLOG The file type CAT-LOG file is to be modified.

CATREC The file type CAT-REC file is to be modified.

PBI The file type PBI file is to be modified
 (also applies to the work file in the case of online backup with HSMS).

DDLTA The file type DDL-TA-LOG file is to be modified.

AT CATALOG *catalog*

Name of the database to which the processed media table belongs.

You can only specify the space clauses PRIMARY, SECONDARY, and [NO] SHARE once.

PRIMARY *allocation*

New primary allocation for files of the specified file type in units of 2K
 (BS2000 halfpage).

allocation must be an integer between 1 and 33 554 430.

If the value specified for *allocation* is too small, SESAM/SQL automatically corrects it to the default value.

PRIMARY *allocation* omitted:

Primary allocation for new files of the specified file type remains unchanged.

SECONDARY *allocation*

New secondary allocation for files of the specified file type in units of 2K
 (BS2000 halfpage) if the file type CATREC or PBI is specified.

allocation must be an integer between 0 and 32 767.

If the value specified for *allocation* is too small, SESAM/SQL automatically corrects it to the default value.

If the file type DALOG or CATLOG is specified, you cannot specify the space clause SECONDARY.

SECONDARY *allocation* omitted:

Secondary allocation for new files of the specified file type remains unchanged.

[NO] SHARE

SHARE indicates that the files of the specified file type are shareable.

NO SHARE indicates that the files of the specified file type are not shareable.

[NO] DEVICE REQUEST

DEVICE REQUEST indicates that SESAM/SQL is to request additional devices for the specified file type at the console once all the entries in the media table have been exhausted.

If SESAM/SQL requests additional devices once the media table has been exhausted then you can make the following entries at the console: ¹

PUBLIC *catid*

The file is created on the specified pubset.

catid BS2000 catalog ID
 The BS2000 catalog ID must be specified.
 You must specify *catid* as an alphanumeric literal.
 If a CATID list was defined then the catalog ID must be specified in the list.

IGNORE

The file is not created.

PRIVATE *catid, dev_type, vsn, vsn...*

The file is created on private disks with the specified Volume Serial Number and the specified device type and is catalogued using the specified catalog ID. A maximum of 6 Volume Serial Numbers can be specified.

catid BS2000 catalog ID
 The BS2000 catalog ID must be specified.
 You must specify *catid* as an alphanumeric literal.
 If a CATID list was defined then the catalog ID must be specified in the list.

dev_type Device type of the private disk.
 You must specify *dev_type* as an alphanumeric literal.

vsn Volume Serial Number.
 You must specify *vsn* as an alphanumeric literal.

NO DEVICE REQUEST indicates that SESAM/SQL is not to request additional devices at the console once all the entries in the media table have been exhausted.

[NO] DEVICE REQUEST omitted:

The current setting for the specified file type is retained.

You can only specify the [NO] DEVICE REQUEST clause once.

¹ TAPE *device_type* can no longer be specified in SESAM/SQL V7.0 and higher.

ADD STOGROUP *stogroup*

A new entry (media record) for the specified file type is added to the media table. *stogroup* is the name of the additional storage group that is to be provided for files of the specified file type. You can qualify the name of the storage group with the database name *catalog*.

There must not yet be any media record designating a magnetic tape device type or a magnetic tape cartridge device type for the specified file type (TAPE clause, see [section “CREATE MEDIA DESCRIPTION - Define file attributes of database-specific files” on page 115](#)).

In addition, please note that the number of media records per file type in the media table is limited: 10 records can be entered for DA-LOG, CAT-LOG and DDL-TA-LOG files, 8 records for PBI files and 2 records for CAT-REC files.

You can only specify the ADD STOGROUP clause once. You cannot specify both the ADD STOGROUP and DROP STOGROUP clauses in the same ALTER MEDIA DESCRIPTION statement.

DROP STOGROUP *stogroup*

An entry (media record) for the specified file type is deleted from the media table. The media record is identified via the name *stogroup* of the corresponding storage group. You can qualify the name of the storage group with the database name *catalog*. The media record to be deleted cannot be the only record for the specified file type. If this is the case, ALTER MEDIA DESCRIPTION is rejected.

You can only specify the DROP STOGROUP clause once. You cannot specify both the ADD STOGROUP and DROP STOGROUP clauses in the same ALTER MEDIA DESCRIPTION statement.

Example

In the example below, the description of the file attributes for the DA-LOG files is modified in the media table of the database ORDERCUST: the primary allocation for DA-LOG files is increased from the default value of 768 2K units to 1500 2K units.



```
ALTER MEDIA DESCRIPTION FOR DALOG AT CATALOG ordercust  
PRIMARY 1500, SHARE
```

See also

CREATE MEDIA DESCRIPTION, DROP MEDIA DESCRIPTION

ALTER PARTITIONING FOR TABLE - Modify the partitioning of a base table

You use ALTER PARTITIONING FOR TABLE to

- add a partition to a base table (ADD)
- alter the partition boundaries of a partition (ALTER)
- delete a partition (DROP)

To execute ALTER PARTITIONING FOR TABLE, the current routing code must either have the special privilege UTILITY or be the owner of the schema to which the base table belongs.

The various forms of this statement are described in separate sections below.

ALTER PARTITIONING FOR TABLE ... ADD PARTITION

This statement adds a partition to a base table.

It converts a non-partitioned base table with a primary key to a partitioned base table.

ALTER PARTITIONING FOR TABLE *table*

ADD PARTITION ON SPACE *space*

SET UPPER LIMIT OF $\left\{ \begin{array}{l} \text{ADDED} \\ \text{PRIOR} \end{array} \right\}$ PARTITION TO VALUE $\left\{ \begin{array}{l} < \\ <= \end{array} \right\}$ (*column_value, ...*)

$\left[\left\{ \begin{array}{l} \text{ONLY WHEN EMPTY} \\ \text{PRESERVE ROWS} \end{array} \right\} \right] \left[\left\{ \begin{array}{l} \text{GENERATE INDEX} \\ \text{NO INDEX} \end{array} \right\} \right]$

table

Name of a partitioned base table or of a non-partitioned base table with a primary key. The table may not be an only CALL DML table or a CALL DML table which is password-protected, and it may not be in the “check pending” state.

ADD PARTITION ON SPACE *space*

Name of an existing user space.

No partition of the *table* table may be located on it.

The owner of the space must also be the owner of the schema to which the base table belongs.

The new partition is created on the specified user space.

The maximum possible number of partitions in a base table (16) may not be exceeded.

SESM/SQL automatically numbers the table’s partitions in ascending order from 1 to 16 in accordance with the upper limits of the partitions.

SET UPPER LIMIT OF $\left\{ \begin{array}{l} \text{ADDED} \\ \text{PRIOR} \end{array} \right\}$ PARTITION TO VALUE $\left\{ \begin{array}{l} < \\ <= \end{array} \right\}$ (*column_value,...*)

In the case of a **partitioned** table the upper limit specified must be different from the existing upper limits of the partitions. The partition with the next highest specified upper from the one specified limit is split.

When ADDED is specified, the new partition is the lower of the two partitions and is assigned the specified upper limit.

When PRIOR is specified, the new partition is the higher of the two partitions. The partition which is to be split is assigned the specified upper limit.

A **non-partitioned** base table is converted into a partitioned table with two partitions. When **ADDED** is specified, the new partition is the lower of the two partitions and is assigned the specified upper limit. The upper limit of the higher partition is obtained from the highest primary key value.

When **PRIOR** is specified, the new partition is the higher of the two partitions. The upper limit of the higher partition is obtained from the highest primary key value. The lower partition is assigned the specified upper limit.

The upper limit is either included or excluded by the preceding comparison operator:

<= Rows whose primary key value is *column_value*,... belong to **this** partition

< Rows whose primary key value is *column_value*,... belong to the **next** partition

The lexicographical rules apply for the comparison, see the “[SQL Reference Manual Part 1: SQL Statements](#)”.

ONLY WHEN EMPTY

The new partition is added only if it remains empty, i.e. if no rows are to be moved.

When indexes are defined in a CALL DML table, the user spaces concerned are assigned the “copy pending” state if they are subject to logical data backup.

PRESERVE ROWS

The rows concerned are retained.

They are moved in accordance with the new upper limits.

When rows are transferred from one partition to another, the row numbers are adjusted in accordance with the prefix `ROW_ID_PREFIX` (see View `SYS.PARTITIONS` in the `SYS_INFO_SCHEMA`, [SQL Reference Manual Part 1: SQL Statements](#)).

The user spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

GENERATE INDEX

When secondary indexes are defined and rows were moved as a result of the PRESERVE ROWS specification, the secondary indexes in the table are reconstructed. This also applies for secondary indexes which were marked as defective before the ALTER PARTITIONING statement.

The index spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

When ONLY WHEN EMPTY was specified or no rows were moved as a result of the PRESERVE ROWS specification, the secondary indexes are reconstructed only if there were already secondary indexes marked as defective beforehand. Otherwise the secondary indexes are not reconstructed.

NO INDEX

When secondary indexes are defined and rows were moved as a result of the PRESERVE ROWS specification, the secondary indexes in the table are marked as defective and not reconstructed.

When ONLY WHEN EMPTY was specified or no rows were moved as a result of the PRESERVE ROWS specification, the secondary indexes are not reconstructed. They retain their original state.

ALTER PARTITIONING FOR TABLE ... ALTER PARTITION

This statement changes the partition boundary of a partition.

ALTER PARTITIONING FOR TABLE *table*

ALTER PARTITION ON SPACE *space*

SET UPPER LIMIT TO VALUE $\left\{ \begin{array}{l} < \\ <= \end{array} \right\} (column_value, \dots)$

$\left[\begin{array}{l} \text{ONLY WHEN EMPTY} \\ \underline{\text{PRESERVE ROWS}} \end{array} \right] \left[\begin{array}{l} \underline{\text{GENERATE INDEX}} \\ \text{NO INDEX} \end{array} \right]$

table

Name of a partitioned base table. It may not be in the “check pending” state.

ALTER PARTITION ON SPACE *space*

Name of the user space with the partition whose partition boundary is to be changed.

SET UPPER LIMIT TO VALUE $\left\{ \begin{array}{l} < \\ <= \end{array} \right\} (column_value, \dots)$

The upper limit specified must be lower than the upper limit of the next highest partition. It must also be greater than the upper limit of the next lowest partition. The upper limit of the highest partition cannot be changed.

The upper limit is either included or excluded by the preceding comparison operator:

<= Rows whose primary key value is *column_value*,... belong to **this** partition

< Rows whose primary key value is *column_value*,... belong to the **next** partition

The lexicographical rules apply for the comparison, see the “[SQL Reference Manual Part 1: SQL Statements](#)”.

ONLY WHEN EMPTY

The change to the partition’s upper limit may not result in rows being moved.

PRESERVE ROWS

The rows concerned are retained.

They are moved in accordance with the new upper limits.

When rows are transferred from one partition to another, the row numbers are adjusted in accordance with the prefix ROW_ID_PREFIX (see View SYS.PARTITIONS in the SYS_INFO_SCHEMA, [SQL Reference Manual Part 1: SQL Statements](#)).

The user spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

GENERATE INDEX

When indexes are defined and rows were moved as a result of the PRESERVE ROWS specification, the indexes in the table are reconstructed. This also applies for indexes which were marked as defective before the ALTER PARTITIONING statement.

The index spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

When ONLY WHEN EMPTY was specified or no rows were moved as a result of the PRESERVE ROWS specification, the indexes are reconstructed only if they were marked as defective beforehand. Otherwise the indexes are not reconstructed.

NO INDEX

When indexes are defined and rows were moved as a result of the PRESERVE ROWS specification, the secondary indexes in the table are marked as defective and not reconstructed.

When ONLY WHEN EMPTY was specified or no rows were moved as a result of the PRESERVE ROWS specification, the indexes are not reconstructed. They retain their original state.

ALTER PARTITIONING FOR TABLE ... DROP PARTITION

This statement deletes a partition.

It converts a partitioned base table to a non-partitioned base table if it contains only one partition following deletion.

```
ALTER PARTITIONING FOR TABLE table
  DROP PARTITION ON SPACE space
  DROP UPPER LIMIT OF { DROPPED }
                       { PRIOR } PARTITION
  [ { DELETE ROWS [DEFERRED]
    { ONLY WHEN EMPTY
    { PRESERVE ROWS } } ] [ { GENERATE INDEX }
                           { NO INDEX } ] ]
```

table

Name of a partitioned base table. It may not be in the “check pending” state.

DROP PARTITION ON SPACE *space*

Name of the user space with the partition which is to be deleted.

DROPPED

The upper limit of the partition which is to be deleted is deleted. The clause may not be specified for the highest partition.

PRIOR

The upper limit of the partition which is to be deleted is used as the upper limit of the next lowest partition. The clause may not be specified for the lowest partition.

ONLY WHEN EMPTY

The partition which is to be deleted may not contain any rows.

PRESERVE ROWS

The rows of the partition which is to be deleted are retained.

When PRIOR is specified, they are moved to the next lowest partition. When DROPPED is specified, they are moved to the next highest partition.

When rows are transferred from one partition to another, the row numbers are adjusted in accordance with the prefix ROW_ID_PREFIX (see View SYS.PARTITIONS in the SYS_INFO_SCHEMA, [SQL Reference Manual Part 1: SQL Statements](#)).

The user spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

The maximum possible number of rows in a partition (c. 268 million) may be exceeded only if the table contains just one partition following the delete operation and is subsequently converted to a non-partitioned base table by SESAM/SQL .

DELETE ROWS [DEFERRED]

The rows of the partition which is to be deleted are deleted.

However, the table may not be referenced by a referential constraint of another base table.

The DEFERRED clause causes the rows to be deleted quickly.

Only the contiguous part of the partition is deleted. Any relocations which exist are retained. The next time the user space is reorganized using the utility statement REORG SPACE all the existing tables and indexes are recovered in the user space. The relocations which have not been deleted then also disappear. Information on the storage structure of base tables is provided in the “[Core manual](#)”.

The user spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

GENERATE INDEX

When indexes are defined and rows were moved as a result of the PRESERVE ROWS specification or DELETE ROWS is specified, the indexes in the table are reconstructed. This also applies for indexes which were marked as defective before the ALTER PARTITIONING statement.

The index spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

When ONLY WHEN EMPTY was specified or no rows were moved as a result of the PRESERVE ROWS specification, the indexes are reconstructed only if they were marked as defective beforehand. Otherwise the indexes are not reconstructed.

When a non-partitioned table is created from a partitioned table by the ALTER PARTITIONING statement, the secondary indexes of the table are reconstructed.

The index spaces affected are placed in the “copy pending” state if they are subject to logical data backup.

NO INDEX

When indexes are defined and rows were moved as a result of the PRESERVE ROWS specification or DELETE ROWS is specified, the secondary indexes in the table are marked as defective and not reconstructed.

When ONLY WHEN EMPTY was specified and no rows were moved as a result of the PRESERVE ROWS specification, the indexes are not reconstructed. They retain their original state.

When a non-partitioned table is created from a partitioned table by the ALTER PARTITIONING statement, the secondary indexes of the table are marked as defective and not reconstructed.

Examples

1. This example adds a partition to the non-partitioned table CUSTOMER.

```
ALTER PARTITIONING FOR TABLE customer
  ADD PARTITION ON SPACE spacenew
  SET UPPER LIMIT OF ADDED PARTITION TO VALUE <= (999)
  ONLY WHEN EMPTY
```

The CUSTOMER table thus becomes a partitioned table with 2 partitions. The lower partition is assigned the upper limit of 999. The upper limit of the higher partition is obtained from the highest primary key value.

2. This example changes the upper limit of the partition created above.

```
ALTER PARTITIONING FOR TABLE customer
  ALTER PARTITION ON SPACE spacenew
  SET UPPER LIMIT TO VALUE <= (500)
  PRESERVE ROWS GENERATE INDEX
```

The rows are moved. Indexes are rebuilt if required.

3. This example deletes a partition in the partitioned table CUSTOMER.

```
ALTER PARTITIONING FOR TABLE customer
  DROP PARTITION ON SPACE spacenew
  DROP UPPER LIMIT OF PRIOR PARTITION
  DELETE ROWS DEFERRED GENERATE INDEX
```

The CUSTOMER table from example 1 thus becomes a non-partitioned table again. The partition is deleted. Relocated data is initially retained. Indexes are rebuilt.

4. Let the partitions of the base table SALES be organized according to the months of the year. Let the primary key be a column *sales_date* of the data type DATE.

```
CREATE TABLE sales ...
USING PARTITION BY RANGE PARTITION 1 VALUE <= DATE'2012-01-31' ON SPACE sp1
USING PARTITION BY RANGE PARTITION 2 VALUE <= DATE'2012-02-29' ON SPACE sp2
...
USING PARTITION BY RANGE PARTITION 11 VALUE <= DATE'2012-11-30'
                                     ON SPACE sp11
USING PARTITION BY RANGE PARTITION 12 VALUE <= () ON SPACE sp12
```

At the start of the year 2013 the data from January 2012 is to be deleted. The space which becomes free is to be used for the data from January 2013. This task can be performed conveniently using the utility statement ALTER PARTITIONING FOR TABLE.

In the first step the following utility statement is used to remove partition 1 from the metadata of the *sales* table. The rows on space *sp1* are deleted.

```
ALTER PARTITIONING FOR TABLE sales
      DROP PARTITION ON SPACE sp1 DELETE ROWS
```

In the second step a new (unlimited) partition is created. The preceding partition is assigned 12/31/2012 as its new upper limit. Any sales which have already been recorded for the year 2013 are transferred to the new partition.

```
ALTER PARTITIONING FOR TABLE sales
      ADD PARTITION ON SPACE sp1
      SET UPPER LIMIT OF PRIOR PARTITION TO VALUE <= DATE'2012-12-31'
      PRESERVE ROWS
```

SESAM/SQL automatically adjusts the partition numbers, so the changed *sales* table contains the following partitions:

```
PARTITION 1 VALUE <= DATE'2012-02-29' ON SPACE sp2
PARTITION 2 VALUE <= DATE'2012-03-31' ON SPACE sp3
...
PARTITION 11 VALUE <= DATE'2012-12-31' ON SPACE sp12
PARTITION 12 VALUE <= () ON SPACE sp1
```

CHECK CONSTRAINTS - Check integrity constraints

You use CHECK CONSTRAINTS to check whether integrity constraints are satisfied. You always need to use CHECK CONSTRAINTS to check the integrity constraints for user spaces and base tables if integrity constraints have been defined for the specified tables and the stored data has been changed in some way after execution of the utility statement LOAD with NO CONSTRAINT CHECK:

SESAM/SQL places the tables involved in the utility statement or the user spaces being processed in the state “check pending” (see [page 13](#)).

You have to execute CHECK CONSTRAINTS before you can process the locked tables again with SQL statements. If the check was successful, SESAM/SQL makes the tables generally available again. If not, the tables remain in the “check pending” state.

If no table in a user space is in the state “check pending”, the user space itself is also no longer in the state “check pending”.

You will find a description of how to handle tables that remain in the state “check pending” after CHECK CONSTRAINTS has been executed in the “[Core manual](#)”.

The current authorization identifier must have the special privilege UTILITY or must own the table or user spaces being checked.

If CHECK CONSTRAINTS is used for partitioned tables (ON TABLE) or partitions on spaces (ON SPACE), all the partitions affected must be available, see [table 3 on page 22](#).

```
CHECK CONSTRAINTS { integrity_constraint_name , ... } [ { SCOPE ALL } ]
                  { ON TABLE table , ... }
                  { ON SPACE space , ... }
```

integrity_constraint_name ,...

Name of one or more table or column constraints to be checked.

You can only specify each *integrity_constraint_name* once.

ON TABLE *table* ,...

Name of one or more base tables. You can only specify each table name once.

All of the defined table and column constraints are checked for several (see [SCOPE PENDING](#)) or all (see [SCOPE ALL](#)) of the specified base tables. If the check is successful, SESAM/SQL places all the tables that are currently in the state “check pending” in the state “space o.k”. If SESAM/SQL discovers that an integrity constraint has been violated, it places all the base tables checked in the state “check pending” or leaves them in the state “check pending”, as appropriate.

ON SPACE *space* ,...

Name of one or more user spaces. Up to 1000 space names may be specified. You can only specify each space name once.

All of the defined table and column constraints are checked for several (see [SCOPE PENDING](#)) or all (see [SCOPE ALL](#)) of the base tables in the specified user space(s). If the check is successful, SESAM/SQL places tables that are currently in the state “check pending” in the state “space o.k”. If SESAM/SQL discovers that an integrity constraint has been violated, it places all base tables of the checked user spaces in the state “check pending” or leaves them in the state “check pending”.

SCOPE ALL

The integrity constraints of all the specified tables are checked.

The following must be observed:

If one of the tables violates an integrity constraint, all of the tables to be checked are placed in the state “check pending”, even if the other tables did not violate any integrity constraint.

SCOPE ALL is only effective in conjunction with the clauses ON TABLE and ON SPACE.

SCOPE PENDING

Of the specified tables or the tables of the specified user spaces, only those in the state “check pending” are checked.

SCOPE PENDING is only effective in conjunction with the clauses ON TABLE and ON SPACE.

Example

In the example below, all the integrity constraints defined for the base table ORDERS are checked when the table has the state “check pending”. The ORDERS table is defined in the schema ORDERPROC of the database ORDERCUST.



```
CHECK CONSTRAINTS ON TABLE ordercust.orderproc.orders
```

If SESAM/SQL discovers that an integrity constraint defined for ORDERS has been violated, ORDERS remains in the state “check pending”. If not, SESAM/SQL places the ORDERS table in the state “space o.k.”.

See also

CHECK FORMAL

CHECK FORMAL - Check the format of tables and indexes

You use CHECK FORMAL to check the formal correctness. The formal check mainly involves checking internal SESAM/SQL blocks and tables.

If an inconsistency is detected, SESAM/SQL creates an error file (see [page 95](#)). SESAM/SQL reports an inconsistency detected during CHECK FORMAL as an SQLSTATE. The space or index are marked as “defective”. If possible, the format check is continued with CHECK FORMAL.

CHECK FORMAL is also allowed for replications and for SESAM backup copies contained in the SQL database catalog. A CHECK FORMAL is possible with foreign copies if the foreign copy is added as a database in the SQL database catalog. To prevent it from losing its property of being a foreign copy, it must be added with ACCESS=READ.

The database whose spaces, tables or indexes are to be checked must be added in the SQL database catalog list with ACCESS=ADMIN in the DBH to ensure that spaces (in the case of defective tables) and defective indexes are actually marked as “defective”. If the database is added with ACCESS=READ, as a backup copy, as a replication or as a foreign copy then spaces (in the case of defective tables) and defective indexes are not marked as “defective”. You will find notes on the error only in the SQLSTATE and in the error file.

The current authorization identifier must have one of the following properties.

- It must have the special privilege UTILITY.
- It must own the space to be checked.
- It must own the space in which the base table or index to be checked is located.

```
CHECK FORMAL { TABLE table
                INDEX index
                SPACE space } [NO ACTION]
```

TABLE *table*

Name of the base table to be checked.

In the case of partitioned tables all partitions in the table are checked. If a partition is not available (see [table 3 on page 22](#)), this is logged in the error file (see [page 95](#)). The check is continued with the next partition.

If an error is detected during the check, the user space in which the table or the partition is located is marked as “defective”.

INDEX *index*

Name of the index to be checked.

If an error is detected during the check, the index is marked as “defective”.

SPACE *space*

Name of the space whose base tables, partitions and indexes are to be checked. In the case of a catalog space, the CATALOG space name must be specified as *special_name*. The space is marked as “defective” if an error is detected when it is checking a base table or partition in the space. If an error is detected when checking a space index, only the index will be marked as “defective”.

NO ACTION

If this clause is specified, spaces are not marked as “defective” in the case of defective tables. You will find notes on the error only in the SQLSTATE and in the error file.

Defective indexes will, however, continue to be marked as “defective”.

You can continue to work with defective indexes; they will no longer be used. Further processing of defective tables may lead to serious errors which can even cause the DBH to crash. A RECOVER should therefore definitely be performed.

Error file for CHECK FORMAL

If needed, SESAM/SQL creates a new error file for CHECK FORMAL or continues the old file. The file has the following name:

catalog.space. EXC.C

catalog is the name of the database.

space is the name of the user space being checked.

In the case of CHECK FORMAL TABLE for a partitioned table, an error file with the space name of the partition is created for each errored partition.

EXC.C Default name ending of the error file for CHECK statements.

If *catalog* is located on a DB user ID, the error file will be created on the DB user ID, provided the necessary preparations have been made, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”. If the error file cannot be created on the DB user ID, it will be created on the DBH user ID.

Example

In the example below, the format of the base table ORDERS is checked. The ORDERS table belongs to the schema ORDERPROC of the database ORDERCUST.



```
CHECK FORMAL TABLE ordercust.orderproc.orders
```

See also

CHECK CONSTRAINTS

COPY - Create a SESAM backup copy

You use COPY to create SESAM backup copies of the entire SESAM/SQL database, the catalog space or one or more user spaces. You can create the SESAM backup copies on disk or magnetic tape cartridge.

You can use COPY to create SESAM backup copies online or offline.

You can find more detailed information about creating SESAM backup copies online and offline in the [“Core manual”](#).

You can use COPY to create SESAM backup copies on magnetic tape cartridge with the help of the HSMS or ARCHIVE software products (see [section “Tape backups with HSMS and ARCHIVE” on page 36](#)).

Backup with HSMS can also be performed from the additional mirror unit of a multi-mirror pair in the Local Symmetrix or from the additional mirror unit of the target unit of a remote copy pair in a Remote Symmetrix (see the [“Core manual”](#)).

When Additional Mirror Units (BCVs) are mentioned in the following, “additional mirror or clone units” are meant.

As starting point for replications, COPY CATALOG and COPY CATALOG_SPACE always create a CAT-REC file with the fixed name *catalog.CAT-REC.COPY* on the medium on which the CAT-REC file is located. No metadata is stored for these additional copies.

If logging has been activated, COPY CATALOG and COPY CATALOG_SPACE always involve switching the CAT-LOG and DA-LOG files. For a database operated without logging, SESAM/SQL creates the CAT-REC file and the CAT-REC copy according to the specifications in the CAT-REC media record in the media table when the first COPY CATALOG or COPY CATALOG_SPACE statement is executed. If no CAT-REC media record exists, SESAM/SQL uses default values (see the [“Core manual”](#)).

If a space contains tables and indexes then this space can still be backed up even if one or more of the indexes are defective. However, when such a space is backed up, SESAM/SQL issues a warning indicating the presence of defective indexes.

A backup which contains defective indexes can be used for a RECOVER or CREATE REPLICATION. The defective indexes must then be reconstructed.

The current authorization identifier must have the special privilege UTILITY or must own the user space involved in the backup operation.

With SESAM/SQL, it is possible to switch from winter time to daylight saving time and vice versa (clock resetting) without interrupting the current session. However, during the critical phase of switching from daylight saving time to winter time (i.e. the last hour of daylight saving time and the first hour of winter time) SESAM backup copies are not permitted. SESAM/SQL rejects any attempt to make a SESAM backup copy during this critical phase of the switch.

Databases on a DB user ID other than the DBH user ID

If a database is located on a DB user ID other than the DBH user ID, SESAM/SQL will try to create the backup files in the DB user ID in the case of COPY on disk. Preparations must have been made for this purpose, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

If not all of the files can be created on the DB user ID, the statement will be rejected.

If the backup files cannot be created on the DB user ID, they will be created on the DBH user ID.

If the backup copy is to be created using the HSMS software product in the case of COPY, the DBH user ID for all database files must be defined as co-owner on the DB user ID.

ARCHIVE can only be used if the database is located on the DBH user ID.

The CAT-REC file and the CAT-REC copy are created on the DB user ID, provided the necessary preparations have been made, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”. It must always be possible to create the CAT-REC file in the DB user ID. If the CAT-REC copy cannot be created on the DB user ID, it will be created on the DBH user ID.

COPY statement

$$\text{COPY } \left\{ \begin{array}{l} \text{CATALOG } \textit{catalog} \\ \text{CATALOG_SPACE } \textit{catalog} \\ \text{SPACE } \textit{space} \text{ , . . . } \end{array} \right\} \left[\left\{ \begin{array}{l} \text{ONLINE} \\ \text{OFFLINE} \end{array} \right\} \right]$$

$$\left[\text{USING } \left\{ \begin{array}{l} \text{STOGROUP } \textit{stogroup} \\ \text{DIRECTORY } \left\{ \begin{array}{l} \textit{hsms_archive_name} \\ \textit{archive_directory_file} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{BY_ADD_MIRROR_UNIT} \\ \text{BY_SRDF_TARGET} \end{array} \right\} \right] \end{array} \right\} \right] \right]$$

[[NO] CHECK FORMAL] [LOG] [EXCEPT NO LOG INDEX SPACE]

CATALOG *catalog*

Name of the database for which a SESAM backup copy is to be created.

Creating a backup copy of the database comprises creating backup copies of the corresponding catalog space and all the user spaces in the database.

If you specify EXCEPT NO LOG INDEX SPACE then no spaces that contain indexes only and are not present in the logical data backup are backed up. If this clause is active then the backup cannot be added to the SQL database catalog list as a backup copy for retrieval purposes.

CATALOG_SPACE *catalog*

Name of the database whose catalog space is to be backed up.

SPACE *space ,...*

Name of the user space(s) for which a backup copy is to be created. You can specify up to 999 space names. You can only specify each space name once. You cannot specify CATALOG as a space name.

The spaces are sorted on their space size in descending order and are backed up in this order.

ONLINE

The SESAM backup copy is created online.

During online backup you can execute SQL statements for reading and modifying data and CALL-DML statements on the user spaces affected by the backup. When ONLINE is specified, the PBI file (in the case of disk copy and ARCHIVE) or the work file (in the case of HSMS) is always created on the user ID in which the DBH was started. The medium defined in the media description for PBI is used. In the case of a mirror disk backup (BY_ADD_MIRROR_UNIT, BY_SRDF_TARGET) no PBI or work file is created.

If the online backup is to take place in an HSMS archive, the database files to be backed up must not be located on private disk.

OFFLINE

The SESAM backup copy is created offline.

SESAM/SQL locks the user spaces affected by the backup during offline backup to prevent other users from making modifications.

USING STOGROUP *stogroup*

The SESAM backup copy is created as a disk copy on the storage group *stogroup*. The SESAM backup copies are created either on the DB user ID or on the DBH user ID.

If the database is located on a DB user ID, the backup files will also be created on the DB user ID provided the DBH user ID is defined as co-owner in the DB user ID or the files were previously created on the DB user ID using the command CREATE-FILE.

If the files were previously created on the DB user ID with the command CREATE-FILE, USING STOGROUP will be ignored.

USING DIRECTORY $\left\{ \begin{array}{l} \textit{hsms_archive_name} \\ \textit{archive_directory_file} \end{array} \right\}$

SESAM/SQL creates a SESAM backup copy on magnetic tape cartridge using the HSMS or ARCHIVE software products.

The backup is performed by calling HSMS or ARCHIVE as a subprogram.

You must specify *hsms_archive_name* or *archive_directory_file* as an alphanumeric literal.

If an HSMS parameter file is found then the backup is performed using HSMS.

hsms_archive_name designates the HSMS archive that is to be used. The length of the name is restricted to 22 characters (or 12 characters without the user ID).

The name of the HSMS parameter file is derived from the name of the HSMS archive or a combination of the configuration name and the DBH name (connection module parameter CNF or NAM), here referred to as *c* and *n*:

hsms_archive_name.HSMS-PAR or SESAM*cn*.HSMS-PAR

First of all, SESAM/SQL searches for the file *hsms_archive_name*.HSMS-PAR, and then, across different sessions, the file SESAM*cn*.HSMS-PAR.

The HSMS parameter file and the HSMS archive must be created on the DBH's BS2000 user ID. The HSMS parameter file can be empty. In this case, it simply has the task of selecting HSMS as the backup procedure.

SESAM/SQL creates a log file under the name *hsms_archive_name*.SYSLST. New information is appended to this file on subsequent COPY runs.

If the database is located on a DB user ID and the backup is to take place using HSMS, the DBH user ID must be defined as co-owner in the DB user ID. It is not enough to create the database files as shareable.

If no HSMS parameter file is present, the backup is performed with ARCHIVE.

archive_directory_file designates an ARCHIVE directory. A parameter file can also be specified for ARCHIVE. Its name is formed in the same way as for HSMS:

archive_directory_file.ARC-PAR or SESAM*cn*.ARC-PAR

The ARCHIVE directory and the archive parameter file must be located on the BS2000 user ID under which the DBH is running.

ARCHIVE creates a log file under the name *archive_directory_file*.SYSLST; this file is extended for further COPY statements.

You can only specify USING DIRECTORY for an ARCHIVE backup in conjunction with databases located on the BS2000 user ID under which the DBH is running.

BY_ADD_MIRROR_UNIT

Specifying **BY_ADD_MIRROR_UNIT** allows you to back up database files which are located on the additional mirror unit of a multi-mirror pair on the Local Symmetrix in an HSMS archive. The user does not need to worry about splitting up or subsequently synchronizing the additional mirror unit.

You may only specify this clause if the backup occurs by means of HSMS and the database files are located on the same additional mirror unit. The user ID on which the DBH runs must have the HSMS-ADMINISTRATION privilege. You can find more detailed information about backing up additional mirror units with HSMS in the “[Core manual](#)”.

BY_SRDF_TARGET

When **BY_SRDF_TARGET** is specified, database files located on the additional mirror unit of a target unit of a remote-copy pair in a Remote Symmetrix can be backed up in an HSMS archive. The user does not need to worry about splitting up or subsequently synchronizing the additional mirror unit which is assigned to the target unit.

You may only specify this clause if the backup occurs by means of HSMS and the database files are located on the same additional mirror unit in the Remote Symmetrix. The HSMS archive must be located on the system to which the source unit is also connected. The user ID on which the DBH runs must have the HSMS-ADMINISTRATION privilege.

You can find more detailed information about backing up additional mirror units with HSMS in the “[Core manual](#)”.

USING clause omitted:

SESAM/SQL creates a SESAM backup copy (disk copy) on the default storage group D0STOGROUP. If the default storage group is not available, COPY is rejected.

[NO] CHECK FORMAL

If you specify CHECK FORMAL, SESAM/SQL automatically executes the utility statement CHECK FORMAL to check the format of the SESAM backup copy if it is created on disk (online or offline).

In the case of online creation of a SESAM backup copy from an additional mirror unit using HSMS, SESAM/SQL performs the formal check on the detached additional mirror unit while the files are backed up to an HSMS archive.

If the source unit or normal unit belongs to a shared pubset, the SESAM/SQL DBH must run on the server which is represented as the master side for the shared pubset.

If the SESAM backup copy is created offline on magnetic tape cartridge, SESAM/SQL performs the formal check of the original before the backup copy is created.

If, during the formal check in the cases described above, an error is found, this means the backup was not successful; the original space is not flagged as “defective”, however.

The specification of CHECK FORMAL together with the ONLINE and USING DIRECTORY clauses is permissible only if either the BY_ADD_MIRROR_UNIT or BY_SRDF_TARGET clause is specified.

Otherwise the format check is not possible when SESAM backup data is created on tape cartridge online. In such a case the utility statement CHECK FORMAL should be issued before the COPY statement.

If [NO] CHECK FORMAL is not specified, the following default settings apply:

If you have specified the clauses ONLINE and USING DIRECTORY, NO CHECK FORMAL is valid. In all other cases, CHECK FORMAL is valid.

LOG

Following COPY CATALOG ... LOG logging is activated for all user spaces and for the catalog space of the database.

LOG reactivates logging for a user space after the SESAM backup copy has been created if logging was previously deactivated for this user space with the SQL statement ALTER SPACE. Logging is still deactivated in the SESAM backup data created.

Furthermore, you can use COPY SPACE ... LOG to activate logging for a user space for which no logging was initially specified with CREATE SPACE NO LOG.

LOG omitted:

The current setting for logging remains valid.

EXCEPT NO LOG INDEX SPACE

Only possible with COPY CATALOG.

If this clause is specified then the spaces for which the following two conditions apply are excluded from the backup:

- The spaces are pure index spaces and
- logical data saving is deactivated for these spaces.

A backup created using COPY CATALOG ... EXCEPT NO LOG INDEX SPACE cannot be added in the SQL database catalog list as a backup copy for retrieval purposes.

You can use a backup performed with COPY CATALOG ... EXCEPT NO LOG INDEX SPACE to create a partial replication. This partial replication can be used for a recovery and can be updated by means of REFRESH REPLICATION. However, this partial replication cannot be used for a retrieval.

BS2000 file names for SESAM backup copies and CAT-REC copy

Catalog space: :*catid:bs2000_user_id.catalog.CATALOG.nnnnnn*

User spaces: :*catid:bs2000_user_id.catalog.space.nnnnnn*

CAT-REC copy :*catid:bs2000_user_id.catalog.CAT-REC.COPY*

nnnnnn is a six-digit integer used to number the SESAM backup copies sequentially. In the case of a backup using ARCHIVE or HSMS this number is not visible.

Managing SESAM backup copies

The catalog table RECOVERY_UNITS (see the “[Core manual](#)”) contains information on the SESAM backup copies of user spaces. You can access this information with the help of the utility monitor (see the “[Utility Monitor](#)” manual) or an ESQL program. You can delete entries for SESAM backup copies that are no longer needed from RECOVERY_UNITS with the utility statement MODIFY.

The CAT-REC file contains information on SESAM backup copies of the database and catalog space. You can also use the utility monitor to access the information in the CAT-REC file (offline update). You can also update the current CAT-REC file using the utility statement MODIFY (online update, see [page 177](#)).

The save versions in the HSMS archive and in the ARCHIVE directory must be maintained separately.

