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.

1.1 Brief product description

SESAM/SQL is the relational database server for BS2000/OSD systems. SESAM/SQL combines the advantages of the relational data model with all the characteristics expected of a system which is subject to high loads in productive operation. On the one hand, this offers simple operation and data which is independent of the physical storage method used and, on the other, it means that the system is suitable for high transaction rates and large volumes of data and possesses outstanding security and availability characteristics.

The SQL interface implemented in SESAM/SQL has been based across the board on the ISO/IEC 9075:1992 standard and on ISO/IEC 9075:1999. This standardized SQL interface means that SESAM/SQL allows you to create portable, future-proof database applications which can be transferred to different database systems and operating systems.

SESAM/SQL fulfils all the demands placed on a modern database system in today's world:

- SESAM/SQL uses SQL, a uniform language and a consistent set of terms for defining, structuring and maintaining a relational database and for creating application programs.
- SESAM/SQL runs on all BS2000/OSD systems and can be used as a powerful SQL server for BS2000/OSD, SINIX/UNIX and MS-DOS/MS-Windows clients.
- SESAM/SQL excels in terms of high availability, security and data integrity.
SESAM/SQL fulfils all the security criteria for level C2 of the Department of Defense (Orange Book) and level F2/Q3 of the Bundesamt für Sicherheit in der Informationstechnik (BSI - The German Federal Authority for Information Technology Security) with the exception of the Security Audit Trail function (SAT) for providing evidence of operations. Trusted communications between the application program and the database system are provided along with trusted authentication of users.

SESAM/SQL supports modern parallel processing techniques for multi-user operation and multi-database processing.

The Universal Transaction Monitor openUTM and the SESAM/SQL database system together form a powerful DB/DC system including fully coordinated transaction processing and restart facilities for online applications.

The product SESAM/SQL-DCN allows transparent, efficient and trusted access to distributed databases in BS2000/OSD networks.

PC users can use SESAM-DBAccess to access SESAM/SQL databases.

A large range of add-on products increases the range of application of SESAM/SQL. These products range from database design tools, programming languages and third and fourth generation software development environments through to easy-to-use products for end users and the use of SESAM/SQL in World Wide Web applications.

**Brief description of CALL DML**

CALL DML is a CALL interface for processing SESAM/SQL databases. CALL DML provides statements for the following functions:

- open and close a CALL DML table, which must be part of a SESAM/SQL database
- add, update and delete data in a CALL DML table
- search and record output in a CALL DML table
- transaction-oriented security and administration of the DBH
1.2 Target group

This manual is intended for all CALL DML application programmers.

This manual provides the user with a description of CALL DML language constructs, a
description of how to create CALL DML programs, and a description of the CALL DML utility
routines together with examples.

In order to better understand the information contained in this manual, it would be helpful
to be familiar with the concept of transactions and have a basic knowledge of the BS2000
operating system and the Universal Transaction Monitor *openUTM*. 
1.3 Summary of the contents of the manuals

The documentation for the SESAM/SQL database system can be found in the following manuals:

- Core Manual
- CALL DML Applications
- Database Operation
- Utility Monitor
- Messages
- Glossary

The following additional documentation is also available:

- Migrating SESAM Databases and Applications to SESAM/SQL-Server
- Performance

The following manual describes how to create ESQL-COBOL programs:

- ESQL-COBOL User Guide

The following manuals describe remote access with SESAM-DBAccess:

- DBAccess V2.0A Client Installation, Administration, ODBC
- DBAccess V2.0A Service Installation, Administration
- DBAccess V2.0A ESQL/C
- SESAM-DBAccess (JDBC) V3.0A

If you are searching for information on a specific topic, you can use the table of contents, the index or the running headers. References to other documents are given in abbreviated form in the text. The complete title of the document referenced is included under ‘Related Publications’ at the back of the manual.
1.3.1 Summary of the contents of this manual

This manual describes the transfer areas at the CALL DML interface. It contains all DML statements and includes information on the old data types.

Examples of the DML statements are included in a separate chapter. All the examples are based on the same CALL DML tables.

Other chapters provide information on programming transactions, on compiling, linking and loading, and on using UTM and DCAM.

A description of the CALL DML utility programs SEDI61 and SEDI63, including examples, has been included in a separate chapter.
1.3.2 Guide to the SESAM/SQL manuals

Core Manual

The “Core Manual” provides an overview of the database system and describes basic principles, concepts and interrelationships. It provides the basis for understanding all the other SESAM/SQL manuals.

SQL Reference Manual
Part 1: SQL Statements and Part 2: Utilities

The SQL Reference Manual, Part 1 deals with the embedding of programs and describes the syntax and semantics of the SQL language constructs in alphabetical order.

The utility statements are not included in this alphabetical list and are dealt with separately in the “SQL Reference Manual, Part 2: Utilities”.

Both simple and complex examples are used to clarify the functions of the SQL language constructs.

The “ESQL-COBOL User Guide” explains how to create ESQL programs.

CALL DML Applications

This manual is aimed at CALL DML programmers and describes the language constructs used in the CALL DML interface and explains how to create CALL DML programs.

Database Operation

This manual is aimed at the system administrator and covers database operation. It includes details on starting and terminating the DBH and DCN and the associated load options and administration statements. The manual also describes the utilities required for database operation.

Utility Monitor

This manual describes how to use the utility monitor and the functions it provides. The utility monitor is a component part of SESAM/SQL and provides a menu-driven interface for creating, loading, backing up and reconstructing a database using SQL statements. The utility monitor also provides simple methods of querying the metadata.
Messages

This manual contains all the messages from the SESAM/SQL database system and the
distribution component SESAM/SQL-DCN. The messages are usually accompanied by
brief texts explaining the meaning and suggesting response measures. The SQL codes and
CALL DML status codes are also listed here.

Glossary

This manual contains the glossary.

Migrating SESAM Databases and Applications to SESAM/SQL-Server

This manual gives an overview of the new concepts and functions contained in
SESAM/SQL-Server V2. The main emphasis is on the relationship to previous versions in
order to facilitate migration to the new SESAM/SQL-Server environment for existing
SESAM/SQL users.

Performance

This manual is aimed at experienced SESAM/SQL users. It describes how users can
identify performance bottlenecks and indicates which parameters can be used to influence
system performance.

1.4 README file

Information on any functional changes and additions to the current product version
described in this manual can be found in the product-specific README file. You will find the
README file on your BS2000 computer under the name SYSRME.product.version.language.
The user ID under which the README file is located can be obtained from your systems
support staff. You can view the README file with the /SHOW-FILE command or in an editor,
and print it on a standard printer using the following command:

/PRINT-DOCUMENT filename, LINE-SPACING=*BY-EBCDIC-CONTROL

or, if SPOOL with a version earlier than 3.0A is used:

/PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=
    PARAMETERS(CONTROL-CHARACTERS=EBCDIC)

1.5 Changes in V3.0 made since V2.2

Changes in V3.0 made since V2.2

A list of the most important changes made in SESAM/SQL-Server V3.0 compared with V2.2 is contained in table 1. The table also shows the manual and chapter /section in which you will find a description of each change. If a topic is described in more than one manual, the one which contains a complete description is listed first. The entries in the “Manual” column have the following meanings:

<table>
<thead>
<tr>
<th>Core</th>
<th>Core Manual</th>
<th>RM P1</th>
<th>Reference Manual, Part 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM P2</td>
<td>Reference Manual, Part 2</td>
<td>DBO</td>
<td>Database Operation</td>
</tr>
<tr>
<td>Utilmon</td>
<td>Utility Monitor</td>
<td>CALL-DML</td>
<td>CALL-DML Applications</td>
</tr>
<tr>
<td>Perf</td>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic</th>
<th>Manual</th>
<th>Chapter/section</th>
</tr>
</thead>
<tbody>
<tr>
<td>System architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task concept, multitask operation</td>
<td>Core</td>
<td>7.1</td>
</tr>
<tr>
<td>Notational conventions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax descriptions are based on SDF Version 4.1A</td>
<td>DBO</td>
<td>1.6</td>
</tr>
<tr>
<td>Working with the SESAM/SQL-DBH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New: special start commands</td>
<td>DBO</td>
<td>2.1</td>
</tr>
<tr>
<td>DBH start statements and options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New DBH-TASKS</td>
<td>DBO</td>
<td>3.2</td>
</tr>
<tr>
<td>In the STORAGE-SIZE option, the STACK-POOL parameter is no longer used</td>
<td>Core</td>
<td>7.1</td>
</tr>
<tr>
<td>In the TRANSFER-CONTAINER option, the THREAD-BUFFER and REORGANISATION parameters are no longer used; the maximum values for the INITIAL and MAXIMUM parameters have been increased</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SESDCN control statements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New option: SESDLG-PASSWORD</td>
<td>DBO</td>
<td>4.5</td>
</tr>
<tr>
<td>Administration using SESADM</td>
<td>Core</td>
<td>7.11</td>
</tr>
<tr>
<td>New start command</td>
<td>DBO</td>
<td>5.1</td>
</tr>
<tr>
<td>Administration statements and administration commands</td>
<td>DBO</td>
<td>5.2</td>
</tr>
<tr>
<td>Various minor changes</td>
<td>DBO</td>
<td>5.2</td>
</tr>
<tr>
<td>SESMON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New: TASKS form</td>
<td>DBO</td>
<td>7.4</td>
</tr>
</tbody>
</table>

Table 1: Changes made in Version 3.0 since Version 2.2 (part 1 of 2)
### Changes in V3.0 made since V2.2

<table>
<thead>
<tr>
<th>Topic</th>
<th>Manual</th>
<th>Chapter/section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified: SERVICE ORDERS, STATEMENTS, SYSTEM-INFORMATION, TRANSACTIONS</td>
<td>DBO</td>
<td>7.4</td>
</tr>
<tr>
<td>Omitted: QUEUES form</td>
<td>DBO</td>
<td>7.4</td>
</tr>
<tr>
<td>Output to file</td>
<td>DBO</td>
<td>7.4</td>
</tr>
<tr>
<td>Modified: DBH record</td>
<td>DBO</td>
<td>7.5</td>
</tr>
<tr>
<td>Modified: output to SYSLST</td>
<td>DBO</td>
<td>7.6</td>
</tr>
<tr>
<td><strong>Error handling</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New: messages in special situations</td>
<td>DBO</td>
<td>8.2</td>
</tr>
<tr>
<td><strong>SEDI70</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New: editing the CAT-LOG and DA-LOG files</td>
<td>DBO</td>
<td>9.3</td>
</tr>
<tr>
<td><strong>CALL-DML</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Open flags allowed has changed</td>
<td>CALL-DML</td>
<td>3.2</td>
</tr>
<tr>
<td><strong>SEDI61</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New start command</td>
<td>CALL-DML</td>
<td>7.1</td>
</tr>
<tr>
<td><strong>SEDI63</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New start command</td>
<td>CALL-DML</td>
<td>7.2</td>
</tr>
<tr>
<td><strong>Distributed processing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication between different versions has been modified</td>
<td>Core</td>
<td>7.4, 7.4</td>
</tr>
<tr>
<td><strong>Files and job variables</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export via WORK files is no longer supported: WORK file has been omitted</td>
<td>Core</td>
<td>7.7, 7.7, 9.5</td>
</tr>
<tr>
<td>Job variables SESAM.START, SESAM.END and SESAM.ERROR are omitted</td>
<td>DBO</td>
<td>7.7, 9.5</td>
</tr>
<tr>
<td><strong>Buffers and containers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack pool and thread buffer are omitted</td>
<td>Core</td>
<td>7.7</td>
</tr>
<tr>
<td><strong>Utility monitor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New start command</td>
<td>Utilmon</td>
<td>3</td>
</tr>
<tr>
<td><strong>UTM interface</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Now described in a separate chapter</td>
<td>Core</td>
<td>9</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brought into line with V3.0</td>
<td>Perf</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Changes made in Version 3.0 since Version 2.2

(part 2 of 2)
1.6 Notational conventions

The DML statements are described by means of syntax diagrams using the following notation:

- → or ← Read direction left to right (default): the right-pointing arrow can be omitted.
- Read direction right to left
- → element Mandatory syntax element
- → element Mandatory syntax element: must be specified at least once and may be repeated
- → element Optional syntax element
- → element1 Syntax elements, just one of which must be specified
- → element2
- → element3
- S Subfunction that is described in a separate syntax diagram
- → ... End of a section of a syntax diagram (continued)
- → ... Continuation of a section of a syntax diagram

These forms of notation can be combined as required.
Each syntax diagram contains a header line which contains the names of the syntax elements, and also their displacement from the start of the statement and their length in the form displacement/length.

*Example:*

Password

The password must be entered at displacement 0 with a length of 3.

0/3

The displacement and length are specified in exactly the same way in the description of “Password”.

Syntax elements of a DML statement may be constants or variables:

- The *constants* must be entered by the user exactly as shown in the syntax diagram.

  *Example:*

  0

  You must enter “0” on issuing the statement.

- With *variables*, the user must substitute the appropriate value for the variable names. Names of variables are shown in lowercase.

  *Example:*

  san

  When you give this statement, you must substitute the appropriate value for “san”, e.g. “AD7”.

The following metacharacters are used to define the acknowledgment, response and inquiry areas:

- Braces enclose a series of values, of which one may occur.

  `{ }`

- Square brackets indicate that the contents of the brackets are optional.

  `[ ]`

- The contents of the field are of no interest to the user.

  `-`

- Repetition of the preceding specification.

  `...`
2 CALL DML interface

SESAM/SQL databases are accessed via application programs that contain DML statements to the SESAM/SQL DBH for retrieving and updating data and for administration.

- Retrieval statements are used to select individual attributes or records.
- Update statements are used to update individual attributes or selected records.
- Administration statements are used to issue commands to the SESAM/SQL DBH and SESAM/SQL-DCN.

Retrieval and update statements access a section of the CALL DML table. This section is referred to as a logical file and represents the user’s view of the CALL DML table at the time the database is accessed.

CALL DML can only be used to process CALL DML tables. In CALL DML tables there is a null attribute value for each attribute. When a CALL DML table is created with SQL, each attribute is automatically assigned a null attribute value (see “SQL Reference Manual, Part 1”, CREATE CALL DML TABLE).

DML statements can be tested with the SESAM utility routine SEDI63. This allows the user to ensure that all DML statements are correct before including them in the application program.
2.1 CALL DML calls

A database is accessed via the CALL interface of the various programming languages.

CALL DML calls can be used to access CALL DML tables in SESAM/SQL databases. A CALL DML call comprises the areas that make up a DML statement:

Statement area: This contains a definition of the statement to be executed by the SESAM/SQL DBH.

Acknowledgment area: The SESAM/SQL DBH returns status messages and, in certain cases, the number of response records here (execution messages in the case of statement execution, or error messages in the case of errors).

Response area: This is where retrieval statements return response records.

Inquiry area: This contains comparison values for retrieval statements in which comparison conditions are applied to attributes or primary key values. For direct updates, it contains the data to be updated or inserted, or the primary key value of a record to be deleted.

The application program transfers the areas of a DML statement for processing SESAM/SQL databases when the connection module is called.

The program must define the transfer areas with a sufficient length for these CALL DML calls. When the areas are passed to the connection module, it must be in the sequence in which they are listed above.

CALL DML calls can call the appropriate connection module at various entry addresses in the application program. The following table shows the possible entry addresses of the connection modules:

<table>
<thead>
<tr>
<th>Connection module</th>
<th>Entry addresses (CALL DML calls)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SESMOD</td>
<td>SESAM, SESPUT, SESGET, SESGETW</td>
</tr>
<tr>
<td>SESUTMC</td>
<td>SESAM</td>
</tr>
<tr>
<td>SESDCAM</td>
<td>SESAM, SESPUT, SESGET, SESGETW</td>
</tr>
</tbody>
</table>

Table 2: Entry addresses in the connection module
The CALL DML calls have the following meanings:

**SESAM**  The application program passes a DML statement to the DBH and fetches an acknowledgment and a response from the DBH.

**SESPUT**  The application program passes a DML statement and continues processing.

**SESGET**  The application program inquires whether a DML statement issued by means of SESPUT has received a response. If no response was returned, status message 83 appears in the acknowledgment area. The application program repeats the inquiry until a response is available.

**SESGETW**  The application program inquires whether a DML statement issued by means of SESPUT has received a response. The application program waits until a response is indicated.
2.2 Format and meaning of the transfer areas

Statement area

The application program places the DML statement in the statement area. The length of the statement area is variable.

Acknowledgment area

The acknowledgment area has a fixed length of 16 bytes.

In the acknowledgment area, the application program passes the file identifier of the logical file on which the DML statement is to operate.

After execution of a DML statement, the SESAM/SQL DBH places information about the execution of the statement in the acknowledgment area. The acknowledgment area has the following format:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>st</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
<td>The information placed in these bytes depends on the statement involved.</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier of the logical file</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td></td>
<td>The information placed in these bytes depends on the statement involved.</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td></td>
<td>The information placed in these bytes depends on the statement involved.</td>
</tr>
</tbody>
</table>

Table 3: Format of the acknowledgment area

The status must be interrogated by the application program and appropriate action taken by the program. Otherwise, a misleading status may occur in the next statement.

The status subnumber provides additional diagnostic information which cannot, however, be interpreted by the application program.
Response area

After a retrieval statement, the DBH places the first response record of the logical file in the response area. In block mode, the first group of response records is placed in the response area.

A response record contains:
– the primary key value (in the default case)
– the attribute values in the order in which the attributes were referenced in the statement, and in the format defined in the attribute catalog.

The maximum number of bytes per response that can be placed in the response area is defined in the open statement. Thus the length of the response area defined in the program must be large enough for this number of bytes.

The remaining responses can be output to the response area by means of the response polling statement.

Inquiry area

In retrieval statements that select records based on comparison values, the comparison values must be placed by the application program in the inquiry area, and their length must be that defined in the attribute catalog.

In direct update statements, the attribute values that are to be updated or inserted are placed in the inquiry area. For deletion, this area contains the primary key value of the record to be deleted.

The order of entries in the inquiry area must be the same as the order in which the attributes are specified in the statement.

Format of statement and inquiry areas

The statement and inquiry areas are of variable length and must therefore include a length field:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>length+4</td>
<td>Length of statement area (entry) + 4 or inquiry area (entry) + 4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>....</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>statement</td>
<td>DML statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inquiry</td>
<td>Inquiry area entries</td>
</tr>
</tbody>
</table>

Table 4: Format of the statement and inquiry areas
The length fields can either contain the maximum value or be set to the actual value dynamically by the program.

Please note:
The connection module transfers to the DBH statement and inquiry areas of the length specified. If the lengths defined are too long, unnecessarily long messages are passed by the application program to the DBH, thereby degrading the performance.

If the inquiry area is not analyzed by a statement, X'FFFFFFFF' can be passed to the DBH instead of the address of the inquiry area. If this cannot be done, an empty inquiry area must be passed with a value of 4 in the length field.

The length specification in the length field of the inquiry area is compared with the lengths of the attributes as defined in the attribute catalog. If the inquiry area is too small, the statement is rejected with a status code.

The minimum length of the inquiry area can be calculated as follows:

\[ 2 \times \text{length of primary key} \]

In a follow-up statement, the inquiry area must be formatted as for the preceding base statement. This applies even if the follow-up statement does not reference all the fields in the inquiry area. In the CALL DML call, the 4-byte length fields must not be transferred with the statement and inquiry area.

The application program can monitor the occurrence of errors by checking the acknowledgment area for particular status codes. The appropriate actions must be included in the program.

None of the four areas may be changed between a SESPUT call and a SESGET call. Otherwise, the SESGET call will be rejected with a status code.
2.3 Mixed operation of SQL and CALL DML interfaces

SESAM/SQL supports both CALL DML and SQL interfaces. Implementing ESQL-COBOL makes it possible to use both interfaces in the same application program. A COBOL program that is to employ both interfaces can sign on to the SESAM/SQL DBH with either a DML statement or an SQL statement:

- CALL DML statements only access resources reserved in the SESAM/SQL DBH specifically by CALL DML statements.
- Similarly, SQL statements can only access resources reserved by SQL statements.

A program is in

- CALL DML mode if the last statement to the SESAM/SQL DBH was a CALL DML statement
- SQL mode if the last statement to the SESAM/SQL DBH was an SQL statement

and is handled like a program with the appropriate interface. The format of the status codes is that of the last mode activated.

The following notes refer to mixed operation of CALL DML and SQL interfaces in the same application program:

- DML statements can only be used to process CALL DML tables.
- An application program that uses both interfaces has only signed off from the SESAM/SQL DBH correctly once the resources of both interfaces have been released.

For more information on the transaction concept, please also refer to the “Core Manual”.

The following applies to application programs that pass CALL DML statements to the SESAM/SQL DBH with SESPUT:

Switching from CALL DML mode to SQL mode is not possible if SESPUT calls are open for the task.
2.4 Examples of the various programming languages

CALL DML call in COBOL programs

Format:

CALL "SESAM" USING statement acknowledgment response inquiry.

The four transfer areas of the CALL DML call must be defined in the WORKING-STORAGE SECTION or, for subprograms, in the LINKAGE SECTION.