Reading SESAM backup copies

The user can read-access a SESAM backup copy of a complete SESAM/SQL database under a separate DBH. This reduces the workload on the current session with the original database. A SESAM backup copy can be accessed for reading from several DBHs simultaneously.

The user can add the SESAM backup copy to the SQL database catalog with the aid of the ADD-SQL-DB-CATALOG-ENTRY statement. This is done by specifying the number of the SESAM backup copy of the catalog space in the COPY-NUMBER operand.

Only DML read access is permitted, i.e. utility statements are not permitted either. The SESAM backup copy which is read accessed remains available without any changes for recoveries that may become necessary at some later stage.

Examples

1. In the example below, a SESAM backup copy of the database ORDERCUST is created, i.e. a backup copy of the catalog space and user spaces of ORDERCUST. The space INDEXSPACE is excluded from the backup because it contains only indexes and is not involved in the logical data backup.



```
COPY CATALOG ordercust USING STOGROUP stogroup3  
EXCEPT NO LOG INDEX SPACE
```

2. In this example, a SESAM backup copy of the catalog space of the database ORDERCUST is created.



```
COPY CATALOG_SPACE ordercust USING STOGROUP stogroup3  
NO CHECK FORMAL
```

3. In this example, a SESAM backup copy of the user space TABLESPACE of the database ORDERCUST is created.



```
COPY SPACE tablespace USING STOGROUP stogroup3
```

In all cases, SESAM/SQL creates the SESAM backup copies (disk copies) on the storage group STOGROUP3. The individual SESAM backup copies are created offline. Their formal correctness is checked in the 1st and 3rd examples.

See also

RECOVER, CREATE REPLICATION

CREATE CATALOG - Create a database (catalog space)

You use CREATE CATALOG to create a database and the corresponding BS2000 file for the catalog space.

On pubsets with "large files" the catalog space can be up to 4 TB in size. Otherwise it can be up to 64 GB in size.

The database's coded character set (CCS; synonym: code table) is used in database operation.

CREATE CATALOG defines the authorization identifier for the universal user. The universal user is a user with comprehensive authorization.

Once CREATE CATALOG has been executed successfully, the catalog space and thus the database represented by the catalog space is added in the SQL database catalog list of the current DBH session.

You cannot execute CREATE CATALOG unless the SQL database catalog list has room for at least one more entry (see the "[Database Operation](#)" manual). Furthermore, the database name specified in the CREATE CATALOG statement cannot already exist in the SQL database catalog.

The CREATE CATALOG statement may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

If a database is to be created on a DB user ID, the database administrator must make preparations for creating the database files, see section "Database files and job variables on foreign user IDs" in the "[Core manual](#)".

The following files are affected:

- the catalog space and, in case of logging, the CAT-REC file; both these files must be on the same BS2000 user ID.
- the CAT-LOG and DA-LOG files if they are to be stored on this DB user ID. Otherwise, these files are created on the DBH user ID.
- the user spaces of this database at the latest before the respective SQL statement CREATE SPACE is executed.

```
CREATE CATALOG catalog [PASSWORD password]
```

```
  [ON USER_ID bs2000_user_id]
```

```
  [CODE_TABLE ccs_name]
```

```
  [CATALOG_SPACE space_parameter_list]
```

```
  [STOGROUP stogroup stogroup_attributes]
```

```
  [MEDIA STOGROUP stogroup stogroup_attributes]
```

```
  USER authorization_identifier [, system_user_id]
```

$$space_parameter_list ::= \left\{ \begin{array}{l} PRIMARY \textit{ allocation} \\ SECONDARY \textit{ allocation} \\ PCTFREE \textit{ percent} \\ [NO] \underline{SHARE} \\ [NO] \underline{DESTROY} \\ NO LOG \end{array} \right\} \dots$$

$$stogroup_attributes ::= \left\{ \begin{array}{l} VOLUMES(\textit{volume_name} \dots) ON \textit{dev_type} \\ PUBLIC \end{array} \right\} [ON \textit{catid}]$$

$$system_user_id ::= \left\{ \begin{array}{l} \textit{utm_user} \\ \textit{bs2000_user} \end{array} \right\}$$

$$\textit{utm_user} ::= \left(\left\{ \begin{array}{l} \textit{hostname} \\ * \end{array} \right\}, \left\{ \begin{array}{l} \textit{utm_application_name} \\ * \end{array} \right\}, \left\{ \begin{array}{l} \textit{utm_userid} \\ * \end{array} \right\} \right)$$

$$\textit{bs2000_user} ::= \left(\left\{ \begin{array}{l} \textit{hostname} \\ * \end{array} \right\}, [*], \left\{ \begin{array}{l} \textit{bs2000_userid} \\ * \end{array} \right\} \right)$$

catalog

Name of the new SESAM/SQL database to be created. The catalog space for this database is created with CREATE CATALOG.

There must not yet be any entry in the SQL database catalog list for *catalog*. In addition, there must be room in the SQL database catalog list for at least one more entry. SESAM/SQL enters *catalog* in the SQL database catalog list as the logical and physical name of the new database (see the “[Database Operation](#)” manual).

SESAM/SQL only uses an existing BS2000 file for the catalog space if the file is empty. However, in this case you must also use CREATE CATALOG to initialize the catalog space.

PASSWORD *password*

BS2000 password for the file of the catalog space. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

password is valid for the entire database, i.e. not only for the catalog space but also for all user spaces created later with CREATE SPACE (see the “[SQL Reference Manual Part 1: SQL Statements](#)”), for all user spaces created automatically, for the CAT-REC file, the CAT-LOG files, the DA-LOG file and the PBI file, as well as for SESAM backup copies.



If the database is to be located on a DB user ID, the following applies:

- in the case of co-ownership, SESAM/SQL assigns the password for the files
- if the database administrator creates the file himself/herself, the password must also be assigned at creation time.

See section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

PASSWORD *password* omitted:

The catalog space is created without a password.

ON USER_ID *bs2000_user_id*

Name of the BS2000 user ID (DB user ID) on which the database is to be created. Preparations must have been made for catalog space and CAT-REC file for logging, see [page 105](#).

ON USER_ID *bs2000_user_id* omitted:

The catalog space is created on the DBH user ID.

CODE_TABLE *ccs_name*

Name of a coded character set (CCS) defined in BS2000 (system component XHCS), up to 8 characters in length.

ccs_name must be specified as an alphanumeric literal consisting of uppercase letters, digits, @, # or \$. The first character may not be a digit.

The CCS names of EBCDIC character sets or self-defined character sets based on EBCDIC character sets can be specified for *ccs_name*.

The value *_NONE_* can also be specified for *ccs_name*. No coded character set is then used for the database. This corresponds to the behavior of SESAM/SQL before V5.0.

The *ccs_name* (see also: ALTER CATALOG) is entered and used the database's metadata, e.g. in the utility statement LOAD for checking against the input file's CCSN.

Introductory information on using coded character sets in SESAM/SQL is provided in the section "Unicode concept in SESAM/SQL" in the "[Core manual](#)".

CODE_TABLE *ccs_name* omitted:

In SESAM/SQL the CCSN *_NONE_* is entered in the database's metadata, see above.

CATALOG_SPACE *space_parameter_list*

You may only specify each of the following space clauses once: PRIMARY, SECONDARY, PCTFREE, [NO] SHARE, [NO] DESTROY, NO LOG.

PRIMARY *allocation*

Primary allocation for the BS2000 file for the catalog space in units of 2K (BS2000 halfpage). *allocation* must be an integer between 1 and 33 554 430.

SESAM/SQL always makes sure that the catalog space is big enough for the catalog tables in which the database metadata is to be stored. This means that SESAM/SQL ignores a primary allocation that is too small.

PRIMARY *allocation* omitted:

SESAM/SQL creates a catalog space which is at least large enough for the catalog tables in which the database metadata is to be stored.

SECONDARY *allocation*

Secondary allocation for the BS2000 file for the catalog space in units of 2K (BS2000 halfpage). *allocation* must be an integer between 0 and 32767. If you specify a *value* smaller than 24, SESAM/SQL automatically uses SECONDARY 24.

SECONDARY *allocation* omitted:
SECONDARY 24 is used.

PCTFREE *percent*

Free space reservation for the BS2000 file for the catalog space expressed as a percentage.

percent must be an integer between 0 and 70.

SESAM/SQL stores the specification for the free space reservation in the database's metadata. This specification is evaluated when the catalog space is reorganized with the utility statement REORG.

PCTFREE *percent* omitted:
PCTFREE 20 is used.

[NO] SHARE

SHARE indicates that the file for the catalog space is shareable.

NO SHARE indicates that the file for the catalog space is not shareable. The current setting is retained for an existing file.

[NO] DESTROY

DESTROY indicates that when the file for the catalog space is deleted, the storage space is to be overwritten with binary zeros.

NO DESTROY indicates that when the file for the catalog space is deleted, only the storage space is released.

The current setting is retained for an existing file.

[NO] DESTROY omitted:

The current setting is retained for an existing file.

NO LOG

The entire SESAM/SQL database is operated without logging.

This applies not only to CAT logging in the catalog space but also to DA logging in all new user spaces created with the SQL statement CREATE SPACE. It also applies for any user spaces with the name D0*authorization_identifier* created automatically. *authorization_identifier* is the authorization identifier with the corresponding CREATE SPACE authorization.

SESAM/SQL does not create a CAT-LOG file. SESAM/SQL creates the CAT-REC file when the first SESAM backup copy of the database or catalog space is created with the COPY utility statement.

SESAM/SQL uses the specifications in the CAT-REC media record in the media table to create the file (see the MEDIA STOGROUP *stogroup* clause).

NO LOG omitted:

The entire SESAM/SQL database is operated with logging.

CAT logging is performed for the catalog space. The default value (with DA logging) is valid for any new user spaces created with CREATE SPACE and user spaces created automatically.

If the database is to be located on a DBH user ID, SESAM/SQL creates the CAT-REC file and the first CAT-LOG file according to the entries in the media table (see the MEDIA STOGROUP *stogroup* clause).

If the database is to be located on a DB user ID, preparations must have been made for the CAT-REC file, see [page 105](#).

Logging cannot be subsequently deactivated for the entire database. However, it can be deactivated for individual user spaces.

CATALOG_SPACE *space_parameter_list* omitted:

SESAM/SQL uses the above-mentioned default values for the individual space clauses.

STOGROUP *stogroup*

Name of a storage group. STOGROUP *stogroup* defines the storage group for the catalog space. The name cannot be D0STOGROUP. You can qualify the name of the storage group with the database name *catalog*. If *stogroup* includes a database name, it must match the database name *catalog*. SESAM/SQL enters the name of the storage group in the database's metadata.

Different *stogroup* names must be used for STOGROUP and MEDIA STOGROUP.

The authorization identifier specified for USER, i.e. the universal user, owns the storage group and is granted the special privilege USAGE for this storage group.

In addition to *stogroup*, SESAM/SQL also enters another storage group, the default storage group D0STOGROUP, in the database's metadata. By default, this storage group comprises

public disks on the default pubset of the user ID under which the DBH is running. The database administrator can use ALTER STOGROUP to change this setting at any time.

The authorization identifier specified for USER is the owner of the default storage group.

PUBLIC is authorized to use the default storage group.

The clause STOGROUP has no effect if the database administrator has already created the catalog space with the CREATE-FILE command. However, the storage group is entered in the database's metadata.

STOGROUP *stogroup* omitted:

SESAM/SQL creates the catalog space on the default storage group.

At least one of the clauses STOGROUP and MEDIA STOGROUP should be specified. Otherwise, SESAM/SQL creates the catalog space, the CAT-REC file and the first CAT-LOG file on the default storage group. With regard to media recovery procedures we recommend not to store all the files on the same storage group.

MEDIA STOGROUP *stogroup*

Name of a storage group. MEDIA STOGROUP *stogroup* defines the storage group for the CAT-REC file. If logging has been specified for the database, this storage group is also used for the first CAT-LOG file. The name cannot be D0STOGROUP. You can qualify the name of the storage group with the database name *catalog*. If *stogroup* includes a database name, it must match the database name *catalog*. SESAM/SQL enters the first media record for the file type CAT-REC and, if appropriate, the file type CAT-LOG in the media table according to the specifications made for MEDIA STOGROUP.

SESAM/SQL creates the CAT-REC file and, if appropriate, the first CAT-LOG file according to these specifications. Default values are used for the file attributes (primary allocation, secondary allocation, etc.) (see the “[Core manual](#)”).

Different *stogroup* names must be used for STOGROUP and MEDIA STOGROUP.

The authorization identifier specified for USER, i.e. the universal user, owns the storage group and is granted the special privilege USAGE for this storage group.

MEDIA STOGROUP *stogroup* omitted:

SESAM/SQL creates the CAT-REC file and, if appropriate, the first CAT-LOG file on the default storage group D0STOGROUP. In this case, the corresponding entries in the media table contain the specification D0STOGROUP and the default values for the attributes of the CAT-LOG and CAT-REC file (see the “[Core manual](#)”).

PUBLIC is authorized to use the default storage group.

You should specify at least one of the clauses STOGROUP and MEDIA STOGROUP as otherwise SESAM/SQL creates the catalog space, the CAT-REC file and the first CAT-LOG file on the default storage group. This is not a good idea with regard to media recovery procedures.

stogroup-attributes ::=

The following attributes may only be allocated once.

VOLUMES (*volume_name* ,...)

Volume serial number of private disks. You must specify *volume_name* as an alphanumeric literal. You can only specify each VSN once. A maximum of 100 disks can be specified.

ON *dev_type*

Device type of the private volumes. You must specify *dev_type* as an alphanumeric literal. All the volumes in a storage group must have the same device type.

The VOLUMES specification is not used if the associated files were already present prior to CREATE CATALOG. However, the specifications are stored in the database's metadata and are effective in further statements (REORG, COPY).

PUBLIC

The storage group comprises all public volumes (pubset) whose catalog ID is the *catlog_id* specified for **ON** *catid*.

ON *catid*

Catalog ID of the pubset in which the files are to be cataloged.

You must specify *catid* as an alphanumeric literal.

In the case of private volumes (i.e. VOLUMES ... specified in the STOGROUP clause), the files are cataloged on this pubset but are created on a private volume. If **PUBLIC** is specified, the files are also created on this pubset.

ON *catid* omitted:

The default pubset is which is assigned to the BS2000 user ID under which the DBH is running is used.

USER *authorization_identifier* [, *system_user_id*]

Definition of the universal user.

authorization_identifier

is the authorization identifier for the universal user.

system_user_id

is the system user ID (BS2000 user, UTM user) for the universal user. If this specification is omitted, SESAM/SQL uses the system user ID of the user that issued the CREATE CATALOG statement.

The universal user is the only user authorized to execute SQL and utility statements for the new database unless the universal user has used the SQL statements **CREATE USER** and **CREATE SYSTEM_USER** to create other SQL users and has granted these users the appropriate privileges with the **GRANT** statement (see the “[Core manual](#)” and the “[SQL Reference Manual Part 1: SQL Statements](#)”).

utm_user ::=

Specification of the UTM user.

hostname

You must specify the symbolic host name *hostname* as an alphanumeric literal.

If DCAM is not available on the host, the host is assigned the name "HOMEPROC".

For UTM-D: Specification of the local host on which the SESAM/SQL database connection was generated.

* All hosts.

utm_application_name

Name of the UTM application. You must specify *utm_application_name* as an alphanumeric literal.

UTM-D: Name of the local UTM application

* All UTM applications

utm_userid

For local UTM system users, you enter the UTM user ID defined with KDCSIGN. For UTM-D, you specify the local UTM session name (LSES). You must specify *utm_userid* as an alphanumeric literal.

* All UTM user IDs

bs2000_user ::=

Specification of the BS2000 user.

hostname

Symbolic host name. You must specify *hostname* as an alphanumeric literal. If DCAM is not available on the host, the host is assigned the name "HOMEPROC".

* All hosts.

bs2000_userid

BS2000 user ID You must specify *bs2000_userid* as an alphanumeric literal.

* All BS2000 user IDs

BS2000 file names for the catalog space and the CAT-REC file

The BS2000 file for the catalog space and, in case of logging, the CAT-REC file are either created on the DBH user ID or on a DB user ID.

Catalog space: :*catid:bs2000_user_id.catalog*.CATALOG

CAT-REC file: :*catid:bs2000_user_id.catalog*.CAT-REC

Examples

1. The example below creates the catalog space for the database ORDERCUST.

```
CREATE CATALOG ordercust PASSWORD 'C''PW#1'''
CODE_TABLE 'EDF041' CATALOG_SPACE SHARE NO DESTROY
MEDIA STOGROUP ordercust.stogroup1 PUBLIC
USER utiuniv
```

The password 'PW#1' is assigned to the catalog space or database. EDF041 is selected as the database's coded character set. As a BS2000 file, the catalog space must be shareable and not be overwritten with binary zeros when it is deleted. The default values assigned by SESAM/SQL apply to all other parameters. SESAM/SQL creates the catalog space on the default storage group D0STOGROUP and on the user ID under which the DBH is running. The CAT-REC file and the first CATALOG file are created on the storage group STOGROUP1. UTIUNIV is specified as the authorization identifier for the universal user. The system user ID of the user who issued the CREATE CATALOG statement is used as the system user ID for the universal user.

2. The example below creates the catalog space for the database ORDERCUST.



```
CREATE CATALOG ordercust
CATALOG_SPACE SHARE DESTROY
STOGROUP stogroup1 PUBLIC
MEDIA STOGROUP stogroup2 PUBLIC
USER utiuniv, (*,*,*)
```

As a BS2000 file, the catalog space must be shareable and be overwritten with binary zeros when it is deleted. The default values assigned by SESAM/SQL apply to all other parameters.

SESAM/SQL creates the catalog space on the default storage group STOGROUP1 and on the user ID under which the DBH is running. The CAT-REC file and the first CATALOG file are created on the storage group STOGROUP2.

UTIUNIV is specified as the authorization identifier for the universal user. All hosts, all UTM user IDs and all BS2000 user IDs are permitted as the system user ID.

See also

ALTER CATALOG

CREATE MEDIA DESCRIPTION - Define file attributes of database-specific files

You use CREATE MEDIA DESCRIPTION to define the file attributes for the files of a certain database-specific or space-specific file type.

CREATE MEDIA DESCRIPTION creates the first entry (media record) in the media table for the database-specific file type involved. The media table is described in the “[Core manual](#)”.

SESAM/SQL recognizes the following database-specific file types:

- CAT-REC file
- CAT-LOG files
- DA-LOG files
- PBI files
(media records also apply to the work file in the case of online backup with HSMS)

SESAM/SQL recognizes the following space-specific file type:

- DDL-TA-LOG files

The current authorization identifier must have the special privilege UTILITY.

```
CREATE MEDIA DESCRIPTION FOR { DALOG
                             CATLOG
                             CATREC
                             PBI } AT CATALOG catalog
```

```
{ PRIMARY allocation
  SECONDARY allocation
  [NO] SHARE } ... [[NO] DEVICE REQUEST] [STOGROUP stogroup] 1
```

¹ TAPE *device type* can no longer be specified in SESAM/SQL V7.0 and higher.

FOR {
 DALOG
 CATLOG
 CATREC
 PBI
 DDLTA }

Indicates the file type for which the first media record is to be created. If a media record already exists for the specified file type in the media table, CREATE MEDIA DESCRIPTION is rejected. Please note that the first media record for the CAT-REC file was created when CREATE CATALOG was executed. If logging has been activated for the database, the first media record for the CAT-LOG files is also created.

DALOG The first media record for the file type DA-LOG file is created.

CATLOG The first media record for the file type CAT-LOG file is created.

CATREC The first media record for the file type CAT-REC file is created.

PBI The first media record for the file type PBI file is created (also applies to the work file in the case of online backup with HSMS).

DDLTA The first media record for the file type DDL-TA-LOG file is created.

AT CATALOG *catalog*

Name of the database to which the processed media table belongs.

You may only specify the space clauses PRIMARY, SECONDARY and [NO] SHARE once:

PRIMARY *allocation*

Primary allocation for files of the specified file type in units of 2K (BS2000 halfpage). *allocation* must be an integer between 1 and 33 554 430.

If the value specified for primary allocation is too small, SESAM/SQL automatically corrects it to the default value.

PRIMARY *allocation* omitted: SESAM/SQL uses the following default values:

- DA-LOG files, CAT-LOG files: 768
- CAT-REC file: 12
- PBI files: 576
- DDL-TA-LOG files: 1536

SECONDARY *allocation*

Secondary allocation for files of the specified file type in units of 2K (BS2000 halfpage) if the file type CATREC or PBI is specified. If the file type DALOG or CATLOG is specified, you cannot specify the space clause SECONDARY.

allocation must be an integer between 0 and 32 767.

If the value specified for secondary allocation is too small, SESAM/SQL automatically corrects it to the default value.

SECONDARY *allocation* omitted: SESAM/SQL uses the following default values:

- DA-LOG file, CAT-LOG file: 0
- CAT-REC file: 12
- PBI files: 72
- DDL-TA-LOG files: 384

[NO] SHARE

SHARE indicates that the files of the specified file type are shareable.

NO SHARE indicates that the files of the specified file type are not shareable.

[NO] DEVICE REQUEST

DEVICE REQUEST indicates that SESAM/SQL requests additional devices for the specified file type at the console once all the entries in the media table have been exhausted.

If SESAM/SQL requests additional devices all the entries in the media table have been exhausted, then the following entries are possible: ¹

PUBLIC *catid*

The file is created on the specified pubset.

catid

BS2000 catalog ID

The BS2000 catalog ID must be specified.

You must specify *catid* as an alphanumeric literal.

If a CATID list was defined then the catalog ID must be specified in the list.

IGNORE

The file is not created.

¹ TAPE *device_type* can no longer be specified in SESAM/SQL V7.0 and higher.

PRIVATE *catid, dev_type, vsn, vsn,...*

The file is created on private disks with the specified Volume Serial Number and the specified device type and is catalogued using the specified catalog ID. A maximum of 6 Volume Serial Numbers can be specified.

catid BS2000 catalog ID
The BS2000 catalog ID must be specified.
You must specify *catid* as an alphanumeric literal. If a CATID list was defined then the catalog ID must be specified in the list.

dev_type Device type of the private disk. You must specify
dev_type as an alphanumeric literal.

vsn Volume Serial Number.
You must specify *vsn* as an alphanumeric literal.

NO DEVICE REQUEST indicates that SESAM/SQL does not request additional devices for the specified file type at the console once all the entries in the media table have been exhausted

STOGROUP *stogroup*

Name of the storage group whose volumes are to be used for storing the files of the specified file type. You can qualify the name of the storage group with the database name *catalog*.

STOGROUP *stogroup* omitted:

SESAM/SQL creates the files of the specified file type on the default storage group D0STOGROUP.

Example

In the example below, the first media record for DA-LOG files is entered in the media table of the database ORDERCUST.



```
CREATE MEDIA DESCRIPTION FOR DALOG AT CATALOG ordercust
SHARE NO DEVICE REQUEST STOGROUP stogroup1
```

This statement defines the storage medium and file attributes for the DA-LOG files of the database ORDERCUST. As indicated by the parameters specified, SESAM/SQL creates the DA-LOG files for the database ORDERCUST on the storage group STOGROUP1 with a default value for the primary allocation of 768 2K-units and the file attribute “shareable”. SESAM/SQL will not request further devices for DA-LOG files at the console once all the entries in the media table have been exhausted.

See also

ALTER MEDIA DESCRIPTION, DROP MEDIA DESCRIPTION

CREATE REPLICATION - Create replication

You use CREATE REPLICATION

- to create a replication from a SESAM backup copy which was created using COPY CATALOG. The SESAM backup copy may be created on magnetic tape cartridge or on disk.
- to create a replication from a foreign copy created from a consistent database using optional means.

This replication is then an image file of the original at the time the SESAM backup copy or the foreign copy was created. You can also create a partial replication by using FOR SPACE to select user spaces.

A replication can only be created from a SESAM backup copy or from foreign copies that were created with SESAM/SQL V9.0.

For information on how to create foreign copies, refer to the “[Core manual](#)”. A space must not have the state “check pending” or “recover pending” if it is to be subsequently incorporated in the replication.

The file names of the replication have the same structure as the file names of a database, with the exception of the CAT-REC file. Its name is *replication.CAT-REC.REPL*.

Replications and partial replications are available for read access. If a replication contains a partitioned table, all the partitions of the table must be contained in the replication. A partial replication must always be complete in itself, i.e. the partial replication must contain all the indexes for all the tables which belong to the partial replication.

You use the utility statement REFRESH REPLICATION to update replications and partial replications. This is the only type of modifications that is permissible for replications. You can use the replication and the utility statement RECOVER USING/TO REPLICATION to repair or reset the original from which the replication was created. You can also use the same statements to create a new original.

You can only use REFRESH REPLICATION for subsequent updates if logging has been activated for all the spaces in the replication. However, you can create a replication without logging.

Replications can also be created from backups with defective indexes. There are two restrictions on the use of replications with defective indexes:

- No RECOVER INDEX is possible with the replication.
- Following a RECOVER INDEX in the original, the affected spaces must first be deleted from the replication using REFRESH REPLICATION FOR SPACE. You can then add them back to the replication using REFRESH SPACE (see section “Replication of a database” in the “[Core manual](#)”).

You can only create a replication on hard disk.

If the “Job Variables” software product is installed in BS2000, SESAM/SQL makes two entries in a job variable available following CREATE REPLICATION. These can be used to ascertain as of which CAT-LOG and DA-LOG file the logging data must be present for the next REFRESH REPLICATION.

The name of the job variable is `SESAM.replication.NEXT-REPL-LOG`.

If the replication is located on a DB user ID, an attempt will be made to create the job variable in the replication user ID as well. Preparations must have been made for this purpose, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”. If the job variable cannot be created on the replication user ID, it will be created on the DBH user ID.

The CREATE REPLICATION statement may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

Read authorizations are necessary for the copy which is used to create the (partial) replication.

After CREATE REPLICATION has been executed, the replication is added in the SQL database catalog list and can be accessed.

If a replication is to be created from a backup copy generated using HSMS then it is necessary to define co-ownership rights in the cases listed below (see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”):

- the replication is to be located on a user ID other than the DBH user ID
- the original database is located on a DB user ID

It may be advisable to assign partially qualified co-ownership. This ensures that co-ownership is defined for all the affected files.

```

CREATE REPLICATION replication [PASSWORD password]
  [ON USER_ID bs2000_user_id]
  [USING MEDIA stogroup_attributes]

FROM [FOREIGN] CATREC catrec_file
  [PASSWORD password]
  [ON USER_ID bs2000_user_id]
[FOR SPACE space,...]
[ { RENAME }
  { COPY } ]

stogroup_attributes ::= { VOLUMES (volume_name,...) ON dev_type
  PUBLIC } [ON catid]

```

replication

Name under which the replication is to be created.

If a non-empty catalog space has already been created under the name *replication*, this must be an old catalog space of a replication. Otherwise CREATE REPLICATION is aborted.

There must as yet be no entry in the SQL catalog list for the specified name. At least one entry must still be available.

PASSWORD *password*

BS2000 password for the replication. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
 - string* contains four printable characters.
- 'X'*hex_string*"
 - hex_string* contains eight hexadecimal characters.
- 'n'
 - n* identifies an integer from - 2147483648 to + 2147483647.

password is valid for the replication and for the CAT-REC file of the replication.



If the replication is to be located on a DB user ID, the following applies:

- in the case of co-ownership, SESAM/SQL assigns the password for the files
- if the database administrator creates the file himself/herself, the password must also be assigned at creation time.

See section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

If RENAME is specified the replication is given the password after it has been renamed.

PASSWORD *password* omitted:

The replication is created without a password. The password of the foreign copy remains valid.

ON USER_ID *bs2000_user_id*

Name of the BS2000 user ID (DB user ID) on which the replication is to be created.

If the replication is created from a disk copy, the database administrator must make preparations for the replication files, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

If the replication is created from a backup copy generated with HSMS then HSMS uses the name of the backup copy plus the suffix “.REPL” when reading in the files. In the second step, SESAM/SQL renames the files to the name of the replication.

On the DB user ID, co-ownership of all the file names used in this process must be accorded to the DBH user ID.

It may be advisable to assign partially qualified co-ownership both for the backup copy and for the replication. This ensures that co-ownership is defined for all the affected files.

You must specify the user ID without a “\$”. In this case, the CAT-REC file name is *replication.CAT-REC.REPL*.

If *bs2000_user_id* is specified together with RENAME, *bs2000_usr_id* must be the same user ID as that specified under FROM CATREC ... ON USER_ID. Otherwise the process will be aborted with SQLSTATE.

ON USER_ID *bs2000_user_id* omitted:

The replication is created on the DBH user ID.

USING MEDIA *stogroup_attributes*

Here you identify the media which are to be used to store the replication files.

USING MEDIA *stogroup_attributes* may not be specified together with **RENAME**. Otherwise the process will be aborted with **SQLSTATE**.

stogroup-attributes ::=

The following attributes may only be allocated once.

VOLUMES (*volume_name* ,...)

Volume serial number of private disks. You must specify *volume_name* as an alphanumeric literal. You can only specify each VSN once. A maximum of 100 disks can be specified.

ON *dev_type*

Device type of the private volumes. You must specify *dev_type* as an alphanumeric literal. All disks must be of the same device type.

The **VOLUMES** specification is not evaluated if the associated files are already present before **CREATE REPLICATION**

PUBLIC

The storage group comprises all public volumes (pubset) whose catalog ID is the *catlog_id* specified for **ON** *catid*.

ON *catid*

Catalog ID of the pubset, in which the files are to be created.

You must specify *catid* as an alphanumeric literal.

On private disks (i.e. when **VOLUMES** is specified), the files are cataloged in this pubset, but created on private disks. If **PUBLIC** is specified, the files are also created on this pubset.

ON *catid* omitted:

The default pubset is which is assigned to the BS2000 user ID under which the DBH is running is used.

USING MEDIA *stogroup_attributes* omitted and no files created:

The default pubset is which is assigned to the BS2000 user ID under which the DBH is running is used.

FROM [FOREIGN] CATREC *catrec_file*

Name of a CAT-REC file or of a CAT-REC copy without BS2000 user ID. This file contains the metadata which is to be used to create the replication. It must be accessible for reading.

FOREIGN

If FOREIGN is specified, a foreign copy of a database is specified through its CAT-REC file. The name format is *catalog*.CAT-REC. The file names of the foreign copy are consequently:

catalog.CATALOG
catalog.space
...

FOREIGN not specified:

The name of the *catrec_file* has the format *catalog*.CAT-REC.COPY. The last backup record in the file must contain a backup of the entire database.

PASSWORD *password*

BS2000 password of the CAT-REC file and the backups or foreign copies which are to be read in. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

ON USER_ID *bs2000_user_id*

Identifier specifying the location of the CAT-REC file and the copies. You must specify the user ID without a "\$".

FOR SPACE *space*,...

Names of the user spaces which are to belong to this partial replication. All the spaces must belong to the database from which the backup copy was derived. You can specify up to 999 space names. You can specify one and the same space only once.

The catalog space and the specified user spaces are then stored in the partial replication. You cannot specify the catalog space as a user space.

When selecting the user spaces you should ensure that the replication remains self-contained. The index space belonging to a table space can be incorporated into the replication. The index space does not, however, have to be incorporated if, for example, it is not subject to the logical data backup. An index space should not be incorporated into the replication without the corresponding table space.

FOR SPACE *space*,... omitted:

All user spaces and the catalog space of the backup are part of the replication if the backup used was a complete catalog backup. From a backup created with COPY CATALOG EXCEPT NO LOG INDEX SPACE, a partial replication is created which does not contain any index spaces without logging.

RENAME

Allowed only with FOREIGN CATREC. Otherwise the process will be aborted with SQLSTATE. RENAME requires exclusive write authorization. The foreign copy must be located on the same user ID as the replication. The foreign copy must not be open in any DBH. There must be no copies with the same name as the replication. This does not apply if the foreign copy and the replication name are identical. Since the foreign copy files are renamed, the files will no longer be available after RENAME. This is also the case if an error occurs.

CREATE REPLICATION cannot be repeated with this function.

COPY

The files are copied.

BS2000 replication file names

The BS2000 file for the catalog space and the replication's CAT-REC file is either created on the DBH user ID or on a DB user ID.

In the first case the files are created by the DBH.

In the second case, preparations must have been made for creating the files, see section "Database files and job variables on foreign user IDs" in the "[Core manual](#)". In the case of a replication from a backup created using HSMS, you should note that the DBH user ID must be defined as a co-owner of all the affected files on the DB user ID.

The replication's CAT-REC file is created with CREATE REPLICATION and contains all the records up to the backup record used to create the replication.

Replication: *:catid:bs2000_user_id.replication.CATALOG*

Space name *:catid:bs2000_user_id.replication.unqu_spacename*

CAT-REC file of the replication :
:catid:bs2000_user_id.replication.CAT-REC.REPL

Auxiliary files for a replication with HSMS:

:catid:bs2000_user_id.catalog.CATALOG.REPL

:catid:bs2000_user_id.catalog.ins_space_name.REPL

Reading the SESAM backup copy

The user can read-access a SESAM backup copy of a complete SESAM/SQL database under a separate DBH. This reduces the workload on the current session with the original database. A SESAM backup copy can be accessed for reading from several DBHs simultaneously.

The user can add the SESAM backup copy to the SQL database catalog with the aid of the ADD-SQL-DB-CATALOG-ENTRY statement. This is done by specifying the number of the SESAM backup copy of the catalog space in the COPY-NUMBER operand.

Only DML read access is permitted, i.e. utility statements are not permitted either. The SESAM backup copy which is read accessed remains available without any changes for recoveries that may become necessary at some later stage.

Example

The example below creates a partial replication from the last SESAM backup copy of the database ORDERCUST. This partial replication contains the catalog space and the user space TABLESPACE.



```
CREATE REPLICATION orderrep
FROM CATREC 'ORDERCUST.CAT-REC.COPY'
FOR SPACE tablespace
```

The replication is based on the status of the ORDERCUST database which is stored in the current CAT-REC copy of the original database.

See also

REFRESH REPLICATION

DROP MEDIA DESCRIPTION - Delete all the media table entries for database-specific file type

You use DROP MEDIA DESCRIPTION to delete all the entries for a certain database-specific or space-specific file type from the media table. The media table is described in the “[Core manual](#)”. DROP MEDIA DESCRIPTION does not affect the associated files, i.e. existing files are not deleted.

SESAM/SQL recognizes the following database-specific file types:

- CAT-REC file
- CAT-LOG files
- DA-LOG files
- PBI files
(media records also apply to the work file in the case of online backup with HSMS)

SESAM/SQL recognizes the following space-specific file type:

- DDL-TA-LOG files

The current authorization identifier must have the special privilege UTILITY.

DROP MEDIA DESCRIPTION FOR $\left\{ \begin{array}{l} \text{DALOG} \\ \text{CATLOG} \\ \text{CATREC} \\ \text{PBI} \\ \text{DDLTA} \end{array} \right\}$ AT CATALOG *catalog*

FOR $\left\{ \begin{array}{l} \text{DALOG} \\ \text{CATLOG} \\ \text{CATREC} \\ \text{PBI} \\ \text{DDLTA} \end{array} \right\}$

Indicates the file type for which all entries (media records) are to be deleted from the media table.

DALOG All media records for the file type DA-LOG file are deleted.

CATLOG All media records for the file type CAT-LOG file are deleted.

CAT-REC All media records for the file type CAT-REC file are deleted.

PBI All media records for the file type PBI file are deleted
(also applies to the work file in the case of online backup with HSMS).

DDLTA All media records for the file type DDL-TA-LOG file are deleted.

AT CATALOG *catalog*

Name of the database to which the processed media table belongs.

Example

In the example below, all media records for the file type PBI files belonging to the database ORDERCUST are deleted.



```
DROP MEDIA DESCRIPTION FOR PBI AT CATALOG ordercust
```

See also

CREATE MEDIA DESCRIPTION, ALTER MEDIA DESCRIPTION

EXPORT TABLE - Export a table from a database to an export file

You can use EXPORT TABLE to export a base table from a SESAM/SQL database to a BS2000 file. This BS2000 file is known as an export file. The export file is created with coded character set name (CCSN).

The table's metadata is stored in the export file. If you wish, you can also export user data. You can then use IMPORT TABLE to generate a new table from this export file.

An export file represents a simple way of copying a base table.

Fewer steps are required than when using the LOAD/UNLOAD statements since the structure of the table is stored in the export file's metadata and does not have to be reproduced in the destination table (see "[Core manual](#)").

The EXPORT/IMPORT TABLE statements allow you to:

- copy a base table within a database
- copy a base table from one database to another
- migrate a base table from one space to another
- migrate a base table to a partitioned table
- modify the number and boundaries of the partitions in a partitioned table
- reconstruct a base table
- reallocate the row numbers of a base table
- restructure a base table
- archive an individual base table

EXPORT/IMPORT TABLE can be used with base tables of any size. There is no limit on the size and the number of objects that are present, for example columns or integrity constraints

The following metadata is stored in the export file:

- The metadata of the base table and all the associated columns.
The metadata for a column also includes the SQL and/or CALL DML default values.
- The metadata for all the indexes in this base table.
- The metadata for all the integrity constraints with the exception of the reference constraints defined in this base table.
This also contains the non-NULL constraint for a column which is handled in the same way as a check constraint.

View definitions and the password catalog of password-protected CALL DML tables are not stored. The password protection has to be redefined using the SESAM/SQL SEPA utility.

EXPORT TABLE can also be performed on a replication or backup copy of the current SESAM/SQL version.

If an export file is saved to magnetic tape cartridge then it may extend over multiple physical volumes.

The number of rows that were copied from the table *table* to the export file is output in the SQLrowcount field in the ESQL-COBOL program's communication area.

When an EXPORT run is aborted, the content of any incomplete export file is logically deleted. Logical deletion prevents an incomplete export file from being mistakenly used for an IMPORT statement.