The following example shows a definition of the data areas and the corresponding CALL DML call:

```
01 STATEMENT-AREA.
   02 STA-LENGTH   PIC 9(4) COMP VALUE 57.
   02 FILLER       PIC XX       VALUE SPACE.
   02 STATEMENT.
      03 PASSWORD   PIC X(3).
      03 OPERATION  PIC X(50).

01 ACKNOWLEDGMENT.
   02 STATU        PIC XX.
   02 FILLER       PIC X(4).
   02 FILE-ID      PIC XX.
   02 FILLER       PIC X(8).

01 RESPONSE.
   02 PRIMARY-KEY  PIC X(8).
   02 RESPONSE-RECORD PIC X(70).

01 INQUIRY-AREA.
   02 INQ-LENGTH   PIC 9(4) COMP VALUE 12.
   02 FILLER       PIC XX       VALUE SPACE.
   02 INQUIRY      PIC X(8).

... PROCEDURE DIVISION.
...  CALL "SESAM" USING STATEMENT ACKNOWLEDGMENT RESPONSE INQUIRY.
  IF STATU NOT = "00"
    THEN
    GO TO STATPROC.
...  STATPROC.
...  User processing of status message
```
The length fields of the statement and inquiry areas must be 2-byte binary fields. They can be defined by means of the following PIC clauses:

```
PIC 9(num) COMP. where 1 ≤ num ≤ 4.
```

### CALL DML call in Assembler programs

```
LA   1,param
L    15,=V(SESAM)
BALR 14,15

Statement section:
```
STM  14,1,SAVE
LA   1,PARAM
L    15,=V(SESAM)
BALR 14,15
LM   14,1,SAVE

CLC  STATUS,C'00'
BE   ...processing
B    ...status handling

Definition section:
```
PARAM DC   A(STA...)
DC   A(ACK...)
DC   A(RES...)
DC   A(INQ...)
SAVE DS   4F
```
CALL DML call in PASCAL programs

Format:

sesam (sp.sta,sp.ack,sp.res,sp.inq);

Definition section:

type area = array(.1..1000.) of char;
field = 1..1004;
spacef = array(.1..2.) of char;
sespar = record

<table>
<thead>
<tr>
<th>statl</th>
<th>: lfield;</th>
<th>{statement area}</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats</td>
<td>: spacef;</td>
<td></td>
</tr>
<tr>
<td>stat</td>
<td>: area;</td>
<td></td>
</tr>
<tr>
<td>ack</td>
<td>: area;</td>
<td>{acknowledgment area}</td>
</tr>
<tr>
<td>resp</td>
<td>: area;</td>
<td>{response area}</td>
</tr>
<tr>
<td>inql</td>
<td>: lfield;</td>
<td>{inquiry area}</td>
</tr>
<tr>
<td>inqs</td>
<td>: spacef;</td>
<td></td>
</tr>
<tr>
<td>inq</td>
<td>: area;</td>
<td></td>
</tr>
</tbody>
</table>
end;

var sp : sespar;

procedure sesam (var s,a,r,i : area); external;
...

Statement section:

sesam (sp.sta,sp.ack,sp.res,sp.inq);
...

The underlined numbers are sample values and must be updated to suit the application.

CALL DML call in FORTRAN programs

Format:

CALLSESAM.(statement,acknowledgment,response,inquiryarea)
The statement areas should be set up with the aid of DIMENSION, COMMON, EQUIVALENCE and DATA.

Database operations can however only be performed successfully on attributes with data formats (such as INTEGER 2, INTEGER or REAL) which correspond with those of FORTRAN (i.e. halfword, word and double word).

**CALL DML call in PL/I programs**

**Format:**

```
CALL SESAM.(STATEMENT,ACKNOWLEDGMENT,RESPONSE,INQUIRY);
```

**Definition section:**

```
DCL SESAM ENTRY OPTIONS (ASSEMBLER):
DCL SESAREA AREA(6000):
  /                                                                  */
DCL 1 STA BASED (ZSTA),                                           /    STATEMENT AREA  */
  5 LENGTH BIN FIXED(15),
  5 SPACE CHAR(2) INIT(' '),
  5 STATEMENT,
  10 PASSWORD CHAR(3),
  10 OPCODE CHAR(1),
  10 TEXT CHAR(LSTA-4):
  /                                                                  */
DCL 1 ACKNOWLEDG,                                                /    ACKNOWLEDGMENT AREA */
  5 STATUS CHAR(2) INIT(' '),
  5 RES1 CHAR(4) INIT(' '),
  5 FILE-ID CHAR(2) INIT(' '),
  5 RES2 CHAR(2) INIT(' '),
  5 ADD-INFO CHAR(2) INIT(' '),
  5 RES3 CHAR(4) INIT(' '):
  /                                                                  */
DCL RESPONSE CHAR(LRSP) BASED (PRSP);                            /    RESPONSE AREA   */
  /                                                                  */
DCL 1 INQ BASED (PINQ),                                          /    INQUIRY AREA    */
  5 LENGTH BIN FIXED(15),
  5 SPACE CHAR(2) INIT(' '),
  5 INQUIRY CHAR(LINQ):
  /                                                                  */
DCL STATEMENTAREA CHAR(2044) VAR;                                 /    INPUT BUFFER    */
DCL INQUIRYAREA CHAR(2044) VAR;                                   /    INPUT BUFFER    */
DCL (LSTA,LRSP,LINQ) BIN FIXED(15);                               /    LENGTH FIELDS    */
DCL (PSTA,PRSP,LINQ) POINTER INIT (NULL);                         /    POINTER -> AREA  */
```
CALL DML call in ALGOL programs

Format:

CALLFO(SESAM,STA,ACK,RES,INQ);

Definition section:

'PROCEDURE' INITFO;'CODE';
'PROCEDURE' SESAM;'CODE',SESMOD;
'PROCEDURE' CALLFO;'CODE';
...

Statement section:

INITFO(1);
CALLFO(SESAM,STA,ACK,RES,INQ);
...

CALL DML call in RPG programs

RPG programs process SESAM/SQL databases by means of EXIT calls.

The transfer areas are formatted as data structures and transferred using the RLABL statement. Both data structure names and partial field names can be specified.

The length fields of the statement and inquiry areas must be 2-byte partial fields which can be set to the appropriate values by means of the Z-ADD statement.

C statements for the call section:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789012345678901234567890123456789012345678901234567890</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>EXIT SESAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>RLABL STATEM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>RLABL ACKNO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>RLABL RESPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>RLABL INQUI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data structure statements for the definition section:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890123456789012345678901234567890123456789012345678901234567890</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISTADS    DS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I          B 1  20STAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I          5 200 STATEM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I          5  7 PASSWO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I          8  8 OPCODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I          ... (redefinition)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IACKNO    DS  
I          1  2 STATUS  
I          ... (redefinition)  

IRESPO    DS  
I          1  8 CUST  
I          9  29 NAME  
I          ... (redefinition)  

INQDS    DS  
I          B 1  20INQL  
I          5 200 INQUI  
I          5  12 PKEY  
I          ... (redefinition)  

Existing RPG application programs not yet generated using the RPG3 compiler can continue to be used via the special file routine SESRPGS.
Eine Dokuschablone von Frank Flachenecker
by f.f. 1992
### 3 DML statements

#### 3.1 Overview of DML statements

<table>
<thead>
<tr>
<th>Function group</th>
<th>Statement</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical file management</td>
<td>Open</td>
<td>Open a logical file</td>
</tr>
<tr>
<td></td>
<td>Close</td>
<td>Close logical files</td>
</tr>
<tr>
<td>Retrieval functions</td>
<td>Search</td>
<td>Define a logical input file by selection, projection</td>
</tr>
<tr>
<td></td>
<td>Search with join</td>
<td>Define a logical input file by selection, projection and join</td>
</tr>
<tr>
<td></td>
<td>Restrict a join cursor file</td>
<td>Select responses after a search with join that has created a cursor file</td>
</tr>
<tr>
<td>Index browsing</td>
<td></td>
<td>Determine frequency of attribute values</td>
</tr>
<tr>
<td>Define comparison values</td>
<td></td>
<td>Replace the default mask character for a masked search</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace the default string identifier for the string search</td>
</tr>
<tr>
<td>Record output</td>
<td></td>
<td>Define a logical input file by projection and selection. Output the significant attribute values. Record output for tables with old data types</td>
</tr>
<tr>
<td>Inquiry</td>
<td></td>
<td>Define a logical input file by projection and selection. Output the significant and null attribute values. Inquiry on tables with old data types</td>
</tr>
</tbody>
</table>

Table 5: Overview of DML statements  
(part 1 of 2)
<table>
<thead>
<tr>
<th>Function group</th>
<th>Statement</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response polling</td>
<td>Process a logical input file after a search, record output or inquiry</td>
<td></td>
</tr>
<tr>
<td>Cursor file handling</td>
<td>Process cursor files after a search</td>
<td></td>
</tr>
<tr>
<td>Direct update functions</td>
<td>Addition</td>
<td>Add records</td>
</tr>
<tr>
<td>Update</td>
<td>Insert, update or delete attribute values or occurrences of a multiple attribute</td>
<td></td>
</tr>
<tr>
<td>Deletion</td>
<td>Delete records</td>
<td></td>
</tr>
<tr>
<td>Follow-up update</td>
<td>Process a logical file after a direct update</td>
<td></td>
</tr>
<tr>
<td>Set and delete deletion identifier</td>
<td>Define deletion identifier for deleting an attribute value or release deletion identifier (for old data types only)</td>
<td></td>
</tr>
<tr>
<td>Inquire on general table</td>
<td>Attribute information</td>
<td>Inquire on the attribute definitions of a table</td>
</tr>
<tr>
<td>Transaction-oriented security</td>
<td>Inquire on the attribute definitions of a table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attribute information for tables with old data types</td>
<td></td>
</tr>
<tr>
<td>Administration</td>
<td>Begin transaction</td>
<td>Initiate a transaction</td>
</tr>
<tr>
<td></td>
<td>End transaction</td>
<td>Terminate a transaction</td>
</tr>
<tr>
<td></td>
<td>Reset transaction</td>
<td>Reset a transaction</td>
</tr>
<tr>
<td></td>
<td>Administrator open</td>
<td>Open communication between administrator program and DBH</td>
</tr>
<tr>
<td></td>
<td>Administration statement for DBH</td>
<td>Control SESAM/SQL DBH</td>
</tr>
<tr>
<td></td>
<td>Administration statement for SESDCN</td>
<td>Control SESAM/SQL DCN</td>
</tr>
</tbody>
</table>

Table 5: Overview of DML statements (part 2 of 2)
3.2 Open

The open statement allows a requester to open a logical file. The logical file is identified by the file identifier. A user can thus open several logical files and process them in parallel. Logical files can only be opened for CALL DML tables (see the “SQL Reference Manual, Part 1”). Otherwise, the open statement is rejected.

In the following sections, the CALL DML table is generally referred to simply as the table.

The function identifiers in the open statement specify which operations can be performed under the specified file identifier and under the requester’s other file identifiers.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The DBH returns the acknowledgment to the statement.

The inquiry and response areas are not used, but must be made available.

Statement area

<table>
<thead>
<tr>
<th>Pass-word</th>
<th>Op. code</th>
<th>Table name</th>
<th>Length of resp.area</th>
<th>Length of inq.area</th>
<th>Func. code</th>
<th>File ident.</th>
<th>End ident.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/17</td>
<td>21/5</td>
<td>26/5</td>
<td>31/1</td>
<td>32/2</td>
<td>34/1</td>
</tr>
</tbody>
</table>

Key

Function identifier

<table>
<thead>
<tr>
<th>X</th>
<th>this file identifier: read</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>this file identifier: update</td>
</tr>
</tbody>
</table>

End identifier

<table>
<thead>
<tr>
<th>9</th>
<th>end of statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>chain statement</td>
</tr>
</tbody>
</table>
**Password (0/3)**

pas  Password for protected CALL DML table, any three-character string for unprotected CALL DML table.

**Operation code (3/1)**

2  Operation code for the open statement

**Table name (4/17)**

`tabname`

Name of the CALL DML table for which the logical file is to be opened. The table name must be entered in the CALL DML table catalog list (see the “Database Operation” manual, ADD-OLD-TABLE-CATALOG-LIST). Table names comprising less than 17 characters must be right-filled with blanks to a length of 17.

**Length of response area (21/5)**

SESAM/SQL knows the maximum length of the response area and can thus create buffers of the required size (see the “Database Operation” manual, TRANSFER-CONTAINER).

`length-R`

Maximum length of response area in bytes:
- The decimal number to be entered here is the longest response this application expects to receive.
- In block mode (&BLNnnn or &BLKnnn), the length must be multiplied by the number of response records per block (nnn).
- The length must be as exact as possible, to avoid excessive use of resources.

Minimum value: $2 \times (primary\ key\ length)$
Maximum value: 32000

Where block mode is used (&BLKnnn or &BLNnnn), the following should be noted:

If the response area is too small to hold a full block, a blocking factor is defined internally based on the size of the response area. The number (nnn) defined in the statement by &BLKnnn or &BLNnnn is ignored in response output.
**Length of inquiry area (26/5)**

SESAM/SQL knows the maximum length of the inquiry area and can thus create buffers of the required size (see the “Database Operation” manual, TRANSFER-CONTAINER).

**length-I**
- Maximum length of inquiry area in bytes:
  - The decimal number to be entered here is the total of the primary key and attribute comparison values.
  - The length must be as exact as possible, to avoid excessive use of resources.
  - Minimum value: $2 \times \text{(primary key length)}$
  - Maximum value: 32000

**Function code (31/1)**

The function code defines, for the period during which the file identifier applies, the functions permitted under this file identifier.

At the same time, a hierarchical structure defines whether, in conjunction with transaction-oriented security, a record is locked exclusively or is shared. The function code X requests exclusive locks.

- **X** Direct updating may be performed under this file identifier.
  - Function identifier X is normally used in online application programs.

- **R** Retrieval functions only are performed under this file identifier.
  - Function R thus provides no security against data being changed under other file identifiers.

**File identifier (32/2)**

The file identifier identifies a logical file under which retrieval and/or direct update functions can be performed.

- **ff** File identifier which must be entered in the acknowledgment area for all subsequent statements.
  - The permitted characters are numbers 0 to 8 and any letter.
End identifier (34/1)

9     Indicates the end of the statement
;
End of statement. The statement is chained to a subsequent open statement and passed in a call to the DBH. The following rules must be observed:

– Open statements can be chained as often as required. Thus several open statements can be sent to the DBH in one call, thereby reducing processing time.
– In the last open statement, 9 must be specified as the end identifier.
– When all open statements have been executed, the application program receives status code 00 and the file identifier of the last open statement.
– Execution of the open statements is terminated if any one of them cannot be carried out. The status code and the file identifier of the failed open statement are output in the acknowledgment area.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>MOD..</td>
<td>Reported by DBH, by SESMOD, by SESDCN or DVS error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DCN..</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dxxx</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: Acknowledgment area after opening the logical file
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(10)</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(20)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(21)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(22)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(25)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(26)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(29)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2I)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2M)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2S)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2U)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2X)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2Y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2Z)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>(\text{MOD}_{..})</td>
<td>Reported by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\text{DCN}_{..})</td>
<td>- DBH,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\text{Dxxx})</td>
<td>- SESMOD,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SESDCN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- or DVS error</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>(ff)</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>(uu)</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7: Acknowledgment area on an error
3.3 Close

The close statement is used to close logical files.

The close statement carries out the following specific functions:

user close  Close all logical files for this requester (user)
file close  Close just one logical file for this requester (user)

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:
On a file close, the application program must supply the file identifier.
The DBH returns the acknowledgment to the statement.

The inquiry and response areas are not used, but must be made available.

User close

The user close closes all logical files for a requester.

If the user close occurs within transaction boundaries, the following applies:

– A user close is only permitted within transaction boundaries if all logical files were also opened within the transaction. If a logical file was opened outside the transaction, the close statement is rejected with status code 8T.

– If the transaction is reset after the close, the close will already have released all resources used by the logical files. Thus the logical files will no longer be available after the reset.

File close

Logical file ff belonging to the requester is closed.

If the file close occurs within transaction boundaries, the following applies:

– A file close within a transaction is permitted if the logical file was also opened in the transaction.

– If the transaction is reset after the close, the close will already have released all resources used by the logical files. Thus the logical files will no longer be available after the reset.
**Statement area**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/17</td>
<td>21/-</td>
</tr>
</tbody>
</table>

User close: pas → 8 → ff → 9

File close:

**Password (0/3)**

pas  Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

**Operation code (3/1)**

8  Operation code for the close statement

**Close function (4/17), (21/-)**

ff  File identifier of the logical file to be closed (file close)

**End identifier (-/1)**

9  Indicates the end of the statement
### Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>( {00, 10, 80, 8N, 8C, 8T, 9U} )</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>[ff]</td>
<td>File identifier (file close)</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>uu</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8: Acknowledgment area
3.4 Search

The search defines a logical input file, also called a view. Records can be selected conditionally based on primary key value or on attribute values (selection).

The search also defines which attributes a record of the logical input file should contain (projection). The records of a logical input file can also be sorted.

The record numbers of a logical input file can also be stored in a cursor file for further processing. This cursor file can be restricted, overwritten or processed by another search.

A search can handle a maximum of 256 attributes or occurrences of a multiple attribute. If this number is not sufficient, it can be increased to a maximum value of 1024 when the SESAM/SQL DBH is loaded (see the “Database Operation” manual, COLUMNS).

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier, and the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values, the application program must make these available in the inquiry area.
## Statement area

<table>
<thead>
<tr>
<th>Password</th>
<th>Op.-code</th>
<th>PK function</th>
<th>Strategy</th>
<th>Subquestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/-</td>
</tr>
</tbody>
</table>

![Diagram showing the relationship between the password, operation code, PK function, strategy, and subquestions.](image-url)

- The diagram illustrates how different values in the password, operation code, and PK function lead to specific strategies and subquestions. Each element is connected with arrows indicating the flow or relationship between them.

- The diagram also shows how the values evolve from one section to another, with arrows pointing to the next step in the process.

- The illustration helps to visualize the logic and decision-making processes involved in managing the password, operation code, and PK function in the context of the strategy and subquestions.
Block mode  Lock mode  End id.
-7  -7  -7  -7  -1

...  &BLKnnn  &PSN000  &RNL000  &RNW000  ;
Key

PK (primary key)  0    all data
    function       1    equal to PK group value
    2    range of PK group values
    3    larger than PK group value
    4    equal to PK value
    5    range of PK values
    6    larger than PK value
    8    equal to record number

Strategy  0    sequential search, output responses
    1    output responses
    Y    output number of responses
    A    restrict cursor file, output responses
    Z    record numbers in cursor file, output number of responses
    P    restrict cursor file, store record numbers in cursor file and
         output number of responses

Subquestions  C    AND operator, projection, selection
    L    OR operator, projection, selection
    U    AND operator, selection
    O    OR operator, selection
    E    projection
    Z    count significant occurrences of a multiple attribute
    T    sort responses on secondary index, projection
    S    sort responses, projection
    R    selection by boundary condition, projection

End id.  9    end of statement
          ;    chain statement

The suggested order of T, R and S subquestions as the last subquestions in the
search, as shown in the diagram, is recommended but not mandatory.

The following rules must always be observed:

- S subquestions must follow U, O, C, L and R subquestions. A maximum of
  6 S subquestions are allowed in a search.
- R subquestions must follow U, O, C and L subquestions and come before
  S subquestions. A maximum of 6 R subquestions are allowed in a search.
- The T subquestion must not immediately precede an O or L subquestion. It is
  not permitted if the search contains R or S subquestions.
Password (0/3)

pas    Password for protected CALL DML table,
       any three-character string for unprotected CALL DML table.

Operation code (3/1)

6    Operation code for the search statement

Primary key function (PK function) (4/1)

The primary key function allows records to be selected by applying conditions to the primary
key or record number. The comparison values are placed in the inquiry area. A primary key
value can be used for selection instead of the whole primary key value:
The primary key group value identifies a group of records whose primary key value contains
the primary key group value in the left of the key. The comparison value in the inquiry area
must be blank-filled to the full length of the primary key. A blank at the end of the comparison
value will therefore not be recognized.

With compound keys, the only primary key group value that may be used is a value of the
compound key attribute AAB or of several compound key attributes (starting at AAB,
ascending AAC, etc.), with the compound key attribute whose value is furthest to the right
in the primary key group value not being of type INTEGER or SMALLINT. It is possible,
however, for the compound key attribute furthest to the right to be only partially covered by
the primary key group value. Blanks must be inserted in the inquiry area for the remaining
compound key attributes.

0    All data:
       All records are selected.

1    Equal to primary key group value:
       A primary key group value must be entered in the inquiry area as comparison value.
       All records containing the primary key group value left-justified in their primary key
       value are selected.

2    Range of primary key group values:
       Two primary key group values that define a range of primary key group values must
       be placed in the inquiry area.
       All records whose primary key values are greater than or equal to the first
       comparison value and less than or equal to the second comparison value are
       selected.
       The first comparison value must not be larger than the second comparison value.
Search

3 Greater than primary key group value:
A primary key group value must be entered in the inquiry area as the comparison value.
All records whose primary key value is greater than the comparison value are selected.

4 Equal to primary key value:
A primary key value must be entered in the inquiry area as the comparison value.
The record whose primary key value is equal to the comparison value is selected.

5 Range of primary key values:
Two primary key values defining a range of primary key values must be entered in the inquiry area as comparison values.
All records whose primary key values are greater than or equal to the first comparison value and less than or equal to the second comparison value are selected.
The first comparison value must not be larger than the second comparison value.

6 Greater than primary key value:
A primary key value must be entered in the inquiry area as the comparison value.
All records whose primary key value is greater than the comparison value are selected.

8 Equal to record number:
A record number must be entered in the inquiry area as the comparison value.
The record with the specified record number is selected.

Strategy (5/1)
The strategy specifies the type of result the search should return. The following types of result are possible:

- Response records in the response area.
- Number of responses in the acknowledgment area.
- Creation of a cursor file containing the record numbers of the response records.
The strategy also defines the search method to be used by the search:

- Strategy 0 defines a sequential search of the table.
- All other strategy specifications (1/Y/Z/A/P) leave it to SESAM/SQL to select the most appropriate search method: sequential search or search via index.

If in any one session the index of more than 4 attributes is defective, the search is sequential.
If the index of an attribute referenced by a T subquestion (see “Search subquestions” on page 45) is defective, the search is terminated with status code 9E.

0 The search processes the records sequentially and returns the first response record in the response area. If block mode (&BLKnnn or &BLNnnn) is in use, the first nnn responses are placed in the response area. Further responses can be retrieved using the response polling statement (xxx719 or xxx799, see section “Response polling” on page 117). The responses are returned in ascending order of primary key values (sequential search).

1 SESAM/SQL determines the search method; otherwise synonymous with strategy 0.

Y The search counts the response records and places the result in the acknowledgment area with status code 10. The response records can be retrieved with updated response polling xxx719 and all successor responses with xxx799.

Z The search counts the response records, stores the record numbers of the response records in a cursor file, and places the number of responses in the acknowledgment area with status code 10. The responses can be retrieved from the cursor file with response polling statement xxx729, and the successor responses with xxx709.

A Precondition:
A cursor file has been created by a preceding search.

If the search specifies the same conditions as those used to create the cursor file, the search returns the response record whose record number is the first in the cursor file (or nnn records in block mode).
If the conditions are different, the records pointed to by the cursor file are tested for these conditions and the first nnn matching responses output. Successor responses can be retrieved with response polling statement xxx799.
P Precondition:
A cursor file has been created by a preceding search.

The search processes only those records whose record numbers are stored in the cursor file. The old cursor file is restricted, the responses to the new search counted and the record numbers of the responses written to the old cursor file. The contents of the old cursor file are overwritten. The number of responses is placed in the acknowledgment area with status code 10. The response records can be retrieved with statement xxx729, and successor responses with xxx709.

Subquestions (6/-)

Subquestions firstly allow the conditions by which records are to be selected to be specified, and secondly allow selection of the attributes whose values are to be output in the response record (projection). The response records can be sorted on output either by attribute values or by index values. With a multiple attribute, the number of occurrences containing a significant value can be counted.

For a detailed description of subquestion elements see “Search subquestions” on page 45.

Block mode (-/7)

The user can define how many of the responses found are to be returned in the response area.

&BLKnnn
nnn responses are placed in the response area. The record number of each record is output.

&BLNnnn
nnn responses are placed in the response area. The responses are output without record numbers.

If neither &BLKnnn nor &BLNnnn is specified, the default is to place just one response record without its record number in the response area.
Lock mode (-/7)

&PSN000
  The response records are output without the primary key value.

&RNL000
  The record accessed by the search within a transaction is not locked.

&RNW000
  The search can read a record that has been locked by another transaction (dirty read). The statement is acknowledged with status code 9S. In block mode, no more responses are output to the response area after a dirty read. Further response records may, however, be read by response polling statement xxx799. If &RNW000 is omitted, a transaction that attempts to access a locked record is placed in a wait state until the record is released.

End identifier (-/1)

9 Indicates the end of the statement
;
End of statement. The statement is linked with a subsequent end TA statement.

Search subquestions

Subquestions within searches perform the following functions:

– Formulation of selection criteria and combinations thereof on which records are selected.

– Defining which attributes are to be projected in the response record.

– Formulation of sort conditions and combinations thereof for response output.

– Counting significant occurrences of a multiple attribute.

– Testing whether an attribute has a (non-)significant value.
Logical relationships between subquestions

Multiple subquestions that produce a selection are logically ANDed or ORed together. Note that unlike boolean logic, the OR in SESAM/SQL is the stronger relationship. As SESAM/SQL does not permit parenthesizing of subquestions, “multiplying out” must be used where necessary to achieve the same effect as parentheses. If several attributes are specified in a condition within a subquestion, SESAM/SQL connects the different attributes together with OR.

Example

a, b, c and d represent subquestions within a search. The logical relationships are represented by AND and OR.

The boolean expression (a AND b) OR (c AND d) is to be represented in a subquestion. To represent it in SESAM/SQL logic, it must be “multiplied out” by relating each element in the first parenthesis with each element in the second parenthesis. The logical operator is the operator between the two expressions in parentheses (OR). This gives the following representation:

(a OR c) AND (a OR d) AND (b OR c) AND (b OR d)

The parentheses in this expression are superfluous in SESAM/SQL logic, as the OR operator creates a stronger relationship than the AND operator.

Maximum number of attributes in a search

A search can reference a maximum of 256 attributes and occurrences of a multiple attribute. This value can be increased to 1024 when the SESAM/SQL DBH is loaded (see the “Database Operation” manual, COLUMNS).

Old subquestion types

Existing application programs may still contain old types of subquestion. In this case, subquestions A, B and D are interpreted as C subquestions, and J, K and M as L subquestions.
<table>
<thead>
<tr>
<th>Subquestion type</th>
<th>Attribute selection</th>
<th>Search condition</th>
<th>Comparison condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>-/1</td>
<td>-/-</td>
<td>-/1</td>
<td>-/2</td>
</tr>
</tbody>
</table>

- **S**
  - S → san
  - san/mmm/
  - 7 → 32
  - 8 → 34

- **R**
  - R → san
  - san/mmm/
  - 7 → 12
  - 8 → 14
  - 31
  - 39
### Key

<table>
<thead>
<tr>
<th>Search condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no condition</td>
</tr>
<tr>
<td>4</td>
<td>string and mask search</td>
</tr>
<tr>
<td>5</td>
<td>conditions</td>
</tr>
<tr>
<td>6</td>
<td>negated conditions</td>
</tr>
<tr>
<td>8</td>
<td>turn off search conditions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparison condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>no comparison condition</td>
</tr>
<tr>
<td>01</td>
<td>= (equal to)</td>
</tr>
<tr>
<td>02</td>
<td>&lt; (less than)</td>
</tr>
<tr>
<td>03</td>
<td>≤ (less than or equal to)</td>
</tr>
<tr>
<td>04</td>
<td>&gt; (greater than)</td>
</tr>
<tr>
<td>05</td>
<td>≥ (greater than or equal to)</td>
</tr>
<tr>
<td>06</td>
<td>≠ (not equal to)</td>
</tr>
<tr>
<td>23</td>
<td>from ... to</td>
</tr>
<tr>
<td>24</td>
<td>outside from ... to</td>
</tr>
<tr>
<td>80</td>
<td>skip one inquiry area entry</td>
</tr>
<tr>
<td>82</td>
<td>skip two inquiry area entries</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fixed combinations from search and comparison conditions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>check for significant value</td>
</tr>
<tr>
<td>200</td>
<td>check for non-significant value</td>
</tr>
<tr>
<td>712</td>
<td>next smallest to comparison value</td>
</tr>
<tr>
<td>714</td>
<td>next largest to comparison value</td>
</tr>
<tr>
<td>731</td>
<td>lowest value</td>
</tr>
<tr>
<td>739</td>
<td>highest value</td>
</tr>
<tr>
<td>732</td>
<td>sort in descending order</td>
</tr>
<tr>
<td>734</td>
<td>sort in ascending order</td>
</tr>
</tbody>
</table>

### Subquestion type (-/1)

**C**: C subquestion:
The C subquestion allows records to be selected conditionally on an attribute value (selection).
The value of the referenced attribute is output for each response record (projection).
A C subquestion is logically ANDed with a preceding subquestion. If the C subquestion is the first subquestion in a search, it is ANDed with the primary key function.
If the same condition is applied to several attributes, the values of all attributes specified for this condition are output. The attributes in the C subquestion are then ORed.
When the attribute values are output, null attribute values are replaced by the default value. To specify multiple attributes in a C subquestion, it is a precondition that the data definitions have the same
- attribute length,
- data type,
- number of decimal places and
- default value.

U U subquestion:
The U subquestion allows records to be selected conditionally based on an attribute value (selection). There is no projection of attribute values in the response record. The other function characteristics are the same as the C subquestion.

L L subquestion:
The L subquestion allows records to be selected conditionally based on an attribute value (selection).
The value of the referenced attribute is output in the response record (projection).
An L subquestion is logically ORed with a preceding subquestion. The L subquestion must not be the first subquestion making a selection.
If the same condition is applied to several attributes, the values of all attributes specified for this condition are output.
When the attribute values are output, null attribute values are replaced by the default value. To specify multiple attributes in an L subquestion, the data definitions must have the same
- attribute length,
- data type,
- number of decimal places and
- default value.

O O subquestion:
The O subquestion allows records to be selected conditionally based on an attribute value (selection). There is no projection of attribute values in the response record. The other function characteristics are the same as the L subquestion.

E E subquestion:
The E subquestion is used only for projection. The attribute values of the attributes listed are placed in the output record. Null attribute values are replaced by default values.

Z Z subquestion:
The Z subquestion counts the occurrences containing a significant attribute value in a multiple attribute.
**T subquestion:**
The T subquestion allows records to be selected on the basis of index values (selection). The value of the referenced index is projected in the response record. The responses are sorted in ascending order of index values (see “Sorting the response records” on page 60). Records where the index attribute does not contain a significant value are not selected.

**S subquestion:**
The S subquestion defines the sort sequence of response records. S subquestions are only permitted with strategy Z or P (creation of a cursor file).
A search may contain a maximum of 6 S subquestions with a total maximum of 6 attributes and 6 sort criteria. The first attribute of the first S subquestion is the high-order sort criterion. Default values are output for null attribute values.

The application program must call the SESORT module. The utility SEDI63 calls SESORT automatically.

A search with S subquestion may not be initiated whilst a sort procedure using the SORTWORK file is still running in BS2000 (e.g. COBOL SORT) for the application program. Otherwise, status 1G results if the SORTWORK file is required for the search to be processed.

**R subquestion:**
The R subquestion allows records to be selected based on so-called boundary conditions. Boundary conditions are:
- smallest attribute value
- largest attribute value
- next smallest attribute value to the comparison value in the inquiry area
- next largest attribute value to the comparison value in the inquiry area

An R subquestion operates on the response set of all preceding C, L, U or O subquestions. If there was no preceding subquestion creating a selection, the R subquestion operates on the whole data base. A search can contain a maximum of 6 R subquestions. Each R subquestion contains one symbolic attribute name and one boundary condition.

Hierarchical structure of R subquestions:
A hierarchical relationship exists between R subquestions. This means that a second or subsequent R subquestion can only be executed if the R subquestion just processed has returned more than one response. If only one response was returned, it is output immediately.

Default values are output for null attribute values.
**Attribute selection (/-)***

The subquestion refers to the attributes with the specified symbolic attribute names (SAN).

- **san** symbol attribute name of an attribute or of the primary key (AAA).
  
  For a compound key, individual compound key attributes (e.g. AAB) or the complete compound key may be specified.

- **san/mmm/**
  
  Symbolic attribute name and occurrence number (mmm) of a multiple attribute.

- **san/mmm-nnn/**
  
  Symbolic attribute name of the multiple attribute of which occurrences mmm to nnn are to be processed.

A search can reference a maximum of 256 attributes and occurrences. This value can be increased to a maximum of 1024 when loading the SESAM/SQL DBH (see the “Database Operation” manual, COLUMNS).

**Search condition and comparison condition**

Conditions applied to attribute values can be constructed using a search condition and one or more comparison conditions.

**Search condition (/-1)**

- **0** No condition applied to attribute value.

- **1** The referenced attributes are tested to see whether a significant value has been stored for any of them in the record. The subquestion is satisfied if at least one of the attributes contains a significant value.

- **2** The referenced attributes are tested to see whether a null value has been stored for any of them in the record. The subquestion is satisfied if at least one of the attributes contains a null value.

- **4** String and mask search:
  
  String and mask searches can only be used on attributes defined with data type CHAR.

  String search:
  
  Only the comparison conditions 01 and 06 are permitted. An attribute value is checked to see whether it contains a string or not. The comparison value in the inquiry area must be enclosed in string identifiers. The length of the string is:

  \[
  1 \leq \text{string length} \leq \text{attribute length}-2.
  \]

  The default string identifier is “%”. It can be changed by means of the ‘set string identifier’ statement (see section “Define comparison values” on page 92).
Mask search:
An attribute value is tested for a specific character in a particular position. The mask character is substituted for non-relevant characters. The subquestion is satisfied if the known positions satisfy the comparison condition.

The default mask character is “?” and can be substituted by a different character by means of the “set mask character” statement.

Mask characters and string identifiers cannot be mixed in one comparison value.

Search condition 4 can only be used with compound keys if all the compound keys have the data type CHAR.

5 The attribute value is tested for the comparison condition. If more than one condition is specified, the subquestion is satisfied providing the attribute value satisfies one of the comparison conditions (OR relationship).

6 The attribute value is tested for the comparison condition. If more than one condition is specified, the conditions are ORed. The negation then refers to the whole comparison expression.

7 Search condition for formulating boundary conditions and sort criteria.

8 Cancel search conditions (see “Flexible construction of subquestions” on page 62)

**Comparison condition (0-2)**

00 no condition; the inquiry area contains no comparison value
01 equal to comparison value in inquiry area
02 less than comparison value in inquiry area
03 less than or equal to comparison value in inquiry area
04 greater than comparison value in inquiry area
05 greater than or equal to comparison value in inquiry area
06 not equal to comparison value in inquiry area
23 greater than or equal to first comparison value and also less than or equal to second comparison value in inquiry area
24 less than first comparison value or greater than second comparison value in inquiry area

*Turn off comparison conditions*

80 skip an inquiry area entry (see “Flexible construction of subquestions” on page 62)
82 skip two inquiry area entries
Formulating boundary conditions

12  next smallest value to comparison value in inquiry area
14  next largest value to comparison value in inquiry area
31  smallest value
39  largest value

Formulating sort criteria

32  sort descending
34  sort ascending

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: Acknowledgment area for response
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0      | 2      | \{1K  \\
|        |        | 60  \\
|        |        | 66  \\
|        |        | 67  \\
|        |        | 6B  \\
|        |        | 6C  \\
|        |        | 6D  \\
|        |        | 6T  \\
|        |        | 6U  \\
|        |        | 6W  \\
|        |        | 6Z  \\
|        |        | 9O  \\
|        |        | 9E  \\
|        |        | 9Q  \\
|        |        | 1F  \\
|        |        | 1G  \\
|        |        | \} only for searches \{ with S subquestions |
| 2      | 4      | -     | -                                |
| 6      | 2      | ff    | File identifier                 |
| 8      | 2      | -     | -                                |
| 10     | 2      | ss    | Status subnumber                |
| 12     | 4      | -     | -                                |

Table 10: Acknowledgment area on error

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0      | 2      | \{61  \\
|        |        | 63  \\
|        |        | 64  \\
|        |        | 6A  \\
|        |        | 6M  \\
|        |        | \} Status code                   |
| 2      | 3      | san   | Symbolic attribute name          |
| 5      | 1      | -     | Blank                            |
| 6      | 2      | ff    | File identifier                  |
| 8      | 2      | -     | -                                |
| 10     | 2      | ss    | Status subnumber                 |

Table 11: Acknowledgment area on error (part 1 of 2)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 11: Acknowledgment area on error (part 2 of 2)
Response area

The user can define in a search statement how many response records are to be placed in the response area (block mode). SESAM/SQL returns responses of the length defined for the response area in the open statement. If the response area is too small for a complete block, the blocking factor is determined internally based on the size of the response area. The number defined by &BLNnnn or &BLKnnn in the search statement is ignored on response output.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number of the response record in block mode &amp;BLKnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No record number is specified in block mode &amp;BLNnnn, or if block mode is not being used.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv]</td>
<td>Primary key value of the length specified in the attribute catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The primary key value is not output if &amp;PSN000 was specified.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv[...]]</td>
<td>Attribute values or number of counted attribute values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute values are output with the length specified in the attribute catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The output sequence depends on the sequence in which the attributes were referenced in the subquestions. If attributes are referenced more than once, their values are also output more than once.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
<td>In block mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Output of response records 2 to nnn; each response record has the format defined previously.</td>
</tr>
</tbody>
</table>

Table 12: Response area
### Inquiry area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>[pkv1</td>
<td>First comparison value for primary key: primary key value for PK function 4/5/6, primary key group value for PK function 1/2/3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rno</td>
<td>Record number for PK function 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for PK function 0.</td>
</tr>
<tr>
<td>-</td>
<td>L(AK)</td>
<td>[pkv2]</td>
<td>Second comparison value for primary key: primary key value for PK function 5, primary key group value for PK function 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other PK functions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv1]</td>
<td>First comparison value for an attribute for comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv2]</td>
<td>Second comparison value for an attribute for comparison conditions 23, 24 and 82.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Comparison values atv1 and atv2 (where relevant) for further subquestions with comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
</tbody>
</table>

Table 13: Inquiry area

The sequence of comparison values in the inquiry area corresponds to the sequence in which the attributes are referenced in the subquestion.

The lengths of the comparison values must always be the same as that specified in the attribute catalog:

- Shorter values for an attribute with data type CHAR must be right-filled with blanks to the full attribute length.
- Shorter values for attributes with data format NUMERIC, DECIMAL, INTEGER or SMALLINT must be filled with leading zeros to the full attribute length.

A comparison value for a compound key (AAA) must be given as the full length key, consisting of the lengths of all compound key attributes (AAB, AAC, ...).
Numeric comparison values must be given as the correct type in the inquiry area:

- unpacked for data type NUMERIC
- packed for data type DECIMAL
- binary for data types INTEGER and SMALLINT

<table>
<thead>
<tr>
<th>Data type</th>
<th>Storage format</th>
<th>Sample value</th>
<th>Compar. value in inquiry area</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC</td>
<td>unpacked</td>
<td>+3210</td>
<td>X’F3F2F1F0´</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3210</td>
<td>X’F3F2F1D0´</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>packed</td>
<td>+3210</td>
<td>X’03210C´</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3210</td>
<td>X’03210D´</td>
</tr>
<tr>
<td>INTEGER</td>
<td>binary</td>
<td>+3210</td>
<td>X’00000C8A´</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3210</td>
<td>X’FFFFF376´</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>binary</td>
<td>+3210</td>
<td>X’0C8A´</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3210</td>
<td>X’F376´</td>
</tr>
<tr>
<td>CHAR</td>
<td>alphanumeric</td>
<td>+3210</td>
<td>X’4EF3F2F1F0´</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3210</td>
<td>X’60F3F2F1F0´</td>
</tr>
</tbody>
</table>

Table 14: Examples of entering numeric comparison values

**Sorting the response records**

The responses returned by a search can be sorted on the following criteria:

- ascending order of primary key values
- ascending order of index values
- ascending or descending order of any attribute values

**Sort in ascending order of primary key values**

A search supplies the response records in ascending order of primary key values when the search is done under strategy 0 (sequential search):

`xxx6x0...9`

**Sort in ascending order of index values**

The T subquestion allows records to be selected conditionally based on the value of an attribute defined as an index. The responses are output in ascending order of the index attribute values. The records in which the attribute does not contain a significant value are not selected.
Sort in ascending or descending order of attribute values

The S subquestion allows the response records from a search to be output in ascending or descending order.

Use of the S subquestion requires that strategy Z or P was used for the search, as these strategies create a cursor file. The response area must be at least 36 bytes long.

A search can contain a maximum of 6 S subquestions, with a total maximum of 6 attributes or occurrences of a multiple attribute, and a maximum of 6 sort criteria. The first attribute of the first S subquestion is the high-order sort criterion.

Handling the S subquestion in the application program:

The first stage is for the application program to pass the search to the connection module SESMOD.

SESAM/SQL creates a sort cursor file containing the record numbers of the response records to be sorted and their attribute values. Status code 1S and the number of response records is placed in the acknowledgment area.