The affected spaces in which the table *table* is located are exclusively locked against modifications by foreign transactions as long as EXPORT TABLE is executing. However, concurrent read access to both the affected spaces and the table is possible. In the case of a partitioned table and if user data is to be exported, all the partitions affected must be available (see [table 3 on page 22](#)).

The current authorization identifier must possess the special privilege UTILITY or must be the owner of the schema to which the table belongs.

If a search condition is defined then the current authorization identifier must possess the SELECT privilege for the corresponding table and all other tables referenced by the search condition. The authorization identifier must also have the EXECUTE privilege for all User Defined Functions (UDFs, see "[SQL Reference Manual Part 1: SQL Statements](#)") which are addressed in the search condition.

```
EXPORT TABLE table INTO FILE file [PASSWORD password] [ { ALL DATA  
WHERE search_condition } ]  
NO DATA
```

table

Name of the table that is to be exported. The table must exist and must be a base table. The table can also be partitioned.

The affected spaces in which the table is located must not be in the "recover pending" state. The table itself must not have the state "check pending" if user data is to be exported.

INTO FILE *file*

Name of the export file. You must specify *file* as an alphanumeric literal.

If the file does not yet exist it is created on the DBH user ID as a SAM file with variable record length and a maximum block length of 32768 bytes.

The CCSN of the database (see CODE-TABLE clause on [page 108](#)) is entered as the CCSN of the export file. If no CCSN is defined for the database, the export file is created without a CCSN.

The file may be present before EXPORT TABLE is run. File attributes are corrected, if need be, and existing data is overwritten by EXPORT TABLE.

If the export file is to be located on a user ID other than the DBH user ID, the database administrator must make preparations for the export file, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

The export file may not be opened by another user.

When the WHERE clause is specified, the export file may not be located on tape.



When a table which contains columns with the data types NATIONAL CHARACTER (VARYING) is exported, the export file can no longer be imported using SESAM/SQL V4.0 or earlier.

PASSWORD *password*

A new export file must be created with password protection.

An existing export file must be opened with the write password.

password can be specified in several different ways:

- 'C'*string*''
string contains four printable characters.
- 'X'*hex_string*''
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

ALL DATA

All the user data is written to the export file.

In the case of a partitioned table all partitions in the table must be available (see [table 3 on page 22](#)).

WHERE *search_condition*

Only the user data that meets the search condition is taken over.

The table *table* may not be a CALL DML only table.

In the case of a partitioned table all the partitions affected in the table must be available (see [table 3 on page 22](#)).

No user variables and no question marks may be used as placeholders in the search condition.

Tables other than the table *table* can also be referenced. However, none of the referenced tables may be a CALL DML only table.

The processing of a search condition for a referenced table other than *table* is performed at the session's default isolation level. However, this does not apply if you have specified a different value using

SET TRANSACTION or the ISOLATION LEVEL pragma.

Other pragmas are also effective with WHERE *search_condition* (see the [“SQL Reference Manual Part 1: SQL Statements”](#)).

NO DATA

No user data is transferred to the export file.

Example

The example below transfers the ORDERS table with all the user data from the ORDERPROC schema in the ORDERCUST database to the export file DAT.EXPORT.ORDERS



```
EXPORT TABLE ordercust.orderproc.orders  
INTO FILE 'DAT.EXPORT.ORDERS' ALL DATA
```

See also

IMPORT TABLE

IMPORT TABLE - Import a table from an export file into a database

You use IMPORT TABLE to create a non-partitioned or partitioned base table whose structure is imported from an export file. The coded character set name (CCSN) of the export file is used for checking against the database's coded character set.

The table is created together with all its columns and the primary key (if present). If you wish, you can also take over user data, integrity constraints or indexes from the export file.

For further information on the export file and the possible uses of the EXPORT/ IMPORT TABLE statement pair, refer to [section "EXPORT TABLE - Export a table from a database to an export file" on page 129](#).

A base table is generated in the specified database from the metadata in the export file as follows from:

- The table type is taken over from the exported table. Only the table type SQL or CALL-DML can be taken over for a partitioned table.
- All the columns of the exported table are taken over.
The data type, the defaults and the column sequence are retained.
- If there is a primary key then this is also taken over.
A primary key must exist for partitioned tables.
The primary key is also imported if NO CONSTRAINT is specified.
The name of the primary key remains unchanged. If the current schema already contains an integrity constraint with the same name then a new name is generated. In this case, a warning is issued.

The password catalog of a password-protected CALL-DML table is not imported. The password protection must be redefined for a non-partitioned table using the SESAM/SQL SEPA utility. No password protection can be defined for a partitioned table using SEPA.

The number of rows that were copied from the export file to the table *table* is output in the SQLrowcount field in the ESQL-COBOL program's communication area.

The IMPORT TABLE statement is executed in the following steps:

- A new table is generated on the relevant user space or spaces. Here all the spaces on which the table is to be generated must be physically available (see [table 3 on page 22](#)).
- The user data is entered (optional)
- The indexes and integrity constraints are generated (optional)

The affected spaces are locked in different ways during the corresponding steps

- First of all the space or spaces on which the table is to be created is/are exclusively locked.

- Read accesses to the space on the part of foreign transactions are possible again once the empty table has been generated. However, the newly created table cannot be accessed.
- The space on which the indexes are to be created is then exclusively locked. This is the case if WITH INDEX or WITH CONSTRAINT is specified.

If logging is active for the space in which the table or a partition of this table is to be created, the space will have the state “copy pending” following IMPORT TABLE. This is only the case if records are transferred to the table.

The current authorization identifier must possess the special privilege UTILITY or must be the owner of the schema to which the table belongs.

If the USING clause is not specified and there is therefore as yet no default space for the owner of the schema, then the current authorization identifier must possess the right to use the default storage group D0STOGROUP.

If an error occurs during the execution of IMPORT TABLE then it is necessary to distinguish between two cases:

- Logical data saving is active for the space and the space has the state “copy pending”: SESAM/SQL has generated the table and entered the user data.
You can now:
 - back up the space with COPY and then generate the indexes and integrity constraints at a later date,
 - back up the space with COPY, delete the table with DROP TABLE and repeat IMPORT TABLE,
 - reset the space to its previous state with RECOVER TO, delete the table with DROP TABLE and repeat IMPORT TABLE.
- The space does not have the state “copy pending”:
You should repair the space with RECOVER SPACE delete the table with DROP TABLE and repeat IMPORT TABLE.

```

IMPORT TABLE table [USING {SPACE space
                             PARTITION BY RANGE partition, ..., last_partition } ]

FROM FILE file [PASSWORD password] { [ALL DATA] [ {NEW ROW_IDS } ]
                                         NO DATA }

[ {WITH INDEX [USING SPACE space] } ] [ {WITH CONSTRAINT } ]
[ NO INDEX ] [ NO CONSTRAINT ] ]

partition ::= PARTITION partno VALUE { <
                                         <= } (column_value, ...) ON SPACE space

last_partition ::= PARTITION partno [VALUE <=( )] ON SPACE space

partno ::= unsigned_integer

column_value ::= { alphanumeric_literal
                    numeric_literal
                    time_literal }

```

table

The table is given the name specified for *table*.

The name does not have to be the same as the name of the exported table.

The table *table* must not yet exist. However, the schema into which the table is to be imported must be present.

USING SPACE *space*

A non-partitioned table is created.

The space *space* must exist and the owner of the space must also be the owner of the schema.

The space may not be in the “recover pending” state.

USING PARTITION BY RANGE *partition*, ..., *last_partition*

A partitioned table is created.

Specifies that a partitioned table with at least 2 and at most 16 partitions is to be created. All partitions in a table must be located on different spaces, and all spaces must already be defined in the database. The table must have a primary key. This can be a single column or a combination of columns.

When ALL DATA is selected for partitioned tables, the NEW ROW_IDS clause must also be used.

$$partition ::= PARTITION partno VALUE \left\{ \begin{array}{l} < \\ <= \end{array} \right\} (column_value [, \dots]) \\ ON SPACE space$$

Defines a partition's properties.

partno is a positive integer from 1 to 16 and is the partition's current number. *partno* must be assigned in ascending order for the individual partitions. If less than 16 partitions are defined, the series of numbers can contain gaps, and the first partition need not begin with 1.

(*column_value,...*) is a sequence of column values which defines the upper limit of the primary key interval for the partition concerned. *column_value* must have a data format that is comparable to the data format of the relevant column of the primary key. The rules for the "default values for table columns" also apply for *column_value*, see the "[SQL Reference Manual Part 1: SQL Statements](#)".

The lower limit of the partition is obtained implicitly from the upper limit of the previous partition or from the lowest primary key in the table (in the first partition). All rows read in from the primary key interval thus defined are assigned to this partition.

The upper limit is either included or excluded by the preceding comparison operator:

- <= Rows whose primary key value is *column_value,...* belong to **this** partition
- < Rows whose primary key value is *column_value,...* belong to the **next** partition

The lexicographical rules apply for the comparison, see the "[SQL Reference Manual Part 1: SQL Statements](#)" manual.

If the primary key consists of a combination of multiple columns, you may only specify at most as many values as there are columns in the primary key. If you specify fewer elements than there are columns defined in the primary key, SESAM/SQL adds the missing values by setting the highest value which matches the data type of the relevant column. The upper limits specified must be strictly in ascending order for the individual partitions.

space specifies the name of the space in which this partition is stored. The space must exist and the space owner must also be the schema owner. The spaces of the partitions of a partitioned table must be disjunctive, i.e. a space may not be used for two partitions. The space may not be in the "recover pending" state.

last_partition := PARTITION *partno* [VALUE <=()] ON SPACE *space*

The same conditions apply for the last partition as for *partition*.

Only the upper limit may not be specified since it is determined here from the highest primary key value. The VALUE clause can therefore also be omitted.

USING not specified

A non-partitioned table is created on the schema owner's default space. If no default space exists yet, it is created implicitly.

FROM FILE *file*

file must be the name of the export file that is to be imported. You must specify *file* as an alphanumeric literal.

If the file and the DBH are located on a different BS2000 user ID then the DBH user ID must have the read authorization.

The export file's CCSN must match the database's CCSN (see CODE-TABLE clause on [page 108](#)). If no CCSN is used for the database, the export file's CCSN is ignored.



An export file which contains a table with the data types NATIONAL CHARACTER (VARYING) can no longer be imported using SESAM/SQL V4.0 or earlier.

PASSWORD *password*

If the export file is password-protected then the read password must be specified.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

[ALL DATA]

All the user data entered in the export file is loaded into the table.

A warning is issued if no user data is present.

NEW ROW_IDS

If the table has a primary key then new row numbers are assigned in ascending linear order for the imported rows.

If the table does not have a primary key then the row numbers from the exported table are retained. A warning is issued.

OLD ROW_IDS

The row numbers of the exported table are taken over.

If a partitioned table is to be created, OLD ROW_IDS may not be specified.

If the table from which the export file was generated was partitioned, the OLD ROW_IDS specification is ignored. A WARNING is issued and NEW ROW_IDS is used.

NO DATA

An empty base table is generated.

WITH INDEX [USING SPACE *space*]

All the indexes are transferred. A warning is issued if the export file does not contain any index-related metadata.

The name of the index is taken over from the exported table. If an index of the same name is already present in the current schema then a new name is generated and a warning is issued.

The statistical information for the index is updated.

The following applies to indexes that are associated with UNIQUE constraints:

- WITH INDEX and WITH CONSTRAINT are specified and the index is explicitly generated in the exported table:
In this case, the index is created in the same space as the other indexes and continues to be considered to be explicitly generated.
- In all other cases:
The index is created in the same space as the table and is considered to be implicitly generated.

If you do not specify an index space, in the case of a non-partitioned table the index is created on the same space as the table. In the case of a partitioned table it is created on the first partition's space.

USING SPACE *space*

If you specify an index space, the index is created in this space.

The owner of the space must also be the owner of the schema.

The index space must not have the state "recover pending".

NO INDEX

Indexes which do not reference any UNIQUE constraint are not transferred.

WITH CONSTRAINT

All UNIQUE and check constraints are transferred. A warning is issued if the export file does not contain any metadata for these constraints.

The name of the UNIQUE constraint is taken over from the exported file.

If a UNIQUE constraint of the same name is already present in the current schema then a new name is generated and a warning is issued.

If the text contains a column name that is qualified by a check constraint, then the schema and table names that are specified there must be identical to the corresponding names in the current table. Otherwise the check constraint is not transferred and a warning is issued.

NO CONSTRAINT

The UNIQUE constraint and check constraint are not transferred.

Examples

In the first example, the ORDERS table from the export database DAT.EXPORT.ORDERS is imported into the EXPORT_SCHEMA of the database ORDERCUST under the name ORDERS_NEW and is saved in the space TABLESPACE. All the user data present in the export file is loaded. The row numbers of the exported table are taken over. The indexes for the ORDERS table are transferred and saved in the space INDEXSPACE. The UNIQUE and check constraints are not transferred.



```
IMPORT TABLE ordercust.export_schema.orders_new
  USING SPACE tablespace
  FROM FILE 'DAT.EXPORT.ORDERS' ALL DATA OLD ROW_IDS
  WITH INDEX USING SPACE indexspace
  NO CONSTRAINT
```

In the second example the ORDERS table is imported from the export file DAT.EXIMPT.ORDERS. Here ORDERS is stored as a partitioned table on the spaces TABLESPACE, TABLESP002 and TABLESP003. All user data contained in the export file is loaded. The row numbers of the exported table are reassigned. The indexes of the ORDERS table are taken over and stored on the space INDEXSPACE. No constraints are transferred to the new table.



```
IMPORT TABLE orders
  USING PARTITION BY RANGE
    PARTITION 01 VALUE <= (299) ON SPACE tablespace,
    PARTITION 02 VALUE <= (399) ON SPACE tablesp002,
    PARTITION 03           ON SPACE tablesp003
  FROM FILE 'DAT.EXIMPT.ORDERS' ALL DATA NEW ROW_IDS
  WITH INDEX USING SPACE indexspace
  NO CONSTRAINT
```

See also

EXPORT TABLE

LOAD - Load user data into a base table

You use LOAD to load user data from a BS2000 SAM file into a base table of a SESAM/SQL database. You can specify the columns of the table into which the data is to be loaded. The number of inserted rows is output in the `SQLrowcount` field of the communication area of the ESQL COBOL program. You can also exclude the first records (e.g. comments or records that have already been read in) from being read in.

SESAM/SQL offers several load formats, namely `TRANSFER_FORMAT`, `UNLOAD_FORMAT`, `DELIMITER_FORMAT`, `CSV_FORMAT` and the standard representation; it is also possible to define the representation of the individual columns in the input file yourself (see also [“LOAD formats” on page 161](#)). If the load format is not specified and if the representation of the individual columns is not described, all the input file values must be in the standard representation, which corresponds to the data type of the assigned column in the base table.

The coded character set name (CCSN) of the input file is used for checking against the database's coded character set.

In `DELIMITER_FORMAT`, in `CSV_FORMAT` and in the self-defined format, strings that are to be loaded are converted to the database's coded character set if necessary.

When used in conjunction with the utility statement UNLOAD, LOAD also makes it easy to transfer user data from one base table to another. In this case, the input file for LOAD is an output file created earlier with UNLOAD. To do this, you can use the four named formats (specifying `UNLOAD_FORMAT` for LOAD corresponds to specifying `LOAD_FORMAT` for UNLOAD). It is important to pay close attention to the processing of NULL values in the case of the standard representation and formats you have defined yourself (see the `WHEN ... THEN NULL` clause).

If user data is to be copied along with the table structure of a base table then the use of the pair of statements `EXPORT/IMPORT TABLE` may be worthwhile. With these statements it is also possible to copy only a base table's metadata. For more detailed information on the possible uses of `EXPORT/IMPORT TABLE` refer to [section “EXPORT TABLE - Export a table from a database to an export file” on page 129](#) and [section “IMPORT TABLE - Import a table from an export file into a database” on page 133](#).

You can choose between the operating modes `OFFLINE` and `ONLINE` when executing the LOAD statement. The major differences apply to the performance, the locking behavior and error processing after a LOAD statement has been aborted.

You can find more detailed information about `OFFLINE/ONLINE` on [page 143](#).

When executing the LOAD statement, SESAM/SQL checks the primary key constraint in each case. If integrity constraints have been defined, you must check that the integrity constraints are observed during the loading procedure or afterwards in each case. Until the integrity constraints have been checked, the base table into which the data has been loaded is in the “check pending” state. It is therefore locked against accesses by other users (see [page 13](#)). Other tables located in the same user space can only be read. In addition, the indexes of the table have to be rebuilt in the case of LOAD OFFLINE. In general, it does not make much sense to check the integrity constraints before the indexes have been rebuilt since a precondition for checking the integrity constraints is that indexes are intact.

In the case of LOAD ONLINE indexes are always automatically adjusted.

The current authorization identifier must have the special privilege UTILITY or must own the schema in which the table into which the data is to be loaded is located.

The various options for representing data in the input record are described in the [section “Data representation in the input and output files for LOAD and UNLOAD” on page 26](#).

```

LOAD [ { OFFLINE } ] [SORTED] FILE file [PASSWORD password] [(load_description, ...)]

[SKIP FIRST number RECORDS]

INTO TABLE table [(load_column, ...)] [load_parameter] ...
{
  TRANSFER_FORMAT
  UNLOAD_FORMAT
  DELIMITER_FORMAT TERMINATED BY delimiter
  CSV_FORMAT DELIMITER delimiter [QUOTE quote] [ESCAPE escape]
}

[USING FILE file [PASSWORD password]]

[COUNTING_FIELD column]

load_description ::= POSITION ( { number } ) [ data_type ] [WHEN null_constraint THEN NULL]

null_constraint ::= {
  { column comparison_op literal
  POSITION ( { number } ) comparison_op { alphanumeric_literal
  national_literal } } }

comparison_op ::= {
  =
  <
  >
  <=
  >=
  <>
}

load_column ::= {
  column
  column(pos_no)
  column(min..max)
}

load_parameter ::= {
  { [NO] OVERWRITE
  { NO INDEX
  GENERATE INDEX } }
  [NO] CONSTRAINT CHECK
}

```

{ OFFLINE }
{ ONLINE }

You specify the LOAD operating mode as OFFLINE or ONLINE. The differences are as follows:

- With LOAD OFFLINE the user space is exclusively locked against modifications during loading. The table itself is exclusively locked against reading; other tables and indexes on the user space can be read by foreign transactions by means of DML. If GENERATE INDEX is specified, the index spaces are exclusively locked when the indexes are being generated (after the primary data has been loaded). The FROM SPACE and FROM COPY_FILE clauses can be specified.

With LOAD ONLINE the user space is not exclusively locked. Only the same locks are requested as for a DML modification statement. In deadlock situations LOAD ONLINE as the utility is preferred as opposed to DML statements so that it can only be aborted (but not reset) when competing with other utilities. If the table goes into “check pending” state, the table space for fixing this state is temporarily exclusively locked. All the processing takes place in a DBH task.

- With LOAD OFFLINE logging (LD backup) for the affected user space is interrupted during execution of the LOAD statement. This means that in the case of a repair the modifications can only be applied up to the beginning of the load operation. In the case of tables located on a user space with logging you must create a SESAM backup copy of the affected user space after the execution of the last LOAD statement which may have been followed by a constraint check and/or an index creation. If you do not do this, the space will remain in the “copy pending” state and you will only be able to process it with utility statements (see [page 18](#)).

In the case of LOAD ONLINE, logging is not interrupted. The loaded rows for a user space run with logging, are logged on the DA-LOG. The affected user space will therefore not be given the state “copy pending”. Replication spaces in particular can be updated following LOAD ONLINE with the statement REFRESH REPLICATION.

Execution of LOAD ONLINE is rejected if the user space or an affected index space is in the “copy pending” state.

- Loading relatively large amounts of data (particularly when loading an empty table) LOAD OFFLINE is faster than LOAD ONLINE. Relatively large means that the number of rows in the input file is high compared to the number of rows already in the base table.

Loading relatively small amounts of data, LOAD ONLINE is faster than LOAD OFFLINE, especially if indexes are to be updated and integrity constraints are to be checked. Relatively small means that the number of rows to be loaded is low compared to the number of rows already in the base table.

These differences in performance are partly due to processing of the indexes (see INDEX clause on [page 152](#)) and the integrity constraints (see CHECK clause on [page 153](#)), see “Performance” manual, chapter “Performance-related aspects of utility functions”.

- If LOAD OFFLINE is aborted with an error after some of the rows have already been loaded, the table space is in the “load running” state and the corresponding indexes are defective. Unaffected objects on the table space can still be read, but not modified. A RECOVER SPACE and, if applicable, a subsequent RECOVER INDEX are required to repair the space.

If LOAD ONLINE is aborted with an error or because of a deadlock after some of the rows have already been loaded, these rows will remain and the table will still be formally correct, although it may be in the “check pending” state if it was not possible to check the integrity constraints.

- In the case of LOAD ONLINE, the input file and error file may not be located on tape.

[SORTED] FILE *file* [PASSWORD *password*]

Name of the input file. You must specify *file* as an alphanumeric literal. In the case of LOAD ONLINE, *file* may not be a tape file.

The input file must be a SAM file with variable-length records and must not be empty. It may not contain several rows with the same primary key value; otherwise which of these rows is loaded is undefined. If the input file is not located in the ID of the DBH, the DBH ID must have read authorization for this file. Otherwise the DBH cannot access the input file.

The input file's coded character set name (CCSN) is checked.

For all LOAD formats it may be an EBCDIC character set, a self-defined character set, or *NONE. For DELIMITER_FORMAT the input file's character set may also be UTFE (see [page 155](#)).

The record length for SAM files is limited. A record in the SAM file is assigned to precisely one row of the base table (exception: CSV format). It may therefore be necessary to distribute the data to be loaded over several input files according to columns, especially when loading into columns of the data type (NATIONAL) CHARACTER VARYING. In this case, loading the data involves executing the utility statement LOAD several times.

The format of a CSV file is described in detail in the “[SQL Reference Manual Part 1: SQL Statements](#)”.

SORTED specified:

SESAM/SQL assumes that the input file is sorted according to primary key values. If the input file is not sorted, an entry is made to the error file for each incorrectly sorted row.

Specifying **SORTED** has no effect if data is being loaded into a table defined without a primary key or if **ONLINE** is specified.

SORTED omitted:

It is of no significance to the execution of the **LOAD OFFLINE** statement whether the input file has been sorted according to primary key values, since SESAM/SQL sorts the file.

In the case of **LOAD ONLINE** a sorting in the input file according to primary key values is not explicitly used. In the case of large load files in particular, sorting can improve the buffer behavior in SESAM/SQL and thus substantially improve the performance.

For tables defined without a primary key, the order in which the rows are inserted into the table is not defined.

PASSWORD *password*

BS2000 password, if the input file is password protected. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

(*load_description* ,...)

List of load descriptions



If neither (*load_description*, ...) nor **TRANSFER_FORMAT**, **UNLOAD_FORMAT** nor **DELIMITER_FORMAT** is specified, all the input file values must be in the standard representation which corresponds to the data type of the assigned column of the base table *table*.

The load descriptions describe the formats of the input values in the input file and the position of the input values in the input record. The following rules apply:

- You must specify either a load description for each column in the input file created from input values of the same type or no load description at all.

- If you specify a list of load descriptions but not a list of load columns (*load_column*, ...), SESAM/SQL assigns the specified load descriptions or positions to the columns in the table *table* one after the other.
- If you specify both a list of load descriptions and a list of load columns, SESAM/SQL assigns a load description to each element in the list of load columns. The list of load columns and the list of load descriptions must therefore contain the same number of elements.
- In the case of a multiple column, you only need to specify the load description for a single column element. This also holds true if you reference the column as a whole via the column name or if a subarea is used to reference several column elements simultaneously. The same applies to the WHEN...THEN NULL clause.
- You cannot specify a list of load descriptions if you specify one of the following formats TRANSFER_FORMAT, UNLOAD_FORMAT, CSV_FORMAT or DELIMITER_FORMAT.

The way in which data is represented for LOAD is described in the [section “Data representation in the input and output files for LOAD and UNLOAD” on page 26](#).

load_description ::=

POSITION (*number*)

Positive integer indicating the position of the input value in the input record. Position specifications that result in overlapping input fields are permitted. SESAM/SQL does not, however, check for overlapping input fields.

Because it is possible to load variable-length records, you must make sure that the position specified in *load_description* is not outside the record.

If you specify a list of load descriptions but not a list of load columns (*load_column*, ...), SESAM/SQL assigns the specified load descriptions or positions to the columns in the table *table* one after the other.

POSITION (*)

You must specify POSITION (*) if it is not possible or desirable to specify an exact position. Each starting position is calculated from the preceding values in the input file.

data_type

Data type of the input value. The data types correspond to the data types at the SQL interface (see the [“SQL Reference Manual Part 1: SQL Statements”](#)).

You cannot specify FLOAT as the *data_type*. You must use REAL or DOUBLE PRECISION instead of FLOAT.

You cannot specify *dimension* in *data_type*.

If you omit *length* or *precision* [,*scale*], SESAM/SQL uses the value 1 for *length* and *precision* and 0 for *scale*. SESAM/SQL ignores any *length* specification made for the data type

(NATIONAL) CHARACTER VARYING.

data_type must be compatible with the data type of the corresponding load column.

If you want to specify the input values for a certain table column in readable form, you must use the data type CHARACTER with a sufficient length specification (see [section “Readable representation of the data in the input and output files” on page 27](#)).

If the input values and the associated column in the table have different data types (CHARACTER (VARYING) or NATIONAL CHARACTER (VARYING)), the following constraints apply on account of data conversion:

- In the case of conversion from CHARACTER (VARYING) to NATIONAL CHARACTER (VARYING), the input value is assigned an EBCDIC code in accordance with the input file’s CCSN. The input file may **not** be assigned CCSN=*NONE.
- In the case of conversion from NATIONAL CHARACTER (VARYING) to CHARACTER (VARYING), the column value is assigned an EBCDIC code in accordance with the database’s CCSN. The database may **not** be assigned CODE_TABLE _NONE_, see the CODE_TABLE clause on [page 108](#).
- If the input value and the associated column in the table have the data type CHARACTER (VARYING) and a coded character set is used for the database, the input file’s CCSN and the database’s CCSN must be identical.
- In the case of conversion from (NATIONAL) CHARACTER (VARYING) to a numeric data type or to a time data type, the input file can be assigned CCSN=*NONE. The database may also be assigned the CODE_TABLE _NONE_.

data_type omitted:

The data must be represented in accordance with the data type of the associated *load_column*, see [table 4 on page 34](#). If no data type or the data type (NATIONAL) CHARACTER VARYING is specified for a *load_column* of the data type (NATIONAL) CHARACTER VARYING in the assigned load presentation, SESAM/SQL expects a 2-byte length field immediately ahead of the value to be read in.



In the case of NATIONAL CHARACTER VARYING the length field contains the length in bytes, not in characters.

WHEN *null_constraint* THEN NULL

Specification of a NULL constraint.

If the NULL constraint is satisfied, the NULL value is placed in the table column involved instead of a column value. You cannot specify a WHEN...THEN NULL clause for columns defined as NOT NULL.

null_constraint ::=

column

Name of the column whose value is to be compared with the literal *literal*.

column must be a *load_column* (see [page 150](#)). You cannot specify a multiple column as *column*.

comparison_op ::= $\left. \begin{array}{l} = < \\ > < = \\ > = \\ < > \end{array} \right\}$

The normal rules for comparing two values apply (see the “[SQL Reference Manual Part 1: SQL Statements](#)”).

literal

Literal.

The data type of *literal* must be comparable with the data type of *load_description*.

POSITION (*number*)

Positive integer indicating the position of the first comparison operand in the input file.

POSITION (*)

The value in the input file defined by the relevant load description serves as the first operand for the NULL constraint.

If load description references several occurrences of a multiple column, you must specify POSITION(*) as the first operand of the NULL constraint.

SESAM/SQL then compares the individual input values for the multiple column with the subsequent literal and, where necessary, assigns NULL for each relevant occurrence.

$\left. \begin{array}{l} \textit{alphanumeric_literal} \\ \textit{national_literal} \end{array} \right\}$

Alphanumeric literal (e.g. '0' or X'F0') or national literal (e.g. N'0' or NX'0030' or U&'0' or U&'0030').

The length for comparison is determined by the OCTET_LENGTH of the literal (see the “[SQL Reference Manual Part 1: SQL Statements](#)”).

SKIP FIRST *number* RECORDS

number is a positive integer.

The first *number* records of the input file are not read.

These can, for example, be comments, header line (see UNLOAD, [page 248](#)) or records that have already been read in (aborted LOAD, see [page 158](#)).

When *number*=0 (default value) all records of the input file are read. *number* must be less than or equal to the number of records in the input file.

When the TRANSFER_FORMAT clause is specified, all descriptor records (at the start of the file) are read and interpreted. These may be followed by the records to be skipped, then come the data records. In other words *number* must be 0 or at least equal to the number of descriptor records.

Skipped records are also counted. The number of a record in the error file therefore always relates to the complete input file.

Information on the use of this clause in the event of an error

If a LOAD ONLINE is aborted with an error (not the same as a consistency check) you will obtain a message in the LOAD error file (see [page 158](#)) about the number of records processed when the loading process was aborted:

n INPUT RECORDS PROCESSED.

This message is used for error tracing and also to complete the aborted loading process without modifying the input file with SKIP FIRST *n* RECORDS. Here *n* always relates to the complete input file, i.e. it also contains the skipped records.

Even when a LOAD OFFLINE is terminated with an error you are shown the message *n* INPUT RECORDS PROCESSED in the LOAD error file.

- No record was loaded for *n*=0; the LOAD must be repeated in full.
If the table is in the “load running” state, the space must be recovered beforehand with RECOVER SPACE.
- *n*>0 means all primary data was loaded in full. You may need to recover the indexes (RECOVER INDEX ON TABLE) or check the integrity constraints (CHECK CONSTRAINTS ON TABLE).

INTO TABLE *table*

Base table into which the data is to be loaded.

In the case of LOAD ONLINE, the specified table may not be an CALL-DML only table.

In the case of a partitioned table all the partitions must be available, see [table 3 on page 22](#)

(*load_column*,...)

List of the columns into which data is to be loaded.

The list of load columns indicates the columns in *table* or the occurrences of a multiple column in *table* into which data is to be loaded.

The following rules apply:

- You cannot specify the same *load_column* more than once in a list of load columns.
- You cannot specify a list of load columns if you specify the TRANSFER_FORMAT clause.
- If the input file is not in transfer format (TRANSFER_FORMAT clause not specified), the following applies:
If you specify neither a list of load columns nor a list of load descriptions, the input file must include values for all the columns in *table*. LOAD expects the input values in the order and with the formats laid down in the column definitions for *table*.
- If you specify a list of load columns but not a list of load descriptions, the input values must have the format defined for the *load_column*, see [table 4 on page 34](#) (exception: DELIMITER_FORMAT).
- If no *load_column* is specified and *table* contains a multiple column, LOAD expects the maximum number of occurrences.
- If you specify both a list of load descriptions and a list of load columns, SESAM/SQL assigns a load description to each element in the list of load columns. The list of load columns and the list of load descriptions must therefore contain the same number of elements.
- If a multiple column is referenced using only the column names and not via the individual occurrence or a range of occurrences, LOAD expects the maximum number of occurrences.
It is recommended that you reference a multiple column using only the column names when the number of occurrences to which values are assigned in the individual input records are different. Occurrences that are not assigned a value must be marked in the input file like NULL values with WHEN *null_constraint* THEN NULL.
- If *table* is defined with a single or compound primary key, you can either not specify any *load_column* or the list of load columns must include the primary key.

load_column ::=

column

Name of the column into which data is to be loaded.

If *column* is a multiple column, the following applies:

col_{num} is the number of existing occurrences for *column*.

col_{max} is the maximum number of occurrences possible for *column*.

column (*pos_no*)

pos_no is a positive integer.

If *pos_no* is greater than $col_{num} + 1$, the $(col_{num} + 1)$ th element of the multiple column *column* is loaded.

Otherwise, the *pos_no*th column element is overwritten if OVERWRITE is specified.

If NO OVERWRITE is valid, an entry is made in the error file (see the USING FILE clause on [page 158](#)).

If *column* is not a multiple column or *pos_no* is greater than col_{max} , an error message is issued.

column (*min* .. *max*)

The value is the aggregate from the column elements *min* to *max* of the multiple column *column*.

min not greater than $col_{num} + 1$: The data is loaded into the column elements *min* to *max*.

min greater than $col_{num} + 1$: The data is loaded into the column elements $col_{num} + 1$ to $col_{num} + 1 + max - min$.

If *column* is not a multiple column, *min* is greater than *max* or *max* is greater than col_{max} , an error message is issued.

pos_no or *min* .. *max* not specified:

If *column* is a multiple column, specifying *column* is the same as specifying *column*(1 .. col_{max}).

column, *column*(*pos_no*) or *column* (*min* .. *max*) for *load_column* more than once if the ranges defined overlap. Additionally, if several subareas of a multiple column are specified in the case of LOAD ONLINE, the resulting occurrence range may not contain any gaps either.

load_parameter ::=

You may only specify each of the following load parameters once:

[NO] OVERWRITE

OVERWRITE indicates that the column values in the rows of the table *table* are to be overwritten by the values from input records with the same primary key value. OVERWRITE causes NULL values in the input record to delete the corresponding column values.

You cannot specify OVERWRITE if the table into which data is to be loaded has been defined without a primary key.

If you use a counting field (see the COUNTING_FIELD clause on [page 160](#)), OVERWRITE is ignored.

If you specify NO OVERWRITE, SESAM/SQL proceeds as follows:

Wherever column values are missing in *table*, SESAM/SQL enters the corresponding input values. If column values already exist at certain positions, SESAM/SQL rejects the corresponding input values and writes the rejected values to the error file (see USING FILE clause on [page 158](#)).

The number of loaded rows in the table *table* is output in the SQLrowcount field of the communication area of the ESQL-COBOL program.

Only the inserted rows are counted, modified rows will not be taken into account.



[NO] OVERWRITE does not specify whether the LOAD may modify rows. Rows with an identical primary key value are always modified. The clause only specifies whether significant column values may also be overwritten.

{ NO INDEX
GENERATE INDEX }

In the case of LOAD OFFLINE, the INDEX clause specifies how the indexes defined in the table *table* should be processed. Specifying this clause has no effect in the case of tables without an index.

- If NO INDEX is specified, the affected indexes are not regenerated and are therefore invalid after LOAD.
It does not usually make sense to specify NO INDEX in conjunction with CONSTRAINT CHECK since intact indexes are a precondition for checking the integrity constraints.
- If GENERATE INDEX is specified, all indexes will be regenerated after loading. In the case of partitioned tables all partitions must be available for this purpose, see [table 3 on page 22](#). Partitions which are not available are not logged in the error file.

In the case of LOAD ONLINE, the INDEX clause must not be specified. The table indexes affected by the loading are always adjusted. Invalid indexes are ignored and will remain invalid after LOAD.

[NO] CONSTRAINT CHECK

The CHECK clause specifies how the integrity constraints that refer to the table *table* are to be processed.

If CONSTRAINT CHECK is specified, all the integrity constraints will be checked. In the case of a partitioned table all partitions must be available for this purpose, see [table 3 on page 22](#). If a partition is not available, it remains in the “check pending” state. Partitions which are not available are not logged in the error file.

If the table was in the “check pending” state prior to LOAD (see [page 13](#)) and if no integrity constraint violations are detected, this state will be cancelled again.

If NO CONSTRAINT CHECK is specified or if errors are detected in the integrity constraint check, the table will be or remain locked if the integrity constraints are defined. It will then be in the “check pending” state. A further load or reparatory measures are, however, possible.

In the case of OFFLINE, the integrity constraints are checked once all rows have been inserted into the table. With ONLINE, in many, but not all, cases the check is carried out as each row is inserted into the table. In these cases the procedure in the case of ONLINE is considerably quicker than the procedure in the case of OFFLINE.

Specifying a CHECK clause does not have any effect in tables without integrity constraints. The default setting for LOAD OFFLINE is NO CONSTRAINT CHECK; the default for LOAD ONLINE is CONSTRAINT CHECK.

TRANSFER_FORMAT

Data is loaded from the transfer file created with the UNLOAD statement.



If the transfer file contains data of the data type NATIONAL CHARACTER (VARYING), it can no longer be loaded using SESAM/SQL V4.0 or earlier.

If the file was previously created from a base table with a primary key using UNLOAD ... TRANSFER_FORMAT, the rows will be sorted according to the values of this primary key. If the primary key of *table* matches the primary key of the table from which the transfer file was created, you can speed up the load operation by specifying LOAD OFFLINE SORTED.

You may not specify any *load_description* or *loadcolumn* in the TRANSFER_FORMAT ... specification. Columns into which data are to be loaded may *not* have the attribute format OLDEST for the TRANSFER_FORMAT specification. This attribute format may only occur in CALL-DML only tables.

UNLOAD_FORMAT

UNLOAD_FORMAT requires that each value in the input file is preceded by an indicator of the type SMALLINT (this is referred to as the UNLOAD_FORMAT). This indicator shows whether the input value is to be interpreted as a NULL value. LOAD interprets an input value as a NULL value if the indicator has a negative value.

You create an input file in UNLOAD_FORMAT with the LOAD_FORMAT clause of the utility statement UNLOAD. The records of a file in UNLOAD_FORMAT are sorted according to the primary key values of the base table from which data was unloaded with UNLOAD ... LOAD_FORMAT. If the primary key of *table* matches the primary key of the table from which the input file in UNLOAD_FORMAT was created, you can speed up the load operation by specifying LOAD OFFLINE SORTED.