The application program can now decide whether the response records are to be sorted or whether the sort cursor file is to be further processed before being sorted:

1. Sorting the response records in the sort cursor file:
   The application program must pass the same search to the sort module SESORT. The response records are then sorted by BS2000 SORT. The application program is unable to intervene. The record numbers of the response records are stored in the cursor file, which can be compressed by a search using strategy A or P, or the responses can be retrieved by means of the statements xxx729 and xxx709 (see section “Response polling” on page 117).

2. Processing the sort cursor file without sorting:
   The sort cursor file, which contains the record numbers and attribute values of the response records, can be compressed by means of statement xxx10K9 (see section “Cursor file handling” on page 125). This creates a cursor file containing only record numbers. This cursor file can then be processed by one of the following statements:
   - Search with strategy A or P
   - Cursor file handling statements
   - Response polling statements for paging through the cursor file
When the sort cursor file is sorted, errors may occur, and are communicated to the application program by status code 1F or 1G. Error messages from BS2000 SORT may also occur.

A search with S subquestion may not be initiated whilst a BS2000 SORT (e.g. COBOL SORT) using the SORTWORK file is still running in the application program.

**Flexible construction of subquestions**

Searches of a specific format can be defined in the application program. Subquestions or conditions that are not required can be disabled. The following options are available:

- Within a subquestion, the current search condition is replaced by search condition 8. Non-relevant inquiry area entries are skipped depending on the specified comparison condition. The default value is output for the disabled subquestion, provided this subquestion contains the projection (C subquestion).

- Specific comparison conditions can be disabled within a subquestion. Comparison conditions requiring a comparison value must be replaced by 80 and comparison conditions requiring two comparison conditions by 82. This means that the comparison condition is disabled and the unwanted inquiry area entries are ignored.
3.5 Search with join

Search with join links two logical files. The values of the join attribute of one logical file are compared with the values of the join attribute of the other logical file. If equal, the records in the two logical files are linked. They can be output in the response area, counted, or their record numbers stored in a join cursor file. As in any other search, records can be selected conditionally based on the primary key values or on attribute values. Attribute values can also be projected from the table into the response record.

It is a prerequisite for search with join that the two join attributes have the same data type and that their full length has been defined as an index.

The join attribute can be:

– a normal attribute
– an occurrence of a multiple attribute
– several occurrences of a multiple attribute
– a compound key attribute
– the primary key or the complete compound key

The two logical files may belong to the same or different CALL DML tables. Where they belong to the same table, the join attributes can be the same attribute. If the two logical files belong to different tables, these tables must be processed by the same SESAM/SQL DBH. They can, however, be assigned to different catalogs. Otherwise, the join will be rejected with status 6U.

Both logical files must have been opened by an open statement (see section “Open” on page 29) before the search with join. The open statement defines the size of the inquiry and response areas. When the search with join is given, the size of the inquiry and response areas of the logical file specified in the acknowledgment area must be of sufficient size.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:
The application program supplies the file identifier; the DBH returns the acknowledgment of the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values, these must be made available in the inquiry area by the application program.
Statement area

```
search1 -> join-expr -> search2 -> ...
```

<table>
<thead>
<tr>
<th>Block mode</th>
<th>Lock mode</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-/7</td>
<td>-/7</td>
<td>-/1</td>
</tr>
</tbody>
</table>

```
... &BLKnnn &PSN000 &RNL000 &RNW000 ; &BLNnnn
```


DML statements

Search with join

\[
\text{search1} \quad \text{and} \quad \text{search2}
\]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/1</td>
<td>-/2</td>
</tr>
</tbody>
</table>

\[
\text{pass} \rightarrow 6 \rightarrow 0 \rightarrow 0 \rightarrow N \rightarrow \# \rightarrow \text{ff} \rightarrow \text{C} \rightarrow \text{L} \rightarrow \text{U} \rightarrow \text{O} \rightarrow \text{E} \rightarrow \text{Z}
\]
Key

PK (primary key) function
0 all data
1 equal to PK group value
2 range of PK group values
3 larger than PK group value
4 equal to PK value
5 range of PK values
6 larger than PK value
8 equal to record number

Strategy
0 sequential search, output responses
1 output responses
Y output number of responses
Z record numbers in cursor file, output number of responses

Subquestions
C AND operator, projection, selection
L OR operator, projection, selection
U AND operator, selection
O OR operator, selection
E projection
Z count significant values of an attribute or occurrence

End id.
9 end of statement
; chain statement

join-expr

Join attribute File id. Join attribute File id.
V→ ( san1 # ff1 = san2 # ff2 ...
               san1/mmm/                  san2/mmm/
               san1/mmm-nnn/           san2/mmm-nnn/...
DML statements

Search with join

Search condition
Comparison condition

-1
-2

...
Search with join

Key

<table>
<thead>
<tr>
<th>File identifiers</th>
<th>ff1</th>
<th>logical file from search1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ff2</td>
<td>logical file from search2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Search condition</th>
<th>0</th>
<th>no condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>conditions</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>negated conditions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparison condition</th>
<th>00</th>
<th>no comparison condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01</td>
<td>= (equal)</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>&lt; (less than)</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>≤ (less than or equal to)</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>&gt; (greater than)</td>
</tr>
<tr>
<td></td>
<td>05</td>
<td>≥ (greater than or equal to)</td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>≠ (not equal to)</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>from ... to</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>outside from ... to</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>skip one inquiry area entry</td>
</tr>
<tr>
<td></td>
<td>82</td>
<td>skip two inquiry area entries</td>
</tr>
</tbody>
</table>

Statement elements

The search with join and its subfunctions contain statement elements that are identical to those in the search statement (see section “Search” on page 37). Only those statement elements are described below that do not occur in the normal search or whose functionality differs.

Primary key function (PK function) (4/1)

The primary key function specifies the condition applied to the primary key value by which to select records.
The search2 specification is independent of that in search1, i.e. any combination is possible.

A detailed description of the individual primary key functions can be found under Search (see “Primary key function (PK function) (4/1)” on page 41).
Strategy (5/1)

The strategy defines the type of results the search is to supply. The following results are possible:

- Response records in the response area
- The number of responses in the acknowledgment area
- Creation of a cursor file containing the record numbers of the response records.

The strategy also determines the search method to be used:

- Strategy 0 causes a sequential search of the table.
- Any other strategy (1/Y/Z) allows SESAM/SQL to select the most appropriate search method: sequential search or search via index.

The following combinations are possible in a search with join:

<table>
<thead>
<tr>
<th>search1</th>
<th>search2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Output response records in the response area</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>– Count the responses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Output number in acknowledgment area</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td>– Count the responses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Output number in acknowledgment area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Store record numbers in join cursor file</td>
</tr>
</tbody>
</table>

Table 15: Combinations in a search with join

The meaning of the strategies is described in detail under Search.

A join cursor file created by strategy Z is a record number cursor file. It is identified by the file identifier that was entered in the acknowledgment area in the search with join statement. The join cursor file can be restricted by a further search with join (see section “Restricting a join cursor file” on page 76).
Search with join

DML statements

Record numbers function (6/1)

N The responses are output with record numbers.

The specification need only be given in search1.

If N is not specified, record numbers are only output if block mode is defined with &BLKnnn.

File identifier (-/-2)

ff File identifier identifying the logical file on which search1 or search2 is based.

Search1 must define a different file identifier from search2. It is, however, permitted for both file identifiers to refer to the same table.

ff1 File identifier of the logical file to which search1 refers.

ff2 File identifier of the logical file to which search2 refers.

In a search with join using strategy Z, the join cursor file is created under the file identifier specified in the acknowledgment area.

Join attribute (-/-)

The full length of the two join attributes that join the two logical files must be defined as an index.

Join attribute of logical file ff1 of search1

san1 Symbolic attribute name of an attribute or of the primary key (AAA).

A compound key attribute can also be a join attribute.

san1/mmm/

Symbolic attribute name and occurrence number (mmm) of a multiple attribute.

san1/mmm-nnn/

Symbolic attribute name of the multiple attribute whose occurrences mmm to nnn form the join attribute.

Join attribute of logical file ff2 of search2

san2 Symbolic attribute name of an attribute or of the primary key (AAA).

A compound key attribute can also be a join attribute.

san2/mmm/

Symbolic attribute name and occurrence number (mmm) of a multiple attribute.

san2/mmm-nnn/

Symbolic attribute name of the multiple attribute whose occurrences mmm to nnn form the join attribute.
## Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0      | 2      | `{00
10
16
1S
9S}` | Status                                                                 |
| 2      | 4      | no        | Number of responses counted or placed in the response area. If the number is greater than or equal to X´FFFFFFFF´, then X´FFFFFFFF´ is entered. |
| 6      | 2      | ff        | File identifier
– of the logical file (strategy 0/1/Y),
– of the cursor file (strategy Z) |
| 8      | 2      | t-length  | Total length of responses                                               |
| 10     | 2      | e-length  | Length of each response                                                 |
| 12     | 4      | rno       | Record number; in block mode, this is the record number of the last response record placed in the response area |

Table 16: Acknowledgment area for response

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0      | 2      | `{1K
60
66
67
6D
6J
6T
6U
6W
6Z
9O
9Q
9E}` | Status |
| 2      | 4      | -         | -       |
| 6      | 2      | ff        | File identifier |

Table 17: Acknowledgment area on error (part 1 of 2)
Search with join

The user can define in a search with join statement how many response records are to be placed in the response area (block mode). SESAM/SQL returns responses of the length defined for the response area in the open statement. If the response area is too small for a complete block, the blocking factor is determined internally based on the size of the response area. The number defined as &BLNnnn or &BLKnnn in the search statement is ignored when the responses are output.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>61</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>63</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6M</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>Blank</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 17: Acknowledgment area on error (part 2 of 2)

Response area

The user can define in a search with join statement how many response records are to be placed in the response area (block mode). SESAM/SQL returns responses of the length defined for the response area in the open statement. If the response area is too small for a complete block, the blocking factor is determined internally based on the size of the response area. The number defined as &BLNnnn or &BLKnnn in the search statement is ignored when the responses are output.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno1]</td>
<td>Record number of the response record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– in block mode &amp;BLKnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– in non-block mode if SNR (record no.) function N is used</td>
</tr>
</tbody>
</table>

Table 18: Acknowledgment area on error

Table 19: Response area (part 1 of 2)
## DML statements

### Search with join

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv1]</td>
<td>Primary key value of the length specified in the attribute catalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The primary key value is not output if &amp;PSN000 was specified.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv1[...] ]</td>
<td>Attribute values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute values are output with the length specified in the attribute catalog. The output sequence depends on the sequence in which the attributes were referenced in the subquestions. If attributes are referenced more than once, their values are also output more than once.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>join-atv</td>
<td>Value of join attribute (only output once in the response)</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>[rno2]</td>
<td>Record number of response record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- in block mode &amp;BLKnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- in non-block mode if SNR (record no.) function N is used</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv2]</td>
<td>Primary key value of the length specified in the attribute catalog</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The primary key value is not output if &amp;PSN000 was specified for lock mode.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv2[...] ]</td>
<td>Attribute values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute values are output with the length specified in the attribute catalog. The output sequence depends on the sequence in which the attributes were referenced in the subquestions. If attributes are referenced more than once, their values are also output more than once.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Output response records 2 to nnn; each response record has the format defined previously.</td>
</tr>
</tbody>
</table>

Table 19: Response area (part 2 of 2)
### Inquiry area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>pkv11</td>
<td>First comparison value for primary key: primary key value for PK functions 4/5/6, primary key group value for PK functions 1/2/3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rno1</td>
<td>Record number for PK function 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for PK function 0.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>pkv12</td>
<td>Second comparison value for primary key: primary key value for PK function 5, primary key group value for PK function 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other PK functions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv11</td>
<td>First comparison value for an attribute for comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv12</td>
<td>Second comparison value for an attribute for comparison conditions 23, 24 and 82.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>[...]</td>
<td>Comparison values atv11 and atv12 (where relevant) for further subquestions with comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>join-atv1</td>
<td>First comparison value for the join attribute for comparison conditions 01 to 06, 23, 24, 80 and 82.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>join-atv2</td>
<td>Second comparison value for the join attribute for comparison conditions 23, 24 and 82.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>pkv21</td>
<td>First comparison value for primary key: primary key value for PK functions 4/5/6, primary key group value for PK functions 1/2/3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rno2</td>
<td>Record number for PK function 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for PK function 0.</td>
</tr>
</tbody>
</table>

Table 20: Inquiry area (part 1 of 2)
The sequence of comparison values in the inquiry area corresponds to the sequence in which the attributes are referenced in the subquestion. The lengths of the comparison values must always be the same as that specified in the attribute catalog, i.e. shorter values such as primary key group values must be blank-filled on the right to the full attribute length. A comparison value for a compound key (AAA) must be given as the full length key, consisting of the lengths of all compound key attributes (AAB, AAC, ...). The data type of numeric comparison values entered in the inquiry area must be correct (see also under Search, “Inquiry area” on page 59).

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv22]</td>
<td>Second comparison value for primary key: primary key value for PK function 5, primary key group value for PK function 2. Omitted for all other PK functions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv21]</td>
<td>First comparison value for an attribute for comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714. Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv22]</td>
<td>Second comparison value for an attribute for comparison conditions 23, 24 and 82. Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Comparison values atv21 and atv22 (where relevant) for further subquestions with comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
</tbody>
</table>

Table 20: Inquiry area (part 2 of 2)
3.6 Restricting a join cursor file

A cursor file created by means of a search with join using strategy Z can be further restricted.

The following options are available:

- Selection by conditions applied to the primary key value or record number via the primary key function
- Selection by applying conditions to attribute values using search1 and search2

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier, and the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values, they must be made available in the inquiry area by the application program.

Statement area

```
search1 V search2 ...
```

Block mode

```
/-7 -/7 -/7 -/7 -/1
```

... 9

- `/BLKnnn`
- `/PSN000`
- `/RLN000`
- `/RNW000`
- `;`
DML statements

Restricting a join cursor file

**Search 1**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/1</td>
<td>-/2</td>
<td>-/-</td>
</tr>
</tbody>
</table>

**Search 2**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-/3</td>
<td>-/1</td>
<td>-/1</td>
<td>-/1</td>
<td>-/1</td>
<td>-/1</td>
<td>-/2</td>
</tr>
</tbody>
</table>
Restricting a join cursor file

DML statements

Key

PK (primary key) function

<table>
<thead>
<tr>
<th>PK function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>all data</td>
</tr>
<tr>
<td>1</td>
<td>equal to PK group value</td>
</tr>
<tr>
<td>2</td>
<td>range of PK group values</td>
</tr>
<tr>
<td>3</td>
<td>larger than PK group value</td>
</tr>
<tr>
<td>4</td>
<td>equal to PK value</td>
</tr>
<tr>
<td>5</td>
<td>range of PK values</td>
</tr>
<tr>
<td>6</td>
<td>larger than PK value</td>
</tr>
<tr>
<td>8</td>
<td>equal to record number</td>
</tr>
</tbody>
</table>

Strategy

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>restrict cursor file, output responses</td>
</tr>
<tr>
<td>P</td>
<td>restrict cursor file, record numbers in cursor file and output number of responses</td>
</tr>
</tbody>
</table>

File id.

<table>
<thead>
<tr>
<th>File id.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ff0</td>
<td>cursor file to be restricted</td>
</tr>
<tr>
<td>ff1</td>
<td>logical file from search1</td>
</tr>
<tr>
<td>ff2</td>
<td>logical file from search2</td>
</tr>
</tbody>
</table>

Subquestions

<table>
<thead>
<tr>
<th>Subquestion</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>AND operator, projection, selection</td>
</tr>
<tr>
<td>L</td>
<td>OR operator, projection, selection</td>
</tr>
<tr>
<td>U</td>
<td>AND operator, selection</td>
</tr>
<tr>
<td>O</td>
<td>OR operator, selection</td>
</tr>
<tr>
<td>E</td>
<td>projection</td>
</tr>
<tr>
<td>Z</td>
<td>count significant values of an attribute or occurrence</td>
</tr>
</tbody>
</table>

End id.

<table>
<thead>
<tr>
<th>End id.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>end of statement</td>
</tr>
<tr>
<td>;</td>
<td>chain statement</td>
</tr>
</tbody>
</table>

Statement elements

The statement and its subfunctions contain statement elements identical to those of the search statement (see section “Search” on page 37). Only those statement elements are described below that do not occur in the normal search or whose functionality differs.

Primary key function (PK function) (-/1)

The primary key function specifies the condition applied to the primary key value in order to select records. The search2 specification is independent of that in search1, i.e. any combination is possible.

A detailed description of the individual primary key functions can be found under Search (see page 41).
Strategy (-/1)

The strategy defines the type of results the search is to supply. The following results are possible:

- Response records in the response area.
- The number of responses in the acknowledgment area.
- Creation of a cursor file containing the record numbers of the response records.

With strategy A and P, SESAM/SQL determines the best search method (sequential or via index).

The following combinations of strategies for search1 and search2 are permitted for restricting the cursor file:

<table>
<thead>
<tr>
<th>search1</th>
<th>search2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>Output response records in response area</td>
</tr>
</tbody>
</table>
| P       | P       | - Count responses  
|         |         | - Output no. of responses in acknowledgment area  
|         |         | - Store record numbers in join cursor file |

Table 21: Combinations of strategies

The meaning of the strategies is described in detail under Search.

Record numbers function (-/1)

N The responses are output with record numbers.

This option need only be specified for search1.

If N is not specified, record numbers are only output if block mode has been defined with &BLKnnn.

File identifier (-/2)

ff0 File identifier of the cursor file to be restricted
ff1 File identifier of the logical file to which search1 refers
ff2 File identifier of the logical file to which search2 refers

ff0 must be specified if the cursor file to be restricted (ff0) is to be kept. The new cursor file is created under the file identifier given in the acknowledgment area.

If ff0 is not specified, the old, restricted cursor file is deleted or overwritten. The new cursor file is created, as above, under the file identifier given in the acknowledgment area.

If ff1 or ff2 is not specified, the file identifier from the acknowledgment is used.
## Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>no</td>
<td>Number of responses counted or placed in the acknowledgment area. If the number is greater than or equal to X’FFFFFFFF’, then X’FFFFFFFF’ is entered.</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– of the logical file (strategy A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– of the cursor file (strategy P)</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 22: Acknowledgment area for response

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1K</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>66</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>67</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6T</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6U</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6W</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9O</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9Q</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9E</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 23: Acknowledgment area on error (part 1 of 2)
Restricting a join cursor file

This statement allows the user to define how many response records are to be placed in the response area. SESAM/SQL returns responses of the length defined for the response area in the open statement. If the response area is too small for a complete block, the blocking factor is determined internally based on the size of the response area. The number defined as &BLNnnn or &BLKnnn in the search statement is ignored when the responses are output.

### Table 24: Acknowledgment area on error

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>61</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>Blank</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Response area

This statement allows the user to define how many response records are to be placed in the response area. SESAM/SQL returns responses of the length defined for the response area in the open statement. If the response area is too small for a complete block, the blocking factor is determined internally based on the size of the response area. The number defined as &BLNnnn or &BLKnnn in the search statement is ignored when the responses are output.

### Table 25: Response area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno1]</td>
<td>Record number of the response record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– in block mode &amp;BLKnnn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– in non-block mode if SNR (record no.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>function N is used</td>
</tr>
</tbody>
</table>

(part 1 of 2)
### Restricting a join cursor file

**DML statements**

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv1]</td>
<td>Primary key value of the length specified in the attribute catalog. The primary key value is not output if &amp;PSN000 was specified for lock mode.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv1[...]]</td>
<td>Attribute values. The attribute values are output with the length specified in the attribute catalog. The output sequence depends on the sequence in which the attributes were referenced in the subquestions. If attributes are referenced more than once, their values are also output more than once.</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>[rno2]</td>
<td>Record number of response record. In block mode &amp;BLKnnn. In non-block mode if SNR (record no.) function N is used.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv2]</td>
<td>Primary key value of the length specified in the attribute catalog. The primary key value is not output if &amp;PSN000 was specified for lock mode.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv2[...]]</td>
<td>Attribute values. The attribute values are output with the length specified in the attribute catalog. The output sequence depends on the sequence in which the attributes were referenced in the subquestions. If attributes are referenced more than once, their values are also output more than once.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: Output response records 2 to nnn; each response record has the format defined previously.</td>
</tr>
</tbody>
</table>

Table 25: Response area (part 2 of 2)
## Inquiry area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv11</td>
<td>First comparison value for primary key: primary key value for PK functions 4/5/6, primary key group value for PK functions 1/2/3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rno1</td>
<td>Record number for PK function 8 Omitted for PK function 0.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv12]</td>
<td>Second comparison value for primary key: primary key value for PK function 5, primary key group value for PK function 2 Omitte3d for all other PK functions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv11]</td>
<td>First comparison value for an attribute for comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714. Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv12]</td>
<td>Second comparison value for an attribute for comparison conditions 23, 24 and 82. Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Comparison values atv11 and atv12 (where relevant) for further subquestions with comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>pkv21</td>
<td>First comparison value for primary key: primary key value for PK function 4/5/6, primary key group value for PK function 1/2/3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rno2</td>
<td>Record number for PK function 8 Omitted for PK function 0.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv22]</td>
<td>Second comparison value for primary key: primary key value for PK function 5, primary key group value for PK function 2 Omitted for all other PK functions.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv21]</td>
<td>First comparison value for an attribute for comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714. Omitted for all other comparison conditions.</td>
</tr>
</tbody>
</table>

Table 26: Inquiry area (part 1 of 2)
Restricting a join cursor file

The sequence of comparison values in the inquiry area corresponds to the sequence in which the attributes are referenced in the subquestion.

The lengths of the comparison values must always be the same as that specified in the attribute catalog, i.e. shorter values such as primary key group values must be blank-filled on the right to the full attribute length.

A comparison value for a compound key (AAA) must be given as the full length key, consisting of the lengths of all compound key attributes (AAB, AAC, ...).

The data type of numeric comparison values entered in the inquiry area must be correct (see also under Search, “Inquiry area” on page 59).

---

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv22]</td>
<td>Second comparison value for an attribute for comparison conditions 23, 24 and 82. Omitted for all other comparison conditions.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Comparison values atv21 and atv22 (where relevant) for further subquestions with comparison conditions 01 to 06, 23, 24, 80, 82, 712 and 714.</td>
</tr>
</tbody>
</table>

Table 26: Inquiry area (part 2 of 2)
3.7 **Index browsing**

Index browsing outputs the frequency of attribute values. It can only be used on attributes whose full or partial length has been inverted, i.e. that have been declared as an index.

Attributes that have been partially inverted must be of the data type CHAR.

One statement processes just one attribute. With a multiple attribute, the frequencies of all significant values of all occurrences are output.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier, and the DBH returns the acknowledgment to the statement.

Response area: The DBH returns the frequency and attribute value for each response.

Inquiry area: If the comparison condition requires comparison values for the attribute values to be counted, they must be placed in the inquiry area.
Index browsing

DML statements

Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/4</td>
<td>7/3</td>
<td>10/1</td>
<td>11/2</td>
<td>-/7</td>
<td>-/1</td>
</tr>
</tbody>
</table>

Password: pas → Op. code (60Bl) → Attr. cond. (san) → Search cond. (0) → Comparison condition (00) → Block mode (4) → End ID (0)

&BLNnnn → ;
**Key**

Search condition

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no condition</td>
</tr>
<tr>
<td>4</td>
<td>string and mask search</td>
</tr>
<tr>
<td>5</td>
<td>conditions</td>
</tr>
</tbody>
</table>

Comparison condition

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>no comparison condition</td>
</tr>
<tr>
<td>01</td>
<td>= (equal to)</td>
</tr>
<tr>
<td>02</td>
<td>&lt; (less than)</td>
</tr>
<tr>
<td>03</td>
<td>≤ (less than or equal to)</td>
</tr>
<tr>
<td>04</td>
<td>&gt; (greater than)</td>
</tr>
<tr>
<td>05</td>
<td>≥ (greater than or equal to)</td>
</tr>
<tr>
<td>06</td>
<td>≠ (not equal to)</td>
</tr>
<tr>
<td>23</td>
<td>from ... to</td>
</tr>
<tr>
<td>24</td>
<td>outside from ... to</td>
</tr>
</tbody>
</table>

End identifier

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>end of statement</td>
</tr>
<tr>
<td>;</td>
<td>chain statement</td>
</tr>
</tbody>
</table>

**Password (0/3)**

**Password**

- **Pas** Password for a protected CALL DML table,
- any three-character string for an unprotected CALL DML table.

**Operation code (3/4)**

**60BI** Operation code for the index browsing statement

**Attribute (7/3)**

**San** Symbolic attribute name of the attribute of whose values the frequencies are to be output. With a multiple attribute, the occurrences cannot be referenced individually. The frequency is output for all significant values of all occurrences.

If san is an attribute with data type NUMERIC, DECIMAL, INTEGER or SMALLINT, its full length must have been inverted.

**Search condition and comparison condition**

Search and comparison conditions allow one or more conditions to be applied to the attribute values to be counted.
Search condition (10/1)

0  No conditions are applied to the attribute value.

4  String search:

Only the comparison conditions 01 and 06 are permitted. An attribute value is tested to see whether it contains a string or not. The comparison value in the inquiry area must be enclosed in string identifiers. The length of the string is: \[1 \leq \text{string length} \leq \text{attribute length}-2\]. The default string identifier is “%”. It can be changed by means of the set string identifier statement (see section “Define comparison values” on page 92).

Mask search:

An attribute value is tested for a specific character in a particular position. Non-relevant characters are substituted with the mask character. The comparison condition is satisfied if the known positions satisfy the comparison condition. The default mask character is “?”. It can be changed by means of the set mask character statement.

A comparison value may contain either mask characters only or string identifiers only, but not both.

5  The attribute value is tested to see if it meets the comparison conditions. If several comparison conditions are specified, the subquestion is satisfied as soon as the attribute value fulfils one of the comparison conditions (OR relationship).

Comparison condition (11/2)

00  no condition; no comparison value in the inquiry area
01  equal to the comparison value in the inquiry area
02  less than the comparison value in the inquiry area
03  less than or equal to the comparison value in the inquiry area
04  greater than the comparison value in the inquiry area
05  greater than or equal to the comparison value in the inquiry area
06  not equal to the comparison value in the inquiry area
23  greater than or equal to the first comparison value and also less than or equal to the second comparison value in the inquiry area
24  less than the first comparison value or greater than the second comparison value in the inquiry area
Block mode (-/7)

The user can find out how many responses have been found and placed in the response area.

&BLNnnn

The response area contains nnn attribute values and their respective frequencies.

If &BLNnnn is omitted, then by default just one attribute value and its frequency is output.

End identifier (-/1)

9 Identifies the end of the statement.

; End of statement. The statement is chained to a following end TA statement.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(00)(10)</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>no</td>
<td>Number of attribute values found</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 27: Acknowledgment area for response
## Index browsing

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{1K, 60, 66, 67, 6B, 6D, 6U, 6W, 6Z, 9E, 9Q}</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 28: Acknowledgment area on error

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{61, 63, 6A}</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 29: Acknowledgment area on error
Response area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>no</td>
<td>Frequency of attribute value (binary)</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode (&amp;BLNnnn): responses 2 to nnn</td>
</tr>
</tbody>
</table>

Table 30: Response area

The responses are sorted in ascending order of the value of the index attribute. Where the attributes are partially inverted, non-inverted positions are replaced by the current mask character.

Inquiry area

Search condition 0 does not require an entry in the inquiry area.

For search conditions 4 and 5, the comparison values must be placed in the inquiry area:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>atv1</td>
<td>First comparison value for the secondary index attribute in comparison conditions 01 to 06, 23 and 24</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>[atv2]</td>
<td>Second comparison value for the secondary index attribute in comparison conditions 23 and 24</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Further atv1/atv2 comparison values if more than one comparison condition was specified for search condition 5</td>
</tr>
</tbody>
</table>

Table 31: Inquiry area

Masked values or strings can also be used as comparison values if search condition 4 was specified.
For partially inverted attributes, the length of the string must not exceed the length of the inverted part of the attribute.
The comparison values must always be specified as the length defined in the attribute catalog. This also applies to the comparison values for a partially inverted attribute. Shorter comparison values must be blank-filled to the full attribute length.

Numeric comparison values in the inquiry area must be of the correct data type (see “Inquiry area” on page 59).
3.8 Define comparison values

In retrieval statements, records can be selected by means of partially known comparison values. The corresponding attribute must, however, be defined as alphanumeric (CHAR).

Masked search

A record is selected if an attribute value contains a particular character in a specified position. The mask character is substituted for non-relevant characters in the comparison value.

The default mask character is the question mark (?). If the attribute value and the comparison value contain a question mark, a different mask character can be defined for the logical file by means of the set mask character function. The new mask character must be entered in the inquiry area.

The new mask character must not be the same as the current string identifier and must not be contained in the comparison value.

The delete mask character function reinstates the default mask character “?”.

String search

A record is selected if an attribute value contains a particular character or character string in a specified position.

The required character or character string is enclosed in string identifiers to form the comparison value.

The default string identifier is the percent character (%). If the attribute value and the search string contain the percent character, the percent character cannot be used as the string identifier. A different one can be defined by means of the set string identifier function. The new string identifier must in this case be placed in the inquiry area.

The string identifier
– must not be the same as the current mask character and
– must not be contained in the search string itself.

The delete string character function reinstates the default string identifier “%”.
Applicability of mask character/string identifier:
The mask character/string identifier is valid
– until a new mask/string character is defined
– until the mask/string character is deleted
– until the logical file is closed.

Contents of transfer areas:
Statement area: The application program supplies the statement.
Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.
Inquiry area: The application program enters the new mask or string character (only for set mask character/string identifier).

**Statement area**

<table>
<thead>
<tr>
<th>Password</th>
<th>Op. code</th>
<th>Function</th>
<th>Type</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
</tr>
<tr>
<td>pas</td>
<td>6</td>
<td>0</td>
<td>F</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Password (0/3)**
pas Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

**Operation code (3/1)**
6 Operation code for the define comparison values function
Define comparison values

Function (5/1)

F Set mask character/string identifier.
   The new mask character/string identifier must be entered in the inquiry area.

E Delete mask character/string identifier.

Type of character (6/1)

S The statement refers to the string identifier.
   If S is not specified, the statement refers to the mask character.

End identifier (-/1)

9 Identifies the end of the statement

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00 60 6E 6Z</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[ss]</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 32: Acknowledgment area
Inquiry area

The inquiry area need only be used in the functions for *set the mask character or string identifier*.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>String identifier, if the statement contained “S” as the type of character. Mask character if the type specifier was omitted from the statement.</td>
</tr>
</tbody>
</table>

Table 33: Inquiry area
3.9 Record output

Record output offers the following functions:

- Selection of records conditionally, based on primary key value
- Projection of attribute sequences; null attribute values are not output
- Optional output of the attribute definition of the projected attributes

Record output returns variable length responses, as null attribute values are suppressed in the response output. If the response area was not defined large enough in the open statement, remainder response polling can be performed (see section “Response polling” on page 117).

Record output can be used, for example, to retrieve continuous text from a table.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:
The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values for the primary key value, the application program must put them in the inquiry area.
### Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/-</td>
<td>-/1</td>
</tr>
</tbody>
</table>

```
pas -> 4 -> 0 -> san1
       -> 1
       -> 2
       -> 3
       -> 4
       -> 5
       -> 6
       -> 8
```

<table>
<thead>
<tr>
<th>Block mode</th>
<th>Lock mode</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-/7</td>
<td>-/7</td>
<td>-/1</td>
</tr>
</tbody>
</table>

```
&BLKnnn  &PSN000  &RLN000  &RNW000  ;
&BLNnnn
```

U1054-J-Z125-11-76
### Key

<table>
<thead>
<tr>
<th>PK (primary key) function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>all data</td>
</tr>
<tr>
<td>1</td>
<td>equal to PK group value</td>
</tr>
<tr>
<td>2</td>
<td>range of PK group values</td>
</tr>
<tr>
<td>3</td>
<td>larger than PK group value</td>
</tr>
<tr>
<td>4</td>
<td>equal to PK value</td>
</tr>
<tr>
<td>5</td>
<td>range of PK values</td>
</tr>
<tr>
<td>6</td>
<td>larger than PK value</td>
</tr>
<tr>
<td>8</td>
<td>equal to record number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SAN, attribute length and attribute values</td>
</tr>
<tr>
<td>2</td>
<td>attribute values only</td>
</tr>
<tr>
<td>3</td>
<td>both attribute definition and attribute values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>End identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>end of statement</td>
</tr>
<tr>
<td>;</td>
<td>chain statement</td>
</tr>
</tbody>
</table>

### Password (0/3)

```
pas      Password for a protected CALL DML table,
         any three-character string for an unprotected CALL DML table.
```

### Operation code (3/1)

```
4      Operation code for the record output statement
```

### Primary key function (PK function) (4/1)

The primary key function allows records to be selected by applying conditions to the primary key or record number. The comparison values are placed in the inquiry area. A primary key group value can be used for selection instead of the whole primary key value. The primary key group value identifies a group of records whose primary key values contain the primary key group value in the left of the key. The comparison value in the inquiry area must be blank-filled to the full length of the primary key.

With compound keys, the only primary key group value that may be used is a value of the compound key attribute AAB or of several compound key attributes (starting at AAB, ascending AAC, etc.), with the compound key attribute whose value is furthest to the right in the primary key group value not being of the data type INTEGER or SMALLINT. It is possible, however, for the compound key attribute furthest to the right to be only partially covered by the primary key group value. Blanks must be inserted in the inquiry area for the remaining compound key attributes.
DML statements

Record output

0  All data:
All records are selected.

1  Equal to primary key group value:
A primary key group value must be entered in the inquiry area as comparison value.
All records containing the primary key group value left-justified in their primary key
value are selected.

2  Range of primary key group values:
Two primary key group values that define a range of primary key group values must
be placed in the inquiry area.
All records whose primary key values are greater than or equal to the first
comparison value and smaller than or equal to the second comparison value are
selected.
The first comparison value must not be larger than the second comparison value.

3  Greater than primary key group value:
A primary key group value must be entered in the inquiry area as the comparison
value.
All records whose primary key value is greater than the comparison value are
selected.

4  Equal to primary key value:
A primary key value must be entered in the inquiry area as the comparison value.
The record whose primary key value is equal to the comparison value is selected.

5  Range of primary key values:
Two primary key values defining a range of primary key values must be entered in
the inquiry area as comparison values.
All records whose primary key values are greater than or equal to the first
comparison value and smaller than or equal to the second comparison value are
selected.
The first comparison value must not be larger than the second comparison value.

6  Greater than primary key value:
A primary key value must be entered in the inquiry area as the comparison value.
All records whose primary key value is greater than the comparison value are
selected.

8  Equal to record number:
A record number must be entered in the inquiry area as the comparison value.
The record with the specified record number is selected.
**Attribute selection (5/-)**

Record output allows attribute sequences to be projected. An attribute sequence is defined by a start and end attribute. The start and end attributes must be specified in ascending sequence.

Any number of attribute sequences can be specified. They must, however, be given in ascending sequence and must not overlap.

An attribute sequence can begin or end with an occurrence of a multiple attribute. The start and end attributes are equal if just one attribute is to be referenced.

- **san1** Symbolic attribute name of the start attribute
- **san1/mmm/** Occurrence of a multiple attribute which is the start of the attribute sequence
- **san2** Symbolic attribute name of the end attribute
- **san2/nnn/** Occurrence of a multiple attribute which is the end of the attribute sequence

The end attribute can be omitted from the last attribute sequence of record output if the last attribute defined in the attribute catalog is to be used as the end attribute.

**Format identifier (-/1)**

The format identifier determines what information is to be output for the projected attributes:

- **1** Symbolic attribute name, attribute length and attribute values are output.
- **2** Only the attribute values or values of occurrences of a multiple attribute are output.
- **3** The whole attribute definition and the attribute values are output.

Attribute definition and attribute value are only output if the referenced attribute has a significant value.
**Block mode (-/7)**

The user can define how many of the responses found are to be placed in the response area.

&BLKnnn

nnn responses are placed in the response area. The record number of each response is output.

&BLNnnn

nnn responses are placed in the response area. The record number of the response records is not output.

If neither &BLKnnn nor &BLNnnn is specified, by default just one response record is placed in the response area with no record number.

---

**Lock mode (-/7)**

&PSN000

The response records are output without primary key values.

&RNL000

The record accessed by record output within a transaction is not locked.

&RNW000

The record output can access a record that has been locked by another transaction (dirty read). In this case, the statement is acknowledged with status code 9S. After a dirty read in block mode, no further responses are placed in the response area. If &RNW000 is omitted, a transaction attempting to access a locked record is placed in a wait state until the record becomes free.

---

**End identifier (-/1)**

9 Identifies the end of the statement

; End of statement. The statement is chained to a following end TA statement.
Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>length</td>
<td>Response length</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 34: Acknowledgment area for response

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>4A</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>length</td>
<td>Response length</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 35: Acknowledgment area on error

Status code 4A is returned if the response to the record output is longer than the response area length defined in the open statement. The symbolic attribute name of the first attribute that cannot fit into the response area is output together with the length of the response placed in the response area.

The remainder of the response can be retrieved by response polling statement xxx739 (see section “Response polling” on page 117).
Status code 9S (read a locked record) may also contain the cause of status code 4A. In this case, the acknowledgment area contains the same information as for status code 4A.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{40, 41, 42, 47, 4B, 4D, 4M, 4Z, 70, 7D, 9O, 9S}</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 36: Acknowledgment area on error

**Response area**

SESAM/SQL places in the response area the attribute values and also the full attribute definition or parts of it, depending on the format identifier.

Information is only output for attributes with a significant value. Attributes with null values are ignored.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
</tbody>
</table>
| 4     | L(AC)  | [pkv] | Primary key value
The primary key value of each response is output with the length specified in the attribute catalog. If &PSN000 is specified, no primary key value is output. |

Table 37: Response area for format identifiers 1 or 2 (part 1 of 2)
### Table 37: Response area for format identifiers 1 or 2 (part 2 of 2)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name, omitted for format identifier 2</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X'00' to X'FF'</td>
<td>Attribute length -1: 1 to 256 bytes; omitted for format identifier 2</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute value is output with the length specified in the attribute catalog. For multiple attributes, the attribute definition and attribute value is output for each occurrence.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record: end identifier that terminates the last attribute value</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: responses 2 - nnn</td>
</tr>
</tbody>
</table>

### Table 38: Response area for format identifier 3 (part 1 of 2)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>rno</td>
<td>Record number, only for block mode &amp;BLKnnn; the record number is not specified if the block mode is &amp;BLNnnn or if no block mode is used.</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The primary key value of each response is output with the length specified in the attribute catalog. If &amp;PSN000 is specified, no primary key value is output.</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>X'0001' to X'0100'</td>
<td>Attribute length: 1 to 256 bytes</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X'00' to X'0F'</td>
<td>Number of decimal places: 0 to 15 places of decimals</td>
</tr>
</tbody>
</table>

Table 37: Response area for format identifiers 1 or 2 (part 2 of 2)

Table 38: Response area for format identifier 3 (part 1 of 2)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>X’11’</td>
<td>Data type: CHAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’21’</td>
<td>NUMERIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’22’</td>
<td>DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’24’</td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’28’</td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’00’</td>
<td>uninterpretable data type</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’02’</td>
<td>Index information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’04’</td>
<td>Index locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’08’</td>
<td>Index is available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Index required</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>i-length</td>
<td>Index length (binary)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’01’ to X’FF’</td>
<td>Number of occurrences of a multiple attribute as defined in attribute catalog; 1 to 255 occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’00’</td>
<td>not a multiple attribute</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>dfc</td>
<td>Default value character</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’80’</td>
<td>Compound key information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’40’</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’00’</td>
<td>Not a compound key</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute value is output with the length specified in the attribute catalog. For multiple attributes, the attribute definition and attribute value is output for each occurrence.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record: end identifier that terminates the last attribute value</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: responses 2 - nnn</td>
</tr>
</tbody>
</table>

Table 38: Response area for format identifier 3 (part 2 of 2)
**Inquiry area**

PK function 0 does not require an entry in the inquiry area.

Comparison values must be placed in the inquiry area for primary key functions 1 to 6 and 8:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv1</td>
<td>For PK functions 1 to 6: primary key (group) value of the primary key length specified in the attribute catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rno</td>
<td>For PK function 8: record number.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv2]</td>
<td>For PK functions 2 and 5: primary key (group) value of the primary key length specified in the attribute catalog.</td>
</tr>
</tbody>
</table>

Table 39: Inquiry area

Numeric comparison values placed in the inquiry area must be of the correct data type (see also under Search, “Inquiry area” on page 59).
3.10 Inquiry

The inquiry comprises the following functions:

– Selection of records conditionally, based on primary key value
– Projection of attribute sequences; for null attribute values, the default value is output
– Optional output of the attribute definition of the projected attributes

Unlike record output (see section “Record output” on page 96), responses are always of a fixed length, as default values are output for null attribute values.

A maximum of 1024 attributes can be specified per statement. With multiple attributes, each occurrence counts as one.

The inquiry can be used where the user only wishes to retrieve records conditionally on primary key values, and not conditionally on the values of other attributes. The inquiry enables records to be generated from the data base structure in the format required for processing by the application program.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values for the primary key value, the application program must put them in the inquiry area.
## Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/-</td>
</tr>
</tbody>
</table>

![Diagram of statement area]

- **Block mode**: 
  - 7
- **Lock mode**: 
  - 7
- **End id.**: 
  - 7
  - 7
  - 7
  - 1

```plaintext
... &BLKnnn &PSN000 &RNL000 &RNW000 ;
&BLNnnn
```
Key

PK (primary key) function  
0  all data  
1  equal to PK group value  
2  range of PK group values  
3  larger than PK group value  
4  equal to PK value  
5  range of PK values  
6  larger than PK value  
8  equal to record number

Format identifier  
0  attribute values only  
3  both attribute definition and attribute values

End identifier  
9  end of statement  
;  chain statement

Password (0/3)

pas  Password for a protected CALL DML table,  
    any three-character string for an unprotected CALL DML table.