The table into which the data is to be loaded must have the same columns with identical formats as the table from which data was written earlier with UNLOAD ... LOAD_FORMAT into the input file for LOAD. No *load_description* may be specified for the UNLOAD_FORMAT ... specification.

DELIMITER_FORMAT TERMINATED BY *delimiter*

SESAM/SQL expects the input file to be in delimiter format. In delimiter format, all values are represented in the input file's coded character set and terminated with a DELIMITER character. For further details on representing values in delimiter format, refer to [section "Readable representation of the data in the input and output files" on page 27](#).

You must not specify a *load_description* with DELIMITER_FORMAT ... If you specify DELIMITER_FORMAT ..., none of the columns into which data is to be loaded may have the "attribute format OLDEST", which may only occur in CALL-DML only tables.

The input file's CCSN is checked:

- If the input file is assigned an EBCDIC character set or a self-defined character set on EBCDIC basis or no character set (CCSN=*NONE), all input values are interpreted as values of the data type CHARACTER and converted to the format of the table column if necessary.
- If the input file is assigned CCSN=UTFE, all input values are interpreted in accordance with UTFE and converted to the format of the table column.

Treatment of NULL values:

- NULL values are not represented in the input file. Only the (non-escaped) DELIMITER character terminating the NULL value is represented. Two consecutive DELIMITER characters within a record or at the end of a record are therefore interpreted as follows: the first DELIMITER character terminates the preceding value and the second DELIMITER character terminates a NULL value which is not represented.
- If the last value in a record is not NULL, the final DELIMITER character can be omitted.

delimiter

Defines a DELIMITER character which terminates each value in the input file. The DELIMITER character must not be a value which is a component of any of the input values.

delimiter can be specified as follows:

- The input file is assigned an EBCDIC character set or a self-defined character set on EBCDIC basis or no character set (CCSN=*NONE):

alphanumeric literal with the length 1.

Examples: TERMINATED BY ';' or TERMINATED BY X'5E'.

For reasons of compatibility, hexadecimal representation can also be specified: 'X"*xx*"' (*x*: a digit in hexadecimal representation).

- The input file has CCSN=UTFE:

National literal of the length 1 which also has a hexadecimal representation in one byte in UTFE (range of values: NX'0000' ... NX'009F').

Examples: TERMINATED BY N';' or TERMINATED BY NX'003B'

or also: TERMINATED BY U&';' or TERMINATED BY U&'003B'

If a single quote (') is defined as a DELIMITER character, you must specify four single quotes.

Examples: TERMINATED BY "" or ... N"" or ...U&"".

CSV_FORMAT DELIMITER *delimiter* [QUOTE *quote*] [ESCAPE *escape*]

SESAM/SQL expects the input file to be in CSV format. In CSV format, all values are represented in the input file's coded character set and concluded with a non-escaped DELIMITER character or an end-of-line character. For further details on representing values in CSV format, refer to [section "Readable representation of the data in the input and output files" on page 27](#).

You must not specify a *load_description* with CSV_FORMAT.

If you specify CSV_FORMAT, none of the columns into which data is to be loaded may have the "attribute format OLDEST", which may only occur in CALL-DML only tables.

The input file's CCSN is checked:

- If the input file is assigned an EBCDIC character set or a self-defined character set on EBCDIC basis or no character set (CCSN=*NONE), all input values are interpreted as values of the data type CHARACTER and converted to the format of the table column if necessary.
- If the input file is assigned CCSN=UTFE, all input values are interpreted in accordance with UTFE and converted to the format of the table column.

Treatment of NULL values:

- NULL values are not represented in the input file. Only the (non-escaped) DELIMITER character terminating the NULL value is represented. Two consecutive DELIMITER characters within a record or at the end of a record are therefore interpreted as follows: the first DELIMITER character terminates the preceding value and the second DELIMITER character terminates a NULL value which is not represented.
- No DELIMITER character need be specified after the last significant value of a line in the input file. Values which are not specified are regarded as NULL values.



The characters specified for DELIMITER, QUOTE and ESCAPE must all be different.

delimiter

Defines the separator (DELIMITER character) which terminates each value in the input file. A DELIMITER character can also be part of a value, see the descriptions of *quote* and *escape* below.

delimiter can be specified as follows:

- The input file is assigned an EBCDIC character set or a self-defined character set on EBCDIC basis or no character set (CCSN=*NONE):

alphanumeric literal with the length 1.

Examples: DELIMITER ';' or DELIMITER X'5E'.

For reasons of compatibility, hexadecimal representation can also be specified: 'X"xx"' (x: a digit in hexadecimal representation).

- The input file has CCSN=UTFE:

National literal of the length 1 which also has a hexadecimal representation in one byte in UTFE (range of values: NX'0000' . . . NX'009F').

Examples: DELIMITER N';' or DELIMITER NX'003B'

or also: DELIMITER U&';' or DELIMITER U&'003B'

If a single quote (') is defined as a DELIMITER character, you must specify four single quotes.

Examples: DELIMITER ""' or ... N""' or ...U&""'.

quote

Defines the QUOTE character which can enclose the values of the input file. QUOTE characters do not belong to the value itself. A QUOTE character within a value must be duplicated. When a value is enclosed in QUOTE characters, it can also contain NEWLINE characters (which are not interpreted as a line break) or DELIMITER characters. A value consisting only of an opening and a closing QUOTE character is interpreted as a value with the length 0.

The information provided for *delimiter* above also applies for *quote*.

escape

Defines the escape character (ESCAPE character) with which ESCAPE sequences consisting of two characters begin in the input file.

ESCAPE sequences enable DELIMITER characters, QUOTE characters and ESCAPE characters to be written as part of a column value and NEWLINE characters to be ignored as a delimiter between two input lines.

The information provided for *delimiter* above also applies for *escape*.

USING FILE *file* [PASSWORD *password*]

Name of the error file. You must specify *file* as an alphanumeric literal.

In the case of LOAD ONLINE, *file* may not be a tape file.

The following error information for the loading process is logged in the error file:

- information on errored records in the input file. This information comprises the record number of the errored record in the input file, possibly information on the column in which loading was errored, and a record-specific SQLSTATE.
- unavailable partitions of a partitioned table, the record numbers in the input file affected by this, and the relevant SQLSTATE.
- a message about the number of records processed when an ongoing loading process was aborted due to an error (not the same as a consistency check):

n INPUT RECORDS PROCESSED

n Number of the last record read in the input file which, provided it was not errored, resulted in a change to the table.

Skipped records are also counted. A record number in the error file therefore always relates to the complete input file.

If the input file is in TRANSFER_FORMAT (transfer file), SESAM/SQL includes header and description records when determining the record number. In the case of transfer files, SESAM/SQL also returns the record number of the SAM file as a record number.

This allows you, when using the file editor EDT, to go directly to the record in the error file that contains information on the errored record in the input file (see the “[EDT \(BS2000\)](#)” manual).

SESAM/SQL only creates or uses the error file if errors occur during the loading process. LOAD appends new information to an existing error file.

The entries from various LOAD statements are separated by two header lines which log statement, date, time and name of the input file.

If you do not want the error file to be located on the DBH user ID, the necessary preparations must have been made, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”.

PASSWORD *password*

BS2000 password for the error file. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

USING FILE *file* omitted:

If required, LOAD creates the error file on the BS2000 user ID of the DBH and assigns it the following default file name:

catalog.space.EXC.L

catalog is the name of the database.

space Name of the user space which contains the destination table for the load operation.

In the case of a partitioned table the space name of the first partition is used. The error file contains the error information for all partitions in the table.

EXC.L Default name ending of the error file for LOAD statements.

If *catalog* is located on a DB user ID, the error file will be created on the DB user ID, provided the necessary preparations have been made, see section "Database files and job variables on foreign user IDs" in the "Core manual". If the error file cannot be created on the DB user ID, it will be created on the DBH user ID.

COUNTING_FIELD *column*

If a single or compound primary key is defined for the table into which data is to be loaded, you can define a key column as a so-called counting field for the load operation. The column for the counting field must be of the type NUMERIC, DECIMAL, SMALLINT or INTEGER. You cannot assign the counting field any values in the input file. The value for the counting field is assigned by SESAM/SQL. If you specify the COUNTING_FIELD clause, SESAM/SQL ignores the specification OVERWRITE.

Counting starts by incrementing the last digit or last decimal digit of the highest value assigned to the counting field so far by 1.

Using a counting field allows SESAM/SQL to ensure the uniqueness of the primary key values.

The following rules apply to the use of a counting field:

- You cannot declare more than one key column as the counting field.
- If all the key columns are assigned values in the input file, you cannot declare a counting field.
- If no key columns are assigned values in the input file, you must declare a counting field.
- If several but not all of the key columns are assigned values in the input file, you can declare a counting field if you wish.
- You must define a default value for key values that are not declared as counting fields and to which no values are assigned in the input file. This default value is then used for creating the primary key value.
- In the case of LOAD ONLINE the counting field may not be contained in either a referential constraint or a check constraint of the table.
- In the case of a partitioned table all partitions must be available (see [table 3 on page 22](#)).
If the SORTED clause is not specified in LOAD OFFLINE, the partitioned table must not contain a column of the type CHARACTER VARYING with a length of more than 253 characters or a column of the type NATIONAL CHARACTER VARYING with a length of more than 126 characters.

COUNTING_FIELD *column* omitted:

No counting field is used.

LOAD formats

The following table contains an overview of the LOAD formats.

The examples for LOAD statements do not contain all the possible clauses (e.g. OVERWRITE, GENERATE INDEX, CONSTRAINT CHECK, USING FILE ..., COUNTING_FIELD ..).

Format	Input file created by	Null values	Statement (example)	Loaded columns
TRANSFER_FORMAT	UNLOAD statement with TRANSFER_FORMAT	Internal representation (Y/N for all columns at the end of each row)	LOAD FILE <i>file</i> INTO TABLE <i>table</i> TRANSFER_FORMAT	All columns defined in the transfer file
UNLOAD_FORMAT	UNLOAD statement with LOAD_FORMAT	Internal representation (negative indicator for corresponding columns)	LOAD FILE <i>file</i> INTO TABLE <i>table</i> UNLOAD_FORMAT	All columns according to <i>table</i> order; column representation ¹ according to definition in <i>table</i> ; with indicator for each column
			LOAD FILE <i>file</i> INTO TABLE <i>table</i> (<i>column, column, ...</i>) UNLOAD_FORMAT	Only the specified columns in this order; column representation as above
DELIMITER_FORMAT	UNLOAD statement with DELIMITER_FORMAT or Editor	Only DELIMITER characters	LOAD FILE <i>file</i> INTO TABLE <i>table</i> DELIMITER_FORMAT TERMINATED BY <i>delimiter</i>	All columns according to <i>table</i> order; column representation readable in <i>file</i> and convertible in <i>table</i> definition
			LOAD FILE <i>file</i> INTO TABLE <i>table</i> (<i>column, column, ...</i>) DELIMITER_FORMAT TERMINATED BY <i>delimiter</i>	Only the specified columns in this order; column representation as above

Table 8: LOAD load formats

(part 1 of 2)

Format	Input file created by	Null values	Statement (example)	Loaded columns
CSV_FORMAT	UNLOAD statement with CSV_FORMAT or editor or calculation program	DELIMITER character only (may be omitted at end of line)	LOAD FILE <i>file</i> INTO TABLE <i>table</i> CSV_FORMAT DELIMITER <i>delimiter</i> QUOTE <i>quote</i> ESCAPE <i>escape</i>	All columns according to <i>table</i> order; column representation readable in <i>file</i> and convertible in <i>table</i> definition
			LOAD FILE <i>file</i> INTO TABLE <i>table</i> (<i>column, column, ...</i>) CSV_FORMAT DELIMITER <i>delimiter</i>	Only the specified columns in this order; column representation as above; no QUOTE or ESCAPE characters
standard format	UNLOAD statement with standard format	NULL values not identifiable!	LOAD FILE <i>file</i> INTO TABLE <i>table</i>	All columns according to <i>table</i> order; column representation same in <i>file</i> and <i>table</i>
			LOAD FILE <i>file</i> INTO TABLE <i>table</i> (<i>column, column, ...</i>)	Only the specified columns in this order; column representation as above
Format defined by user	UNLOAD statement with standard format or format defined by user or Editor	If <i>null_constraint</i> applies	LOAD FILE <i>file</i> (<i>load_description, load_description</i> WHEN <i>null_constraint</i> THEN NULL,...) INTO TABLE <i>table</i>	All columns according to <i>table</i> order; column representation according to <i>load_description</i> and convertible in <i>table</i> definition
			LOAD FILE <i>file</i> (<i>load_description, load_description</i> WHEN <i>null_constraint</i> THEN NULL,...) INTO TABLE <i>table</i> (<i>column, column, ...</i>)	Only the specified columns in this order; column representation as above

Table 8: LOAD load formats

(part 2 of 2)

¹ Column value representation in the load file

Examples

In the examples below, data from an input file is loaded into the CUSTOMERS table. The CUSTOMERS table belongs to the schema ORDERPROC of the database ORDERCUST.

1. The data is located in the input file SES.CUSTOMER.DEL. The values are separated by the DELIMITER character “;”.



```
LOAD OFFLINE SORTED FILE 'SES.CUSTOMER.DEL'
INTO TABLE ordercust.orderproc.customers
OVERWRITE GENERATE INDEX CONSTRAINT CHECK
DELIMITER_FORMAT TERMINATED BY ';'

```

Data is loaded from the file SES.CUSTOMER.DEL in the OFFLINE mode, which is sorted by the primary key values. Column values in the rows of the CUSTOMERS table are overwritten by the values in the input records that have the same primary key value. The indexes defined for the CUSTOMERS table are rebuilt. The integrity constraints defined for the CUSTOMERS table are checked. The number of loaded records is output in the SQLrowcount field of the diagnostic area.

After the load operation, the CUSTOMERS table is in the state “copy pending” and, if integrity constraints are violated, also in the state “check pending”.

The input file SES.CUSTOMER.DEL contains the following data:

```
100;Siemens AG;Otto-Hahn-Ring 6;81739;Munich;D;089/636-8;Electrical
101;Login GmbH;Rosenheimer Str.34;81667;Munich;D;089/4488870;PC Networks
102;JIKO GmbH;Posener Str. 12;30659;Hanover;D;0551/123874;Import/Export
103;Plenzer Trading;Paul-Heyse-Str. 12;80336;Munich;D;089/923764;Fruit market
____Freddy's Fishery                Hirschgartenstr. 12
12;12587;Berlin;D;016/5739921;Unit retail;
105;The Poodle Parlor;Am Muehlentor 26;41179;Moenchengladbach;D;040/873562;Service
106;Foreign Ltd.;26 West York St.;;New York, NY;USA;001703/2386532;Commercial agency
107;Externa & Co KG;Bernner Weg 78;03000;Bern 33;CH;;;Lawyer

```

2. The data is contained in readable and formatted form in the input file SES.CUSTOMERDATA. The load description for the LOAD statement assigns the data type CHARACTER to the input values represented in readable form.

```
LOAD ONLINE SORTED FILE 'SES.CUSTOMERDATA'  
  (POSITION(1) INTEGER,  
   POSITION(5) CHARACTER(40),  
   POSITION(45) CHARACTER(40),  
   POSITION(85) CHARACTER(5) WHEN Zip='00000' THEN NULL,  
   POSITION(90) CHARACTER(40),  
   POSITION(130) CHARACTER(3),  
   POSITION(133) CHARACTER(25),  
   POSITION(158) CHARACTER(50))  
INTO TABLE ordercust.orderproc.customers  
(cust_num, company, street, zip, city, country, cust_tel, cust_info)  
OVERWRITE CONSTRAINT CHECK
```

Data from the input file SES.CUSTOMERDATA which is sorted according to primary key values, is loaded in the ONLINE mode. A WHEN...THEN NULL clause is defined for position 85 in the input file, which is assigned to the ZIP column of the CUSTOMERS table: If the value 00000 is located at positions 85 to 89 in the input file, the NULL value is entered at the corresponding position in the CUSTOMERS table.

Column values in the rows of the CUSTOMERS table are overwritten by the values in the input records that have the same primary key value. The integrity constraints defined for the CUSTOMERS table are checked.

Advantages of LOAD ONLINE:

- the indexes are updated
- logging is not interrupted
- DML applications can access the base table in parallel

If integrity constraints are violated, the CUSTOMER table will be in the “check pending” state after loading.

The input file SES.CUSTOMERDATA has the following contents:

____Siemens AG		Otto-Hahn-Ring 6	
81739Munich	D	089/636-8	Electrical
____Login GmbH		Rosenheimer Str.34	
81667Munich	D	089/4488870	PC Networks
____JIKO GmbH		Posener Str. 12	
30659Hanover	D	0551/123874	Import-Export
____Plenzer Trading		Paul-Heyse-Str. 12	
80336Munich	D	089/923764	Fruit market
____Freddy's Fishery		Hirschgartenstr. 12 12	
12587Berlin	D	016/5739921	Unit retail
____The Poodle Parlor		Am Muehlentor 26	
41179Moenchengladbach	D	040/873562	Service
____Foreign Ltd.		26 West York St.	
00000New York, NY	USA001703/2386532		Commercial agency
____Externa & Co KG		Berner Weg 78	
03000Bern 33	CH		Lawyer

The primary keys are of the type INTEGER and must be entered in the input file in binary form. In the above example, they are represented by the string “____”.

See also

UNLOAD, EXPORT/IMPORT TABLE

MIGRATE - Migrate databases and tables

MIGRATE makes migration possible, i.e.

- conversion of a SESAM/SQL database created with version 1.1 or earlier into a base table of the current SESAM/SQL version (referred to as a SESAM/SQL-Server table below).
- conversion of a CALL DML/SQL table into an SQL table.
- conversion of a CALL DML only table into a CALL DML/SQL table.

Executing the utility statement MIGRATE interrupts logging for the user space involved. This means that when a user space is repaired, only modifications made up to the point at which migration was started can be applied.

If logging is activated for the user space affected by MIGRATE, the database administrator must create a SESAM backup copy of the user space involved with COPY after migration is performed. Otherwise, the user space remains in the state “copy pending” and can only be processed with utility statements (see [section “Space states and permissible utility statements” on page 13](#)).

Converting a database created with SESAM/SQL V1.1 or earlier into a SESAM/SQL-Server table

The starting point for converting a database created with SESAM/SQL V1.1 or an earlier version (referred to as a V1 database below) into a SESAM/SQL-Server table is a backup file (DB-SIB file) of the database to be converted created beforehand with SESASB. If this database was created with a password catalog, a backup file of the password catalog (PK-SIB file) created beforehand with the utility SEPA is also needed for the conversion procedure.

The column names of the V1 database to be converted must observe the rules for column names of a SESAM/SQL-Server table. Otherwise, the column names must be modified accordingly with SESASB before conversion is performed.

The current authorization identifier must have the special privilege UTILITY or must satisfy one of the following conditions:

- If the USING SPACE clause is not specified and the default space *D0authorization_identifier* of the owner of the schema to which the SESAM/SQL-Server table is to belong does not exist:
The current authorization identifier must own the schema to which the SESAM/SQL-Server table is to belong. The current authorization identifier must also have the special privilege USAGE for the default storage group D0STOGROUP.
- If the USING SPACE clause is not specified, but the default space *D0authorization_identifier* owns the schema to which the SESAM/SQL-Server table is to belong:
The current authorization identifier must own the schema to which the SESAM/SQL-Server table is to belong.
- If the USING SPACE clause is specified:
The current authorization identifier must own the schema to which the SESAM/SQL-Server table is to belong. Furthermore, the owner of this schema must also be the owner of the user space specified in the USING SPACE clause.

```
MIGRATE v1dbsib [PASSWORD_CATALOG v1pksib] [PASSWORD password]
```

```
[ { WITH INDEX }  
  { NO INDEX } ] TO [CALL DML] TABLE table
```

```
[USING SPACE space]
```

v1dbsib

Name of the DB-SIB file for the database created with version 1.1 or an earlier version that is to be converted. You must specify *v1dbsib* as an alphanumeric literal.

PASSWORD_CATALOG *v1pksib*

Name of the PK-SIB file for the password catalog of the V1 database to be converted. You must specify *v1pksib* as an alphanumeric literal.

You must specify PASSWORD_CATALOG *v1pksib* if a password-protected database is to be converted into a CALL DML table (see TO CALL DML TABLE [page 169](#)).

PASSWORD_CATALOG *v1pksib* cannot be specified in the following cases:

- The database is to be converted into an SQL table (see TO TABLE [page 169](#)).
- The database to be converted was defined without a password catalog.
- The password catalog has been destroyed.

PASSWORD *password*

BS2000 password for the DB-SIB file and, if one exists, the PK-SIB file of the database to be converted. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

WITH INDEX specified:

All the indexes defined for the database to be converted are built. SESAM/SQL assigns the names for these indexes. SESAM/SQL places the indexes in the user space in which the resulting base table is stored.

SESAM/SQL issues a warning if, after conversion, an error occurs during index definition or generation. The migrated data remains intact and the user space involved is in the state it would be in if WITH INDEX had not been specified. A user space for which logging has been activated is therefore placed in the state "copy pending". In this case, you must create a SESAM backup copy of the user space with the utility statement COPY before you can define the indexes.

NO INDEX specified:

The indexes of the database to be converted are not included.

TO [CALL DML] TABLE *table*

table

Name of the SESAM/SQL-Server table into which the database is to be converted. The database and the schema in which *table* is to be located must already exist. *table* must be different from all the table names and view names of this schema, i.e. the table *table* must not yet exist. You can qualify *table* with a schema name.

CALL DML specified:

The database is converted into a CALL DML table.

A default value character for representing non-significant values must be defined for each attribute of the database to be converted.

CALL DML omitted:

The database is converted into an SQL table.

The database to be converted cannot include any “old attribute formats”, i.e. attribute formats from SESAM V13.1 or an older version.

USING SPACE *space*

Name of the user space in which *table* is to be located. The user space must already be defined for the corresponding database.

You can qualify the unqualified space name with the name of the database. This database name must be the same as the database name of the table.

USING SPACE *space* omitted:

table is stored in the default space `D0authorization_identifier` of the current authorization identifier. If this default space does not yet exist, it is generated on the default storage group `D0STOGROUP`.

Handling of the compound key during conversion

If a V1 database with a compound key is converted into a SESAM/SQL-Server table, SESAM/SQL uses the name of the compound key as the name for the corresponding primary key constraint of this SESAM/SQL-Server table.

To avoid name conflicts when two or more V1 databases with the same compound key names are converted into SESAM/SQL-Server tables, SESAM/SQL assigns a new, unique name for a primary key constraint (if the derived name already exists) to all but the first of these SESAM/SQL-Server tables. The database administrator can obtain information on these names via the appropriate information schema (see the [“SQL Reference Manual Part 1: SQL Statements”](#)).

If two V1 databases have the same compound key name, this name can only be used for the corresponding primary key constraints without changing it if the V1 databases are converted into tables that are located in different schemas.

Converting a CALL DML/SQL table into an SQL table

When a CALL DML/SQL table is converted into an SQL table, SESAM/SQL removes all non-significant values and deletes the password catalog. It is then no longer possible to access this table using CALL DML statements.

The current authorization identifier must have the special privilege UTILITY or must own the schema to which the table to be converted belongs.

When a partitioned CALL-DML/SQL table is converted into an SQL table, all partitions must be available (see [table 3 on page 22](#)).

```
MIGRATE [CALL DML] TABLE table
```

table

Name of the CALL DML/SQL table to be converted into an SQL table.

Converting a CALL DML only table into a CALL DML/SQL table

If all old attribute formats, i.e. attribute formats of SESAM Version 13.1 or earlier, in a table have been changed using ALTER COLUMN, the table nevertheless remains a CALL DML only table. It continues to be accessible with CALL DML only.

If you want to use SQL to access the table, you must perform a MIGRATE. MIGRATE converts a CALL DML only table into a CALL DML/SQL table.

```
MIGRATE CALL DML ONLY TABLE table
```

table

Name of the CALL DML only table which is to be converted to a CALL DML/SQL table. The table must no longer contain any “old attribute formats”.

MODIFY - Maintain metadata for SESAM backup copies

You can use MODIFY to delete the following elements:

- Records from the RECOVERY_UNITS and DA_LOGS catalog tables
- Recovery unit and CAT-LOG records from the current CAT-REC file

After the following functional descriptions, the various forms of the utility statement MODIFY will be explained.

delete records from the catalog tables RECOVERY_UNITS and DA_LOGS

The catalog table RECOVERY_UNITS contains records with information on the SESAM backup copies for user spaces.

SESAM/SQL adds a new record to the RECOVERY_UNITS catalog table for each SESAM backup copy created with the utility statement COPY.

Each record includes the name of the user space, the time stamp assigned to the SESAM backup copy and information on the DA-LOG file created after the backup.

The catalog table DA_LOGS contains records with information on existing DA-LOG files. Each record in DA_LOGS includes the number of the corresponding DA-LOG file, information on when the file was created, as well as information on all the user spaces for which modifications have been logged in the DA-LOG file.

MODIFY allows you to maintain the catalog tables RECOVERY_UNITS and DA_LOGS by deleting the records that are no longer required. MODIFY provides you with two ways of doing this:

- delete records in RECOVERY_UNITS that reference SESAM backup copies of a certain user space
- delete records in RECOVERY_UNITS and DA_LOGS regardless of their assignment to certain user spaces

If you use MODIFY to delete records from the catalog table RECOVERY_UNITS or DA_LOGS, the corresponding SESAM save files of the user spaces or DA-LOG files remain intact.

For reasons of data protection, you should delete any SESAM backup copies and DA-LOG files that you no longer need from disk in the BS2000 system with the DELETE-FILE ... OPTION=*DESTROY-ALL command. SESAM backup copies and DA-LOG files on magnetic tape cartridge are part of the corresponding HSMS archive or ARCHIVE directory and must therefore be managed using HSMS and ARCHIVE tools (see the [“HSMS \(BS2000\)”](#) and [“ARCHIVE \(BS2000\)”](#) manuals).

At least the last two records of any space are retained in the RECOVERY_UNITS table. One of these records refers to the last SESAM backup copy of the space. A RECOVER can be performed on the basis of the record which describes the last SESAM backup copy. The required records in the DA_LOGS table are also retained.

Foreign copies represent a special case here. When you work with foreign copies, no new records are created in the RECOVERY_UNITS table. You can specify the UNRESTRICTED clause in order to delete records from the DA_LOGS catalog table. Records are then deleted from DA_LOGS whether or not there is a last RECOVERY_UNITS record.

**CAUTION!**

UNRESTRICTED deletes all the records with the specified age in the RECOVERY_UNITS and DA_LOGS, up to but excluding the last in each case. This record is retained for technical reasons.

A RECOVER is no longer possible if the required files have been deleted in the catalog tables RECOVERY_UNITS and DA_LOGS. The oldest record in DA_LOGS required for a foreign copy of a space is the record that was created in the space when it was closed after the last update. If the foreign copy is generated during or after a session in which the space was not opened or was used for read access only then the restart point is not updated in the space.

The catalog tables RECOVERY_UNITS and DA_LOGS are described in more detail in the [“Core manual”](#).

Delete records from the CAT-REC file

The CAT-REC file (catalog recovery file) contains recovery unit records for the catalog space and the associated CAT-LOG records.

SESAM/SQL adds a new recovery unit record to the CAT-REC file for each SESAM backup copy of the entire database or of the catalog space which was created with the utility statement COPY.

Each recovery unit record contains, among other things, the name of the backup copy of the catalog space and the time stamp of the SESAM backup.

Recovery unit records of the CAT-REC file correspond to the records of the RECOVERY_UNITS catalog table, but relate to the catalog space.

SESAM/SQL adds a new CAT-LOG record to the CAT-REC file for each change of the CAT-LOG file. All modifications which relate to the catalog space are logged in the CAT-LOG file. Each CAT-LOG record contains, among other things, the name of the CAT-LOG file, the creation date, and the version number of the associated SESAM backup copy. CAT-LOG records of the CAT-REC file correspond to the records of the DA_LOGS catalog table, but relate to the catalog space.

MODIFY enables you to maintain the database's current CAT-REC file by deleting records which are no longer required (online update of the CAT-REC file).

When you delete records from the CAT-REC file using MODIFY, the associated SESAM backup files of the catalog and of the catalog space are retained, as are the CAT-LOG files.

For reasons of data protection, you should delete any SESAM backup copies and CAT-LOG files that you no longer need from disk in the BS2000 system with the DELETE-FILE ... OPTION=*DESTROY-ALL command. SESAM backup copies on magnetic tape cartridge are part of the associated HSMS archive or ARCHIVE directory and must therefore be managed using HSMS and ARCHIVE (see the "[HSMS \(BS2000\)](#)" and "[ARCHIVE \(BS2000\)](#)" manuals).

The most recent recovery unit record and the associated CAT-LOG records are retained in the CAT-REC file. A RECOVER can be performed on the basis of the record which describes the last SESAM backup copy. The CAT-LOG records required for this are also retained.

Foreign copies represent a special case here. When you work with foreign copies, no recovery unit records are created in the CAT-REC file. CAT-LOG records cannot be deleted without the corresponding recover unit record.

The CAT-REC file is described in more detail in the "[Core manual](#)".

Delete records in RECOVERY_UNITS for a specific user space

The current authorization identifier must have the special privilege UTILITY or must own the specified user space.

```
MODIFY RECOVERY SPACE space DELETE { ALL
                                       AGE n
                                       DATE 'year-month-day' }
```

space

Name of the user space for which records are to be deleted from RECOVERY_UNITS.

DELETE ALL

All the records for *space* are deleted from RECOVERY_UNITS. At least the last two records are, however, retained. One of the remaining records refers to the SESAM backup copy of the user space *space* that was created last.

DELETE AGE *n*

n is a positive integer.

All the records for *space* whose time stamp indicates a date older than *n* days ago are deleted from RECOVERY_UNITS. At least the last two records are, however, retained. One of the remaining records refers to the SESAM backup copy of the user space *space* that was created last.

DELETE DATE '*year-month-day*'

DATE '*year-month-day*' is a date.

All the records for *space* whose time stamp indicates a date older than '*year-month-day*' are deleted from RECOVERY_UNITS. At least the last two records are, however, retained. One of the remaining records refers to the SESAM backup copy of the user space *space* that was created last.

Delete records in RECOVERY_UNITS and DA_LOGS regardless of their assignment to certain user spaces

The current authorization identifier must have the special privilege UTILITY.

```
MODIFY RECOVERY DELETE [UNRESTRICTED] { ALL
                                         AGE n
                                         DATE 'year-month-day' } AT CATALOG catalog
```

DELETE ALL

At least the last two records are retained in the RECOVERY_UNITS for each user space. One of the remaining records refers to the SESAM backup copy of the user space that was created last. The records in DA_LOGS for the DA-LOG files created after these SESAM backup copies are also retained.

All the other records in RECOVERY_UNITS and DA_LOGS are deleted. ALL must not be specified if UNRESTRICTED was selected.

DELETE AGE *n*

n is a positive integer.

At least the last two records are retained in the RECOVERY_UNITS for each user space. One of the remaining records refers to the SESAM backup copy of the user space that was created last. The records in DA_LOGS for the DA-LOG files created after these SESAM backup copies are also retained.

All the other records in RECOVERY_UNITS and DA_LOGS whose time stamp indicates a date older than *n* days ago are deleted.

DELETE DATE '*year-month-day*'

DATE '*year-month-day*' is a date.

At least the last two records are retained in the RECOVERY_UNITS for each user space. One of the remaining records refers to the SESAM backup copy of the user space that was created last. The records in DA_LOGS for the DA-LOG files created after these SESAM backup copies are also retained.

All the other records in RECOVERY_UNITS and DA_LOGS whose time stamp indicates a date older than '*year-month-day*' are deleted.

UNRESTRICTED

UNRESTRICTED deletes all the records with the specified age in the RECOVERY_UNITS and DA_LOGS, up to but excluding the last in each case. This record is retained for technical reasons.

**CAUTION!**

RECOVER is no longer possible if required entries in the catalog table DA_LOGS have been deleted through the specification of UNRESTRICTED.

AT CATALOG *catalog*

Name of the database whose catalog tables RECOVERY_UNITS and DA_LOGS are to be processed with MODIFY.

Examples (delete records from the catalog tables)

1. With the exception of the remaining records for the last created SESAM backup copy of the user space TABLESPACE, this example deletes all the records for SESAM backup copies of TABLESPACE from the catalog table RECOVERY_UNITS of the database ORDERCUST.



```
MODIFY RECOVERY SPACE tablespace DELETE ALL
```

2. In this example, all the records for the SESAM backup copies of BLOBSPACE are deleted from the RECOVERY_UNITS catalog table of the ORDERCUST database if their creation date is more than 15 days ago, with the exception of the remaining records that reference the last SESAM backup copy created for the user space BLOBSPACE. The corresponding records for the last BLOBSPACE SESAM backup copy created are retained in RECOVERY_UNITS even if it was created more than 15 days ago.

```
MODIFY RECOVERY SPACE blobSPACE DELETE AGE 15
```

3. In this example, the remaining records referencing the last SESAM backup copy created for a user space remain in RECOVERY_UNITS for all the user spaces of the database ORDERCUST. The records in DA_LOGS for the DA-LOG files created after these SESAM backup copies are also retained. All the other records in the catalog tables RECOVERY_UNITS and DA_LOGS that belong to the database ORDERCUST are deleted.

```
MODIFY RECOVERY DELETE ALL AT CATALOG ordercust
```


Delete recovery unit and CAT-LOG records from the current CAT-REC file

The current authorization identifier must have the special privilege UTILITY.

```
MODIFY RECOVERY CATALOG_SPACE catalog DELETE { ALL
                                                AGE n
                                                DATE 'year-month-day' }
```

catalog

Logical name of the database whose current CAT-REC file is to be processed with MODIFY (online update of the CAT-REC file). The database must be entered in the database catalog. The database may already be active.

DELETE ALL

All recovery unit records and CAT-LOG records for *catalog* are deleted from the CAT-REC file. However, the latest recovery unit record and the associated CAT-LOG records are retained so that a recovery is still possible with SESAM/SQL.

DELETE AGE *n*

n is a positive integer.

All recovery unit and CAT-LOG records for *catalog* whose time stamps specify a date that is more than *n* days ago are deleted from the CAT-REC file.

However, the latest recovery unit record and the associated CAT-LOG records are retained so that a recovery is still possible with SESAM/SQL.

DELETE DATE '*year-month-day*'

DATE '*year-month-day*' is a date.

All recovery unit rows and CAT-LOG records for *catalog* whose time stamps specify a date that is older than '*year-month-day*' are deleted from the CAT-REC file.

However, the latest recovery unit record and the associated CAT-LOG records are retained so that a recovery is still possible with SESAM/SQL.

Example (delete records from the CAT-REC file)

In this example, the remaining recovery unit record in the CAT-REC file referencing the last SESAM backup copy created for a catalog space is retained for the ORDERCUST database. The associated CAT-LOG records in the CAT-REC file are also retained. All other recovery unit records and CAT-LOG records in the CAT-REC file which belong to the ORDERCUST database are deleted.

```
MODIFY RECOVERY CATALOG_SPACE ordercust DELETE ALL
```

RECOVER - Reset and repair a database, rebuild indexes

You can use RECOVER to

- reset or repair parts of a SESAM/SQL database or the entire database
- rebuild indexes.

The different RECOVER statements are described in separate sections below.

Reset and repair of user spaces and the database

You can use RECOVER to repair, reset or rebuild the following units:

- individual user space
- a space set
- a space list
- the database catalog space
- the entire database

For this you use:

- SESAM backup copies using the utility statement COPY
- foreign copies created using any BS2000 means (e.g. TimeFinder)
- replications created using the utility statement CREATE REPLICATION

When simultaneously updating various user spaces, you can achieve a higher level of parallelism by running several RECOVER statements that are executed in parallel. For further information on this, see the “[Core manual](#)”, section “Execution of utility statements by SESAM/SQL”.

A unit of several user spaces with a common time stamp can be designated as a space set. Spaces with the same time stamp are created on a backup with a COPY CATALOG statement or when a number of spaces are backed up with one COPY statement. The recovery of a space set as a unit is only possible if logging was not interrupted for any of the user spaces in this space set.

A space list designates a number of user spaces from the backup of an entire database or a space set. The spaces in a space list must all have the same time stamp. The recovery of the spaces in a space list is only possible if logging was not interrupted for any of the user spaces in this space list.

If you use a foreign copy for RECOVER then you must copy the files for the foreign copy to the corresponding spaces yourself. If the FOREIGN COPY clause is specified then only the modifications logged in the logging files are applied to the spaces on the basis of the restart information present in the spaces. This does not affect SESAM backup copies. In order to read in the foreign copy, you must close the affected space or the database.

Space list in the case of foreign copy:

You can specify a space list in the case of RECOVER with foreign copy. In the case of foreign copies of different spaces, the spaces must have the same backup time stamp. You can ensure this if you use the administration statement PREPARE-FOREIGN-COPY before you create the foreign copies.

A replication can be stored on any user ID. The user ID in question is specified via a clause in the statement.

In the case of SESAM backups to disk, the required save file is always first searched for on the DB user ID. If it is not present on this user ID, it is read in from the DBH user ID. If SESAM backup copies are present on both the DB and DBH user IDs then those from the DB user ID are used.

In the case of SESAM backups on magnetic tape cartridge via HSMS or ARCHIVE, all the information concerning the location of the backup is stored in the catalog table RECOVERY_UNITS and in the associated HSMS archive or ARCHIVE directory.

If you use an online backup created with ARCHIVE for RECOVER then the status at the beginning of the copy is recovered from the PBI file that was also backed up.

Using the utility monitor form INF.5.2, you can output the required logging files to the screen, to a file or to job variables (see the “[Utility Monitor](#)” manual).

In order to repair a database's catalog space or the entire database using RECOVER, the catalog recovery file (CAT-REC file) must be present. It is advisable to keep a duplicate of the CAT-REC file (see “[Core manual](#)”). However, if the CAT-REC file should prove to be defective, please contact the service.