Operation code (3/1)

5  Operation code for the inquiry statement

Primary key function (PK function) (4/1)

The primary key function allows records to be selected by applying conditions to the primary key or record number. The comparison values are placed in the inquiry area (see “Inquiry area” on page 116). A primary key group value can be used for selection instead of the whole primary key value.

The primary key group value identifies a group of records whose primary key values contain the primary key group value in the left of the key. The comparison value in the inquiry area must be blank-filled to the full length of the primary key.

With compound keys, the only primary key group value that may be used is a value of the compound key attribute AAB or of several compound key attributes (starting at AAB, ascending AAC, etc.), with the compound key attribute whose value is furthest to the right in the primary key group value not being of data type INTEGER or SMALLINT. It is possible, however, for the compound key attribute furthest to the right to be only partially covered by the primary key group value. Blanks must be inserted in the inquiry area for the remaining compound key attributes.
0  All data:
   All records are selected.

1  Equal to primary key group value:
   A primary key group value must be entered in the inquiry area as comparison value.
   All records containing the primary key group value left-justified in their primary key
   value are selected.

2  Range of primary key group values:
   Two primary key group values that define a range of primary key group values must
   be placed in the inquiry area.
   All records whose primary key values are greater than or equal to the first
   comparison value and less than or equal to the second comparison value are
   selected.
   The first comparison value must not be larger than the second comparison value.

3  Greater than primary key group value:
   A primary key group value must be entered in the inquiry area as the comparison
   value.
   All records whose primary key value is greater than the comparison value are
   selected.

4  Equal to primary key value:
   A primary key value must be entered in the inquiry area as the comparison value.
   The record whose primary key value is equal to the comparison value is selected.

5  Range of primary key values:
   Two primary key values defining a range of primary key values must be entered in
   the inquiry area as comparison values.
   All records whose primary key values are greater than or equal to the first
   comparison value and less than or equal to the second comparison value are
   selected.
   The first comparison value must not be larger than the second comparison value.

6  Greater than primary key value:
   A primary key value must be entered in the inquiry area as the comparison value.
   All records whose primary key value is greater than the comparison value are
   selected.

8  Equal to record number:
   A record number must be entered in the inquiry area as the comparison value.
   The record with the specified record number is selected.
Format identifier (5/1)
The format identifier determines the information output for the projected attributes:
0 Only the attribute values or values for the occurrences of a multiple attribute are output.
3 The full attribute definition and the attribute values are output.
If an attribute does not have a significant value, the default value is output instead.

Attribute selection (6/-)
An inquiry allows attribute sequences to be projected.
An attribute sequence is defined by a start and end attribute. The start and end attributes must be specified in ascending sequence.
Any number of attribute sequences can be specified, and in any sequence. Attribute sequences may also overlap.
An attribute sequence can begin or end with an occurrence of a multiple attribute. The start and end attributes are equal if just one attribute is to be referenced.
san1 Symbolic attribute name of the start attribute
san1/mmm/
   Occurrence of a multiple attribute which is the start of the attribute sequence
san2 Symbolic attribute name of the end attribute
san2/nnn/
   Occurrence of a multiple attribute which is the end of the attribute sequence
With format identifier 0, the first and last attributes of each group must be present and authorized to be read. All attributes in an attribute sequence for which read authorization does not exist are ignored.

Block mode (-/7)
The user can define how many of the responses found are to be placed in the response area.
&BLKnnn
   nnn responses are placed in the response area. The record number of each response is output.
&BLNnnn
   nnn responses are placed in the response area. The record numbers of the response records are not output.
If neither &BLKnnn nor &BLNnnn is specified, by default just one response record is placed in the response area with no record number.

**Lock mode (-/7)**

&PSN000
The response records are output without primary key values.

&RNL000
The record accessed by inquiry within a transaction is not locked.

&RNW000
The inquiry can access a record that has been locked by another transaction (dirty read). The statement is acknowledged with status code 9S. After a dirty read in block mode, no further responses are placed in the response area. If &RNW000 is omitted, a transaction attempting to access a locked record is placed in a wait state until the record becomes free.

**End identifier (-/1)**

9 Identifies the end of the statement
;
End of statement. The statement is chained to a following end TA statement.

### Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>[00] [10]</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 40: Acknowledgment area for response output
### Table 41: Acknowledgment area on error with response output

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>57, 5A, 5B, 5C, 9O, 9S</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>[san]</td>
<td>Symbolic attribute name (omitted for status code 9S)</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

### Table 42: Acknowledgment area on error without response output (part 1 of 2)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>50, 51, 53, 54, 57, 5B, 5D, 5M, 5Z, 70, 7D, 9O</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>[san]</td>
<td>Symbolic attribute name, only with 51, 53, 54, 5M</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
</tbody>
</table>

Table 42: Acknowledgment area on error without response output (part 1 of 2)
Response area

SESAM/SQL places in the response area the attribute values and possibly also the full attribute definition, depending on the format identifier.

For attributes with null values, the default value is output.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKn; the record number is not specified if the block mode is &amp;BLN or if no block mode is used.</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>[pkv]</td>
<td>Primary key value The primary key value of each response is output with the length specified in the attribute catalog. If &amp;PSN000 is specified, no primary key value is output.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value The attribute value is output with the length specified in the attribute catalog.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: responses 2 - nnn</td>
</tr>
</tbody>
</table>

Table 43: Response area for format identifier 0

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>[pkv]</td>
<td>Primary key value The primary key value of each response is output with the length specified in the attribute catalog. If &amp;PSN000 is specified, no primary key value is output.</td>
</tr>
</tbody>
</table>

Table 44: Response area for format identifier 3 (part 1 of 3)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>X’0001´ to X’0100´</td>
<td>Attribute length: 1 to 256 bytes</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X’00´ to X’0F´</td>
<td>Number of decimal places: 0 to 15 places of decimals</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X´11´</td>
<td>Data type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´21´</td>
<td>CHAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´22´</td>
<td>NUMERIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´24´</td>
<td>DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´28´</td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uninterpretable data type</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X´02´</td>
<td>Index information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´04´</td>
<td>Index locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´08´</td>
<td>Index is available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Index required</td>
</tr>
<tr>
<td></td>
<td>i-length</td>
<td></td>
<td>Index length (binary)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X´01´ to X´FF´</td>
<td>Number of occurrences of a multiple attribute as defined in attribute catalog; 1 to 255 occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>Not a multiple attribute</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>dfc</td>
<td>Default value character</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X´80´</td>
<td>Compound key information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´40´</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>Not a compound key</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>L(AC)</td>
<td>atv</td>
<td></td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute value is output with the length specified in the attribute catalog. For multiple attributes, the attribute definition and attribute value is output for each occurrence.</td>
</tr>
<tr>
<td></td>
<td>[...]</td>
<td></td>
<td>Attribute values 2 - n of the response</td>
</tr>
</tbody>
</table>

Table 44: Response area for format identifier 3
PK function 0 does not require an entry in the inquiry area. Comparison values must be placed in the inquiry area for primary key functions 1 to 6 and 8:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record: end identifier that terminates the last attribute value</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: responses 2 - nnn</td>
</tr>
</tbody>
</table>

Table 44: Response area for format identifier 3

Inquiry area

PK function 0 does not require an entry in the inquiry area.

Comparison values must be placed in the inquiry area for primary key functions 1 to 6 and 8:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv1</td>
<td>For PK functions 1 to 6: primary key (group) value of the primary key length specified in the attribute catalog</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rno</td>
<td>For PK function 8: record number</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv2]</td>
<td>For PK functions 2 and 5: primary key (group) value of the primary key length specified in the attribute catalog</td>
</tr>
</tbody>
</table>

Table 45: Inquiry area

Numeric comparison values placed in the inquiry area must be of the correct data type (see also under Search, “Inquiry area” on page 59).
3.11 Response polling

The retrieval statements search, record output and inquiry select records based on specified conditions, and by default return a result of the first response found in the application program response area. For block modes &BLKnnn or &BLNnnn, the first nnn responses are output.

With the response polling statement, all subsequent responses are polled successively.

The number of responses output by a response polling statement depends on the block mode used by the base statement.

Response polling can also be used in updated form, whereby the new comparison values for the primary key value must be entered in the same way as in the base statement. The range of primary key values of the records to be selected can thus be extended or restricted.

Strategy Z or P used with a search enables cursor files to be created. These only contain the record numbers of the selected records. The response polling statement allows the complete records whose numbers are stored in the cursor file to be retrieved. Polling conditions allow the user to page through the cursor file at will, e.g. to call off responses in any sequence.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the responses. These have the same format as the responses in the base statement. If the base statement uses block mode, nnn responses are also returned by response polling.

Inquiry area: With updated response polling, the new comparison value(s) for the primary key must be entered here.
Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/7</td>
<td>-/-1</td>
</tr>
</tbody>
</table>

Key

Polling condition
- 0: next response from cursor file
- 1: first response from the set of responses
- 2: first response from cursor file
- 3: remainder response polling following record output
- 5: previous response from cursor file
- 6: last response from cursor file
- 9: next response from set of responses

End identifier
- 9: End of statement
- ;: chain statement

Password (0/3)

pas  Any three-character string, as password protection has already been performed by the base statement.

Operation code (3/1)

7  Operation code for the response polling statement
Polling condition (4/1)

Response polling after a base statement of search without cursor file creation, record output or inquiry as base statement

1  Poll first response:
   The first response from the base statement is output with the content of the inquiry area unchanged.

   Depending on the primary key function in the base statement, one or two new comparison values can be entered in the inquiry area. The first response that satisfies the new primary key value condition is then output (updated response polling).

9  Poll next response:
   The next response is output.

   This may have been preceded either by a base statement or by response polling with polling condition 1 or 9.

Response polling after record output as the base statement

3  Poll remainder of response after record output if the response was longer than the response area (status code 4A or 9S).

Paging in a cursor file

2  Poll first response:
   The first response from the cursor file is output with the content of the inquiry area unchanged.

   Depending on the primary key function in the base statement, one or two new comparison values can be entered in the inquiry area. The first response from the cursor file that satisfies the new condition is then output.

   This statement may have been preceded by response polling with response condition 0, 2, 5 or 6. The base statement must have been a search with strategy Z or P.

0  Poll next response:
   The next response is output from the cursor file,

   This statement may have been preceded by response polling with polling condition 0, 2, 5 or 6. The base statement must have been a search with strategy Z or P.
6 Poll last response:
The last response from the cursor file is output with the content of the inquiry area unchanged.
Depending on the primary key function in the base statement, one or two new comparison values can be entered in the inquiry area. The last response from the cursor file that satisfies the new condition is then output (updated response polling).
This statement may have been preceded by response polling with polling condition 0, 2, 5 or 6. The base statement must have been a search with strategy Z or P.

5 Poll preceding response:
The content of the inquiry area must remain unchanged. The preceding response in the cursor file is output.
This statement may have been preceded by response polling with polling condition 0, 2, 5 or 6. The base statement must have been a search with strategy Z or P.

Lock mode (5/7 or -/7)

&RL000
The record accessed by response polling within a transaction is not locked.

&RNW000
The response polling statement can access a record that has been locked by another transaction (dirty read). The statement is acknowledged with status code 9S. After a dirty read in block mode, no further responses are placed in the response area.
If &RNW000 is omitted, a transaction attempting to access a locked record is placed in a wait state until the record becomes free.
If the base statement was a search with join, the same lock mode must be used in response polling as was used in the search with join.

End identifier (-/1)

9 Identifies the end of the statement
;
End of statement. The statement is chained to a following end TA statement.

If a record number cursor file has been processed by cursor file handling statements (see section “Cursor file handling” on page 125), only the response polling statements xxx729 or xxx769 can be used immediately afterwards.

If a record number cursor file has been processed by response polling statements for paging in the cursor file, the next record number must not subsequently be read by xxx11R9. The cursor file must be read from the beginning with xxx10R9 or xxx12R9.
Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>[r-no]</td>
<td>Number of responses, only if base statement was search, not with status code 16</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 46: Acknowledgment area for response

If the open statement did not define a response area large enough to hold a complete block, then each response poll only returns the number of responses that can be held in the response area. A status code of 00 is returned. The next response polling statement returns the next responses.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>70</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7T</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 47: Acknowledgment area on error
Response area

The format of the response area corresponds with that of the base statement:

- Search (see page 58)
- Search with join (see page 72)
- Index browsing (see page 91)
- Record output (see page 103 or page 312)
- Inquiry (see page 114 or page 324)

Inquiry area

For updated response polling, the new comparison values for the primary key function must be placed in the inquiry area:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv1</td>
<td>For PK functions 1 to 6 (base statement): primary key (group) value of the primary key length specified in the attribute catalog</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[rno1]</td>
<td>For PK function 8 (base statement): record number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Omitted for PK function 0.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>pkv2</td>
<td>For PK functions 2 and 5 (base statement): primary key (group) value of the primary key length specified in the attribute catalog</td>
</tr>
</tbody>
</table>

Table 48: Inquiry area

Numeric comparison values placed in the inquiry area must be of the correct data type (see “Inquiry area” on page 59).
Paging in a cursor file in block mode

In a search that creates a cursor file, block mode can be activated using &BLKnnn or &BLNnnn. nnn response records will then be output for each response poll.

Within a block, the response records are always output in the sequence in which they are held in the cursor file, irrespective of the read direction.

If exactly nnn response records are output for a response poll, SESAM/SQL returns status code 00 in the acknowledgment area. The cursor file can be read by the response polling statements xxx729, xxx709, xxx769 and xxx759.

If the cursor file has been processed to the stage where response polling returns less than nnn or even no response records, then SESAM/SQL indicates status code 10 in the acknowledgment area. The cursor file can be reread from the beginning using xxx729 or from the end using xxx769.

Example of paging in a cursor file

A cursor file has been created with blocking factor 3 in the search (xxx60Z&BLK0039). The response records whose record numbers are held in the cursor file have the following primary key values:

P00333, P00708, P01000, P03674, P05408, P05583, P09980, P11444, P11500, P12921

During response polling, the content of the inquiry area remains unchanged, i.e. no updated response polling is executed.
Response polling

Effect of the possible response polling statements:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Paging with no direction change</th>
<th>Paging with direction change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>only forwards only backwards</td>
<td>forwards, backwards forwards</td>
</tr>
<tr>
<td>Statement 1:</td>
<td>xxx729 xxx769</td>
<td>xxx729 xxx769</td>
</tr>
<tr>
<td>Response:</td>
<td>P00333 P00708 P01000</td>
<td>P11444 P11500 P12921</td>
</tr>
<tr>
<td>Acknowledgment:</td>
<td>Status 00</td>
<td>Status 00</td>
</tr>
<tr>
<td>Statement 2:</td>
<td>xxx709 xxx759</td>
<td>xxx709 xxx759</td>
</tr>
<tr>
<td>Response:</td>
<td>P03674 P05408 P05583 P09980</td>
<td>P03674 P05408 P05583 P09980</td>
</tr>
<tr>
<td>Acknowledgment:</td>
<td>Status 00</td>
<td>Status 00</td>
</tr>
<tr>
<td>Statement 3:</td>
<td>xxx709 xxx759</td>
<td>xxx709 xxx709</td>
</tr>
<tr>
<td>Response:</td>
<td>P09980 P11444 P11500</td>
<td>P00708 P01000 P03674</td>
</tr>
<tr>
<td>Acknowledgment:</td>
<td>Status 00</td>
<td>Status 00</td>
</tr>
<tr>
<td>Statement 4:</td>
<td>xxx709 xxx759</td>
<td>xxx709 xxx709</td>
</tr>
<tr>
<td>Response:</td>
<td>P12921 P00333</td>
<td>- -</td>
</tr>
<tr>
<td>Acknowledgment:</td>
<td>Status 10</td>
<td>Status 10</td>
</tr>
</tbody>
</table>

The results of using updated response polling are similar, except that the set of response records is a subset of the response records referred to by the cursor file as a whole.
3.12 Cursor file handling

A cursor file can be created for any logical file. This cursor file, like the logical file, is identified by the file identifier.

The cursor file handling statements enable the user to process a record number cursor file created by a search:
- Output the record numbers in the cursor file or join cursor file
- Write record numbers in the cursor file
- Delete record numbers from the cursor file

The user can also create and process a cursor file with records of any contents (user cursor file):
- Write variable-length records in a cursor file
- Delete records from a cursor file
- Read records in a cursor file

This type of user cursor file can be used, for example, for intermediate storage of records.

Searches with S subquestions generate a so-called sort cursor file containing for each record the record number and associated attribute values. The user can compress the sort cursor file into a pure record number cursor file.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: For the statements that read a cursor file, the DBH places the record numbers or the records read in the response area.

Inquiry area: For the statements that write to a cursor file, the application program must make the record numbers or records available in the inquiry area.
Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
</tr>
</tbody>
</table>

```
pas 1 0 R 9
   S
   D
   K
   J
   R
   1
   R
   S
   D
   2
   R
   D
   4
   R
   5
   S
```
Key

<table>
<thead>
<tr>
<th>Strategy, function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0R</td>
<td>Read first and subsequent record numbers in record number cursor file</td>
</tr>
<tr>
<td>0S</td>
<td>Write record numbers to new record number cursor file</td>
</tr>
<tr>
<td>0D</td>
<td>Delete complete cursor file</td>
</tr>
<tr>
<td>0K</td>
<td>Compress sort cursor file</td>
</tr>
<tr>
<td>1R</td>
<td>Read next record numbers in record number cursor file</td>
</tr>
<tr>
<td>1S</td>
<td>Write record numbers to existing record number cursor file</td>
</tr>
<tr>
<td>1D</td>
<td>Delete unread records or record numbers</td>
</tr>
<tr>
<td>2R</td>
<td>Read only the first record number in record number cursor file</td>
</tr>
<tr>
<td>2D</td>
<td>Delete records or record numbers that have been read</td>
</tr>
<tr>
<td>4R</td>
<td>Read first record in user cursor file</td>
</tr>
<tr>
<td>4S</td>
<td>Write first record to new user cursor file</td>
</tr>
<tr>
<td>5R</td>
<td>Read next record in user cursor file</td>
</tr>
<tr>
<td>5S</td>
<td>Write record to existing user cursor file</td>
</tr>
</tbody>
</table>

Password (0/3)

pas       Password for a protected CALL DML table,
               any three-character string for an unprotected CALL DML table.

Operation code (3/1)

1       Operation code for the cursor file handling statements

Strategy and function (4/1 and 5/1)

A combination of strategy and function allows the user to specify how he wishes to process a record number, user or sort cursor file.

Reading a record number cursor file

0R       The cursor file is read starting with the first record number. The response area is filled with record numbers from the cursor file to the length specified in the open statement.

1R       Follow-up statement following xxx10R9:
               The next record numbers are read. The response area is filled with the record numbers read.
               Follow-up statement following xxx12R9:
               The next record number is read. This one record number is made available in the response area.
2R  The first record number in the cursor file is read. This record number is made available in the response area.

**Read a user cursor file**

4R  The first record in the user cursor file is read and written to the response area.
5R  The next record is read and stored to the response area.

**Read a join cursor file**

JR  The record numbers of the first component of the join cursor file are read, starting with the first record number. The next record numbers are read in the next call. The response area is filled in the same way as for 0R or 1R.

**Write to a record number cursor file**

0S  A new cursor file is created with the record numbers specified in the inquiry area. If a cursor file already exists with the same file identifier, it is deleted.
1S  An existing cursor file is extended with the record numbers in the inquiry area.

**Write to a user cursor file**

4S  A new cursor file is created and the record in the inquiry area inserted in it. Any existing cursor file with the same file identifier is deleted.
5S  An existing user cursor file is extended with the record in the inquiry area.

The records in a user cursor file are of variable length. Thus each record in the inquiry area must be preceded by a 2-byte record length field containing the actual record length + 4, followed by two blanks. This is also necessary where one or more record numbers are written to a record number cursor file.

When a join cursor file is read, only the first component is processed.

**Delete a record number or user cursor file**

0D  The whole cursor file is deleted.
1D  All unread records or record numbers in the cursor file are deleted.
2D  All records or record numbers that have been read are deleted from the cursor file.

Deletion of single records or record numbers using 1D or 2D is only possible if preceded by at least one read.

**Compress a sort cursor file**

0K  A sort cursor file created by a search with an S subquestion is compressed into a record number cursor file.
End identifier (6/1)

9 Indicates end of statement

If a record number cursor file has been read by means of response polling statements in order to page through the cursor file (see section “Response polling” on page 117), the next record number cannot be read by xxx11R9. The cursor file must be read from the beginning using xxx10R9 or xxx12R9.

If a record number cursor file has been processed using the cursor file handling statements, the only response polling statements that can be used immediately afterwards are xxx729 (poll first response) or xxx769 (poll last response).

When a join cursor file is read, only the first component is processed.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>\001\01\</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>r-no</td>
<td>Number of responses</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
</tbody>
</table>

Table 49: Acknowledgment area for response
Cursor file handling

DML statements

### Response area

The response area is only filled by statements that read a cursor file (function R).

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>12 13 15 18 19 1Z</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 50: Acknowledgment area on error

### Inquiry area

The record numbers or records to be written to the cursor file must be made available in the inquiry area (function S).

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>length</td>
<td>Length of the actual inquiry area (binary)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>X’4040´</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>{rno[...]} {v-rec}</td>
<td>One or more record numbers to be written to a record number cursor file. A variable-length record to be written to a user cursor file.</td>
</tr>
</tbody>
</table>

Table 52: Inquiry area
3.13 Addition

The addition statement allows the user to add records to a table. A record consists of at least a primary key value.

The DBH checks the addition to see whether the specified primary key value already exists in the CALL DML table. If it does, the addition is rejected.

When making an addition, the user can use a compound key attribute as an automatic count field. The value of the count field is incremented by one for each addition (see “Automatic count field in a compound key” on page 139).

The addition statement belongs to the group of direct updating statements.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Inquiry area: The application program supplies the record to be added.

Response area: The application program gives the value of the automatic count field (if used).
Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/-</td>
<td>-/1</td>
</tr>
</tbody>
</table>

Block mode

- /7

End id.

/-1

...
DML statements

Key

PK function  C  Primary key at any position in the input record
        4  Primary key at the beginning of the input record
Attribute update function  0  Add attribute value
                      #  Compound key attribute for automatic count field
                      8  Skip attribute
End identifier  9  End of statement
                ;  Chain statement

Password (0/3)

pas  Password for a protected CALL DML table,
    any three-character string for an unprotected CALL DML table.

Operation code (3/1)

9  Operation code for the direct update statement addition

Primary key function (PK function) (4/1)

C  The primary key value or the attribute values of the compound key are located
    anywhere in the input record. A value must be specified for each compound key
    attribute. The null value can also be entered for individual compound key attributes,
    but the compound key as a whole must be unique.

4  The primary key value or the attribute values of the compound key are at the
    beginning of the input record. The attribute values for the compound key must be
    given in the sequence of the symbolic attribute names AAB, AAC etc.

Update authorization (5/1)

The statement logs a direct update under the file identifier ff in the acknowledgment area,
and obtains update authorization.

X  After execution of the statement, update authorization is cancelled again.

V  For compatibility reasons, this former update authorization is still syntactically
    permitted, but has the same meaning as X.
Update record (6/1)

N  Add a new record to the table:

  The table is first checked to see if a record with the specified primary key or record number already exists. If so, the addition is rejected. If not, the record is added with the attribute values from the input record.

Attribute selection (-/-)

Specifying symbolic attribute names defines the attributes to which attribute values are to be assigned in a new record. The attribute value for each named attribute must be made available in the inquiry area.

san  Symbolic attribute name of an attribute or of the primary key (AAA).
  For a compound key, individual compound key attributes with the symbolic attribute names AAB, AAC etc. can be specified instead of the higher-level symbolic attribute name AAA.

san/mmm/
  Symbolic attribute name and occurrence number (mmm) of a multiple attribute.

san/mmm-nnn/
  Symbolic attribute name of a multiple attribute for whose occurrences mmm to nnn attribute values are to be inserted in the new record.

A maximum of 512 attributes can be specified in a direct updating statement. This includes compound key attributes, irrespective of whether they are specified individually or by the symbolic attribute name AAA. Each separate occurrence of a multiple attribute counts as one.

Each attribute and each occurrence of a multiple attribute can only be referenced once in a statement. Ranges of occurrences must not overlap.

With primary key function 4, no primary key and thus also no compound key attributes may be included.
**Attribute update function (-/1)**

The attribute update function defines how the specified attribute is to be treated when a record is added.

0  Addition of the attribute value in the inquiry area for the named attribute

#  The compound key attribute is used as an automatic count field. The attribute is incremented by 1 for the addition. If the compound key attributes to the left of the count field have an as yet unallocated combination of attribute values, the count field is given a value of 1.

If the count field overflows, the addition is rejected with status code 9D.

This function is only permitted for compound key attributes of the data type DECIMAL, NUMERIC, INTEGER or SMALLINT (see “Automatic count field in a compound key” on page 139)

The value in the inquiry area is ignored.

8  Skip attribute. This function is not possible for compound key attributes.

**Block mode (-/7)**

The user can define how many records are to be passed to the DBH in one statement.

&BLNnnn

    nnn records are passed to the DBH in the inquiry area for addition to the table.

If &BLNnnn is omitted, the default is to add one record per statement.

If a status code other than 00 is returned on adding a record, block mode processing is terminated. The number of correctly executed additions is returned in the acknowledgment area.

If transaction-oriented security is in force and block mode is used outside transaction boundaries, the entire block is bracketed as one system transaction. If a status occurs, the transaction is closed.

**End identifier (-/1)**

9  Indicates end of statement

;  End of statement. The statement is chained to a subsequent begin TA, end TA or reset TA statement.
**Acknowledgment area**

The format of the acknowledgment area is the same for the addition, update and delete statements. The following description applies to the update (see section “Update” on page 141) and delete (see section “Deletion” on page 150) statements as well as to the addition statement.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>...</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>[r-length]</td>
<td>Response length; only applies to addition with PK function C and automatic count field</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[no]</td>
<td>For block mode &amp;BLNnnn: number of additions/updates/deletes executed</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode this is the record number of the last record added/updated/deleted</td>
</tr>
</tbody>
</table>

Table 53: Acknowledgment area after successful direct updating
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>90</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>91</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>93</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>94</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>95</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>96</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>97</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>98</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>99</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9E</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9H</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9I</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9K</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9O</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9Q</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9R</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9U</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9Z</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>san</td>
<td>Symbolic attribute name of the attribute at which the error occurred; reported by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DCN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>[r-length]</td>
<td>Response length; only applies to addition with PK function C and automatic count field</td>
</tr>
</tbody>
</table>

Table 54: Acknowledgment area on error after direct updating (part 1 of 2)
### Response area

The response area is used for output of the count field value for additions using primary key function C and attribute update function #:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>count</td>
<td>Value of compound key attribute used as automatic count field.</td>
</tr>
</tbody>
</table>

Table 55: Response area

### Inquiry area

The input records to be added to the table must be made available in the inquiry area.

Numeric attribute values placed in the inquiry area must be of the correct data type (see under Search, “Inquiry area” on page 59).

The attribute values of a numeric compound key attribute must not be negative.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be added (must be specified once only)</td>
</tr>
<tr>
<td></td>
<td>L(AC)</td>
<td>[atv]</td>
<td>Attribute value to be inserted in the new record</td>
</tr>
<tr>
<td></td>
<td>L(AC)</td>
<td>ckv</td>
<td>Compound key attribute value of the record to be added</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>For block mode: input records 2 to nnn</td>
</tr>
</tbody>
</table>

Table 56: Inquiry area for primary key function C
Addition

In a table with a compound key, one of the compound key attributes can be used as an automatic count field. The compound key attribute must, however, be one of the data types NUMERIC, INTEGER, SMALLINT or DECIMAL.

The count field always refers to records with the same base. By “records with the same base” we mean records where the values for the compound key attributes to the left of the count field are the same.

An addition causes the count field to be given the currently highest value plus one for this base. For every further addition of a record with the same base, the value of the count field is incremented by one. This technique enables a self-generating primary key to be implemented for a table with a compound key.

The compound key attribute used as an automatic count field can be given an explicit value for an addition, just like a “normal” compound key attribute. If the count field mechanism is used subsequently, the base value for incrementing is the highest value of the attribute, irrespective of whether it was generated by the user or automatically.

Addition with automatic count field incrementing after deletion

When records have been deleted from the table by a delete statement, these records are initially simply flagged internally for deletion. When another record is added with automatic count field incrementing, the count field values of the flagged records are not reassigned. The count field of the new record to be added is given the highest value before the deletion, plus one.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be added</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv[...]]</td>
<td>Attribute value(s) of the record to be added</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>For block mode: input records 2 to nnn</td>
</tr>
</tbody>
</table>

Table 57: Inquiry area for primary key function 4
**Example**

<table>
<thead>
<tr>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial situation:</strong></td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>0002</td>
</tr>
<tr>
<td></td>
<td>0003</td>
</tr>
<tr>
<td></td>
<td>0004</td>
</tr>
<tr>
<td>A database contains records with the compound key count attribute values shown on the right. The records whose count attribute value is underlined are deleted.</td>
<td></td>
</tr>
<tr>
<td><strong>Resulting situation:</strong></td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The database now only contains the records with the following values for the count attribute:</td>
<td></td>
</tr>
</tbody>
</table>

**Case A**

Addition:
- with automatic count field incrementing
  | 0001 | 0001 |
  | 0005 | 0004 |

or:

**Case B**

First addition:
- explicit, with count field value 0002
  | 0001 | 0001 |
  | 0002 | 0002 |
  | | 0004 |

Second addition:
- with automatic count field incrementing
  | 0001 | 0001 |
  | 0002 | 0002 |
  | 0003 | 0004 |
  | 0005 | 0005 |
3.14 Update

The update statement allows existing records to be updated. It comprises the following functions:

- Addition, update and deletion of attribute values
- Addition, update and deletion of occurrences of a multiple attribute
- Insert occurrences in front of an existing occurrence of a multiple attribute
- Append occurrences after the last significant occurrence of a multiple attribute

The update statement also allows attribute values to be deleted by overwriting the current attribute value with the null attribute value.

The update statement is one of the group of direct updating statements.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Inquiry area: The application program supplies the new data for the record to be updated.

The response area is not used.
Statement area

<table>
<thead>
<tr>
<th>Password code</th>
<th>Op. code</th>
<th>PK function</th>
<th>Update auth.</th>
<th>Record update</th>
<th>Attribute selection</th>
<th>Attr. update</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/-</td>
<td>-/-</td>
</tr>
</tbody>
</table>

Block mode

End id.

-7

-1

... 9

&BLNnnn

;
Key

PK function  C  Primary key at any location in the input record
               4  Primary key at the beginning of the input record
               8  Record number at the beginning of the input record

Record update function  A  Update an existing record
                         0  Update or add a record

Attribute update function  0  Update or add an attribute value
                             L  Delete an attribute value
                             N  Insert values for occurrences
                             A  Update the value of an occurrence
                             H  Append values to last significant occurrence
                             8  Skip attribute

End identifier  9  End of statement
                 ;  Chain statement

Password (0/3)

pas  Password for protected CALL DML table,
      any three-character string for unprotected CALL DML table.

Operation code (3/1)

9  Operation code for the direct updating statement update

Primary key function (PK function) (4/1)

C  The primary key value or the attribute values of the compound key are located
   anywhere in the input record. A value must be specified for each compound key
   attribute. The null value can also be entered for individual compound key attributes
   provided the standard default value character was used when the CALL DML table
   was first loaded. The compound key as a whole must be unique.

8  The record is accessed by means of the record number, which is entered in the
    inquiry area.

4  The primary key value or the attribute values of the compound key are at the
    beginning of the input record. The attribute values for the compound key must be
    given in the sequence of the symbolic attribute names AAB, AAC etc.
Update
dml statements

Update authorization (5/1)
The statement logs a direct update under the file identifier ff in the acknowledgment area, and obtains update authorization.

X After execution of the statement, update authorization is cancelled again.

V For compatibility reasons, this former update authorization is still syntactically permitted, but has the same meaning as X.

Record update function (6/1)

A Update an existing record:
All attribute values except for the primary key value can be added, updated or deleted.
An attempt to update a non-existent record is rejected with a status code.

0 Update or add a record:
The attribute values in an existing record are overwritten by those in the input record.
If an attempt is made to update a non-existent record, a new record is added with the primary key value and the attribute values from the input record.

Attribute selection (-/-)
Specifying symbolic attribute names defines the attributes to which attribute values are to be assigned in a new record. The attribute value for each named attribute must be made available in the inquiry area.

san Symbolic attribute name of an attribute or of the primary key (AAA).
For a compound key, individual compound key attributes with the symbolic attribute names AAB, AAC etc. can be specified instead of the higher-level symbolic attribute name AAA. The value of the primary key cannot be changed.

san/mmm/
Symbolic attribute name and occurrence number (mmm) of a multiple attribute.

san/mmm-nnn/
Symbolic attribute name of a multiple attribute for whose occurrences mmm to nnn attribute values are to be updated or inserted.

san/+nn/
Symbolic attribute name of a multiple attribute to which nn occurrences are to be appended.
A maximum of 512 attributes can be specified in a direct updating statement. This includes compound key attributes, irrespective of whether they are specified individually or using the symbolic attribute name AAA. Each separate occurrence of a multiple attribute counts as one.

With primary key function 4, no primary key and thus also no compound key attributes may be included.

**Attribute update function (-/1)**

The attribute update function defines whether the value of the specified attribute is to be updated, deleted or added. A value of the length of the attribute must be placed in the inquiry area for each attribute update function. For attribute update function L or 8, this entry is ignored. For all other attribute update functions, the value in the inquiry area is either added to the record or replaces an existing attribute value.

0  Updating or adding an attribute value:

  With a multiple attribute, the value of occurrence mmm or of occurrences mmm to nnn is updated if it contains a significant value. Otherwise the value(s) is (are) inserted in the multiple attribute starting from the first free, i.e. null occurrence. The new attribute values, to the full length of the attribute, must be placed in the inquiry area.

  L  A significant value of the attribute or occurrence of the multiple attribute is deleted. A null value remains unchanged. If an occurrence of a multiple attribute is deleted, all following significant attributes move back. A field of the length of the attribute must be reserved in the inquiry area, although it is not used by the statement.

  8  The attribute is ignored. Any value can be entered for the attribute in the inquiry area, and is also ignored. Attribute update function 8 cannot be used with compound key attributes or with the primary key.

*Only for multiple attributes:*

N  The value of an occurrence or the values of several occurrences are inserted in front of occurrence mmm if the latter has a significant value. Otherwise the values are inserted starting with the first free occurrence.

A  Significant values of occurrence mmm or of occurrences mmm to nnn are updated. If an occurrence does not have a significant value, the update is rejected with a status code.

H  nn values are appended starting with the first free occurrence.
Block mode (-/7)

The user can define how many records are to be passed to the DBH in one statement.

\&BLNnnn

\text{n}nn records are passed to the DBH in the inquiry area for updating in the table.

If \&BLNnnn is omitted, the default is to process one record per statement.

If a status code other than 00 is returned on updating a record, block mode processing is terminated. The number of correctly executed updates is returned in the acknowledgment area.

If transaction-oriented security is in force and block mode is used outside transaction boundaries, the entire block is bracketed as one system transaction. If a status occurs, the transaction is closed.

End identifier (-/1)

9 Indicates end of statement

; End of statement. The statement is chained to a subsequent begin TA, end TA or reset TA statement.

Acknowledgment area

The format of the acknowledgment area is identical to that of the addition statement; see “Acknowledgment area” on page 136.
Inquiry area

The primary key or record number of the record to be updated, and also the new attribute values, are entered in the inquiry area. Numeric attribute values entered in the inquiry area must be of the correct data type (see under Search, “Inquiry area” on page 59).

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be updated (must be specified once only)</td>
</tr>
<tr>
<td></td>
<td>L(AC)</td>
<td>[atv]</td>
<td>Attribute value to replace the value in the existing record</td>
</tr>
<tr>
<td></td>
<td>L(AC)</td>
<td>ckv</td>
<td>Compound key attribute value of the record to be updated</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>For block mode: input records 2 to nnn</td>
</tr>
</tbody>
</table>

Table 58: Inquiry area for primary key function C

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>rno</td>
<td>Record number of the record to be updated</td>
</tr>
<tr>
<td>4</td>
<td>L’PKV-4’</td>
<td>-</td>
<td>The record number must be padded out with any text to the full length of the primary key as specified in the attribute catalog.</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv[...]]</td>
<td>Attribute value(s) to replace those in the existing record</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>For block mode: input records 2 to nnn</td>
</tr>
</tbody>
</table>

Table 59: Inquiry area for primary key function 8

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be updated</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[atv[...]]</td>
<td>Attribute value(s) of the record to be updated</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>For block mode: input records 2 to nnn</td>
</tr>
</tbody>
</table>

Table 60: Inquiry area for primary key function 4
Deletion by changing to the null attribute value

Individual attribute values can be deleted in a record by replacing them with the null attribute value. The value of the primary key cannot be deleted in this way.

The null attribute value is the attribute value in the value range of the attribute that consists only of the default value character in printable form. The characters that can be used to form the null attribute value depend on the data type as follows:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Permitted default value characters</th>
<th>Standard default value char.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>any printable character ≤ X'40'</td>
<td>X'40' (blank)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>single-digit positive number +0 to +9 or single-digit negative number -1 to -9; with NUMERIC and DECIMAL, -0 is also possible</td>
<td>+0</td>
</tr>
</tbody>
</table>

Table 61: Characters for the null attribute value

<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value character</th>
<th>Null attribute value (printable)</th>
<th>Null attribute value (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>1</td>
<td>F'1111111111'</td>
<td>X'423A35C7'</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>F'2222222222'</td>
<td>X'0D3ED78E'</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>F'3333333333'</td>
<td>X'13DE4355'</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>F'4444444444'</td>
<td>X'1A7DAF1C'</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>F'5555555555'</td>
<td>X'211D1AE3'</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>F'6666666666'</td>
<td>X'27BC86AA'</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>F'7777777777'</td>
<td>X'2E5BF271'</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>F'8888888888'</td>
<td>X'34FB5E38'</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>F'9999999999'</td>
<td>X'3B9AC9FF'</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>F'¬1111111111'</td>
<td>X'BDC5CA39'</td>
</tr>
<tr>
<td></td>
<td>-2</td>
<td>F'¬2222222222'</td>
<td>X'F2C12382'</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>F'¬3333333333'</td>
<td>X'EC21BCAB'</td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td>F'¬4444444444'</td>
<td>X'58250E4'</td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>F'¬5555555555'</td>
<td>X'DEE2E51D'</td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>F'¬6666666666'</td>
<td>X'DB437956'</td>
</tr>
<tr>
<td></td>
<td>-7</td>
<td>F'¬7777777777'</td>
<td>X'D1A40D8F'</td>
</tr>
<tr>
<td></td>
<td>-8</td>
<td>F'¬8888888888'</td>
<td>X'CB04A1C8'</td>
</tr>
<tr>
<td></td>
<td>-9</td>
<td>F'¬9999999999'</td>
<td>X'C4653601'</td>
</tr>
</tbody>
</table>

Table 62: Summary of the null attribute values for the binary data types

INTEGER and SMALLINT

(part 1 of 2)
<table>
<thead>
<tr>
<th>Data type</th>
<th>Default value character</th>
<th>Null attribute value (printable)</th>
<th>Null attribute value (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>1</td>
<td>H’11111´</td>
<td>X´2B67´</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>H´22222´</td>
<td>X´56CE´</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>H´3333´</td>
<td>X´0D05´</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>H´4444´</td>
<td>X´115C´</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>H´5555´</td>
<td>X´15B3´</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>H´6666´</td>
<td>X´1A0A´</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>H´7777´</td>
<td>X´1E61´</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>H´8888´</td>
<td>X´22B8´</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>H´9999´</td>
<td>X´270F´</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>H´-11111´</td>
<td>X´D499´</td>
</tr>
<tr>
<td></td>
<td>-2</td>
<td>H´-22222´</td>
<td>X´A932´</td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>H´-3333´</td>
<td>X´F2FB´</td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td>H´-4444´</td>
<td>X´EEA4´</td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>H´-5555´</td>
<td>X´EA4D´</td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>H´-6666´</td>
<td>X´E5F6´</td>
</tr>
<tr>
<td></td>
<td>-7</td>
<td>H´-7777´</td>
<td>X´E19F´</td>
</tr>
<tr>
<td></td>
<td>-8</td>
<td>H´-8888´</td>
<td>X´DD48´</td>
</tr>
<tr>
<td></td>
<td>-9</td>
<td>H´-9999´</td>
<td>X´D8F1´</td>
</tr>
</tbody>
</table>

Table 62: Summary of the null attribute values for the binary data types INTEGER and SMALLINT (part 2 of 2)

Example

Default value character: -5

Null attribute value for

data type NUMERIC for a 4-character attribute: X´F5F5F5D5´
data type DECIMAL for a 4-character attribute: X´05555D´
data type SMALLINT: X´EA4D´
data type INTEGER: X´DEE2E51D´
3.15 Deletion

The deletion statement allows the user to delete complete records from the table.
The deletion statement is one of the group of direct update statements.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:
The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Inquiry area: The application program supplies the primary key value or the record number of the record to be deleted.

The response area is not used.

**Statement area**

<table>
<thead>
<tr>
<th>Password code</th>
<th>Op. code</th>
<th>PK function</th>
<th>Update auth.</th>
<th>Record update</th>
<th>Attribute selection</th>
<th>Attribute update</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/3</td>
<td>-/1</td>
</tr>
</tbody>
</table>

Block mode: End id.

/-7            /

/-1            /
Key

PK function  C  Primary key at any location in the input record  
              4  Primary key at the beginning of the input record  
              8  Record number at the beginning of the input record  
End identifier  9  End of statement  
                ;  Chain statement

Password (0/3)

pas  Password for a protected CALL DML table,  
     any three-character string for an unprotected CALL DML table.