For partitioned tables all partitions must be available in the event of resetting with recovery of indexes

(RECOVER ... TO and NO INDEX not specified) and when integrity constraints are defined for the table (see [table 3 on page 22](#)).

The table below provides a detailed view of the possible RECOVER statements and their syntax combinations (x indicates the permitted combinations).

Statement	USING					TO				USING TO [ANY] <i>timestamp_literal</i>				RESTART	ADJUST	further possible clauses		
	COPY_FILE	COPY_NUMBER	<i>timestamp_literal</i>	FOREIGN COPY	REPLICATION	COPY_FILE	COPY_NUMBER	[ANY] <i>timestamp_literal</i>	FOREIGN COPY	REPLICATION	COPY_FILE	COPY_NUMBER	<i>timestamp_literal</i>				FOREIGN COPY	REPLICATION
RECOVER CATALOG	x	x	x	x	x	x	x	x ¹	x	x	x ¹	x ¹	x ¹	x ¹	x	-	-	PASSWORD GENERATE INDEX ² SCOPE PENDING NO INDEX
RECOVER CATALOG_SPACE	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	PASSWORD GENERATE INDEX ² SCOPE PENDING NO INDEX
RECOVER SPACE <i>space</i>	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	PASSWORD GENERATE INDEX ² SCOPE PENDING NO INDEX
RECOVER SPACE <i>space, space, ...</i>	-	-	x	x	x	-	-	x	x	x	-	-	-	-	-	-	-	PASSWORD GENERATE INDEX ² SCOPE PENDING NO INDEX
RECOVER SPACESET	-	-	x	-	-	-	-	x	-	-	-	-	-	-	-	-	-	PASSWORD GENERATE INDEX ² SCOPE PENDING NO INDEX

Table 9: Overview of the possible RECOVER statements and their syntax combinations

¹ If *timestamp_literal* is specified, ANY is also permissible

² GENERATE INDEX ON NO LOG INDEX SPACE

RECOVER SPACE

In order to execute RECOVER SPACE, the current authorization identifier must either have the special privilege UTILITY or must own the user space affected by RECOVER.

This section describes the following RECOVER statements:

- RECOVER of a space using SESAM backup copies or foreign copies
- RECOVER of a space using replications

When a user space is repaired by means of RECOVER *space* USING, the space is placed in the “copy pending” state if it is subject to the logical data backup and if the changes must be applied for a partitioned table on this space with CREATE INDEX or DROP INDEX.

RECOVER SPACE using SESAM backup copies or foreign copies

```
RECOVER SPACE space [ { [USING rec_unit] [TO rec_unit]
                        RESTART
                        ADJUST
                      } ]
```

```
[PASSWORD password] [NO INDEX]
```

```
rec_unit ::= { COPY_FILE file
                COPY_NUMBER number
                timestamp_literal
                FOREIGN COPY
              }
```

```
timestamp_literal ::= TIMESTAMP 'year-month-day hour:minute:second'
```

RECOVER SPACE *space*

Name of the user space to be repaired or reset.

You may not specify "CATALOG" as the name of the user space.

In the case of RECOVER SPACE and RECOVER SPACE ... USING the following conditions must be satisfied for the user space *space*:

- For SESAM backups, the required SESAM backup copy of *space* must be present and must be entered in the catalog table RECOVERY_UNITS. There is no entry for foreign copies in the catalog table RECOVERY_UNITS. The restart information for applying the modifications logged in the logging files is stored in the space itself.
- Logging must not have been interrupted for the user space *space* since the creation of the required backup copy (e.g. by LOAD OFFLINE, IMPORT TABLE, MIGRATE or ALTER SPACE ... NO LOG).
- All DA-LOG files created after the required backup copy for *space* must be present.

RECOVER SPACE *space* without USING, TO, RESTART or ADJUST

SESAM/SQL repairs the user space *space* using the last SESAM backup copy created for this user space.

USING *rec_unit*

SESAM/SQL repairs the user space *space* using the SESAM backup copy indicated by *rec_unit* or the foreign copy.

USING may not specify a mark.

If a foreign copy is to be used, then the space must have been present for logging at the time when the foreign copy was created. The DA-LOG entry that is recorded as the restart point in the foreign copy must still be entered in the database's metadata.

Spaces that do not fulfill these conditions remain unchanged on a RECOVER SPACE ... USING FOREIGN COPY. Such spaces can only be subsequently accessed if they have not been modified since the time the foreign copy was created. If this is not the case, an error message informing you that the space is inconsistent is issued when you attempt to access such a space. If the space was not present in logging at the time the foreign copy was created then only RECOVER SPACE ... TO FOREIGN COPY is possible.

TO *rec_unit*

SESAM/SQL resets the user space *space* to the SESAM backup copy specified by *rec_unit* or to the foreign copy. If a mark is specified in TO, the modifications logged in the log files are applied on the basis of the previous SESAM backup up to the mark.

**CAUTION!**

You cannot undo a RECOVER SPACE ... TO.

It is not possible to achieve a more recent state than that identified by *rec_unit*.

This may lead to inconsistencies between the SESAM backup copy or foreign copy of a user space that has been read in and the corresponding metadata in the catalog space. In this case, SESAM/SQL adjusts the data in the reset user space to match the metadata in the catalog space. The “[Core manual](#)” explains in detail how SESAM/SQL makes this adjustment.

**CAUTION!**

The data in a table may be lost when RECOVER SPACE ... TO is performed. To adjust a table or index to the corresponding metadata in the catalog space, the table or index must be deleted and then rebuilt.

If the space is present in logging at the time RECOVER SPACE... TO was performed and if the indexes have been rebuilt then the space is placed in the state “copy pending”.

USING *rec_unit* TO *timestamp_literal* ¹

This is a resetting to the state specified in TO.

If both USING and TO are specified, USING must specify a backup and only the time stamp specification of a backup or mark is permitted for TO. The state specified in TO is recovered by applying the modifications logged in the log files on the basis of the backup specified in USING. The backup specified in USING must be older than the time stamp specified in TO.

**CAUTION!**

The notes and warnings described in RECOVER SPACE ... TO also apply here.

¹ **TIMESTAMP** 'year-month-day hour:minute:second' is described in more detail in the manual “[SQL Reference Manual Part 1: SQL Statements](#)” under “Time stamp”.

RESTART

If a repair operation is interrupted because of missing DA-LOG file(s), SESAM/SQL resumes repairing the user space *space* at the place the repair operation was interrupted.

If execution of the utility statement RECOVER was not interrupted due to missing DA-LOG file(s) but for other reasons, RECOVER SPACE ... RESTART may be rejected. In this case, you will have to restart the repair of the user space again using RECOVER.

ADJUST

If a repair operation is interrupted because of missing DA-LOG file(s), SESAM/SQL adjusts the current state of the user space *space* achieved by the repair operation to the current metadata of the catalog space. The “[Core manual](#)” explains in detail how SESAM/SQL makes this adjustment.

If execution of the utility statement RECOVER was not interrupted due to missing DA-LOG file(s) but for other reasons, RECOVER SPACE ... ADJUST may be rejected. In this case, you will have to restart the repair of the user space again using RECOVER.

rec_unit ::=

Backup copy or mark.

The SESAM backup copy specified by *rec_unit* or the mark specified by *rec_unit* must be entered in the catalog table RECOVERY_UNITS.

There is no entry for foreign copies in the catalog table RECOVERY_UNITS. The restart information for applying the modifications logged in the logging files is stored in the space itself.

COPY_FILE file

File name of a SESAM backup copy created with the utility statement COPY. You must specify *file* as an alphanumeric literal.

For RECOVER SPACE ... TO COPY_FILE, you can reset to the file *file* even if *file* is not entered or is no longer entered in the catalog table RECOVERY_UNITS.

For RECOVER SPACE ... TO COPY_FILE, you can also specify '*DUMMY' as *file*. In this case, SESAM/SQL resets to the empty user space and ensures that the reset user space and the metadata in the catalog space are consistent. How SESAM/SQL does this is described in the [“Core manual”](#).



CAUTION!

If you specify RECOVER SPACE ... TO COPY_FILE '*DUMMY' then all the tables and indexes in the reset user space are initialized, but the tables do not contain any user data. Indexes only contain data if they belong to tables located on other user spaces.

Specifying '*DUMMY' is, for example, useful if there is no SESAM backup copy for a defective user space or if a space's table contents are to be deleted.

COPY_NUMBER number

Version number of a SESAM backup copy created with the utility statement COPY. *number* is an integer with a maximum of six digits.

The SESAM backup copy indicated by *number* must be entered in the catalog table RECOVERY_UNITS.

timestamp_literal

Time stamp which indicates the time of a SESAM backup copy or a mark set in LOAD OFFLINE, MIGRATE or IMPORT. The copy or mark must be known in the RECOVERY_UNITS table.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

You can find more detailed information on the use of marks in the [“Core manual”](#). No mark can be specified in the case of RECOVER SPACE ... USING.

FOREIGN COPY

If this clause is specified then only the modifications logged in the logging files are applied but no SESAM backup copies are read in. It is up to the user to copy the files of the foreign copy onto the relevant spaces. The space must be closed while the foreign copy is read in. For information on creating a foreign copy, refer to the “[Core manual](#)”.

PASSWORD *password*

BS2000 *password* You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

You use PASSWORD *password* to specify the password for all the SESAM backup copies that SESAM/SQL requires during execution of the RECOVER statement.

You must always specify the BS2000 password for a SESAM backup copy if the BS2000 password of the database was changed after the SESAM backup copy was created, i.e. if the BS2000 passwords for the database and the SESAM backup copy are no longer the same.

SCOPE PENDING

If you specify this clause the space will be repaired or reset if it cannot be opened for update processing, because, for example, it had previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Even a space containing a table, which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

SCOPE PENDING is not permitted in the case of foreign copies.

NO INDEX

This clause identifies indexes as defective and as not having been rebuilt if they have become invalid as a result of resetting a space.

Indexes become invalid if the base table and an associated index are located on different user spaces or if the table is partitioned and not all the user spaces affected are reset simultaneously.

NO INDEX is only permitted with RECOVER SPACE ... TO, RECOVER SPACE USING ... TO and RECOVER SPACE... ADJUST.

RECOVER SPACE using replications

```
RECOVER SPACE space { USING replication_description [TO timestamp_literal]
                       TO replication_description }
```

```
[SCOPE PENDING] [NO INDEX]
```

```
replication_description ::=
```

```
  REPLICATION replication
  [PASSWORD password]
  [ON USER_ID bs2000_user_id]
  { COPY
    RENAME }
```

RECOVER SPACE *space*

Name of the user space to be repaired or reset. "CATALOG" may not be specified as the name.

Precondition for RECOVER SPACE *space* USING:

- The replication must be available.
- All DA-LOG files created after the replication and recorded in the catalog table DA_LOGS must be available.
- It is crucial that logging was not interrupted for the user space *space*.

USING *replication_description*

SESAM/SQL repairs the user space *space* from the specified replication.

USING *replication_description* TO *timestamp_literal* ¹

The state specified in TO is recovered by applying modifications logged in the DA-LOG files on the basis of the replication specified in USING. The replication specified in USING must be older than the time stamp specified in TO.

¹ **TIMESTAMP** 'year-month-day hour:minute:second' is described in more detail in the manual "[SQL Reference Manual Part 1: SQL Statements](#)" under "Time stamp".

timestamp_literal

Time stamp which indicates the time of a SESAM backup copy or a mark set in LOAD OFFLINE, MIGRATE or IMPORT. The copy or mark must be known in the RECOVERY_UNITS table.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

You can find more detailed information on the use of marks in the [“Core manual”](#).

**CAUTION!**

You cannot undo a RECOVER SPACE USING ... TO statement. It is not possible to achieve a more recent state than that identified by *replication_description*.

This may lead to inconsistencies between the replication space that has been read in and the corresponding metadata in the catalog space. In this case, SESAM/SQL adjusts the data in the reset user space to match the metadata in the catalog space. The [“Core manual”](#) explains in detail how SESAM/SQL makes this adjustment.

**CAUTION!**

The data in a table may be lost when RECOVER SPACE USING ... TO is performed. To adjust a table or index to the corresponding metadata in the catalog space, the table or index must be deleted and then rebuilt.

RECOVER SPACE USING ... TO with a replication always causes the specified space and, if applicable, dependent index spaces to be placed in the state “copy pending” (see also the [“Core manual”](#), section “Replication of a database”).

TO *replication_description*

SESAM/SQL resets the user space *space* to the state of the specified replication.

**CAUTION!**

The notes and warnings described in RECOVER SPACE USING ... TO also apply here.

replication_description ::= =

Replication or partial replication which is to repair, reset or create an independent original.

REPLICATION *replication*

replication is the physical name of the replication.

PASSWORD *password*

BS2000 password of the replication. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

ON USER_ID *bs2000_user_id*

User ID of the replication. You must specify the user ID without a "\$".

COPY

The specified space of the replication is copied onto the catalog space. If the replication space has to be copied into a user ID other than the DBH user ID, you must ensure that the necessary files have already been created there. Alternatively, you can also define the DBH user ID as co-owner for the catalog files in the DB user ID.

The storage group specifications for the user space are ignored.

The replication can continue to be used and updated for retrieval applications. The replication itself is not updated in this RECOVER.



If COPY is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication user space continues to be available for use. You can rerun the utility statement once the cause of the problem has been remedied (e.g. the missing log files are made available). If the cause cannot be remedied (e.g. the space has been reset in the original), the user space must be repaired using a SESAM backup copy or reset using RECOVER SPACE ... TO REPLICATION.

RENAME

The affected space of the specified replication is renamed to the catalog space. It must not be available in the original, otherwise it will not be possible to rename it. After renaming, it will be missing from the replication and can be added again using REFRESH SPACE.

RENAME can only be used if the replication and the catalog are located on the same BS2000 user ID. If this user ID is different to the DBH user ID, the DBH user ID must be defined as co-owner for all the files of the replication and the catalog in the DB user ID. The replication may not be added in any SQL database catalog list.



If RENAME is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication user space will no longer be available. The affected user space must be repaired using SESAM backup copies.

SCOPE PENDING

If you specify this clause the space will be repaired or reset if it cannot be opened for update processing, because, for example, it had previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Even a space containing a table, which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

NO INDEX

This clause identifies indexes as defective and as not having been rebuilt if they have become invalid as a result of resetting a space. Indexes become invalid if the base table and an associated index are located on different user spaces or if the table is partitioned and not all the user spaces affected are reset simultaneously. NO INDEX is only permitted with RECOVER SPACE ... TO and RECOVER SPACE USING ... TO.

You can find further information about the concept and application in the “[Core manual](#)”.

RECOVER space set or space list

In order to execute RECOVER SPACESET or RECOVER SPACE with a space list, the current authorization identifier must either have the special privilege UTILITY or must own the user spaces identified by the corresponding time stamp.

This section describes the following RECOVER statements:

- RECOVER for a space set
- RECOVER a space list using SESAM backup copies or foreign copies
- RECOVER of a space list using replications

When user spaces are repaired by means of RECOVER space set / space list USING, the space concerned is placed in the “copy pending” state if it is subject to the logical data backup and if the changes must be applied for a partitioned table on this space with CREATE INDEX or DROP INDEX but not all partitions of the partitioned table are contained in the RECOVER statement.

Space set and space list

A space set consists of several user spaces. A space set is produced by means of one COPY statement of several user spaces and is identified by a time stamp. The combined backup represents a consistent backup copy of these spaces. When a space set is recovered, SESAM/SQL takes into account the consistency between the tables and their indexes.

The spaces that have been backed up using COPY CATALOG or COPY *space list* belong to a combined SESAM backup.

Those spaces that have been backed up together following a PREPARE-FOREIGN-COPY are considered as the combined backed up spaces of a foreign copy.

The spaces of a replication are also consistent with each other and together also constitute a space set.

In the case of SESAM backups, you can specify a space set via the time stamp of the combined backup. A subset can, however, also be specified as a space list. In the case of foreign copies and replications, a space set can only be specified as a space list via the list of spaces.

Only the spaces of a space set may be listed in a space list.

RECOVER for a space set

```
RECOVER SPACESET AT CATALOG catalog [ { USING timestamp_literal
                                         TO timestamp_literal } ]
```

```
[PASSWORD password] [SCOPE PENDING] [NO INDEX]
```

```
timestamp_literal ::= TIMESTAMP 'year-month-day hour:minute:second'
```

RECOVER SPACESET AT CATALOG *catalog*

Name of the database to which the space set to be repaired or reset belongs.

USING *timestamp_literal*¹

SESAM/SQL repairs all the user spaces for which a SESAM backup copy created at the time indicated by *timestamp_literal* exists. SESAM/SQL uses these SESAM backup copies, which were all created with the same COPY statement, for the repair operation.

All the required DA-LOG files must be present.

User spaces are excluded from the repair if they are not present in the logical data backup or if logical data saving was interrupted since the creation of the SESAM backup copy.

In this case, a warning is issued.

¹ `TIMESTAMP 'year-month-day hour:minute: second'` is described in more detail in the manual [“SQL Reference Manual Part 1: SQL Statements”](#) under “Time stamp”.

TO *timestamp_literal*

SESAM/SQL resets all the user spaces for which a SESAM backup copy created at the time indicated by *timestamp_literal* exists. SESAM/SQL resets to this SESAM backup copy.

All the SESAM backup copies identified by *timestamp_literal* were created using the same COPY statement.



CAUTION!

You cannot undo a RECOVER SPACESET ... TO statement.

It is not possible to achieve a more recent state than that identified by *timestamp_literal*.

This may lead to inconsistencies between the SESAM backup copies that have been read in and the corresponding metadata in the catalog space. In this case, SESAM/SQL adjusts the data in the reset user space to match the metadata. The “[Core manual](#)” explains in detail how SESAM/SQL makes this adjustment.



CAUTION!

The data in a table may be lost when RECOVER SPACESET ... TO is performed. To adjust a table or index to the corresponding metadata in the catalog space, the table or index must be deleted and then rebuilt.

If the space is present in logging at the time RECOVER SPACESET ... TO was performed and if the indexes have been rebuilt then the space is placed in the state “copy pending”.

If the reset is unsuccessful for some of the spaces in a space list then these are marked as defective. You can query the state of the spaces after a RECOVER in the view SYS_SPACE_PROPERTIES of SYS_INFO_SCHEMA. The spaces that have been marked as defective can be gathered together in a space list and reset separately once the reason for the abort has been eliminated.

timestamp_literal

Time stamp which indicates the time of a SESAM backup copy. At least one of the copies identified in this way must be known in the table RECOVERY_UNITS.

For more information about the time of copy creation, refer to the utility statement “[COPY - Create a SESAM backup copy](#)” on page 97.

PASSWORD *password*

BS2000 password You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

You use **PASSWORD** *password* to specify the password for all the SESAM backup copies that SESAM/SQL requires during execution of the RECOVER statement.

You must always specify the BS2000 password for a SESAM backup copy if the BS2000 password of the database was changed after the SESAM backup copy was created, i.e. if the BS2000 passwords for the database and the SESAM backup copy are no longer the same.

SCOPE PENDING

If you specify this clause the spaces will be repaired if they cannot be opened for update processing, because, for example, they have previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Spaces containing a table which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

SCOPE PENDING is only permitted in the case of RECOVER SPACESET ... USING.

NO INDEX

This clause identifies indexes as defective and as not having been rebuilt if they have become invalid as a result of resetting a space.

Indexes become invalid if the base table and an associated index are located on different user spaces or if the table is partitioned and not all the user spaces affected are reset simultaneously.

NO INDEX is only permitted with RECOVER SPACESET ... TO.

RECOVER a space list using SESAM backup copies or foreign copies

```
RECOVER SPACE space, space ... { USING { timestamp_literal } }
                                { TO   { timestamp_literal } }
```

```
[PASSWORD password] [SCOPE PENDING] [NO INDEX]
```

```
timestamp_literal ::= TIMESTAMP 'year-month-day hour:minute:second'
```

RECOVER SPACE *space*, *space* ...

Names of the user spaces that are to be repaired or reset. The specified user spaces must have the same backup time. You can specify up to 999 space names. "CATALOG" may not be specified as the name of a user space.

USING *timestamp_literal*¹

SESAM/SQL repairs the user spaces *space*, *space* ... for which there is a SESAM backup copy created at the time specified by *timestamp_literal*. All these SESAM backup copies were created with the same COPY statement.

All the required DA-LOG files must be present.

User spaces are excluded from the repair if they are not present in the logical data backup or if logical data saving was interrupted since the creation of the SESAM backup copy.

In this case, a warning is issued.

USING FOREIGN COPY

If this clause is specified then only the modifications logged in the logging files are applied but no SESAM backup copies are read in. It is up to the user to copy the files of the foreign copy onto the relevant spaces. The space must be closed while the foreign copy is read in. For information on creating a foreign copy, refer to the "[Core manual](#)".

The foreign copies must have been created together after the administration statement PREPARE-FOREIGN-COPY.

¹ `TIMESTAMP 'year-month-day hour:minute:second'` is described in more detail in the manual "[SQL Reference Manual Part 1: SQL Statements](#)" under "Time stamp".

TO *timestamp_literal*

SESAM/SQL resets the user spaces *space, space ...* to the state of the SESAM backup copies identified by *timestamp_literal*. All the SESAM backup copies identified by *timestamp_literal* were created using the same COPY statement.

The required SESAM backup copies of *space, space ...* must be present and entered in the catalog table RECOVERY_UNITS.

**CAUTION!**

You cannot undo a RECOVER SPACE ... TO.

It is not possible to achieve a more recent state than that identified by *timestamp_literal*.

This may lead to inconsistencies between the SESAM backup copy that has been read in and the corresponding metadata in the catalog space. In this case, SESAM/SQL adjusts the data in the reset user space to match the metadata in the catalog space. The “[Core manual](#)” explains in detail how SESAM/SQL makes this adjustment.

**CAUTION!**

The data in a table may be lost when RECOVER SPACE ... TO is performed. To adjust a table or index to the corresponding metadata in the catalog space, the table or index must be deleted and then rebuilt.

If the space is present in logging at the time RECOVER SPACE... TO was performed and if the indexes have been rebuilt then the space is placed in the state “copy pending”.

If the reset is unsuccessful for some of the spaces in a space list then these are marked as defective. You can query the state of the spaces after a RECOVER in the view SYS_SPACE_PROPERTIES of SYS_INFO_SCHEMA. The spaces that have been marked as defective can be gathered together in a space list and reset separately once the reason for the abort has been eliminated.

TO FOREIGN COPY

It is up to the user to copy the files of the foreign copy onto the relevant spaces. The space must be closed while the foreign copy is read in. For information on creating a foreign copy, refer to the “[Core manual](#)”.

**CAUTION!**

The notes and warnings described previously in RECOVER SPACE ... TO also apply here.

timestamp_literal

Time stamp which indicates the time of a SESAM backup copy. All of the copies identified in this way must be known in the RECOVERY_UNITS table.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

PASSWORD *password*

BS2000 password You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- '*n*'
n identifies an integer from - 2147483648 to + 2147483647.

You use PASSWORD *password* to specify the password for all the SESAM backup copies that SESAM/SQL requires during execution of the RECOVER statement.

You must always specify the BS2000 password for a SESAM backup copy if the BS2000 password of the database was changed after the SESAM backup copy was created, i.e. if the BS2000 passwords for the database and the SESAM backup copy are no longer the same.

SCOPE PENDING

If you specify this clause the spaces will be repaired if they cannot be opened for update processing, because, for example, they have previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Spaces containing a table which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

SCOPE PENDING is not permitted in the case of foreign copies and RECOVER SPACE ... TO.

NO INDEX

This clause identifies indexes as defective and as not having been rebuilt if they have become invalid as a result of resetting a space. Indexes become invalid if the base table and an associated index are located on different user spaces or if the table is partitioned and not all the user spaces affected are reset simultaneously.

NO INDEX is only permitted with RECOVER SPACE ... TO.

RECOVER of a space list using replications

```
RECOVER SPACE space, space ... { USING replication_description }
                                { TO replication_description }
```

```
[SCOPE PENDING] [NO INDEX]
```

```
replication_description ::=
```

```
  REPLICATION replication
  [PASSWORD password]
  [ON USER_ID bs2000_user_id]
  { COPY }
  { RENAME }
```

```
RECOVER SPACE space, space ...
```

Names of the user spaces that are to be repaired or reset.

All specified spaces must be current replication spaces

(see the “[Utility Monitor](#)” manual, mask COP.4.6).

You can specify up to 999 space names. “CATALOG” may not be specified as the name of a user space.

Precondition for RECOVER SPACE *space* USING:

- The replication must be available.
- All DA-LOG files created after the replication and recorded in the catalog table DA_LOGS must be available.
- The spaces must be present in the logging, logging may not be interrupted.

USING *replication_description*

SESAM/SQL repairs the user spaces *space*, *space* ... with the specified replication.

TO *replication_description*

SESAM/SQL resets the user spaces *space*, *space* ... to the state of the specified replication.

**CAUTION!**

You cannot undo a RECOVER SPACE ... TO. It is not possible to achieve a more recent state than that identified by *replication_description*.

This may lead to inconsistencies between the replication space that has been read in and the corresponding metadata in the catalog space. In this case, SESAM/SQL adjusts the data in the reset user space to match the metadata in the catalog space. The “[Core manual](#)” explains in detail how SESAM/SQL makes this adjustment.



CAUTION!

The data in a table may be lost when RECOVER SPACE ... TO is performed. To adjust a table or index to the corresponding metadata in the catalog space, the table or index must be deleted and then rebuilt.

RECOVER SPACE ... TO with a replication always means that these spaces and, if applicable, dependent index spaces are placed in the state “copy pending” (see also the “[Core manual](#)”, section “Replication of a Database”).

replication_description ::= =

Replication or partial replication which is to repair, reset or create an independent original.

REPLICATION *replication*

replication is the physical name of the replication.

PASSWORD *password*

BS2000 password of the replication. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

ON USER_ID *bs2000_user_id*

User ID of the replication. You must specify the user ID without a “\$”.

COPY

The specified spaces of the replication are copied onto the catalog spaces. If the replication spaces have to be copied into a user ID other than the DBH user ID, you must ensure that all the necessary files have already been created there.

Alternatively, you can also define the DBH user ID as co-owner for the catalog files in the DB user ID.

The storage group specifications for the user space are ignored.

The replication can continue to be used and updated for retrieval applications. The replication itself is not updated in this RECOVER.



If COPY is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication user spaces continue to be available for use. You can rerun the utility statement once the cause of the problem has been remedied (e.g. the missing log files are made available). If the cause cannot be remedied (e.g. a space has been reset in the original), the spaces must be repaired using SESAM backup copies or reset using RECOVER SPACE ... TO REPLICATION.

RENAME

The affected spaces of the specified replication are renamed to the catalog spaces. These files must not be available in the original, otherwise it will not be possible to rename them. After rename, the spaces will be missing from the replication. They can be added again using REFRESH SPACE.

RENAME can only be used if the replication and the catalog are located on the same BS2000 user ID. If this user ID is different to the DBH user ID, the DBH user ID must be defined as co-owner for all the files of the replication and the catalog in the DB user ID. The replication may not be added in any SQL database catalog list.



If RENAME is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication spaces will no longer be available. The spaces must then be repaired using SESAM backup copies.

SCOPE PENDING

If you specify this clause the spaces will be repaired if they cannot be opened for update processing, because, for example, they have previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Spaces containing a table which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

SCOPE PENDING is only permitted in the case of RECOVER SPACE ... USING.

NO INDEX

This clause identifies indexes as defective and as not having been rebuilt if they have become invalid as a result of resetting a space. Indexes become invalid if the base table and an associated index are located on different user spaces or if the table is partitioned and not all the user spaces affected are reset simultaneously.

NO INDEX is only permitted with RECOVER SPACE ... TO.

You can find further information about the concept and application in the [“Core manual”](#).

RECOVER CATALOG_SPACE

The RECOVER CATALOG_SPACE statement may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

This section describes the following RECOVER statements:

- RECOVER of a catalog space using SESAM backup copies or foreign copies
- RECOVER of a catalog space using replications

RECOVER CATALOG_SPACE using SESAM backup copies or foreign copies

```
RECOVER CATALOG_SPACE catalog [ { USING rec_unit
                                TO rec_unit } ]
```

```
[PASSWORD password]
```

```
rec_unit ::= { COPY_FILE file
               COPY_NUMBER number
               timestamp_literal
               FOREIGN COPY }
```

```
timestamp_literal ::= TIMESTAMP 'year-month-day hour:minute:second'
```

RECOVER CATALOG_SPACE *catalog*

Name of the database whose catalog space is to be fully repaired.

In the case of RECOVER CATALOG_SPACE *catalog* and RECOVER CATALOG_SPACE *catalog* USING, the following conditions must be fulfilled for the catalog space:

- The CAT-REC file must exist.
- In the case of a SESAM backup, the required SESAM backup copy of the catalog space must exist and must be entered in the CAT-REC file.
- The CAT-LOG files created after the required SESAM backup copy or the foreign copy of the catalog space and entered in the CAT-REC file must exist.

- If a foreign copy is to be used then the restart point must still be present in the CAT-REC file. You can use any CAT-REC file state that was created during or after the generation of the foreign copy of the catalog space. The catalog space is repaired to the state of the CAT-REC file.

RECOVER CATALOG_SPACE *catalog* without USING or TO

To perform the repair of the catalog space, SESAM/SQL uses the SESAM backup copy of the catalog space indicated by the last entry in the CAT-REC file (i.e. the last SESAM backup copy created) and the following CAT-LOG files.

USING *rec_unit*

To perform repair, SESAM/SQL uses the SESAM backup copy of the catalog space indicated by *rec_unit* or the foreign copy and the CAT-LOG files created after the backup copy.

Repair of the catalog space is only possible with the current CAT-REC file. A RECOVER CATALOG_SPACE ... USING on the basis of an older CAT-REC file corresponds to a reset of the catalog space to the state of the used CAT-REC file.

If a foreign copy is to be used then the catalog space must have been present in logging at the time this foreign copy was created. The restart point in the CAT-REC file must still be present. Logging must not have been interrupted since the foreign copy was created.

If the catalog space was not present in logging at the time the foreign copy was created then only RECOVER CATALOG_SPACE ... TO FOREIGN COPY is possible.

TO *rec_unit*

SESAM/SQL performs a reset to the specified backup of the catalog space.



CAUTION!

If the catalog space is reset separately then subsequent access to the user spaces is generally impossible because their modification time stamp no longer corresponds to that recorded in the catalog space.

RECOVER CATALOG_SPACE ... TO leaves the user spaces unchanged and these have to be repaired separately.



CAUTION!

You cannot undo a RECOVER TO.

All the records following the record referencing *rec_unit* are deleted from the CAT-REC file. It is not possible to achieve a more recent state than that identified by *rec_unit*.

However, before performing the reset, you can back up the CAT-REC file, the CAT-LOG files and the DA-LOG files outside of SESAM/SQL.

rec_unit ::=

Backup copy that is to be used for repair and reset. Alongside SESAM backup copies, you can also use foreign copies.

COPY_FILE *file*

File name of a SESAM backup copy created with the utility statement COPY. The file must be entered in the CAT-REC file.

You must specify *file* as an alphanumeric literal.

COPY_NUMBER *number*

Version number of a SESAM backup copy created with the utility statement COPY. *number* is an integer with a maximum of six digits.

The SESAM backup copy indicated by *number* must be entered in the CAT-REC file.

*timestamp_literal*¹

Time stamp which indicates the time of a SESAM backup copy. The copy must be entered in the CAT-REC file.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

FOREIGN COPY

If this clause is specified then no SESAM backup copies are read in. Only modifications logged in the log files are applied in the case of USING. It is up to the user to copy the files of the foreign copy onto the relevant spaces. The database must be closed while the foreign copy is read in.

For information on creating a foreign copy, refer to the [“Core manual”](#).

PASSWORD *password*

BS2000 password You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

You use **PASSWORD** *password* to specify the password for all the SESAM backup copies that SESAM/SQL requires during execution of the RECOVER statement.

¹ **TIMESTAMP** '*year-month-day hour:minute: second*' is described in more detail in the manual [“SQL Reference Manual Part 1: SQL Statements”](#) under “Time stamp”.

You must always specify the BS2000 password for a SESAM backup copy if the BS2000 password of the database was changed after the SESAM backup copy was created, i.e. if the BS2000 passwords for the database and the SESAM backup copy are no longer the same.

RECOVER CATALOG_SPACE using replications

```
RECOVER CATALOG_SPACE catalog { USING replication_description
                                TO replication_description }
```

replication_description ::=

```
REPLICATION replication
[PASSWORD password]
[ON USER_ID bs2000_user_id]
[WITH CATREC catrec_file
 [PASSWORD password] [ON USER_ID bs2000_user_id]]
{ COPY
  RENAME }
```

RECOVER CATALOG_SPACE *catalog*

Name of the database whose catalog space is to be recreated, reset or repaired using the replication. An entry must exist for *catalog* in the SQL database catalog.

Precondition for RECOVER CATALOG_SPACE *catalog* USING:

- The CAT-REC file must exist.
- The replication must be available.
- All CAT-LOG files created after the replication of the catalog space and recorded in the CAT-REC file must be available.
- The catalog must be in logging.

USING *replication_description*

SESAM/SQL places the catalog space in the state described in the specified CAT-REC file.

The WITH CATREC clause must be specified.

TO *replication_description*

SESAM/SQL resets the catalog space to the state of the specified replication or recreates the database in this state.

The WITH CATREC clause may not be specified.

**CAUTION!**

If the catalog space is recreated or reset, the spaces will still be missing or the spaces will not generally match the original any more. They can be (re)created with RECOVER SPACE *space-list* TO REPLICATION.

replication_description ::=

Replication or partial replication that is to be used for repair or reset.

REPLICATION *replication*

replication is the physical name of the replication.

PASSWORD *password*

BS2000 password of the replication. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n is an integer between - 2147483648 and + 2147483647.

ON USER_ID *bs2000_user_id*

User ID of the replication. You must specify the user ID without a "\$".

WITH CATREC *catrec_file*

CAT-REC file or CAT-REC copy of the original. You may not specify any BS2000 user ID here. The specified file must be derived from the database which was used to create the replication and may not be older than the replication relating to the last REFRESH REPLICATION.

PASSWORD *password*

BS2000 password of the CAT-REC file You must enter *password* as an alphanumeric literal (see above "PASSWORD *password*").

ON USER_ID *bs2000_user_id*

Identification on which CAT-REC and log files are located. You must specify the user ID without a "\$".

COPY

The catalog space of the specified replication is copied into the catalog space file. If the catalog space is not located on the DBH user ID, the file must already be created. Alternatively, you can also define the DBH user ID as co-owner for the catalog files in the DB user ID.

You may not add the replication to the SQL database catalog list with `STATUS=ACTIVE`.

The replication can continue to be used and updated for retrieval applications. The replication itself is not updated in this `RECOVER`.

The storage group specifications for the catalog space and the user spaces are ignored.



If `COPY` is rejected for one of the reasons that prevents a `REFRESH REPLICATION`, the replication continues to be available for use.

You can rerun the utility statement once the cause of the problem has been remedied (e.g. the missing log files are made available). If the cause cannot be remedied (e.g. a space has been reset in the original), the database must be repaired using the `SESAM` backup copies or it can be reset using `RECOVER CATALOG_SPACE ... TO REPLICATION`.

RENAME

The catalog space of the specified replication is renamed to the name of the catalog space. The catalog space must not be available in the original, otherwise it will not be possible to rename it.

After rename, the original catalog space will be missing in the replication and the whole replication must be recreated using `CREATE REPLICATION`.

`RENAME` can only be used if the replication and the catalog are located on the same `BS2000` user ID. If this user ID is different to the `DBH` user ID, the `DBH` user ID must be defined as co-owner for all the files of the replication and the catalog in the `DB` user ID. The replication may not be added in any `SQL` database catalog list.



If `RENAME` is rejected for one of the reasons that prevents a `REFRESH REPLICATION`, the catalog space of the replication will no longer be available, thus rendering the replication defective. The database must then be repaired using the `SESAM` backup copies and the replication must be recreated.

RECOVER CATALOG

The RECOVER CATALOG statement may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

This section describes the following RECOVER statements:

- RECOVER of a catalog using SESAM backup copies or foreign copies
- RECOVER of a catalog using replications

RECOVER CATALOG using SESAM backup copies or foreign copies

```
RECOVER CATALOG catalog [ { USING rec_unit
                             TO rec_unit
                             [USING rec_unit] TO [ANY] timestamp_literal } ]
[PASSWORD kennwort] [SCOPE PENDING] [GENERATE INDEX ON NO LOG INDEX SPACE]
```

$$rec_unit ::= \left\{ \begin{array}{l} COPY_FILE \textit{file} \\ COPY_NUMBER \textit{number} \\ \textit{timestamp_literal} \\ FOREIGN COPY \end{array} \right\}$$

$$timestamp_literal ::= \text{TIMESTAMP 'year-month-day hour:minute:second'}$$

RECOVER CATALOG *catalog*

Name of the database which is to be repaired as a whole with all the user spaces.

The following prerequisites apply for RECOVER CATALOG if RECOVER CATALOG *catalog* TO *rec_unit* was not used to reset to a full database backup:

- The CAT-REC file must exist.
- In the case of a SESAM backup, the required SESAM backup copy of the catalog space must exist and must be entered in the CAT-REC file.
- All the CAT-LOG files, which are created after the required SESAM backup copy or the foreign copy of the catalog space and that are entered in the CAT-REC file, must exist.
- In the case of a SESAM backup, the required SESAM backup copies of all the user spaces which have been entered in the catalog table RECOVERY_UNITS must exist.

- All the DA-LOG files created after the required SESAM backup copies or the foreign copy of the individual user spaces and which are entered in the DA_LOG catalog table must exist.
- If a foreign copy is to be used then the restart point must still be present in the CAT-REC file. You can use any CAT-REC file state that was created during or after the generation of the foreign copy of the catalog space. The catalog space is repaired to the state of the CAT-REC file.
- Logging of the user spaces must not have been interrupted since the required SESAM backup copy or the foreign copy was created (e.g. with LOAD OFFLINE, MIGRATE).

Spaces that are not subject to logical data saving are unaffected by a RECOVER CATALOG. They must then be retrieved separately. Index spaces represent an exception here. They can be retrieved using GENERATE INDEX ON NO LOG INDEX SPACE. In this case the indexes are rebuilt.

RECOVER CATALOG *catalog* without a USING or TO clause

SESAM/SQL initially repairs the catalog space of the database *catalog* using the last SESAM backup copy of the catalog space created and the CAT-LOG files created after this SESAM backup copy. Once the catalog space has been successfully repaired, SESAM/SQL repairs all the user spaces for which logging was activated using the last SESAM backup copy created. SESAM/SQL doesn't treat all those user spaces for which logging has not been activated, and they must individually be repaired or reset later if they were defective.

USING *rec_unit*

SESAM/SQL repairs the catalog space of the database *catalog* using the SESAM backup copy of the catalog space indicated by *rec_unit* or on the basis of the foreign copy of the catalog space and the CAT-LOG files created after this backup copy. The subsequent repair of the user spaces is performed just as it is if you specify RECOVER CATALOG *catalog* without the USING clause. However, in the case of a foreign copy (*rec_unit* = FOREIGN_COPY), no user spaces are read in.

TO *rec_unit*

SESAM/SQL resets the catalog space to the state of the backup specified by *rec_unit*. The last backups of the user spaces recorded in the catalog space are then read in.

If you use a backup in which the user spaces were saved before the catalog space then the possibly existing modifications logged in the logging files will be applied to the spaces. If, in the case of a foreign copy, the user spaces were backed up after the catalog space then their restart information will not be present in the DA_LOGS table. A repair is not possible then.

TO ANY *timestamp_literal*¹

SESAM/SQL resets the entire database to the state at the time specified.

Transactions which were open at this time are rolled back. The time specified is freely selectable. However, it may not be a time before the catalog was created (CREATE CATALOG). It may also not be in the critical hour when the switchover from daylight savings time to winter time takes place.

SESAM/SQL reads in the last SESAM backup copy of the catalog space which was made before this time and applies the changes to the CAT-LOG files created after this SESAM backup copy up to the time specified. The last backups of the user spaces recorded in the catalog space are then read in. The changes to these spaces are also applied up to the time specified using the DA-LOG files.

No user spaces for which logging has not been activated are treated, and they must be individually repaired or reset later if they were defective.

USING *rec_unit* **TO** [ANY] *timestamp_literal*

SESAM/SQL resets the entire database to the state at the time specified.

Transactions which were open at this time are rolled back. The basis here is the SESAM backup copy or foreign copy of the catalog space indicated by *rec_unit*. If ANY is specified, the time specified is freely selectable, otherwise *timestamp_literal* must specify the time of a SESAM backup copy of the catalog space. However, it may not be a time before the catalog was created (CREATE CATALOG) It may also not be in the critical hour when the switchover from daylight savings time to winter time takes place.

The backup specified with **USING** *rec_unit* must be older than the timestamp specified with **TO** [ANY] *timestamp_literal*.

SESAM/SQL reads in the specified SESAM backup copy of the catalog space and applies the changes to the CAT-LOG files created after this SESAM backup copy up to the time specified.

¹ **TIMESTAMP** 'year-month-day hour:minute:second' is described in more detail in the manual "[SQL Reference Manual Part 1: SQL Statements](#)" under "Time stamp".

The subsequent repair of the user spaces is performed in the same way as if you specify RECOVER CATALOG TO ANY *timestamp_literal* without the USING clause. However, in the case of a foreign copy (*rec_unit* = FOREIGN_COPY), no user spaces are read in.

**CAUTION!**

You cannot undo a RECOVER TO statement (in none of the three cases described above).

All the records following the record referencing *rec_unit* are deleted from the CAT-REC file. The corresponding CAT-LOG files are overwritten. It is not possible to achieve a more recent state than that identified by *rec_unit*.

However, before performing the reset, you can back up the CAT-REC file, the CAT-LOG files and the DA-LOG files outside of SESAM/SQL.

rec_unit ::=

Backup copy to be used for repair or reset.

Alongside SESAM backup copies, it is also possible to use foreign copies.

COPY_FILE *file*

File name of a SESAM backup copy of the catalog space created with the utility statement COPY. The file must be entered in the CAT-REC file. You must specify *file* as an alphanumeric literal.

COPY_NUMBER *number*

Version number of a SESAM backup copy of the catalog space created with the utility statement COPY. *number* is an integer with a maximum of six digits. The SESAM backup copy indicated by *number* must be entered in the CAT-REC file.

timestamp_literal

Time stamp designating the time of a SESAM backup copy of the catalog space. The copy must be entered in the CAT-REC file.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

FOREIGN COPY

This clause specifies that no SESAM backup copies are read in. The foreign copy files must be copied onto the relevant spaces before launching the RECOVER statement. The catalog must be closed while the foreign copy is read in. The modifications logged in the logging files are applied in the case of USING. For information on creating a foreign copy, refer to the [“Core manual”](#).

PASSWORD *password*

BS2000 password You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

You use **PASSWORD** *password* to specify the password for all the SESAM backup copies that SESAM/SQL requires during execution of the RECOVER statement.

You must always specify the BS2000 password for a SESAM backup copy if the BS2000 password of the database was changed after the SESAM backup copy was created, i.e. if the BS2000 passwords for the database and the SESAM backup copy are no longer the same.

SCOPE PENDING

If you specify this clause the catalog space will first be repaired. The spaces will be repaired if they cannot be opened for update processing because, for example, they had previously been set as defective because of an error, an inconsistency is detected between catalog and space or a DMS error is issued when opening the file. Spaces containing a table which has been marked as defective during previous error processing, will be repaired. Formal correctness is not checked. A space containing only indexes is not repaired if one or more indexes are defective.

**CAUTION!**

If **SCOPE PENDING** is specified in the case of **RECOVER CATALOG TO**, the catalog space will always be reset. The procedure for the user spaces is the same as that described in **RECOVER SPACE TO**.

SCOPE PENDING is not permitted in the case of foreign copies.

GENERATE INDEX ON NO LOG INDEX SPACE

No backups of spaces that contain only indexes and are not present in the logical data backup are read in. These spaces are reset and the indexes are rebuilt.

When **SCOPE PENDING** is specified at the same time, indexes are only built if the index space cannot be opened. Defective indexes on the space are tolerated and do not cause the index space to be rebuilt.

RECOVER CATALOG using replications

```
RECOVER CATALOG catalog { USING replication_description
                          TO replication_description }
```

```
[GENERATE INDEX ON NO LOG INDEX SPACE]
```

```
replication_description ::=
```

```
  REPLICATION replication
  [PASSWORD password]
  [ON USER_ID bs2000_user_id]
  [WITH CATREC catrec_file
   [PASSWORD password] [ON USER_ID bs2000_user_id]]
  { COPY
    RENAME }
```

RECOVER CATALOG *catalog*

Name of the database to be recreated in full or partially, reset or repaired using the (partial) replication. An entry must exist for *catalog* in the SQL database catalog.

Preconditions for RECOVER CATALOG *catalog* USING:

- The CAT-REC file must exist.
- The replication must be available.
- All the CAT-LOG files that were created after the replication of the catalog space and are entered in the CAT-REC file must exist.
- All the DA-LOG files that were created after a replication and are entered in the DA-LOGS file must exist.
- If the USING clause is used to apply remaining modifications logged in the log files, the REFRESH REPLICATION restrictions (see [page 219](#)) take effect.

USING *replication_description*

SESAM/SQL repairs the database to the state which is described in the specified CAT-REC file.

The WITH CATREC clause must be specified.

TO *replication_description*

SESAM/SQL resets the database to the state of the specified replication or recreates the database in this state.

The WITH CATREC clause may not be specified.

**CAUTION!**

You cannot undo a RECOVER TO.

All the records following the record referencing *rec_unit* are deleted from the CAT-REC file. The corresponding CAT-LOG files are overwritten. It is not possible to achieve a more recent state than that identified by *rec_unit*.

However, before performing the reset, you can back up the CAT-REC file, the CAT-LOG files and the DA-LOG files outside of SESAM/SQL.

replication_description ::=

Replication or partial replication which is to repair, reset or create an independent original. In the case of a partial replication, only those user spaces are repaired, reset or created which are current replication spaces in the partial replication (see the “[Utility Monitor](#)” manual, mask COP.4.6). The other spaces must be reset or repaired using RECOVER SPACE.

REPLICATION *replication*

replication is the physical name of the replication.

PASSWORD *password*

BS2000 password of the replication. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n is an integer between - 2147483648 and + 2147483647.

ON USER_ID *bs2000_user_id*

User ID of the replication. You must specify the user ID without a “\$”.

WITH CATREC *catrec_file*

CAT-REC file or CAT-REC copy of the original. You may not specify any BS2000 user ID here. The specified file must be derived from the database which was used to create the replication and may not be older than the replication relating to the last REFRESH REPLICATION.

PASSWORD *password*

BS2000 password of the CAT-REC file You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n is an integer between - 2147483648 and + 2147483647.

ON USER_ID *bs2000_user_id*

Identification on which CAT-REC and log files are located. You must specify the user ID without a "\$".

COPY

The specified replication spaces are copied onto the spaces of the (defective) catalog. If you need to copy the replication into a user ID other than the DBH user ID, all the necessary files must have already been created. Alternatively, you can also define the DBH user ID as co-owner for the catalog files in the DB user ID. The storage group specifications for the users are ignored.

You may not add the replication to the SQL database catalog list with STATUS=ACTIVE.

The replication can continue to be used and updated for retrieval applications. The replication itself is not updated in this RECOVER.



If COPY is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication continues to be available for use.

You can rerun the utility once the cause of the problem has been remedied (e.g. missing log files are made available). If the cause cannot be remedied (e.g. a space has been reset in the original), the database must be repaired using SESAM backup copies or it can be reset using RECOVER CATALOG ... TO REPLICATION.

RENAME

The specified replication is renamed to the name of the original catalog. The original must not be available any more, otherwise it will not be possible to rename it. After rename, the original replication will be missing and must be recreated using CREATE REPLICATION.

RENAME can only be used if the replication and the catalog are located on the same BS2000 user ID. If this user ID is different to the DBH user ID, the DBH user ID must be defined as co-owner for all the files of the replication and the catalog in the DB user ID. The replication may not be added in any SQL database catalog list.



If RENAME is rejected for one of the reasons that prevents a REFRESH REPLICATION, the replication will no longer be available. The database must then be repaired using SESAM backup copies.

GENERATE INDEX ON NO LOG INDEX SPACE

Spaces which contain only indexes and are not part of the logical data backup cannot be included in the replication. The index spaces are reset and the indexes are created again.

If a new independent database has been created, the SESAM backup copies of the original cannot be accessed. For this reason, you must reassign the storage group specifications in the new database, create a new copy of the database and delete the metadata of the old SESAM backup copies.

Rebuild indexes

You can use RECOVER INDEX to rebuild indexes. You can rebuild

- an individual index
- all the indexes that belong to a certain base table
- all the indexes that belong to the base tables in a certain user space

If a user space for which logging has been activated contains an index that is to be rebuilt with RECOVER INDEX, RECOVER INDEX interrupts logging for this user space. After execution of RECOVER INDEX, the user space is placed in the state “copy pending” (see [page 13](#)). You must make a SESAM backup copy of the user space with COPY since the newly built index is the starting point for further logging.

In order to execute RECOVER INDEX (without specification of ON TABLE or ON SPACE), the current authorization identifier must have the special privilege UTILITY or must own the schema to which the index is assigned.

In order to execute RECOVER INDEX ON TABLE *table*, the current authorization identifier must have the special privilege UTILITY or must own the schema to which *table* belongs.

In order to execute RECOVER INDEX ON SPACE *space*, the current authorization identifier must have the special privilege UTILITY or must own the user space *space*.

$$\text{RECOVER INDEX } \left\{ \begin{array}{l} \textit{index} \\ \text{ON TABLE } \textit{table} \\ \text{ON SPACE } \textit{space} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{SCOPE ALL} \\ \text{SCOPE PENDING} \end{array} \right\} \right]$$

RECOVER INDEX *index*

Name of the index to which the statement is to apply. The index must already exist.

ON TABLE *table*

Name of a base table. The statement applies to all the indexes defined for this base table.

In the case of a partitioned table all partitions must be available (see [table 3 on page 22](#)).

ON SPACE *space*

Name of a user space. The statement applies to all the indexes defined for tables in this user space.

SCOPE ALL

SESAM/SQL rebuilds all the specified indexes.

Because of the space needed for sorting when the indexes are rebuilt, it is a good idea only to use SCOPE ALL when absolutely necessary.

SCOPE PENDING

Of the indexes specified, SESAM/SQL only rebuilds those that are not available.

Examples

1. In the example below the database ORDERCUST is repaired.

```
RECOVER CATALOG ordercust
```

The database is repaired using the last SESAM backup copy of the catalog space created and the last created SESAM backup copies of the user spaces TABLESPACE and BLOBSPACE.

2. In the example below the database ORDERCUST is repaired.



```
RECOVER CATALOG ordercust USING COPY_NUMBER 2
```

First of all the catalog space is repaired using the SESAM backup copy with version number 2. The user spaces TABLESPACE and BLOBSPACE are then repaired using the last SESAM backup copy to be created for each of them. The metadata of the repaired catalog space is used for the repair of the user spaces. The space INDEXSPACE is not repaired since it is not present in the logical data saving.

3. In the example below, the database ORDERCUST is reset to a previous state.



```
RECOVER CATALOG ordercust TO COPY_NUMBER 2  
GENERATE INDEX ON NO LOG INDEX SPACE
```

First of all the catalog space is reset using the SESAM backup copy with version number 2. The user spaces TABLESPACE and BLOBSPACE are then repaired on the basis of the metadata of the reset catalog space using the last SESAM backup copy to be created for each of them.

No backup of the space INDEXSPACE is read in since it contains only indexes and is not present in the logical data backup. The space INDEXSPACE is reset to a previous state and the indexes are rebuilt.

See also

COPY, CREATE/REFRESH REPLICATION, CREATE CATALOG, CREATE/DROP INDEX

REFRESH REPLICATION - Update a replication

REFRESH REPLICATION allows you to update replications:

- an entire (partial) replication by applying the modifications logged in the log files
- a selection of spaces of a (partial) replication by applying the modifications logged in the log files and thereby reduce the (partial) replication accordingly

If the replication can no longer be updated as a whole with REFRESH REPLICATION (e.g. because logging is interrupted, see below), REFRESH REPLICATION FOR SPACE must be used.

REFRESH REPLICATION FOR SPACE updates the catalog space of the replication and all the specified user spaces of the replication. User spaces of the replication that are not specified are logically deleted from the replication (these are known as “previous replication spaces”). The replication thus becomes a partial replication of the original replication (see also “[Core manual](#)”, section “Updating and extending the replication”).

You can find information about current and previous replication spaces via the utility monitor, mask COP.4.6 (see “[Utility Monitor](#)” manual).

The following rules apply to updating a replication:

- If a replication contains a partitioned table, all the partitions of the table must be contained in the replication. A partial replication must always be complete in itself, i.e. the partial replication must contain all the indexes for all the tables which belong to the partial replication.
- All replication spaces must be run in the original with logging and logging must not have been interrupted.

Reasons for an interruption to logging are:

- explicit switching off of the logical data backup by ALTER SPACE NO LOG
- implicit interruption of logical data backup by the utility statements LOAD OFFLINE, MIGRATE, IMPORT TABLE, RECOVER INDEX, RECOVER TO
- If user spaces were deleted from the original database (DROP SPACE), the corresponding replication spaces must be removed from the replication. They are not specified in REFRESH REPLICATION FOR SPACE for this reason. This also applies if the spaces have been created again in the original under the same name.
- Spaces in the original database may not have an older state than those of the replication (e.g. after resetting a space). The corresponding replication spaces must be removed from the replication. They are not specified in REFRESH REPLICATION FOR SPACE for this reason.

- The CAT-LOG and DA-LOG files of the original must be available as of the state of the replication. The DBH must be able to access them. The job variable *userid.SESAM.replication.NEXT-REPL-LOG* contains the number and sub-number of the first files to be made available.
- There must be a CAT-REC file of the original database available which is more up to date than the CAT-REC file of the replication. This can be created using COPY CATALOG, COPY CATALOG_SPACE or the administration statement CHANGE-CATALOG.
- It is not possible to access the replication during execution of the statement. Nor may it be added in the SQL database catalog list of any other DBH and opened there.

The REFRESH REPLICATION statement may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

In the case of REFRESH REPLICATION and REFRESH REPLICATION FOR SPACE, the following backup units of the replication are updated one after the other:

- CAT-REC file
- Catalog space
- User spaces

The following referential constraints apply during a REFRESH REPLICATION or REFRESH REPLICATION FOR SPACE statement:

- The replication including all spaces, the CAT-REC file of the replication and the CAT-REC copy of the original are exclusively locked.
- It is not possible to access the replication.
- CDML applications with the status 9U/9R are rejected.

If the statement is aborted due to missing log files, you can finish updating the replication by executing the statement again once the files have been made available. If the statement is aborted because the DBH is terminated or the DBH is performing an internal restart, it is normally necessary to recreate the replication.

```
REFRESH REPLICATION replication [FOR SPACE space[, space ...]]  
FROM CATREC catrec_file  
[PASSWORD password]  
[ON USER_ID bs2000_user_id]
```

REFRESH REPLICATION *replication*
Name of replication to be updated.

FOR SPACE *space*

Name or names of the user spaces of the replication to be updated. If a list is specified, each specified name may only be specified once.

You can specify up to 999 space names. You cannot specify the catalog space as a user space.

User spaces that are available in the replication and that are not specified in FOR SPACE are logically deleted from the replication.

FROM CATREC *catrec_file*

Name of the CAT-REC file or CAT-REC copy, which describes the state to which the replication is to be updated. The name may not contain a BS2000 user ID.

The CAT-REC file must be derived from the same original database as the replication. You can either use CHANGE-CATLOG or COPY or the BS2000 command COPY-FILE to produce the file.

If *catalog.CAT-REC[.COPY]* is specified as the CAT-REC file name, the logging files must also be present under the names *catalog.version.C.nnnn* and *catalog.version.D.nnnn*.

If *catalog.CAT-REC* (without *.COPY*) is specified, the original database may not be open.

PASSWORD *password*

BS2000 password for the CAT-REC file You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

password is valid for the CAT-REC file and the associated CAT-LOG and DA-LOG files of the original database.

ON USER_ID *bs2000_user_id*

Name of the BS2000 user ID on which the CAT-REC copy is stored, provided it is not located on the DBH user ID. The CAT-LOG and DA-LOG files must also be stored on this user ID. The files must be shareable and assigned write permission. You must specify the user ID without a "\$".

If the CAT-REC copy is located on the DBH user ID, no user ID will be specified.

Examples

1. The example below updates the replication ORDERREP with the current state of the ORDERCUST database.



```
REFRESH REPLICATION orderrep
FROM CATREC 'ORDERCUST.CAT-REC.COPY'
```

2. The example below updates the user space TABLESPACE in the replication ORDERREP032 with the current state of the database ORDERCUST. The only other replication space BLOBSpace is logically deleted from the replication because it is not specified.



```
REFRESH REPLICATION orderrep032 FOR SPACE tablespace
FROM CATREC 'ORDERCUST.CAT-REC.COPY'
```

See also

CREATE REPLICATION, REFRESH SPACE, RECOVER USING/TO REPLICATION

REFRESH SPACE - Extend a replication

REFRESH SPACE allows you to recreate replication spaces from a backup copy and update them to the current state of the replication by applying the modifications logged in the log files.

If replication spaces have been deleted from the replication (e.g. following a temporary interruption to logging in the original) or recreated in the original, you can add these spaces to the (partial) replication using REFRESH SPACE (see also “[Core manual](#)”, section “Updating and extending the replication”).

REFRESH SPACE does not check whether specified spaces are already contained in the replication. Spaces that already exist are overwritten. If a replication space is recreated, the medium of the replication catalog space will be used.

You can find information about current and previous replication spaces via the utility monitor, mask COP.4.6 (see “[Utility Monitor](#)” manual).

The following rules apply to extending a replication:

- All spaces must be run in the original with logging and the logging procedure must not have been interrupted.
- The DBH must be able to access the backup copies used. The replication must be updated to such an extent that the backup is known there.
- If several spaces are specified in a statement, all backups must have the same time stamp.
- All DA-LOG files created from the time of backup up to the current state of the replication must be made available. The DBH must be able to access them. You can determine which files are required via the RECOVERY_UNITS in the INFORMATION_SCHEMA.
- It is not possible to access the replication during execution of the statement. Nor may it be added in the SQL database catalog list of any other DBH and opened there.

In the case of REFRESH SPACE, the specified user spaces are added to a replication. This does not affect the state of the replication in terms of how up to date it is.

In order to execute REFRESH SPACE, the current authorization identifier must have the special privilege UTILITY or must own all the affected user spaces.

The following referential constraints apply during a REFRESH SPACE statement:

- The replication including all spaces and the CAT-REC file of the replication are exclusively locked.
- It is not possible to access the replication.
- CDML applications with the status 9U/9R are rejected.

If a statement is aborted because of missing log files, you can rerun expansion of the replication by executing the statement again once the files have been made available. If a statement aborts because the DBH is terminated or because the DBH is performing an internal restart, it is usually necessary to recreate the replication.

```
REFRESH SPACE space[, space ...]
```

```
  USING rec_unit
```

```
  [PASSWORD password]
```

```
  [ON USER_ID bs2000_user_id]
```

$$rec_unit ::= \left\{ \begin{array}{l} COPY_FILE \textit{file} \\ COPY_NUMBER \textit{number} \\ \textit{timestamp_literal} \\ FOREIGN_COPY \end{array} \right\}$$

```
timestamp_literal ::= TIMESTAMP 'year-month-day hour:minute:second'
```

```
REFRESH SPACE space[, space ...]
```

Name or names of the user spaces, which are to be added to the replication. You can specify up to 999 space names.

USING *rec_unit*

SESAM backup copy or foreign copy specification, on the basis of which the replication spaces are to be recreated and brought up to the state of the replication by applying the modifications logged in the log files.

COPY_FILE *file*

File name of a SESAM backup copy created with the utility statement COPY. You must specify *file* as an alphanumeric literal.

The copy file must be older than the current state of the replication.

You may only specify COPY_FILE when adding a single space.

COPY_NUMBER *number*

Version number of a SESAM backup copy created with the utility statement COPY. *number* is an integer with a maximum of six digits. The copy number must be known in the RECOVERY_UNITS table of the replication. You may only specify COPY_NUMBER when adding a single space.

*timestamp_literal*¹

Time stamp indicating the time of SESAM backup copies. The copies must be known in the RECOVERY_UNITS table of the replication.

You can also specify *timestamp_literal* when adding several spaces specified in a space list. The copies must have the same time stamp in this case.

For more information about the time of copy creation, refer to the utility statement [“COPY - Create a SESAM backup copy” on page 97](#).

FOREIGN COPY

When FOREIGN COPY is specified, only logging information is applied and no SESAM backup copies are read in. It is up to the user to copy the files of the foreign copy onto the relevant replication spaces. The space must be closed while the foreign copy is read in. For information on creating a foreign copy, refer to the [“Core manual”](#).

The foreign copy must be older than the current state of the replication.

You can also specify FOREIGN COPY when adding several spaces specified in a space list. The foreign copies must have the same time stamp in this case.

¹ **TIMESTAMP** 'year-month-day hour:minute:second' is described in more detail in the manual [“SQL Reference Manual Part 1: SQL Statements”](#) under “Time stamp”.

PASSWORD *password*

Password under which the SESAM backup was created. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

ON USER_ID *bs2000_user_id*

Name of the BS2000 user ID on which the backup or one of the required log files is located. These files are searched for firstly in this user ID and then in the DBH user ID. If this clause is not specified, all files must be available in the DBH user ID.

Example

The example below recreates the replication space BLOBSPACE of ORDERREP032 from the SESAM backup with the version number 1.



```
REFRESH SPACE orderrep032.blobspace  
USING COPY_NUMBER 1
```

See also

CREATE/REFRESH REPLICATION, RECOVER USING/TO REPLICATION

REORG - Reorganizing spaces and base tables

You use REORG to reorganize the catalog space, individual user spaces or also individual base tables on user spaces.

Reorganization means that SESAM/SQL stores logically related storage areas in physically contiguous storage areas. During reorganization, SESAM/SQL takes into account any free space reservation defined earlier using the PCTFREE clause in the statement CREATE CATALOG for the catalog space or in the SQL statements CREATE SPACE or ALTER SPACE for a user space.

REORG SPACE also covers REORG STATISTICS (see the [“SQL Reference Manual Part 1: SQL Statements”](#) manual).

You can also use REORG SPACE to move a user space to a new storage group.

REORG ONLINE TABLE enables an individual (partitioned) base table or an individual partition of a partitioned table to be reorganized. REORG ONLINE TABLE can be executed for this table parallel to other statements and does not require any exclusive locks.

REORG [CATALOG_]SPACE - Reorganize the catalog space and user spaces

Databases on a DB user ID are treated like databases on the DBH user ID, provided the DBH user ID is defined as co-owner of the database files in the DB user ID.

If the DBH user ID is not co-owner of the database files, you must specify COPY in order to reorganize databases on a DB user ID.

SESAM/SQL carries out the reorganization of a user space in two phases:

- in the first phase the work file is set up with the reorganized blocks of the space. During this phase, the user has read access to the space by means of DML statements.
- in the second phase the work file is renamed with the name of the space (RENAME) or copied to the space (COPY). During this phase the space cannot be accessed at all. The time during the second phase when the space cannot be accessed can be kept to a minimum if the user prevents the space from being copied. This is possible by explicitly specifying RENAME or by not specifying any work file.

It is not possible to access a catalog space which is in the process of being reorganized. During reorganization of a user space on the other hand, the user space can still be accessed in the first phase of the reorganization.

REORG SPACE is also allowed for replications.

- Only the user spaces of a replication can be reorganized, not the catalog space.
- The space is exclusively locked throughout the entire reorganization.
- As the storage group definitions in the metadata apply to the original and not generally to the replication, the metadata are not evaluated for creating the standard work file in REORG [CATALOG_]SPACE of replications with RENAME, but rather the standard work file is created on the medium on which the replication space is located.

You can specify a tape file as the work file during the reorganization of a user space if the space only contains tables and no indexes. In this case, reorganization is performed in three phases:

- In the first phase, all the tables of the space are exported to the work file. During this phase, DML statements can be used to access the original space.
- In the second phase, the space is logically deleted. It is not possible to access the space during this phase.
- In the third phase, all the tables are imported from the work file into the space. It is not possible to access the space during this phase.

The current authorization identifier must have the special privilege UTILITY or must own the user space to be reorganized.

```

REORG { SPACE space [ { OLD ROW_IDS } ]
      { CATALOG_SPACE catalog } } [ USING FILE file [ PASSWORD password ] ]

[ { COPY
  { RENAME } } ] [ MINIMIZE ]

```

CATALOG_SPACE *catalog*

Name of the database whose catalog space is to be reorganized.

SPACE *space*

Name of the user space to be reorganized.

NEW ROW_IDS

The row numbers of tables with a primary key are reorganized. This means that the indexes of the space's tables are now invalid. These indexes must be repaired with RECOVER INDEX. If logical data saving is activated for the space, the space subsequently has the state "copy pending".

NEW ROW_IDS is ignored for tables without a primary key. In this case, the row numbers are retained.

OLD ROW_IDS

The row numbers are taken over from the old table.

USING FILE *file* [PASSWORD *password*]

Name of the work file. You must specify *file* as an alphanumeric literal.

If a work file which does not yet exist is specified in USING FILE, SESAM/SQL will create the file on the media that apply for the standard work file under the specified name. If the filename is specified with Catid, the file will be created on the pubset designated by the Catid.

If the database is located on a DB user ID, the work file must also be created on the DB user ID for RENAME.

When you specify primary and secondary allocation for the work file, the values specified should be oriented on the corresponding values for the space (catalog space or user space) being reorganized. If SESAM/SQL is to take a larger free space reservation for the space into account during reorganization, we recommend that you select correspondingly larger values for primary and secondary allocation. In any case, the values for primary and secondary allocation for the work file should be divisible by 12.

You can use a tape file as the work file if the specified space contains only tables. Indexes cannot be reorganized via a tape file. Partitioned CALL DML tables cannot be reorganized via a tape file when NEW ROW_IDS is specified. If the space is present in the logical data backup, it has the state "copy pending" after REORG has been executed.

You can use the pair of statements

EXPORT/IMPORT TABLE to restructure a space that contains tables in such a way that the indexes are subsequently located on a separate space.

The catalog space cannot be reorganized via a tape file because it contains indexes. If the work file is on tape and RENAME is specified, the statement will be rejected.

You enter the name of the work file as an alphanumeric literal:

```
'[:catalog_id:]bs2000_user id.unqual_filename'
```

catid

BS2000 catalog ID

You must specify *catid* as an alphanumeric literal.

If a CATID list was defined then the catalog ID must be specified in the list.

catid omitted:

The default catalog ID assigned to the BS2000 user ID under which the DBH is running is used.

bs2000_userid

BS2000 user ID on which the work file is created.

bs2000_user id not specified:

The work file is created on the BS2000 user ID under which the DBH is running.

unqual_filename

Unqualified file name according to BS2000 conventions

PASSWORD *password*

BS2000 password for the work file. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

USING FILE *file* omitted:

SESAM/SQL uses a default work file with the following name:

- Name of the work file for the catalog space:
:catid:dbh_userid.catalog.CATALOG.REORG
- Name of the work file for user spaces:
:cat_id:dbh_userid.catalog.space.REORG

In the case of COPY, the standard work file is created on the default pubset of the DBH user ID.

In the case of RENAME the standard work file is created on the pubset or private volume specified in the space's storage group. This moves a space to another storage group if this group has previously been reassigned for this space with ALTER SPACE.

If the user has already created a file that fulfills the name conventions for the standard work file on a pubset which can be accessed by the DBH, this file will be used as the standard work file.

If the database is located on a DB user ID, the work file will be created on the DB user ID provided the necessary preparations have been made, see section "Database files and job variables on foreign user IDs" in the "Core manual". If the work file cannot be created on the DB user ID, the statement will be rejected.

COPY

In the case of COPY, the reorganized work file is copied into the space file, i.e. the space retains its original position (except in the case of work files on tape). One standard work file created by SESAM/SQL or one work file specified in USING FILE, but not created by the user is deleted again. If the USING FILE specification is missing and COPY is specified, the standard work file will be created and copied in the DBH user ID.

COPY is the standard value in the USING FILE specification.

RENAME

In the case of RENAME, the existing space file is deleted and the reorganized work file is renamed into the space file, i.e. the space is now located where the work file was. RENAME is the standard value if USING FILE is not specified.

If the database is located on a DB user ID, the DBH user ID must be defined as co-owner of the files in the DB user ID. If, in this case, the work file only has been created as shareable by the user in the DB user ID, but the DBH is not co-owner, the REORG statement will only fail when the renaming is executed.

MINIMIZE

When MINIMIZE is specified, the storage space of the space file no longer required after reorganization is released.

Example

The example below reorganizes the space TABLESPACE. The file DAT.REORG.TABLE is used as the work file.



```
REORG SPACE tablespaces USING FILE 'DAT.REORG.TABLE'
```


REORG ONLINE TABLE - Reorganize a base table

SESAM/SQL reorganizes a base table by modifying and copying the blocks within the user space online.

As no exclusive transaction locks are requested for this purpose, other DML applications can both read and modify the base table.

The indexes of a base table are not affected by the reorganization.

REORG ONLINE TABLE is not permitted for the tables of replications.

The current authorization identifier must have the special privilege UTILITY or must own the schema in which the table to be reorganized is located.

```
REORG ONLINE TABLE table [ON SPACE space]
```

TABLE *table*

Name of the base table which is to be reorganized.

The table must have a primary key.

In the case of partitioned tables a single partition can also be reorganized. The partition is defined by the ON SPACE *space* clause.

ON SPACE *space*

Name of the space on which the base table or partition is located.

If this clause is not specified for a partitioned table, all the partitions of the base table are reorganized one after the other.

Example

In this example the partition of the SERVICE tables which is located on user space TABLESPACE is reorganized.



```
REORG ONLINE TABLE service ON SPACE tablespace
```

UNLOAD - Unload user data from a table

You can use UNLOAD to unload user data from a table in a SESAM/SQL database into a BS2000 SAM file.

You can unload all the data from a table (UNLOAD TABLE) or unload data from selected columns of the table (UNLOAD DATA).

UNLOAD ONLINE enables you to unload user data from base tables or views, to restrict the data which is to be unloaded by means of a search condition, and to define a collation sequence for the output file.

UNLOAD OFFLINE enables you to unload user data from base tables. You can also use UNLOAD to unload data in TRANSFER_FORMAT from:

- a user space that could not be repaired FROM SPACE clause, user space is in the state “recover pending”).
- a backup copy (FROM COPY_FILE clause)

The number of loaded rows is output in the `SQLrowcount` field of the communication area of the ESQL-COBOL program.

SESAM offers several output formats, namely TRANSFER_FORMAT, LOAD_FORMAT, DELIMITER_FORMAT, CSV_FORMAT and the standard representation; it is also possible to define the representation of the individual columns yourself (see also [“UNLOAD formats” on page 253](#)). If, in the latter case, no representation is specified for certain columns, the values will be output in the standard representation which corresponds to the data type of the assigned column in the table. If there is no output format specified or the representation of certain columns is not described, all the values in the standard representation of the data type of the assigned column will be output in each case. The output file is created with coded character set name (CCSN). In DELIMITER_FORMAT, in CSV_FORMAT and in the self-defined format the data to be output is converted to the output file’s coded character set if necessary.

In conjunction with the utility statement LOAD, UNLOAD also makes it easy to transfer user data from one table to another. In this case, the output file for UNLOAD serves as the input file for a subsequent LOAD statement. You can use the four named formats for this (the specification LOAD_FORMAT in the case of UNLOAD corresponds to the specification UNLOAD_FORMAT in the case of LOAD). You should, however, be careful with the standard representation and with output formats you have defined yourself (especially if an output format has not been explicitly specified), because you must pay close attention to the treatment of NULL values (see the WHEN NULL THEN clause).

If user data is to be copied along with the table structure of a base table then the use of the pair of statements EXPORT/IMPORT TABLE may be worthwhile.

With these statements it is also possible to copy only a base table's metadata. For more detailed information on the possible uses of EXPORT/IMPORT TABLE refer to [section "EXPORT TABLE - Export a table from a database to an export file" on page 129](#) and [section "IMPORT TABLE - Import a table from an export file into a database" on page 133](#).

The current authorization identifier must have the special privilege UTILITY or must own the schema in which the table from which data is being unloaded is located.

When a search condition or collation sequence is specified, the current authorization identifier must have the table privilege SELECT for the table concerned and all other tables addressed in the search condition. The authorization identifier must also have the EXECUTE privilege for all User Defined Functions (UDFs, see ["SQL Reference Manual Part 1: SQL Statements"](#)) which are addressed in the search condition or in the collation sequence.

In addition, the statement UNLOAD ... FROM COPY_FILE may only be executed on certain system user IDs. The SESAM/SQL system administrator specifies these user IDs with the DBH option ADMINISTRATOR or with the administration statement MODIFY-ADMINISTRATION.

```

UNLOAD [ { OFFLINE } ] { TABLE }
      [ { ONLINE } ] { DATA } table [(unload_column , ...)]

      [FROM { SPACE space
            { COPY_FILE file [PASSWORD password] } } ]

      INTO FILE file [PASSWORD password]
      { [(unload_description , ...)]
        TRANSFER_FORMAT
        LOAD_FORMAT
        DELIMITER_FORMAT TERMINATED BY delimiter [ { EBCDIC }
                                                    { UTFE } ] }
      { CSV_FORMAT DELIMITER delimiter [QUOTE quote] [ESCAPE escape]
        [ { EBCDIC }
          { UTFE } ] [WITH HEADER] }

      [WHERE search_condition]

      [ORDER BY { { column
                  { column(pos_no)
                  { column_no
                  { expression
                  } } } } } ] [ { ASC
                                { DESC } } ], ... ]

      unload_column ::= { column
                        { column(pos_no)
                        { column(min..max)
                        } } }

      unload_description ::= POSITION ( { number
                                       { *
                                       } ) [data_type]
                               [WHEN NULL THEN null_descriptor]

      null_descriptor ::= { literal
                          { POSITION(number) = alphanumeric_literal
                          } }

```

OFFLINE

When UNLOAD OFFLINE is specified, SESAM/SQL locks the user space to prevent other users making modifications. The FROM SPACE and FROM COPY_FILE clauses can be specified.

When UNLOAD OFFLINE is specified, all records are unloaded from a base table.

In this case UNLOAD OFFLINE is significantly quicker than UNLOAD ONLINE .

UNLOAD OFFLINE also enables the FROM SPACE or FROM COPY_FILE clause to be used to download data from base tables when their user spaces cannot be accessed during normal SESAM/SQL operation.

ONLINE

When UNLOAD ONLINE is specified, the user space is not locked exclusively. Only the same locks are requested as for a DML search statement. All the processing takes place in a DBH task.

When UNLOAD ONLINE is specified, the data to be unloaded can be restricted by a search condition. A collation sequence can also be defined for the output file. The contents of both the base tables and the views can be unloaded.

When UNLOAD ONLINE is specified, the output file and error file may not be located on tape.

TABLE

The data of all the columns is written to the output file.

Using the list of unload columns (see *unload_column ,...*), you can assign an unload description (see *unload_description ,...*) to columns. This allows you to define an unload description for a column that is different from that specified in the column definition in the database and to define how NULL values are to be represented in the output file.

The columns are unloaded in the order of the columns in the definition of the base table from which the data is to be output. This applies particularly to the columns that are not specified in the list of output columns.

DATA

Only data from the columns included in the list of unload columns (see *unload_column ,...*) is written to the output file. The output order corresponds to the order of the columns in this list. You can only specify UNLOAD DATA if you have specified a list of unload columns.

table

Name of the base table or view from which data is to be unloaded.

When UNLOAD ONLINE is specified, the table *table* may not be an CALL DML only table.

In the case of a partitioned table all the partitions must be available, see [table 3 on page 22](#)

When *table* identifies a view, the ONLINE parameter must be specified; in this case the TRANSFER_FORMAT operand may not be specified.

(unload_column ,...)

List of columns in *table* from which data is to be unloaded.

SESAM/SQL unloads the data in the order in which the unload columns are specified.

The following points must be taken into consideration:

- You cannot specify an unload column more than once in the list of unload columns.
- You must specify *(unload_column ,...)* if you specify UNLOAD DATA.
- If you specify a list of unload descriptions *(unload_description, ...)*, the number of unload columns specified must be the same as the number of specified unload descriptions.
- If you specify UNLOAD TABLE, you can only specify a list of unload columns if you also specify a list of unload descriptions.
- For multiple columns, SESAM/SQL always outputs the maximum number of occurrences if no list of unload columns is specified or if the multiple column is only referenced by the column name in the list of unload columns.
If not all the occurrences of the multiple column are supplied with values, SESAM/SQL treats the missing values like NULL values.

unload_column::=

column

Name of the column from which data is to be unloaded.

If *column* is a multiple column, the following applies:

col_{num} is the number of existing occurrences for *column*.

col_{max} is the maximum number of occurrences possible for *column*.

column (pos_no)

pos_no is a positive integer.

If *pos_no* is less than or equal to col_{num} , the *pos_no*th column element of the multiple column *column* is written to the output file, otherwise *column(pos_no)* is treated like a NULL value.

If *column* is not a multiple column or if *pos_no* is greater than col_{max} , an error message is issued.

column (min .. max)

The value is the aggregate from the column elements *min* to *max* of the multiple column *column*.

If *column* is not a multiple column or *min* is not less than *max*, an error message is issued.

pos_no or *min .. max* not specified:

If *column* is a multiple column, specifying *column* is the same as specifying *column(1 .. col_{max})*.

column, *column(pos_no)* or *column(min .. max)*

for *load_column* more than once if the ranges defined overlap.

FROM SPACE *space*

Name of the user space that is in the state “recover pending”. The user space cannot be defective.

You can only specify this clause in conjunction with the UNLOAD OFFLINE TABLE and TRANSFER_FORMAT clauses.

FROM COPY_FILE *file* [PASSWORD *password*]

BS2000 file name of a SESAM backup copy of a user space created on disk. The backup copy must not be defective. You must specify *file* as an alphanumeric literal.

You can only specify this clause in conjunction with the UNLOAD OFFLINE TABLE and TRANSFER_FORMAT clauses.

PASSWORD *password*

BS2000 password assigned with CREATE CATALOG. You must specify *password* as an alphanumeric literal.

password can be specified in several different ways:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

INTO FILE *file* [PASSWORD *password*]

Name of the BS2000 file into which the data is written. You must specify *file* as an alphanumeric literal. In the case of UNLOAD ONLINE, *file* may not be a tape file.

The output file is a SAM file which is formatted to the maximum length of the rows to be output depending on requirements. The maximum length of a row is calculated on the basis of the definition length and the output length of the columns. Furthermore, the output file is already created in the anticipated size, i.e. the number of rows of the table to be unloaded is taken into account.

SAM files can be created in units of 4KB blocks up to a maximum block size of 32 KB. The output file is created in whichever block size has the least cut off whilst attempting to select the largest possible block size of the output file. The maximum possible cut off per row is therefore 4095 Bytes (= 4KB - 1 Byte).

The output file with the database's CCSN (see the CODE-TABLE clause on [page 108](#)) is created for all UNLOAD formats with the exception of DELIMITER_FORMAT and CSV_FORMAT. If no CCSN is used for the database, the output file is also created without a CCSN.

In DELIMITER_FORMAT and CSV_FORMAT the output file's CCSN is determined via the EBCDIC/UTFE parameter (see [page 245](#)).

If the output file cannot be created, the UNLOAD statement will be aborted. If the UNLOAD statement is aborted because the theoretically possible length of an output row exceeds the maximum value, it is recommended that you use the main EXPORT TABLE function, unload the data with UNLOAD DATA or use the CSV_FORMAT with ESCAPE characters.

SESAM/SQL creates the output file on the BS2000 user ID under which the DBH is running and assigns it the specified file name. If you want the output file to be located on a different user ID, you must make the necessary preparations, see section "Database files and job variables on foreign user IDs" in the "[Core manual](#)".

PASSWORD *password*

BS2000 password for a password-protected output file. You must specify *password* as an alphanumeric literal.

password can be specified as follows:

- 'C'*string*"
string contains four printable characters.
- 'X'*hex_string*"
hex_string contains eight hexadecimal characters.
- 'n'
n identifies an integer from - 2147483648 to + 2147483647.

(unload_description, ...)

List of unload descriptions.



If neither *(unload_description, ...)* nor `TRANSFER_FORMAT`, `LOAD_FORMAT` nor `DELIMITER_FORMAT` is specified, all the values will be output in the standard representation that corresponds to the assigned column of the base table *table* in each case.

The unload descriptions describe the formats and positions of the output values and the representation of the NULL value in the output file. If you specify a list of unload columns *(unload_column, ...)*, SESAM/SQL assigns the specified unload descriptions to the elements in the list of unload columns one after the other.

The following points must be taken into consideration:

- If, in the case of UNLOAD TABLE, not all of the columns of the base table *table* are contained in the list of unload descriptions, the values of the missing columns will be output in the standard representation, which corresponds to the data type of the respective column.
- If, for UNLOAD TABLE, you only specify a list of unload descriptions but not a list of unload columns *(unload_column, ...)*, the number of unload descriptions must be the same as the number of columns in *table*. SESAM/SQL assigns the specified unload descriptions to the columns in *table* one after the other.
- If, during conversion into the unload description, characters or decimal places are truncated or decimal places are rounded, SESAM/SQL issues a warning. It does not, however, abort execution of UNLOAD because the above-described behavior may be desired.
SESAM/SQL aborts UNLOAD if data is lost during conversion.
- If you reference a multiple column in the list of unload columns using only the column name, or if you specify a range of occurrences, SESAM/SQL assumes that all the occurrences of this multiple column have the same format. Therefore, in the above-mentioned cases, you only need to specify one format, which is then used as the unload description for the occurrences.

You will find a description of the standard representation of the data for UNLOAD in [section “Standard representation of the data in the input and output files” on page 33](#).

unload_description ::=

POSITION (*pos_no*)

Positive integer indicating the position of the output value in the output file. *pos_no* cannot reference a position outside of the output record.

Position specifications that result in overlapping output fields are permitted. SESAM/SQL does not, however, check for overlapping output fields.

SESAM/SQL evaluates specifications of positions in the output record in accordance with the list of unload descriptions or list of unload columns (*unload_column*, ...), as appropriate. This may result in positions further to the right in the input file being evaluated earlier than positions further to the left. If output fields overlap, the value of a column specified earlier in the list of unload columns is overwritten by the value of a column specified later.

For UNLOAD TABLE, you must also take the position specifications for columns not included in the list of unload descriptions, which SESAM/SQL calculates internally, into consideration.

POSITION (*)

You must specify POSITION (*) if it is not possible or desirable to specify an exact position. The starting positions are calculated from all the previously defined output positions.

data_type

Data type of the output value.

You cannot specify FLOAT as the *data_type*. You must use REAL or DOUBLE PRECISION instead of FLOAT.

You cannot specify *dimension* (number of column elements for multiple columns) in *data_type*.

If you omit *length* or *precision* [*scale*], SESAM/SQL uses the value 1 for *length* and *precision* and 0 for *scale*.

data_type must be compatible with the data type of the corresponding unload column.

If you want to unload the values for a certain table column in readable form, you must use the data type CHARACTER with a sufficiently large length specification (see [section "Readable representation of the data in the input and output files" on page 27](#)).

data_type omitted:

SESAM/SQL uses the data type of the corresponding unload column.

If no unload description is specified for an unload column of the data type (NATIONAL) CHARACTER VARYING, output is performed using variable-lengths records. SESAM/SQL places a 2-byte length field in front of the output value.



In the case of NATIONAL CHARACTER VARYING the length field contains the length in bytes, not in characters.

WHEN NULL THEN *null_descriptor*

This clause defines how the NULL value is represented for the corresponding *unload_description* in the output file. You can only specify this clause if the unload column assigned to the unload description has not been defined as NOT NULL.

If NULL values occur in an unload column for which no WHEN NULL THEN clause has been specified, SESAM/SQL writes the NULL value to the output file in accordance with the unload description as follows:

- for (NATIONAL) CHARACTER:
as a string consisting of blanks '␣' (X'40') or N'␣' (NX'0020')
- for (NATIONAL) CHARACTER VARYING:
X'0000' is the value for the length field
- for NUMERIC: X'F0'
- for DECIMAL: X'0C'
- for INTEGER, SMALLINT: binary zero
- for REAL: X'00000000'
- for DOUBLE PRECISION: X'0000000000000000'
- for DATE: X'0001 0001 0001'
- for TIME(3): X'0000 0000 0000 0000'
- for TIMESTAMP(3): X'0001 0001 0001 0000 0000 0000 0000'

The following applies to unload columns defined with (CALL-DML) default value characters:

If an output file with the above substitute values (non-significant values) is used as the input file for the utility statement LOAD specified without a WHEN ... THEN NULL clause, SESAM/SQL only interprets these non-significant values as NULL values if the (CALL-DML) default value character is the standard (CALL-DML) default value character. In this case, it is recommended that you unload the data in UNLOAD_FORMAT, TRANSFER_FORMAT or DELIMITER_FORMAT or declare a suitable *null_descriptor* in the WHEN NULL THEN ... clause. You must then specify the appropriate clauses when loading the data with LOAD.

You can only specify a literal without a position specification as the *null_descriptor* in an unload description that references several occurrences of a multiple column. If you specified a position, it would not be clear which occurrences contain a NULL value.

null_descriptor ::=

literal

Literal.

SESAM/SQL converts the value specified for *literal* into the unload description of the assigned column using an appropriate data type.

If the unload description is (NATIONAL) CHARACTER VARYING(*n*), *literal* can have up to *n* characters. Output is always performed using the actual length of *literal*.

POSITION(*pos_no*)

Positive integer indicating the position of the NULL descriptors in the output file.

pos_no cannot reference a position outside of the output record.

alphanumeric_literal

Alphanumeric literal.

TRANSFER_FORMAT

The data is output to a so called transfer file.



If data of the data type NATIONAL CHARACTER (VARYING) is output in TRANSFER_FORMAT, the transfer file can no longer be loaded using SESAM/SQL V4.0 or earlier.

LOAD_FORMAT

The data is output to a file that is suitable for use as an input file for the utility statement LOAD with UNLOAD_FORMAT.

In output files in LOAD_FORMAT, SESAM/SQL places an indicator of the type SMALLINT in front of each output value. This indicator shows whether the subsequent output value is to be interpreted as a NULL value. If an output value is to be interpreted as a NULL value, the corresponding indicator contains a negative value.

DELIMITER_FORMAT TERMINATED BY *delimiter* $\left\{ \begin{array}{l} \text{EBCDIC} \\ \text{UTFE} \end{array} \right\}$

SESAM/SQL outputs data in delimiter format. In delimiter format, all values are represented in the output file's coded character set and terminated with a DELIMITER character. If necessary the data to be output is converted to the output file's character set in accordance with the database's CCSN and output. For further details on representing values in delimiter format, refer to [section "Readable representation of the data in the input and output files" on page 27](#).

NULL values are not represented in the output file. Only the DELIMITER character terminating the NULL value is represented. Two consecutive DELIMITER characters within a record or at the end of a record should therefore be interpreted as follows: the first DELIMITER character terminates the preceding value and the second DELIMITER character terminates a NULL value which is not represented.

If a table you wish to unload contains a column with the data type (NATIONAL) CHARACTER VARYING and some of the records contain values with a length 0 (corresponds to an empty string ""), these records are entered in the UNLOAD error file, see the section [“Error file in the case of UNLOAD” on page 251](#).

The (possibly converted and edited) values are written to the output file in a significant length. Each output row is written to a SAM record. If the maximum length of a SAM record is exceeded in the process, the UNLOAD statement is aborted with SQLSTATE.

delimiter

Defines a DELIMITER character which is to terminate each value in the output file. The DELIMITER character must not be a value which is a component of any of the output values.

SESAM/SQL checks whether the DELIMITER character occurs in a value to be output. If it does, then this record is written not to an output file but to the UNLOAD exception file.

The specification of *delimiter* and of *quote* and of *escape* (in CSV_FORMAT, see [page 247](#)) depends on the following parameter.

EBCDIC

The CCSN of the database is entered as the output file's CCSN (see the CODE-TABLE clause on [page 108](#)). Data of the data type NATIONAL CHARACTER (VARYING) is converted to the output file's (EBCDIC) character set and output. Data of the data type CHARACTER (VARYING) is output unchanged. Data of other data types is converted to the output file's (EBCDIC) character set and output.

(The special EBCDIC character set is irrelevant here since in the event of conversion of these (numeric or time) data types the required characters are all contained in the EBCDIC core and are represented in an identical manner.)

If no CCSN is defined for the database, the output file is created without CCSN. In this case data of the data type NATIONAL CHARACTER (VARYING) cannot be output. Data of the data type CHARACTER (VARYING) is output unchanged. Data of other data types is converted to EBCDIC and output.

If conversion is not possible, an entry is written to the UNLOAD error file.

In this case *delimiter*, *quote* and *escape* (CSV_FORMAT) are alphanumeric literals with the length 1.

Examples: TERMINATED BY ';' or TERMINATED BY X'5E'

If a single quote (') is to be defined, you must specify four single quotes (TERMINATED BY ''').

For reasons of compatibility, hexadecimal representation can also be specified: 'X"xx"' (x: a digit in hexadecimal representation).



Output files in delimiter format with CCSN EBCDIC are CSV files (CSV: Comma Separated Values). CSV files can be used both in SESAM/SQL (see also the table function CSV in the [“SQL Reference Manual Part 1: SQL Statements”](#) and the utility functions LOAD and UNLOAD) and also in external software products.

UTFE

The output file is created with the CCSN UTFE. Data of the data type CHARACTER (VARYING) is converted to the output file's UTFE character set in accordance with the database's CCSN (see the CODE-TABLE clause on [page 108](#)) and output. Data of other data types is converted to the output file's UTFE character set and output.



If data of the data type NATIONAL CHARACTER VARYING in DELIMITER_FORMAT or in CSV_FORMAT is output to an output file with the CCSN UTFE, the output value can be abbreviated if, after conversion, it is greater than the maximum length of the SAM file.

If no CCSN is defined for the database, data of the data type CHARACTER (VARYING) cannot be output. Data of other data types is converted to the output file's UTFE character set and output.

If conversion is not possible, an entry is written to the UNLOAD error file.

In this case *delimiter*, *quote* and *escape* (CSV_FORMAT) are alphanumeric literals with the length 1 which also have hexadecimal representation in one byte in UTFE (range of values: NX'0000' . . . NX'009F').

Examples: TERMINATED BY ';' or TERMINATED BY NX'003B'
or also: TERMINATED BY U&';' or TERMINATED BY U&'003B'

If a single quote (') is to be defined, you must specify four single quotes (TERMINATED BY N''' or ...U&''').

CSV_FORMAT DELIMITER *delimiter* [QUOTE *quote*] [ESCAPE *escape*]

{
 EBCDIC
 UTFE
}

[WITH HEADER]

SESAM/SQL outputs data in CSV format. In CSV format, all values are represented in the output file's coded character set and terminated with a DELIMITER character. If necessary the data to be output is converted to the output file's character set in accordance with the database's CCSN and output. For further details on representing values in CSV format, refer to [section "Readable representation of the data in the input and output files" on page 27](#).

NULL values are not represented in the output file. Only the DELIMITER character terminating the NULL value is represented. Two consecutive DELIMITER characters within a record or at the end of a record should therefore be interpreted as follows: the first DELIMITER character terminates the preceding value and the second DELIMITER character terminates a NULL value which is not represented.

When a table which is to be unloaded contains a column of the data type (NATIONAL) CHARACTER VARYING and some rows have the length 0 (corresponds to the empty string ""), these rows are represented in the output file by two consecutive QUOTE characters if a QUOTE character is specified. If no QUOTE character is specified, these rows are entered in the UNLOAD error file.

The (possibly converted and edited) values are written to the output file in a significant length. Each output row is written to a SAM record. If the maximum length of a SAM record is exceeded in the process, an ESCAPE character is written as the last character of the SAM record and the output record is continued in the next SAM record. If no ESCAPE character is specified, the UNLOAD statement is aborted with SQLSTATE.

When a column value which is to be unloaded contains DELIMITER characters, this value is enclosed in QUOTE characters in the output file if a QUOTE character is specified. If an ESCAPE character is specified, the DELIMITER character is prefixed with this ESCAPE character.

If neither a QUOTE nor an ESCAPE character is specified, the row is entered in the UNLOAD error file.

When a column value which is to be unloaded contains QUOTE characters, the QUOTE character is prefixed with an ESCAPE character if an ESCAPE character is specified. If only one QUOTE character is specified, such a value in the output file is enclosed in QUOTE characters, and the QUOTE character is duplicated.

When a column value which is to be unloaded contains ESCAPE characters, the ESCAPE character is prefixed with an ESCAPE character in the output file.

When a column value which is to be unloaded contains NEWLINE characters, the column value in the output file is enclosed in QUOTE characters if a QUOTE character is specified. If no QUOTE character is specified, the row is entered in the UNLOAD error file.

delimiter

Defines a DELIMITER character which is to terminate each value in the output file. The DELIMITER character may also be a value which is a component of any of the output values when a QUOTE character and/or an ESCAPE character is also specified.

quote

Defines the QUOTE character in which the values of the output file can be enclosed.

escape

Defines the escape character (ESCAPE character) with which ESCAPE sequences in the output file can begin.



The characters specified for DELIMITER, QUOTE and ESCAPE must all be different.

The specification of *delimiter*, *quote* or *escape* depends on the parameter EBCDIC/UTFE, see [page 245](#).

WITH HEADER

This parameter causes a header line with the column names in CSV format to be output in the first record of the output file. This first record has the same format as the following records containing the output data.

In the case of multiple columns, the column name is output for every occurrence concerned. The occurrence number is enclosed in parentheses and appended to the column number.

Column names have the data type CHARACTER. Consequently a CCSN must be defined when UTFE is specified for the database.

If writing the first record leads to an entry in the error file, the statement is aborted with SQLSTATE.



When user data is loaded from a CSV input file with LOAD, the header line must be skipped using SKIP FIRST *number* RECORDS, see [page 149](#).

When reading using the table function CSV(), the header line can be skipped by means of the WITH ORDINALITY parameter and a corresponding WHERE clause, see the "[SQL Reference Manual Part 1: SQL Statements](#)".

WHERE *search_condition*

Only the user data that satisfies the search condition is unloaded.

No user variables and no question marks may be used as placeholders in the search condition.

Tables other than the specified table *table* can also be referenced.

However, none of the referenced tables may be a CALL DML only table.

The processing of a search condition for a other referenced tables is performed at the application's default isolation level. However, this does not apply if you have specified a different value using SET TRANSACTION or the ISOLATION LEVEL pragma.

Other pragmas are also effective with WHERE *search_condition* (see the "[SQL Reference Manual Part 1: SQL Statements](#)"). When the WHERE clause is specified, the ONLINE parameter must also be specified.

ORDER BY

The ORDER BY clause defines the order in which the records read are to be written to the output file. The rows are sorted according to the values in the column specified first. If records occur which have the same values in the first column according to the comparison rules, these will be sorted according to the second column, and so on. In SESAM/SQL, NULL values are considered smaller than all non-NULL values for sorting purposes.

The order of rows with the same value in all the sort columns is undefined.

When the ORDER BY clause is specified, the ONLINE parameter must also be specified.

column

Name of the column which is to be used for sorting. *column* must be an unqualified column name, excluding the table name. It must belong to the output quantity which is generated with the WHERE clause.

column(pos_no)

Element of a multiple column according to which the table is to be sorted. *pos_no* is an unsigned integer which indicates the position number of the column element in the multiple column. The column element must belong to the output quantity which is generated with the WHERE clause.

column_number

Number of the column to be used as the basis for sorting.

column_no is an unsigned integer where: $1 \leq \text{column_no} \leq \text{number of output columns}$.
column_number can be an atomic column or a multiple column with the dimension 1.

expression

You can sort not only according to the output columns, but also according to further expressions, such as UPPER(*column*).

The following conditions must be satisfied:

- *expression* may not consist just of a literal.
- *expression* may not contain a subquery or set function.
- No columns of the table *table* may be used in *expression* even if they are not contained in the list of output columns.

ASC / DESC

The values of the associated column are sorted in ascending (ASC) or descending (DESC) order.

Error file in the case of UNLOAD

In the case of UNLOAD, SESAM/SQL creates a new file if required or updates an existing file. The file has the following name:

catalog.space.EXC.U in the case of UNLOAD OFFLINE and
file.EXC.U in the case of UNLOAD ONLINE

catalog is the name of the database.

space Name of the user space containing the table into which the records are to be loaded.

In the case of a partitioned table the space name of the first partition is used. The error file contains the error information for all partitions in the table.

file Name of the file which is specified in the INTO FILE clause.

EXC.U Default name ending of the exception file for UNLOAD statements.

New entries are appended to an existing exception file. The entries from various UNLOAD statements are separated by a header line which logs the date and time. If an exception file contains at least one record, a warning is issued.

If *catalog* is located on a DB user ID, in the case of UNLOAD OFFLINE the error file will be created on the DB user ID provided the necessary preparations have been made, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”. If the error file cannot be created on the DB user ID, it will be created on the DBH user ID.

When UNLOAD ONLINE is specified, the error file is created on the same ID as the output file *file*. If this ID is not the same as the DBH ID, preparations must be made for this, see section “Database files and job variables on foreign user IDs” in the “[Core manual](#)”. If the error file cannot be created on this ID, it is created on the DBH ID.

Format of the error file with UNLOAD

With UNLOAD the error file consists of two headers with the statement name and date/time when the statement was executed, plus the name of the output file.

These are followed by error-specific value lines in the following format:

```
<counter> <number> <sqlstate> <column> [<value>]
```

where

- `<counter>` specifies the record in the output file after which the record written to the error file is to be entered in the output file
- `<number>` specifies the number of the record processed
- `<sqlstate>` specifies the SQLSTATE
- `<column>` specifies the table column
- `<value>` specifies the column value (hexadecimal / printable)

The following types of entry can occur in the UNLOAD error file:

- Values which cannot be transliterated or converted (SQLSTATE 22SB2); values which in CSV format contain NEWLINE characters without a QUOTE character being defined (SQLSTATE 22SB5).

These are presented in the error file in hexadecimal and printable format with a maximum length of 32000 bytes (16000 characters). If more than one value in a record cannot be transliterated or converted, an entry is entered in the error file for each of these values. The `<number>` enables you to determine that the error entries refer to the same record.

- Values containing the DELIMITER character (SQLSTATE 22SB3).
These are presented in the error file in printable format and with a maximum length of 16000 characters.
- Columns of the data type (NATIONAL) CHARACTER VARYING which contain a value of the length NULL (SQLSTATE 22SB4).
An entry is written to the error file, but without `<value>`.
- In the case of UNLOAD with a user-defined format, a numeric value is either fallen below or exceeded.
SQLSTATE 22003 is entered in the error file in the following format:
`<counter> 22003 <column>`
- In the case of UNLOAD with a user-defined format, the column value is longer than the area defined in the output record.
Warning 01004 is entered in the error file in the following format:
`<counter> 01004 <column>`
- In the case of UNLOAD with a user-defined format, decimal places are rounded or truncated.
Warning 01SB3 is entered in the error file in the following format:
`<counter> <number> 01SB3 <column>`



When a warning has been entered in the error file, the record has also been written to the output file. `<counter> +1` designates the record in the output file.

UNLOAD formats

The following table contains an overview of the UNLOAD formats.

The examples for UNLOAD statements do not take into account all possible clauses (e.g. FROM SPACE, FROM COPY FILE ...).

Format	Statement (example)	Unloaded columns	Null values	Processed by means of
TRANSFER_ FORMAT	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i> TRANSFER_FORMAT	All columns according to order of <i>table</i>	Internal representation (Y/N for all columns at the end of a row)	LOAD statement with TRANSFER_ FORMAT
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> TRANSFER_FORMAT	Only the specified rows in this order		
LOAD_ FORMAT	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i> LOAD_FORMAT	All columns according to order of <i>table</i>	Internal representation (negative indicator before corresponding column; full attribute length, except in case of (N)VARCHAR)	LOAD statement with UNLOAD_ FORMAT
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> LOAD_FORMAT	Only the specified rows in this order		
DELIMITER_ FORMAT	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i> DELIMITER_FORMAT TERMINATED BY <i>delimiter</i>	All columns according to order of <i>table</i>	Only delimiter characters	LOAD statement with DELIMITER_ FORMAT or Editor
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> DELIMITER_FORMAT TERMINATED BY <i>delimiter</i>	Only the specified rows in this order		

Table 10: UNLOAD output formats

(part 1 of 2)

Format	Statement (example)	Unloaded columns	Null values	Processed by means of
CSV_FORMAT	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i> CSV_FORMAT DELIMITER <i>delimiter</i> QUOTE <i>quote</i> ESCAPE <i>escape</i>	All columns according to order of <i>table</i>	Only DELIMITER characters	LOAD statement with CSV_FORMAT or editor or calculation program
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> CSV_FORMAT DELIMITER <i>delimiter</i>	Only the specified rows in this order		
Standard format	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i>	All columns according to order of <i>table</i>	Blank or null in full attribute length, except in the case of (N)VARCHAR; no longer identifiable as null value in case of LOAD!	LOAD statement with standard format or format defined by user
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i>	Only the specified rows in this order		
Format defined by user	UNLOAD TABLE <i>table</i> INTO FILE <i>file</i> (<i>unload_description, unload_description WHEN NULL THEN null_descriptor, ...</i>)	All columns according to order of <i>table</i>	As <i>null_descriptor</i> , if this is defined; otherwise as with standard format	LOAD statement with format defined by user or Editor
	UNLOAD DATA <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> (<i>unload_description, unload_description WHEN NULL THEN null_descriptor, ...</i>)	Only the specified rows in this order		
Mix of standard format and format defined by user	UNLOAD TABLE <i>table</i> (<i>column, column, ...</i>) INTO FILE <i>file</i> (<i>unload_description, unload_description WHEN NULL THEN null_descriptor, ...</i>)	All columns with format defined by user for the specified columns; standard format for all other columns	As <i>null_descriptor</i> , if this is defined; otherwise as with standard format	LOAD statement with format defined by user

Table 10: UNLOAD output formats

(part 2 of 2)

Examples

In the examples below, data is unloaded from the base table CUSTOMERS (see the example for the utility statement LOAD on [page 163](#)). The table CUSTOMERS is defined in the schema ORDERPROC of the database ORDERCUST.

1. The data is output (offline) to the output file SES.CUSTOMER.DEL in delimiter format. This output file can subsequently be used as an input file for a LOAD utility statement with the clause DELIMITER_FORMAT TERMINATED BY ';'.
If no value is stored for ZIP in the CUSTOMERS table, the DELIMITER character only is entered at the appropriate location in the output file.



```
UNLOAD TABLE ordercust.orderproc.customer
INTO FILE 'SES.CUSTOMER.DEL'
DELIMITER_FORMAT TERMINATED BY ';'

```

The output file has the following contents after the UNLOAD statement has been executed:

```
100;Siemens AG;Otto-Hahn-Ring 6;81739;Munich;D;089/636-8;Electrical
101;Login GmbH;Rosenheimer Str.34;81667;Munich;D;089/4488870;PC Networks
102;JIKO GmbH;Posener Str. 12;30659;Hanover;D;0551/123874;Import/Export
103;Plenzer Trading;Paul-Heyse-Str. 12;80336;Munich;D;089/923764;Fruit market
____Freddy's Fishery                Hirschgartenstr. 12
12;12587;Berlin;D;016/5739921;Unit retail;
105;The Poodle Parlor;Am Muehlentor 26;41179;Moenchengladbach;D;040/873562;Service
106;Foreign Ltd.;26 West York St.;New York, NY;USA;001703/2386532;Commercial agency
107;Externa & Co KG;Bernner Weg 78;03000;Bern 33;CH;;Lawyer

```

The number of unloaded records is output in the SQLrowcount field of the diagnostic area.

2. The data is output online to the file SES.CUSTOMERDATA.OUT1 in unload format. In the table only the German customers are selected for unloading here. These customers are sorted in the output file according to ZIP code and (when the ZIP codes are identical) according to company names.
The output file can be used as the input file for a subsequent LOAD utility statement with the UNLOAD_FORMAT clause.

```
UNLOAD ONLINE TABLE ordercust.orderproc.customers
INTO FILE 'SES.CUSTOMERDATA.OUT1' LOAD_FORMAT
WHERE country = 'D' ORDER BY zip,company

```

3. The data is unloaded (offline) to the file SES.CUSTOMERDATA.OUT2. The values of the primary key CUST_NUM for the table CUSTOMERS, as well as the values for ZIP are stored in the output file in readable form. Please note that the CHARACTER unload description assigned to an unload column of the type INTEGER must have a length of at least CHARACTER(10). If there is no value for ZIP stored in the table CUSTOMERS, the string "XXXXX" is entered at the appropriate position in the output file.



```
UNLOAD TABLE ordercust.orderproc.customer
(cust_num, company, street, zip, city, country, cust_tel,
cust_info)
INTO FILE 'SES.CUSTOMERDATA.OUT2'
  (POSITION(1) CHARACTER(10),
   POSITION(11) CHARACTER(40),
   POSITION(51) CHARACTER(40),
   POSITION(91) CHARACTER(5) WHEN NULL THEN 'XXXXX',
   POSITION(96) CHARACTER(40),
   POSITION(136) CHARACTER(3),
   POSITION(139) CHARACTER(25),
   POSITION(164) CHARACTER(50))
```

After UNLOAD has been executed, the file SES.CUSTOMERDATA.OUT2 has the following contents:

____Siemens AG		Otto-Hahn-Ring 6	
81739Munich	D	089/636-8	Electrical
____Login GmbH		Rosenheimer Str.34	
81667Munich	D	089/4488870	PC Networks
____JIKO GmbH		Posener Str. 12	
30659Hanover	D	0551/123874	Import-Export
____Plenzer Trading		Paul-Heyse-Str. 12	
80336Munich	D	089/923764	Fruit market
104 Freddy's Fishery		Hirschgartenstr. 12	
12587Berlin	D	016/5739921	Unit retail
____Poodle Parlor		Am Muehlentor 26	
41179Moenchengladbach	D	040/873562	Service
____Foreign Ltd.		26 West York St.	
00000New York, NY	USA001703/2386532		Commercial agency
____Externa & Co KG		Berner Weg 78	
03000Bern 33	CH		Lawyer

See also

LOAD, EXPORT/IMPORT TABLE

4 Appendix

4.1 Syntax overview

You will find a list of rules for all the syntax elements used in this manual below listed in alphabetical order. Syntax elements described in the “[SQL Reference Manual Part 1: SQL Statements](#)”, not in this manual, are indicated by the additional text “(Part 1)”.



Any square brackets shown here in italics are special characters, and must be specified in the statement.

$$\textit{alphanumeric_literal} ::= \left\{ \begin{array}{l} \textit{'[character...]'[{separator...'[character...]'...]} \\ \textit{X'[hex hex]...'[separator...'[hex hex]...']} \end{array} \right\} \text{ (Part 1)}$$

authorization_identifier ::= *unqual_name* (Part 1)

$$\textit{bs2000_user} ::= \left(\left\{ \begin{array}{l} \textit{hostname} \\ * \end{array} \right\}, [\textit{*}], \left\{ \begin{array}{l} \textit{bs2000_user_id} \\ * \end{array} \right\} \right)$$

catalog ::= *unqual_name* (Part 1)

catrec_file ::= *unqual_name* (Part 1)

catalog ::= *unqual_name* (Part 1)

$$\textit{column_value} ::= \left\{ \begin{array}{l} \textit{alphanumeric_literal} \\ \textit{national_literal} \\ \textit{numeric_literal} \\ \textit{time_literal} \end{array} \right\} \text{ (Part 1)}$$
$$\textit{comparison_op} ::= \left\{ \begin{array}{l} = \\ < \\ > \\ <= \\ >= \\ <> \end{array} \right\}$$

data_type ::= (Part 1)

$$\left\{ \begin{array}{l}
 \left[\left\{ \begin{array}{l} [dimension] \\ (dimension) \end{array} \right\} \right] \text{CHARACTER} [(length)] \\
 \left\{ \begin{array}{l} \text{CHARACTER VARYING}(max) \\ \text{VARCHAR}(max) \end{array} \right\} \\
 \left[\left\{ \begin{array}{l} [dimension] \\ (dimension) \end{array} \right\} \right] \left\{ \begin{array}{l} \text{NATIONAL CHARACTER} \\ \text{NCHAR} \end{array} \right\} [(cu_length \text{ [CODE_UNITS]})] \\
 \left\{ \begin{array}{l} \text{NATIONAL CHARACTER VARYING} \\ \text{NCHAR VARYING} \\ \text{NVARCHAR} \end{array} \right\} (cu_max \text{ [CODE_UNITS]}) \\
 \left[\left\{ \begin{array}{l} [dimension] \\ (dimension) \end{array} \right\} \right] \left\{ \begin{array}{l} \text{SMALLINT} \\ \text{INTEGER} \\ \text{NUMERIC} [(precision[, scale])] \\ \text{DECIMAL} [(precision[, scale])] \\ \text{REAL} \\ \text{DOUBLE PRECISION} \\ \text{FLOAT} [(stellen)] \\ \text{DATE} \\ \text{TIME}(3) \\ \text{TIMESTAMP}(3) \end{array} \right\}
 \end{array} \right\}$$

default ::= DEFAULT (Part 1)

$$\left\{ \begin{array}{l}
 \textit{alphanumeric_literal} \\
 \textit{numeric_literal} \\
 \textit{time_literal} \\
 \text{CURRENT_DATE} \\
 \text{CURRENT_TIME}(3) \\
 \text{LOCALTIME}(3) \\
 \text{CURRENT_TIMESTAMP}(3) \\
 \text{LOCALTIMESTAMP}(3) \\
 \text{USER} \\
 \text{CURRENT_USER} \\
 \text{SYSTEM_USER} \\
 \text{NULL} \\
 \text{REF}(\textit{table})
 \end{array} \right\}$$

fixed_pt_number ::=
$$\left[\left\{ \begin{array}{l} + \\ - \end{array} \right\} \right] \left\{ \begin{array}{l} \textit{unsigned_integer} [\textit{unsigned_integer}] \\ \textit{unsigned_integer} \\ \textit{unsigned_integer} \end{array} \right\} \text{ (Part 1)}$$

floating_pt_number ::= *fixed_pt_number* E { + }] *unsigned_integer* (Part 1)

index ::= [[*catalog*] *unqual_schema_name*] *unqual_index_name* (Part 1)

integer ::= [{ + }] *unsigned_integer* [.] (Part 1)

integrity_constraint_name ::= [[*catalog*] *unqual_schema_name*] *unqual_constraint_name* (Part 1)

last_partition ::= PARTITION *partno* [VALUE <=()] ON SPACE *space* (Part 1)

literal ::= { *alphanumeric_literal*
national_literal
special_literal
numeric_literal
time_literal } (Part 1)

load_column ::= { *column*
column(*pos_no*)
column(*min..max*) }

load_description ::= POSITION ({ *number* }) [*data_type*] [WHEN *null_constraint* THEN NULL]

load_parameter ::= { [NO] OVERWRITE
{ NO INDEX
GENERATE INDEX }
[NO] CONSTRAINT CHECK }

max ::= *unsigned_integer* (Part 1)

min ::= *unsigned_integer* (Part 1)

national_literal ::=

{ N' [*character* . . .] ' [{ *separator* . . . ' [*character* . . .] ' } . . .]
NX' [*4hex* . . .] ' [{ *separator* . . . ' [*4hex* . . .] ' } . . .]
U&' [{ *character*
esc 4hex
esc+ 6hex
esc esc } . . .] ' [{ *separator* . . . ' [{ *character*
esc 4hex
esc+ 6hex
esc esc } ' . . .] [UESCAPE '*esc*'] } (Part 1)

$$\text{null_constraint} ::= \left\{ \begin{array}{l} \text{column comparison_op literal} \\ \text{POSITION} \left(\left\{ \begin{array}{l} \text{number} \\ * \end{array} \right\} \right) \text{ comparison_op} \left\{ \begin{array}{l} \text{alphanumeric_literal} \\ \text{national_literal} \end{array} \right\} \end{array} \right\}$$

$$\text{null_descriptor} ::= \left\{ \begin{array}{l} \text{literal} \\ \text{POSITION}(\text{number}) = \text{alphanumeric_literal} \end{array} \right\}$$

$$\text{numeric_function} ::= \left(\begin{array}{l} \text{ABS} (\text{ausdruck}) \\ \text{CEIL[ING]} (\text{ausdruck}) \\ \text{FLOOR} (\text{ausdruck}) \\ \text{MOD} (\text{dividend}, \text{divisor}) \\ \text{SIGN} (\text{ausdruck}) \\ \text{TRUNC} (\text{ausdruck}) \\ \left\{ \begin{array}{l} \text{CHAR_LENGTH} \\ \text{CHARACTER_LENGTH} \end{array} \right\} (\text{ausdruck} [\text{USING} \left\{ \begin{array}{l} \text{CODE_UNITS} \\ \text{OCTETS} \end{array} \right\}]]) \\ \text{OCTET_LENGTH} (\text{ausdruck}) \\ \text{POSITION} (\text{ausdruck} \text{ IN } \text{ausdruck} [\text{USING} \text{CODE_UNITS}]) \\ \text{JULIAN_DAY_OF_DATE} (\text{ausdruck}) \\ \text{EXTRACT} (\text{bestandteil} \text{ FROM } \text{ausdruck}) \end{array} \right) \quad \text{(Part 1)}$$

$$\text{numeric_literal} ::= \left\{ \begin{array}{l} \text{integer} \\ \text{fixed_pt_number} \\ \text{floating_pt_number} \end{array} \right\} \quad \text{(Part 1)}$$

$$\text{partition} ::= \text{PARTITION } \text{partno} \text{ VALUE} \left\{ \begin{array}{l} < \\ <= \end{array} \right\} (\text{column_value}, \dots) \text{ ON SPACE } \text{space} \quad \text{(Part 1)}$$

$$\text{partno} ::= \text{unsigned_integer} \quad \text{(Part 1)}$$

$$\text{pos_no} ::= \text{unsigned_integer} \quad \text{(Part 1)}$$

$$\text{pragma} ::= \text{--\%PRAGMA } \text{pragma_text}, \dots \text{end_of_line} \quad \text{(Part 1)}$$

$$\text{rec_unit} ::= \left\{ \begin{array}{l} \text{COPY_FILE } \text{file} \\ \text{COPY_NUMBER } \text{number} \\ \text{timestamp_literal} \\ \text{FOREIGN COPY} \end{array} \right\}$$

$$\text{regular_name} ::= \text{letter} \left[\left\{ \begin{array}{l} \text{letter} \\ \text{digit} \\ _ \end{array} \right\} \right] \dots \quad \text{(Part 1)}$$

$$\text{replication} ::= \text{catalog}$$

```

replication_description ::=
    REPLICATION replication
        [PASSWORD password]
        [ON USER_ID bs2000_user_id]
        [WITH CATREC catrec_file]
        PASSWORD user_id ON USER_ID bs2000_user_id]
        { COPY
          RENAME }

```

schema ::= [catalog.]*unqual_schema_name* (Part 1)

```

search_condition ::= {
    predicate
    search_condition { AND
                       OR } search_condition
    NOT search_condition
    (search_condition)
}

```

space ::= [catalog.]*unqual_space_name* (Part 1)

```

space_parameter_list ::= {
    PRIMARY allocation
    SECONDARY allocation
    PCTFREE percent
    [NO] SHARE
    [NO] DESTROY
    NO LOG
} ...

```

special_name ::= "character.." (Part 1)

stogroup ::= [catalog.]*unqual_stogroup_name* (Part 1)

```

stogroup_attributes ::=
    VOLUMES(volume_name ,...) ON dev_type } [ON catalog_id]
    PUBLIC

```

```

system_user_id ::= {
    utm_userr
    bs2000_user
}

```

```

table ::= {
    [[catalog.]unqual_schema_name.]unqual_base_table_name
    [[catalog.]unqual_schema_name.]unqual_view_name
    correlation
} (Part 1)

```

$$time_function ::= \left\{ \begin{array}{l} CURRENT_DATE \\ CURRENT_TIME (3) \\ LOCALTIME (3) \\ CURRENT_TIMESTAMP (3) \\ LOCALTIMESTAMP (3) \\ DATE_OF_JULIAN_DAY(expression) \end{array} \right\} \text{ (Part 1)}$$

$$time_literal ::= \left\{ \begin{array}{l} DATE 'year-month-day' \\ TIME 'hour:minute:second' \\ timestamp_literal \end{array} \right\} \text{ (Part 1)}$$

timestamp_literal ::= `TIMESTAMP 'year-month-day hour:minute:second'` (Part 1)

$$unload_column ::= \left\{ \begin{array}{l} column \\ column(number) \\ column(min..max) \end{array} \right\}$$

unload_description ::= `POSITION ($\left\{ \begin{array}{l} number \\ * \end{array} \right\}$) [data_type] [WHEN NULL THEN null_descriptor]`

unqual_base_table_name ::= *unqual_name* (Part 1)

unqual_constraint_name ::= *unqual_name* (Part 1)

unqual_index_name ::= *unqual_name* (Part 1)

$$unqual_name ::= \left\{ \begin{array}{l} regular_name \\ special_name \end{array} \right\} \text{ (Part 1)}$$

unqual_schema_name ::= *unqual_name* (Part 1)

unqual_space_name ::= *unqual_name* (Part 1)

unqual_stogroup_name ::= *unqual_name* (Part 1)

unsigned_integer ::= *digit*... (Part 1)

$$utm_user ::= \left(\left\{ \begin{array}{l} hostname \\ * \end{array} \right\}, \left\{ \begin{array}{l} utm_application_name \\ * \end{array} \right\}, \left\{ \begin{array}{l} utm_userid \\ * \end{array} \right\} \right)$$

4.2 SQL keywords

In SESAM/SQL there are words that are reserved as keywords for SQL and utility statements. These keywords cannot be used as the names of views, tables, columns, etc. in SQL or utility statements or when working with the utility monitor, unless you specify the keyword in the form of a special name.

The synonym processing feature provided by the ESQL precompiler is a convenient way of replacing keywords or of redefining names.

You can use the precompiler option SOURCE-PROPERTIES to set the ESQL-DIALECT parameter to ISO, OLD or ALL-FEATURES. This determines whether the SQL dialect ISO, OLD or FULL has to be used.

The table below lists the reserved keywords and indicates the SQL dialect in which they are valid.

Keyword	ISO	OLD	FULL
ABS	X		X
ABSOLUTE	X		X
ACTION	X		X
ADD	X		X
ALL	X	X	X
ALLOCATE	X		X
ALTER	X		X
AND	X	X	X
ANY	X	X	X
ARE	X		X
AS	X	X	X
ASC	X	X	X
ASSERTION	X		X
AT	X		X
AUTHORIZATION	X	X	X
AVG	X	X	X
BEGIN	X	X	X
BETWEEN	X	X	X
BIT	X		X

Table 11: SESAM/SQL keywords

(part 1 of 10)

Keyword	ISO	OLD	FULL
BIT_LENGTH	X		X
BLOB	X		X
BOTH	X		X
BY	X	X	X
CALL	X		X
CASCADE	X		X
CASCADED	X		X
CASE	X		X
CAST	X		X
CATALOG	X		X
CEIL	X		X
CEILING	X		X
CHAR	X	X	X
CHARACTER	X	X	X
CHARACTER_LENGTH	X		X
CHAR_LENGTH	X		X
CHECK	X	X	X
CLOSE	X	X	X
COALESCE	X		X
COLLATE	X		X
COLLATION	X		X
COLUMN	X		X
COMMIT	X	X	X
CONNECT	X		X
CONNECTION	X		X
CONSTRAINT	X		X
CONSTRAINTS	X		X
CONTINUE	X	X	X
CONVERT	X		X
COPY			X
CORRESPONDING	X		X

Table 11: SESAM/SQL keywords

(part 2 of 10)

Keyword	ISO	OLD	FULL
COUNT	X	X	X
CREATE	X	X	X
CROSS	X		X
CURRENT	X	X	X
CURRENT_CATALOG	X		X
CURRENT_DATE	X		X
CURRENT_ISOLATION_LEVEL			X
CURRENT_REFERENCED_CATALOG			X
CURRENT_SCHEMA	X		X
CURRENT_TIME	X		X
CURRENT_TIMESTAMP	X		X
CURRENT_USER	X		X
CURSOR	X	X	X
DATA	X		X
DATE	X		X
DATE_OF_JULIAN_DAY			X
DAY	X		X
DEALLOCATE	X		X
DEC	X	X	X
DECIMAL	X	X	X
DECLARE	X	X	X
DECLARE	X	X	X
DECRYPT	X	X	X
DEFAULT	X	X	X
DEFERRABLE	X		X
DEFERRED	X		X
DELETE	X	X	X
DESC	X	X	X
DESCRIPTOR	X		X
DIAGNOSTICS	X		X
DIRECTORY			X

Table 11: SESAM/SQL keywords

(part 3 of 10)

Keyword	ISO	OLD	FULL
DISCONNECT	X		X
DISTINCT	X	X	X
DOMAIN	X		X
DOUBLE	X	X	X
DROP	X		X
ELSE	X		X
ENCRYPT	X	X	X
END	X	X	X
END-EXEC		X	
ESCAPE	X	X	X
EXCEPT	X		X
EXCEPTION	X		X
EXEC	X	X	X
EXECUTE	X	X	X
EXISTS	X	X	X
EXP	X		X
EXPORT			X
EXTERNAL	X		X
EXTRACT	X		X
FALSE	X		X
FETCH	X	X	X
FIRST	X	X	X
FLOAT	X	X	X
FLOOR	X		X
FOR	X	X	X
FOREIGN	X	X	X
FOUND	X	X	X
FROM	X	X	X
FULL	X		X

Table 11: SESAM/SQL keywords

(part 4 of 10)

Keyword	ISO	OLD	FULL
GET	X		X
GLOBAL	X		X
GO	X	X	X
GOTO	X	X	X
GRANT	X	X	X
GROUP	X	X	X
HAVING	X	X	X
HEX_OF_VALUE	X	X	X
HOLD	X		X
HOUR	X		X
IDENTITY	X		X
IMMEDIATE	X	X	X
IMPORT			X
IN	X	X	X
INDICATOR	X	X	X
INITIALLY	X		X
INNER	X		X
INPUT	X		X
INSERT	X	X	X
INT	X	X	X
INTEGER	X	X	X
INTERSECT	X		X
INTERVAL	X		X
INTO	X	X	X
IS	X	X	X
ISOLATION	X		X
JOIN	X		X
JULIAN_DAY_OF_DATE			X

Table 11: SESAM/SQL keywords

(part 5 of 10)

Keyword	ISO	OLD	FULL
KEY	X	X	X
LANGUAGE	X	X	X
LAST	X	X	X
LEADING	X		X
LEFT	X		X
LEVEL	X	X	X
LIKE	X	X	X
LIKE_REGEX	X		X
LOAD			X
LOCAL	X		X
LOCALTIME	X		X
LOCALTIMESTAMP	X		X
LOWER	X		X
MATCH	X		X
MAX	X	X	X
MIGRATE			X
MIN	X	X	X
MINUTE	X		X
MOD	X		X
MODIFY			X
MODULE	X	X	X
MONTH	X		X
NAMES	X		X
NATIONAL	X		X
NATURAL	X		X
NCHAR	X		X
NEW	X		X
NEXT	X	X	X
NO	X		X

Table 11: SESAM/SQL keywords

(part 6 of 10)

Keyword	ISO	OLD	FULL
NORMALIZE	X		X
NOT	X	X	X
NULL	X	X	X
NULLIF	X		X
NUMERIC	X	X	X
NVARCHAR			X
OCTET_LENGTH	X		X
OF	X	X	X
OLD	X		X
ON	X	X	X
ONLY	X	X	X
OPEN	X	X	X
OPTION	X	X	X
OR	X	X	X
ORDER	X	X	X
OUTER	X		X
OUTPUT	X		X
OVERLAPS	X		X
PARTIAL	X		X
PERMIT		X	X
POSITION	X		X
POWER	X		X
PRECISION	X	X	X
PREPARE	X	X	X
PRESERVE	X		X
PRIMARY	X	X	X
PRIOR	X	X	X
PRIVILEGES	X	X	X
PROCEDURE	X	X	X
PUBLIC	X	X	X

Table 11: SESAM/SQL keywords

(part 7 of 10)

Keyword	ISO	OLD	FULL
READ	X	X	X
REAL	X	X	X
RECOVER			X
REF	X		X
REFERENCES	X	X	X
REFRESH			X
RELATIVE	X		X
REORG			X
REP_OF_VALUE	X	X	X
RESTORE		X	X
RESTRICT	X		X
RETURN	X	X	X
REVOKE	X		X
RIGHT	X		X
ROLLBACK	X	X	X
ROWS	X		X
SCHEMA	X	X	X
SCOPE	X		X
SCROLL	X	X	X
SECOND	X		X
SECTION	X	X	X
SELECT	X	X	X
SESSION	X		X
SESSION_USER	X		X
SET	X	X	X
SIGN			X
SIZE	X		X
SMALLINT	X	X	X
SOME	X	X	X
SORTED			X

Table 11: SESAM/SQL keywords

(part 8 of 10)

Keyword	ISO	OLD	FULL
SQL	X	X	X
SQLCODE		X	X
SQLERROR	X	X	X
SQLSTATE	X		X
SQRT	X		X
STORE		X	X
SUBSTRING	X		X
SUM	X	X	X
SYSTEM	X		X
SYSTEM_USER	X		X
TABLE	X	X	X
TEMPORARY	X	X	X
THEN	X		X
TIME	X		X
TIMESTAMP	X		X
TIMEZONE_HOUR	X		X
TIMEZONE_MINUTE	X		X
TO	X	X	X
TRAILING	X		X
TRANSACTION	X	X	X
TRANSLATE	X		X
TRANSLATION	X		X
TRIM	X		X
TRUE	X		X
TRUNC			X
UESCAPE	X		X
UNION	X	X	X
UNIQUE	X	X	X
UNKNOWN	X		X
UNLOAD			X

Table 11: SESAM/SQL keywords

(part 9 of 10)

Keyword	ISO	OLD	FULL
UPDATE	X	X	X
UPPER	X		X
USAGE	X		X
USER	X	X	X
USING	X	X	X
VALUE	X		X
VALUES	X	X	X
VALUE_OF_HEX	X	X	X
VALUE_OF_REP	X	X	X
VARCHAR	X		X
VARYING	X		X
VIEW	X	X	X
WHEN	X		X
WHenever	X	X	X
WHERE	X	X	X
WITH	X	X	X
WITHOUT	X		X
WORK	X	X	X
WRITE	X	X	X
YEAR	X		X
ZONE	X		X

Table 11: SESAM/SQL keywords

(part 10 of 10)

Related publications

You will find the manuals on the internet at <http://manuals.ts.fujitsu.com>. You can order printed versions of manuals which are displayed with the order number.

SESAM/SQL-Server (BS2000)

Core manual

User Guide

SESAM/SQL-Server (BS2000)

SQL Reference Manual Part 1: SQL Statements

User Guide

SESAM/SQL-Server (BS2000)

CALL-DM Applications

User Guide

SESAM/SQL-Server (BS2000)

Database Operation

User Guide

SESAM/SQL-Server (BS2000)

Utility Monitor

User Guide

SESAM/SQL-Server (BS2000)

Messages

User Guide

SESAM/SQL-Server (BS2000)

Performance

User Guide

ESQL-COBOL (BS2000)

ESQL-COBOL for SESAM/SQL-Server

User Guide

SESAM-DBAccess

Server-Installation, Administration (available on the manual server only)

BS2000OSD/BC

Commands

User Guide

ARCHIVE (BS2000)

User Guide

EDT (BS2000)

Statements

User Guide

HSMS (BS2000)

Hierarchical Storage Management System

User Guide

SECOS (BS2000)

Security Control System - Access Control

User Guide

XHCS (BS2000)

8-Bit Code and Unicode Processing in BS2000

User Guide

Index

In the index, **bold** page numbers refer to the main sources of the index entries, while *italicized* page numbers refer to examples. The collation sequence is as follows: symbols come before digits which come before letters. A punctuation mark is a symbol.

A

ADD PARTITION (Utility) **81**
ADD PARTITION ON SPACE (clause) **81**
ADD STOGROUP (clause) **79**
ALL DATA (clause) **131, 137**
ALLOWED-USAGE (HSMS operand) **39**
ALTER CATALOG (utility) **73**
ALTER DATA FOR TABLE (utility) **74**
ALTER MEDIA DESCRIPTION (Utility) **79**
ALTER MEDIA DESCRIPTION (utility) **76**
ALTER PARTITION (Utility) **84**
ALTER PARTITION ON SPACE (clause) **84**
ALTER PARTITIONING FOR TABLE **81, 84, 86**
ALTER PARTITIONING FOR TABLE (Utility) **80**
ALTER_CODE_TABLE (clause) **73**
altering
 media table **79**
anonymizing
 data **74**
ARCHIVE **97**
ARCHIVE parameter file
 creating **56**
 description **58**
 error situations **62**
 example **64**
 general rules **61**
 parameter summary **57**
 syntax rules **61**
ARCHIVE parameters
 COMPRESS **60**
 DEVICE **59**
 DIRSAVE **59**
 DRIVES **60**
 ENVIRONMENT-ATTRIBUTES **60**

 OPERATOR **59**
 RETPD **60**
 SAVE-ACL **59**
 SPACE **60**
 STREAM **59**
ARCHIVE run
 controlling **56**
 reserving resources **58**
ARCHIVE-NAME (HSMS operand) **39**
AT CATALOG (clause) **77, 116, 128, 176**
attribute format OLDEST **155, 156**
authorization
 ALTER CATALOG **73, 74**
 ALTER MEDIA DESCRIPTION **76**
 CHECK CONSTRAINTS **91**
 CHECK FORMAL **94**
 COPY **97**
 CREATE MEDIA DESCRIPTION **115**
 DROP MEDIA DESCRIPTION **127**
 EXPORT TABLE **130**
 IMPORT TABLE **134**
 LOAD **141**
 MIGRATE **167, 170**
 MODIFY **174, 175, 177**
 RECOVER INDEX **217**
 REORG **229**
 UNLOAD **235**
automatic counting field see counting field
availability
 logical **22**
 partition **22**
 physical **22**

B

backup archive 37
backup copy 211
BACKUP-FILES (HSMS statement) 37, 50
base table
 archiving 129
 copying 129
 importing 133
 migrating 129
 reallocating row numbers 129
 reconstructing 129
 reorganizing 227
BS2000 password 107, 121, 124, 159, 186, 194,
 197, 204, 212, 222, 231, 239, 240
BS2000 user ID 67, 113, 231
BY_ADD_MIRROR_UNIT 101
BY_SRDF_TARGET 101

C

CALL DML only table 155, 156, 170
 see table style
CALL DML table 169
CALL DML/SQL table 170
 see table style
CAT-LOG file 76
CAT-REC copy 51
CAT-REC file 76, 97, 114, 115, 127, 179, 206,
 214, 221
 foreign user ID 98
 offline update 103
 online update 103, 173, 177
 replication 119, 122
Catalog space 114
catalog space 13, 14
 create 73, 75
 creating 105, 114
 maximum size 105
 reorganizing 227
CATALOG_SPACE (clause) 108, 229
CATLOG file 97, 115, 127
CCSN 129, 133, 140, 144, 234
CHARACTER 34, 243

character
 DELIMITER 157
 ESCAPE 157, 248
character set
 coded 73, 105, 108
CHARACTER VARYING 34, 243
CHECK CONSTRAINTS (Utility) 91, 93
CHECK FORMAL (clause) 102
CHECK FORMAL (Utility) 94, 96
 replication 94
CHECK FORMAL (utility) 66
check pending 24, 25
check pending (space state) 13
CHECK pragma 24
checking
 formal correctness (table, index) 94, 96
 integrity constraint 91, 93
co-owner 52
Co-ownership 98, 107, 120, 189, 190, 200, 207,
 215, 216, 228, 232
co-ownership 36
 backups with HSMS 52
code table see coded character set
CODE_TABLE (clause) 108
coded character set 73, 105, 108
Coded Character Set Name 144, 234
coded character set name 129, 133, 140
column
 multiple 150, 238
column values
 shuffling 74
compound key 169
COMPRESS (ARCHIVE parameter) 60
COMPRESS-FILES (HSMS operand) 40
CONSTRAINT CHECK (load parameter) 153
controlling
 ARCHIVE run 56
 HSMS run 36
converting
 CALL DML only table 170
 CALL DML/SQL table 170
 database 166
COPY (clause) 125, 189, 200, 207, 215, 232
COPY (Utility) 97, 104, 166

- copy pending (space state) [13](#)
- COPY_FILE (clause) [185](#), [204](#), [211](#), [225](#)
- COPY_NUMBER (clause) [185](#), [204](#), [211](#), [225](#)
- counting field [160](#)
- COUNTING FIELD (clause) [160](#)
- CREATE CATALOG (Utility) [105](#), [114](#)
- CREATE CATALOG (utility) [73](#), [75](#)
- CREATE MEDIA DESCRIPTION (Utility) [115](#),
[118](#)
- CREATE REPLICATION (Utility) [119](#), [126](#)
- CREATE-ARCHIVE (HSMS statement) [37](#)
- creating
 - ARCHIVE parameter file [56](#)
 - HSMS archive [37](#)
 - HSMS parameter file [43](#)
- CSV format [28](#), [30](#)
- CSV_FORMAT (clause) [156](#), [247](#)
- D**
- D0STOGROUP [110](#), [118](#)
- DA_LOGS [171](#), [172](#)
 - modifying [175](#), [176](#)
- DA-LOG file [76](#), [97](#), [115](#), [127](#)
- data
 - anonymizing [74](#)
- data representation
 - overview [34–35](#)
 - printable [26](#)
- data type [26](#), [33](#)
- database [73](#), [105](#)
 - changing properties [73](#)
 - creating [105](#), [114](#)
 - foreign user ID [98](#), [105](#)
 - modify properties [73](#)
 - repairing [89](#), [178](#), [218](#)
- database file
 - foreign user ID [52](#), [65](#), [67](#), [95](#), [98](#), [105](#), [107](#),
[120](#), [131](#), [158](#), [231](#), [240](#), [251](#)
- database-specific file [76](#), [115](#), [127](#)
- DATE [32](#), [35](#), [243](#)
- daylight saving time [97](#)
- DB identifier [98](#), [105](#), [107](#), [122](#), [179](#)
- DBH user ID. [105](#)
- DCAM [113](#)
- DDL-TA-LOG file [76](#), [115](#), [127](#)
- DECIMAL [34](#), [243](#)
- default storage group [110](#), [118](#)
- default value character [243](#)
- DELETE AGE (clause) [174](#), [175](#), [177](#)
- DELETE ALL (clause) [174](#), [177](#)
- DELETE DATE (clause) [174](#), [175](#), [177](#)
- DELIMITER [157](#)
- DELIMITER character [27](#), [28](#), [29](#), [30](#), [155](#), [245](#),
[248](#)
- delimiter format [27](#), [29](#), [255](#)
- DELIMITER_FORMAT (clause) [155](#), [244](#)
- description ARCHIVE parameter file [58](#)
- DESTROY (clause) [109](#)
- DEVICE (ARCHIVE parameter) [59](#)
- DEVICE REQUEST (clause) [78](#), [117](#)
- DIRECTORY-NAME (HSMS operand) [40](#)
- DIRSAVE (ARCHIVE parameter) [59](#)
- DOUBLE PRECISION [33](#), [35](#), [243](#)
- DRIVES (ARCHIVE parameter) [60](#)
- DROP MEDIA DESCRIPTION (Utility) [127](#), [128](#)
- DROP PARTITION (Utility) [86](#)
- DROP PARTITION ON SPACE (clause) [86](#)
- DROP STOGROUP (clause) [79](#)

E

EDT [158](#)
ENVIRONMENT (HSMS operand) [39](#)
ENVIRONMENT-ATTRIBUTES (ARCHIVE parameter) [60](#)
error file [65, 66, 95](#)
 foreign user ID [65, 95, 158, 159, 251](#)
error situation
 ARCHIVE parameter file [62](#)
 HSMS parameter file [49](#)
ESCAPE [157, 248](#)
escape character (ESCAPE) [157, 248](#)
example [72](#)
 ALTER MEDIA DESCRIPTION [79](#)
 ALTER PARTITIONING FOR TABLE [89](#)
 ARCHIVE parameter file [64](#)
 CHECK CONSTRAINTS [93](#)
 CHECK FORMAL [89, 96](#)
 COPY [104](#)
 CREATE CATALOG [73, 75, 114](#)
 CREATE MEDIA DESCRIPTION [118](#)
 CREATE REPLICATION [126](#)
 delimiter format [255](#)
 DROP MEDIA DESCRIPTION [128](#)
 EXPORT TABLE [132](#)
 HSMS parameter file [49](#)
 IMPORT TABLE [139](#)
 LOAD [165](#)
 MODIFY [176, 177](#)
 RECOVER [218](#)
 REFRESH REPLICATION [222](#)
 REFRESH REPLICATION FOR SPACE [222](#)
 REFRESH SPACE [226](#)
 REORG [232](#)
 UNLOAD [255, 256](#)
EXCEPT NO LOG INDEX SPACE (clause) [103](#)
Exception file [159, 251](#)
exception file see error file
export file [129](#)
 CCSN [129, 131, 133](#)
 creating table from [133](#)
 foreign user ID [131](#)
EXPORT TABLE (Utility) [129, 132](#)

exporting

 metadata [129](#)
 user data [129](#)

F

file

 database-specific [76](#)
 error [65, 95, 251](#)
 input [65, 140, 144](#)
 output [65, 234, 240](#)
 space-specific [76](#)
FILL (SQL dialect) [263](#)
FOR SPACE (clause) [125](#)
FOREIGN [124](#)
foreign copy [119](#)
 RECOVER SPACE [182, 203](#)
FOREIGN COPY (clause) [225](#)
foreign user ID
 CAT-REC file [98](#)
 database [98, 105](#)
 error file [65, 95, 158, 159, 251](#)
 export file [131](#)
 output file [65, 240](#)
 replication [120](#)
 work file [231](#)
format check (table, index) [94, 96](#)
format description, data [33](#)
free space reservation [109, 227](#)
FROM CATREC (clause) [124](#)
FROM COPY_FILE (clause) [239](#)
FROM FILE (clause) [137](#)
FROM SPACE (clause) [239](#)

G

general rules

 ARCHIVE parameter file [61](#)
 HSMS parameter file [49](#)
GENERATE INDEX (load parameter) [152](#)
GENERATE INDEX ON NO LOG INDEX SPACE (clause) [212, 216](#)

H

hexadecimal encryption of SQL data types 34

host name 113

HSMS archive

changing system 54

changing user ID 52

co-ownership 52

creating 37

modifying archive directory 42

modifying attributes 42

HSMS backup

changing pubset 52

changing user ID 52

HSMS operands

ALLOWED-USAGE 39

ARCHIVE-NAME 39

COMPRESS-FILES 40

DIRECTORY-NAME 40

ENVIRONMENT 39

LOCATION 40

OPERATION-CONTROL 40

OWNER-FIELD 39

RETENTION-PERIOD 40

S2-DEVICE-TYPE 40

TAPE-CONTROL 41

USER-ACCESS 39

HSMS parameter file

creating 43

error situations 49

example 49

general rules 49

parameter description 45

parameter overview 44

syntax rules 48

HSMS parameters

HSMS-ARCHIVE-USERID 47

MAIL 48

PARALLEL-RUNS 48

REORGANIZE-SPACE 48

REQUEST-NAME 47

SAVE-ACL 47

SAVE-CATREC-COPY 46

SAVE-DIRECTORY 47

SAVE-FILES 45

HSMS run

controlling 36

HSMS statement

BACKUP-FILES 37, 50

CREATE-ARCHIVE 37

MODIFY-ARCHIVE 42

MODIFY-ARCHIVE-ATTRIBUTES 42

MODIFY-TAPE-CONTROL 37

RESTORE-FILES 37, 50

SHOW-ARCHIVE 51

HSMS-ARCHIVE-USERID (HSMS

parameter) 47

I

IMPORT TABLE (Utility) 133, 139

importing

base table 133

user data 133

index 133

rebuilding 217

INDEX (clause) 95

Input file 144

input file 65, 140

INTEGER 34, 243

Integrity constraint 91, 141

integrity constraint 133

checking 91, 93

INTO FILE (clause) 131, 240

INTO TABLE (clause) 149

ISO (SQL dialect) 263

J

job variable 120, 179

K**keyword**

SQL 263

L

- link name 67
- literal 148
- LOAD (Utility) 91, 140, 163, 234
- LOAD (utility) 66
- load column 150
- load description 145
- load format 33, 140
- LOAD formats 161
- load operation see loading
- load parameter 152
- load running (space state) 13
- LOAD_FORMAT (clause) 244
- loading user data 140, 163
- LOCATION (HSMS operand) 40
- lock 13
- LOG (clause) 102
- Logging 102, 143, 166
- logging 178
 - activating 102
 - deactivating 109
- logging file 179

M

- MAIL (HSMS parameter) 48
- media 115
- media record 127, 128
 - creating 115, 118
- MEDIA STOGROUP (clause) 111
- media table 76, 79, 115, 118
 - modify 76
- metadata exporting 129
- MIGRATE (Utility) 166
- migration 166, 170
- MINIMIZE (clause) 232
- MODIFY (Utility) 171, 176
- modify media table 76
- MODIFY-ARCHIVE (HSMS statement) 42
- MODIFY-ARCHIVE-ATTRIBUTES (HSMS statement) 42
- MODIFY-TAPE-CONTROL (HSMS statement) 37
- modifying database properties 73
- multiple column 150, 238

N

- name conflicts (during conversion) 169
- NATIONAL CHARACTER 34, 243
- NATIONAL CHARACTER VARYING 34, 243
- NEWLINE character 28
- NO CHECK FORMAL (clause) 102
- NO CONSTRAINT CHECK (load parameter) 91, 153
- NO CONSTRAINTS (clause) 139
- NO DATA (clause) 132, 138
- NO DESTROY (clause) 109
- NO DEVICE REQUEST (clause) 78, 117
- NO INDEX (clause) 138, 168, 186, 190, 194, 197, 201
- NO INDEX (load parameter) 152
- NO LOG (clause) 109
- NO OVERWRITE (load parameter) 152
- NO SHARE (clause) 78, 109, 117
- non-significant value 243
- notational conventions 9
- NULL value 148, 164, 243
- NUMERIC 34, 243

O

- occurrence 150, 238
- OFFLINE (clause) 99, 237
- offline backup 97
- OLD (SQL dialect) 263
- OLDEST 155, 156
- ON SPACE (clause) 92, 217
- ON TABLE (clause) 92, 217
- ON USER_ID (clause) 108, 122, 124, 189, 199, 206, 214, 222, 226
- ONLINE (clause) 99, 237
- online backup 97
- OPERATION-CONTROL (HSMS operand) 40
- OPERATOR (ARCHIVE parameter) 59
- output file 65, 234, 240
 - foreign user ID 65, 240
- output format 234, 242
- output formats 33
- OVERWRITE (load parameter) 152
- OWNER-FIELD (HSMS operand) 39

P

PARALLEL-RUNS (HSMS parameter) 48
partial replication 119, 125, 219
partition 22
 availability 22
Partitioned table 133, 135, 138, 149, 152, 159,
 160, 170, 217, 237, 251
partitioned table 22, 91, 139
password (BS2000) 107, 121, 124, 159, 186,
 194, 197, 204, 212, 222, 231, 240
PASSWORD (clause) 107, 121, 124, 131, 137,
 159, 168, 186, 194, 197, 204, 212, 222, 226,
 231, 239, 240
password catalog 167
password see password (BS2000)
PASSWORD_CATALOG (clause) 167
PBI file 76, 115, 127
PCTFREE (clause) 109, 227
POSITION 146, 242, 244
pragma
 CHECK 24
precompiler options
 SOURCE-PROPERTIES 263
predecessor version 166
PRIMARY (clause) 77, 108, 116
primary allocation
 defining 77, 116
Primary key 133
primary key
 multi-column 136
printable
 representation (of the data) 26
PUBLIC (clause) 112, 123
pubset, changing on HSMS backup 52

Q

QUOTE character 28

R

REAL 33, 35, 243
RECOVER (Utility) 89, 178, 218
RECOVER CATALOG (Utility) 208
 using backup copies 208
 using replications 213
RECOVER CATALOG_SPACE (Utility) 202
 using backup copies 202, 205
RECOVER INDEX (Utility) 217
recover pending (space state) 13
RECOVER SPACE (Utility) 181, 195, 198
 using backup copies 181
 using replications 187
RECOVER SPACESET (Utility) 192
RECOVERY_UNITS 171, 172
 deleting entries 103
 modifying 174, 175, 176
REFRESH REPLICATION (Utility) 219, 222
REFRESH SPACE (Utility) 223, 226
RENAME (clause) 125, 190, 200, 207, 216, 232
REORG (Utility) 227, 232
reorg pending (space state) 14
REORGANIZE-SPACE (HSMS parameter) 48
reorganizing
 base table 227
 database 227
 user space 227
repairing
 catalog space 178
 database 89, 178, 218
 space list 178
 space set 178
 user space 178
replication 189, 199, 206, 214
 CHECK FORMAL 94
 creating 119
 extending 223
 foreign user ID 120
 updating 219
represent
 date see data representation
representation, standard 33
REQUEST-NAME (HSMS parameter) 47

- reserving
 - resources (ARCHIVE run) 58
- resetting
 - catalog space 178
 - database 178
 - space list 178
 - space set 178
 - user space 178
- resources, reserving (ARCHIVE run) 58
- RESTORE-FILES (HSMS statement) 37, 50
- RETENTION-PERIOD (HSMS operand) 40
- RETPD (ARCHIVE parameter) 60
- S**
- S2-DEVICE-TYPE (HSMS operand) 40
- SAVE-ACL (ARCHIVE parameter) 59
- SAVE-ACL (HSMS parameter) 47
- SAVE-CATREC-COPY (HSMS parameter) 46
- SAVE-DIRECTORY (HSMS parameter) 47
- SAVE-FILE (HSMS parameter) 45
- SCOPE ALL (clause) 218
- SCOPE PENDING (clause) 186, 190, 194, 197, 200, 212, 218
- Search condition 132
- SECONDARY (clause) 77, 109, 117
- secondary allocation
 - defining 77, 117
- SECOS, co-ownership 52
- SECURE-RESOURCE-ALLOCATION 58
- separator (DELIMITER character) 157
- SESAM backup copies
 - managing 103
- SESAM backup copy 185, 204, 211, 225, 239
 - BS2000 file name 103
 - creating 97, 104, 126
- SESAM/SQL-Server 7
- SESAM/SQL-Server table 166
- SHARE (clause) 78, 109, 117
- SHOW-ARCHIVE (HSMS statement) 51
- shuffling column values 74, 75
- SKIP FIRST RECORDS (clause) 149
- SMALLINT 34, 243
- SOURCE-PROPERTIES (precompiler option) 263
- space
 - state 13
 - states 13
 - SPACE (ARCHIVE parameter) 60
 - SPACE (clause) 95, 229
 - space defect (space state) 13, 14
 - space list 178
 - repairing 178
 - space o.k (space state) 13
 - space parameter (clause) 77, 108, 116
 - space set 178, 191
 - space states
 - overview 15–20
 - space-specific file 76
 - special name 263
 - SQL data type, hexadecimal encryption 34
 - SQL database catalog 105
 - SQL dialect 263
 - SQL keyword 263
 - SQL table 169, 170
 - standard representation 33
 - state see space state
 - STOGROUP (clause) 110, 118
 - Storage group 110, 118
 - STREAM (ARCHIVE parameter) 59
 - symbolic host name. 113
 - syntax elements (overview) 257–262
 - syntax rules
 - ARCHIVE parameter file 61
 - HSMS parameter file 48
 - system user ID 112

T

table 13, 217
 CALL DML only 170
 CALL DML/SQL 170
 exporting 129
 importing 133
 modifying partitions 129
 partitioned 22, 91, 133, 135, 138, 139, 149,
 152, 159, 160, 170, 217, 237, 251
 SQL 170
 state “check pending” 13
 TABLE (clause) 94
 TAPE-CONTROL (HSMS operand) 41
 TIME(3) 32, 35, 243
 Timestamp 185, 188, 193, 197, 204, 211, 225
 TIMESTAMP(3) 32, 35, 243
 TO ... TABLE (clause) 169
 transfer file 158, 244
 TRANSFER_FORMAT (clause) 154, 244

U

universal user 105, 110, 112
 UNLOAD (Utility) 140, 234, 255
 UNLOAD (utility) 66
 unload column 238, 241
 unload description 241
 UNLOAD formats 253
 UNLOAD_FORMAT (clause) 154
 unloading
 user data 234, 255
 UNRESTRICTED (clause) 175, 176
 USER (clause) 112
 user data
 exporting 129
 importing 133
 loading 140, 163
 unloading 234, 255
 user ID (BS2000) 67, 113, 231
 changing on HSMS backup 52
 user space 13, 217
 reorganizing 227
 repairing 178
 resetting 178
 USER-ACCESS (HSMS operand) 39

USING DIRECTORY (clause) 100
 USING FILE (clause) 158, 230
 USING MEDIA (clause) 123
 USING PARTITION BY RANGE (clause) 135
 USING SPACE (clause) 135, 169
 USING STOGROUP (clause) 99
 utility functions (overview) 12
 utility monitor 219, 223
 utility see utility statement
 utility statement 7, 11, 69, 70
 UTM application name 113
 UTM user 113

V

value
 non-significant 243
 VOLUMES ... ON (clause) 112, 123

W

WHEN ... THEN NULL (clause) 148, 243
 WHEN NULL THEN (clause) 243
 WHERE, search_condition 132
 winter time 97
 WITH CATREC (clause) 206, 214
 WITH CONSTRAINTS (clause) 138
 WITH INDEX (clause) 138, 168
 work file
 foreign user ID 231
 work file (REORG) 230

X

XHCS 73, 108