Operation code (3/1)

9  Operation code for the direct updating statement deletion

Primary key function (PK function) (4/1)

C  The primary key value or the attribute values of the compound key are located in  
   any order in the input record. A value must be specified for each compound key  
   attribute. The null value can also be entered for individual compound key attributes,  
   provided the standard default value character was used when the CALL DML table  
   was first loaded. The compound key as a whole must be unique.  
8  The record is accessed by means of the record number, which is entered in the  
   inquiry area.  
4  The primary key value or the attribute values of the compound key are at the  
   beginning of the input record. The attribute values for the compound key must be  
   given in the sequence of the symbolic attribute names AAB, AAC etc.

Update authorization (5/1)

The statement logs a direct update under the file identifier ff in the acknowledgment area,  
and obtains update authorization.  
X  After execution of the statement, update authorization is cancelled again.  
V  For compatibility reasons, this former update authorization is still syntactically  
   permitted, but has the same meaning as X.

Record update(6/1)

L  Delete an entire record
Deletion

**Attribute selection (-/3)**

`san` Symbolic attribute name of an attribute or of the primary key (AAA).
For a compound key, individual compound key attributes with the symbolic attribute names AAB, AAC etc. can be specified instead of the higher-level symbolic attribute name AAA.

**Attribute update function(-/1)**

`L` A significant value of the attribute or occurrence of the multiple attribute is deleted. A null value remains unchanged. If an occurrence of a multiple attribute is deleted, all subsequent attributes with a significant value move up.
A field with the length of the attribute must be reserved in the inquiry area. It is not, however, used by the statement.

**Block mode (-/7)**

The user can define how many records in the table are to be deleted via a statement to the DBH.

`&BLNnnn` The DBH deletes a maximum of nnn records with one statement.
If `&BLNnnn` is omitted, the default is to delete one record per statement.
If a status code other than 00 is returned on deleting a record, block mode processing is terminated. The number of correctly executed record deletions is returned in the acknowledgment area.
If transaction-oriented security is in force and block mode is used outside transaction boundaries, the entire block is bracketed as one system transaction. If a status occurs, the transaction is closed.

**End identifier (-/1)**

`9` Indicates end of statement
`;` End of statement. The statement is chained to a subsequent begin TA, end TA or reset TA statement.

**Acknowledgment area**

The format of the acknowledgment area for the deletion statement is identical to that of the addition statement; see “Acknowledgment area” on page 136.
## Inquiry area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ckv</td>
<td>Value of a compound key attribute in the record to be deleted</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>ckv[...]</td>
<td>Attribute values of the remaining compound key attributes</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: primary key value or attribute values of the compound key for records 2 to nnn to be deleted</td>
</tr>
</tbody>
</table>

Table 63: Inquiry area for primary key function C

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>rno</td>
<td>Record number of the record to be deleted</td>
</tr>
<tr>
<td>4</td>
<td>L´ PKV -4´</td>
<td>-</td>
<td>The record number must be padded with any text to the full length of the primary key as defined in the attribute catalog.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: record numbers of records 2 to nnn to be deleted, padded as necessary to the full primary key length</td>
</tr>
</tbody>
</table>

Table 64: Inquiry area for primary key function 8

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv</td>
<td>Primary key value of the record to be deleted</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>In block mode: primary key value of records 2 to nnn to be deleted</td>
</tr>
</tbody>
</table>

Table 65: Inquiry area for primary key function 4
3.16 Follow-up update

The follow-up update statement resumes operation following one of the direct updating base statements. These base statements may be

- addition (see section “Addition” on page 131),
- update (see section “Update” on page 141), or
- deletion (see section “Deletion” on page 150).

The follow-up update is a simplified form of the direct updating statements. Details of the attributes to be added, updated or deleted are omitted, as they are already defined in the base statement.

The follow-up update can be used from the second record or, in block mode, from the second group of records.

The follow-up update statement saves on processing time compared with a base statement.

Transfer areas:

The transfer areas of the follow-up update are supplied and analyzed in the same way as the base statement.

Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/1</td>
<td>-/7</td>
<td>-/1</td>
</tr>
</tbody>
</table>

\[\text{Diagram}\]
Key

PK function

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Primary key at any location in the input record</td>
</tr>
<tr>
<td>4</td>
<td>Primary key at the beginning of the input record</td>
</tr>
<tr>
<td>8</td>
<td>Record number at the beginning of the input record</td>
</tr>
</tbody>
</table>

Record update function

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Delete an existing record</td>
</tr>
<tr>
<td>A</td>
<td>Update an existing record</td>
</tr>
<tr>
<td>0</td>
<td>Update or add a record</td>
</tr>
<tr>
<td>N</td>
<td>Add a record</td>
</tr>
</tbody>
</table>

End identifier

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>End of statement</td>
</tr>
<tr>
<td>;</td>
<td>Chain statement</td>
</tr>
</tbody>
</table>

Password (0/3)

pas  Any three-character string, as password protection has already been dealt with in the base statement.

Operation code (3/1)

7  Operation code for the direct updating statement follow-up update

Primary key function (PK function) (4/1)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>The primary key value or the attribute values of the compound key are located in any position in the input record. A value must be specified for each compound key attribute. The null value can also be entered for individual compound key attributes, provided the standard default value character was used when the CALL DML table was first loaded. The compound key as a whole must be unique.</td>
</tr>
<tr>
<td>8</td>
<td>The record is accessed by means of the record number, which is entered in the inquiry area.</td>
</tr>
<tr>
<td>4</td>
<td>The primary key value or the attribute values of the compound key are at the beginning of the input record. The attribute values for the compound key must be given in the sequence of the symbolic attribute names AAB, AAC etc.</td>
</tr>
</tbody>
</table>
The primary key function of the follow-up update must be compatible with the primary key function of the base statement. Permitted combinations:

<table>
<thead>
<tr>
<th>Base statement</th>
<th>Follow-up update</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 66: Combinations of base statements and follow-up statements

**Update authorization (5/1)**

The statement logs a direct update under the file identifier ff in the acknowledgment area, and obtains update authorization.

X After execution of the statement, update authorization is cancelled again.

V For compatibility reasons, this former update authorization is still syntactically permitted, but has the same meaning as X.

**Record update function (6/1)**

A Update an existing record:
- All attribute values except for the primary key value can be added, updated or deleted.
- An attempt to update a non-existent record is rejected with a status code.

0 Update or add a record:
- If the record already exists, the attribute values are overwritten by those in the input record.
- If an attempt is made to update a non-existent record, a new record is added with the primary key value and attribute values from the input record.

L Delete a complete record

N Add a record to the table:
- The table is first checked to see if a record with the specified primary key or record number already exists. If so, the addition is rejected. If not, the record is added with the attribute values from the input record.
**Block mode (-/7)**

The user can define how many records are to be passed to the DBH in one statement.

&BLNnnn

nnn records are updated, added or deleted in one statement.

If &BLNnnn is omitted, the default is to process one record per statement.

If a status code other than 00 is returned on a follow-up update, block mode processing is terminated. The number of correctly executed updates is returned in the acknowledgment area.

If transaction-oriented security is in force and block mode is used outside transaction boundaries, the entire block is bracketed as one system transaction. If a status occurs, the transaction is closed.

**End identifier (-/1)**

9 Indicates end of statement

; End of statement. The statement is chained to a subsequent begin TA, end TA or reset TA statement.

**Acknowledgment area**

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>......</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>[r-length]</td>
<td>Response length, only for addition with PK function C and automatic count field</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[no]</td>
<td>In block mode &amp;BLNnnn: number of additions/updates/record deletions executed</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, record number of the record last added/updated/deleted</td>
</tr>
</tbody>
</table>

Table 67: Acknowledgment area after successful direct update
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>70 71 91 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F 9G 9H 9I 9J 9K 9L 9M 9N 9O 9P 9Q 9R 9S 9T 9U</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>san...</td>
<td>Symbolic attribute name of the attribute at which the error occurred; reported by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>- DBH,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOD...</td>
<td>- SESMOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINK</td>
<td>- SESLINK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DCN...</td>
<td>- SESDCN</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>[r-length]</td>
<td>Response length, only for addition with PK function C and automatic count field in block mode</td>
</tr>
</tbody>
</table>

Table 68: Acknowledgment area on error after direct update (part 1 of 2)
Response area

The response area is used to output the count field value when primary key function C and attribute update function # were used in an addition base statement:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>count</td>
<td>Value of the compound key attribute used as an automatic count field</td>
</tr>
</tbody>
</table>

Table 69: Response Area

Inquiry area

The inquiry area of the follow-up update must have the same format as the inquiry area for the base statement:

– addition (see page 138)
– update (see page 147)
– deletion (see page 153)
3.17 Attribute information

The attribute information statements provides information about the definition of one or more attributes. The following information can be obtained:

- symbolic and/or verbal attribute name
- data type
- attribute length and number of decimal places
- details of the index number of occurrences of a multiple attribute
- default value character
- details of the compound key

Unlike the record output (see section “Record output” on page 96) and inquiry (see section “Inquiry” on page 107) statements, only the above information is output for the attribute information statements. The attribute values cannot be obtained.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:
The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: The DBH returns the requested information.

The inquiry area is not used.
Statement area

<table>
<thead>
<tr>
<th>Pass-word</th>
<th>Op. code</th>
<th>Information function</th>
<th>Attribute selection:</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1 6/-</td>
<td>-/1</td>
</tr>
</tbody>
</table>

```
pas 3 1 1 san 9
    2 3 san1 san2
    4  v a n
    2  van
    4  vgn
```  

Key

<table>
<thead>
<tr>
<th>Information function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Information about the symbolic attribute name</td>
</tr>
<tr>
<td>2</td>
<td>Information about the verbal attribute name</td>
</tr>
<tr>
<td>3</td>
<td>Information about the symbolic and verbal attribute name</td>
</tr>
<tr>
<td>A</td>
<td>Information about the complete attribute definition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>End identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>End of statement</td>
</tr>
<tr>
<td>;</td>
<td>chain statement</td>
</tr>
</tbody>
</table>

Password (0/3)

`pas` Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

Operation code (3/1)

3 Operation code for the attribute information statement
**Information function (4/1)**

The information function defines what information is to be output about an attribute.

1. Output symbolic attribute name
2. Output verbal attribute name
3. Output symbolic and verbal attribute names
A. Output complete attribute definition, including the symbolic and verbal attribute names

**Attribute selection (5/-)**

Attribute selection requires a code number indicating the type of attribute name to follow, and the attribute name itself.

{id} attribute name

1. san Symbolic attribute name
3. san1 Symbolic attribute name of the start attribute in an attribute sequence
   san2 Symbolic attribute name of the end attribute in an attribute sequence
2. van Verbal attribute name: verbal attribute names less than 31 characters long must be blank-filled on the right to the full length of 31 characters.
4. vgn Group name for verbal attribute names: common part of verbal attribute names all beginning with the same character string, blank-filled on the right to the full length of 31 characters if necessary.

The maximum number of attributes is limited by the capacity of the response area (max. 32000 bytes).

**End identifier (-/1)**

9 Indicates end of statement
;
End of statement. The statement is chained to a following end TA statement.

**Acknowledgment area**

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
</tbody>
</table>

Table 70: Acknowledgment area for response
### Table 70: Acknowledgment area for response

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>r-length</td>
<td>Response length (binary)</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 71: Acknowledgment area on error

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>({1K, 30, 31, 33, 3B, 3V, 3Z})</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>({\text{van}, \text{san}})</td>
<td>Verbal attribute name for status code 31, 3V Symbolic attribute name for status code 31, 33</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>r-length</td>
<td>Response length (binary)</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Response area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Responses for attributes 2 to n</td>
</tr>
</tbody>
</table>

Table 72: Response area for information function 1

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31</td>
<td>van</td>
<td>Verbal attribute name</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Responses for attributes 2 to n</td>
</tr>
</tbody>
</table>

Table 73: Response area for information function 2

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>van</td>
<td>Verbal attribute name</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>[...]</td>
<td>Responses for attributes 2 to n</td>
</tr>
</tbody>
</table>

Table 74: Response area for information function 3

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>van</td>
<td>Verbal attribute name</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>X’0001´ to X’0100´</td>
<td>Attribute length: 1 to 256 bytes</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>X’00´ to X’0F´</td>
<td>Number of decimal places: 0 to 15 decimal places</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>X’11´</td>
<td>Data type: CHAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’21´</td>
<td>NUMERIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’22´</td>
<td>DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’24´</td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’28´</td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’00´</td>
<td>uninterpretable data format</td>
</tr>
</tbody>
</table>

Table 75: Response area for information function A (part 1 of 2)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>1</td>
<td>X’02´</td>
<td>Index information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’04´</td>
<td>Index locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’08´</td>
<td>Index available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Index required</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>i-length</td>
<td>Index length (binary)</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>X’01´ to X´FF´</td>
<td>Number of occurrences of a multiple attribute defined in the attribute catalog:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>1 to 255 occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not a multiple attribute</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>dfc</td>
<td>Default character</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>X´80´</td>
<td>Compound key information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´40´</td>
<td>Compound key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not a compound key</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>[...]</td>
<td>Responses for attributes 2 to n</td>
</tr>
</tbody>
</table>

Table 75: Response area for information function A
3.18 Statements for transaction-oriented security

The following statements are involved in transaction-oriented security:

- The “begin transaction” (BTA) statement opens a transaction for the application program with SESAM/SQL DBH.

- The “end transaction” (ETA) statement closes the transaction for the DBH.

  BTA and ETA form what are called the transaction boundaries:
  The statements between BTA and ETA are either executed in full or not at all.
  ETA implicitly closes all logical files that have not yet been closed in the transaction. It is thus not absolutely necessary to close logical files explicitly.

- Reset transaction (RTA) terminates processing within a transaction boundary. The following action is taken for the transaction:
  - all direct and follow-up updates performed are reset
  - all locks and save data are released
  - base statements (for search and direct updating) are invalidated.

  No follow-up statements are permitted after RTA. It does not matter if the base statement was issued before or in the reset transaction.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:

  The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

The inquiry and response areas are not used.
Statement area

<table>
<thead>
<tr>
<th>Password</th>
<th>Operation code</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/3</td>
<td>6/1</td>
</tr>
<tr>
<td>pas</td>
<td>90B</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>90C</td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>90R</td>
<td>9</td>
</tr>
</tbody>
</table>

Password (0/3)

pas Any three-character string for a protected or unprotected CALL DML table.

Operation code (3/3)

90B Operation code for the “begin transaction” statement (BTA).
90C Operation code for the “end transaction” statement (ETA).
90R Operation code for the “reset transaction” statement (RTA).

End identifier (6/1)

9 Indicates end of statement
; End of statement;
the following combinations are permitted:
– BTA; open1; open2; etc.
– BTA; statement
– ETA; BTA (not under UTM)
– ETA; user close
– ETA; logical file close
### Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>({00, 90, 91, 9K, 9N, 9R})</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>({MOD, DCN})</td>
<td>reported by the DBH, by SESMOD, SESDCN</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[ss]</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 76: Acknowledgment area
3.19 Administrator Open

The administrator open statement opens communication between an administrator program and the SESAM/SQL DBH. This statement may only be issued in programs that have not opened any logical files and in which administration in the DBH is permitted (see the “Database Operation” manual).

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The DBH returns the acknowledgment to the statement.

The inquiry and response areas must be made available, but are not used.

Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/17</td>
<td>21/5</td>
<td>26/5</td>
<td>31/1</td>
<td>32/2</td>
<td>34/1</td>
</tr>
</tbody>
</table>

adm → 2 → length-r → length-i → A → ff → 9

tabname

Password (0/3)
adm Administrator password

Operation code (3/1)

2 Operation code for the administrator open statement
Table name (4/17)

For local access, 17 blanks can be entered instead of the table name.

**tabname**

For SESAM/SQL DCN: name of a table operating under the DBH to be administered.

The table name must be held in the CALL DML catalog list (see the “Database Operation” manual, ADD-OLD-TABLE-CATALOG-LIST).

Data base names less than 17 characters long must be right-filled with blanks to the full length of 17.

Length of response area (21/5)

SESAM/SQL knows the maximum length of the response area and can therefore work out the buffer size required (see the “Database Operation” manual, TRANSFER-CONTAINER).

**length-r**

Maximum length of response area in bytes:

- The decimal number to be entered is the largest response length expected by the application program.
- Minimum value: 0
- Maximum value: 32000

Length of inquiry area (26/5)

SESAM/SQL knows the maximum length of the inquiry area and can therefore work out the buffer size required.

**length-i**

Maximum length of inquiry area in bytes:

- The decimal number to be entered is the total length of the primary key and attribute comparison values.
- Minimum value: 0
- Maximum value: 32000

Function code (31/1)

A Administration of the SESAM/SQL DBH
File identifier (32/2)

ff File identifier under which the SESAM/SQL DBH is administered.
Permitted characters are the numbers 0 to 8 and any letter.

End identifier (34/1)

9 Indicates the end of the administrator open statement

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{00, 20, 2B, 2M, 2X, 2Y}</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>{MOD, DCN, Dxxx}</td>
<td>reported by the DBH,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– SESMOD,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– SESDCN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– DMS error</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 77: Acknowledgment area
3.20 Administration statements for the DBH

The administration statement allows administration commands to be sent to the SESAM/SQL DBH.

The following administration activities can be carried out:
– Output of information about SESAM/SQL operation
– Control of SESAM/SQL operation
– Output of diagnostic documentation

A detailed description of the administration commands can be found in the “Database Operation” manual.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The DBH returns the acknowledgment to the statement.

Response area: The responses to administration commands are output in the response area.

The statement does not use the inquiry area.

Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/3</td>
<td>6/-</td>
<td>-/2</td>
</tr>
</tbody>
</table>

adm ➔ 010 ➔ admin ➔ 9..

Password (0/3)

adm Administrator password;
this password must be identical to the password specified for the DBH option ADMINISTRATOR when the DBH was started. If restrictions for the system administrator were set for this DBH option, these restrictions also apply to the current user.
DML statements

Operation code (3/3)

010       Operation code for DBH administration

Administration command (6/-)

Only SEND-MSG formats are permitted at the CALL DML interface. The administration commands are described in the “Database Operation” manual.

End identifier (-/2)

9.. Indicates the end of the administration command

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{00 01 0A 0B 0K 0Y 0Z}</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>a-length</td>
<td>Response length</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[ss]</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 78: Acknowledgment area

Response area

The response is the message from the respective administration command from the terminal.
A detailed description of the administration commands can be found in the “Database Operation” manual.
The response is edited for printing and can be printed in units of 79 characters per line by means of the WRTRD macro. The contents of the response area must not be analyzed, as the format may change from version to version.

### 3.21 Administration statement for SESDCN

The SESDCN administration statement allows administration commands to be sent to SESAM/SQL-DCN.

The following administration activities can be carried out:

- Output of information about SESAM/SQL-DCN operation
- Control of SESAM/SQL-DCN operation
- Output of diagnostic documentation

A detailed description of the SESDCN administration commands can be found in the “Database Operation” manual.

**Contents of transfer areas:**

**Statement area:** The application program supplies the statement.

**Acknowledgment area:** SESDCN returns the acknowledgment to the statement.

**Response area:** The responses to administration commands (SESDCN administration) are output in the response area.

The statement does not use the inquiry area.

#### Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/3</td>
<td>6/2</td>
<td>8/1 9/8</td>
<td>-/1</td>
<td>-/1</td>
<td>-/1</td>
<td>-/1/≤65</td>
</tr>
</tbody>
</table>

adm → 020 → length-r → P → entry → C → entry → D → entry → A → admin → 9 →
Password (0/3)

adm   Administrator password; this password must be identical to the password entered for the SESDCN option ADMINISTRATOR when the DBH was started.

Operation code (3/3)

020   Operation code for SESDCN administration

Length of response area (6/2)

length-r
   Maximum value: 32000

Processor entry (8/9)

P      Identifier of the processor entry
entry   Symbolic device name of the processor on which the executing SESDCN is located;
        Content: A-Z, 0-9, or blanks.
        The first character must be a letter. If the content is shorter than 8 characters, the
        field must be right-filled with blanks.

If no processor is entered, DBCON uses the HOME processor.

Configuration entry (-/2)

C      Identifier for the configuration entry;
        Content: A-Z, 0-9, or blanks.
entry   Name of the configuration in which SESDCN is running;
        Content: A-Z, 0-9, or blanks.

If no configuration is entered, DBCON uses the HOME configuration.

DCN entry (-/2)

D      Id for the DCN entry
entry   Communication name of the SESDCN that is to perform administration;
        Content: A-Z, 0-9, or blanks.

If no communication name is entered, DBCON uses blanks.
DCN administration command (-/≤65)

A

Id for the SESDCN administration command

admin

The administration commands are described in the “Database Operation” manual; Maximum length of an entry: 64.

End identifier (-/2)

9_=' Indicates the end of the administration command

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>[00]</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[OA]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0Y]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[02]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>MOD_</td>
<td>Identifier</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>[ss]</td>
<td>[ss]</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 79: Acknowledgment area
Response area

Output is in table form with “num” of rows. Each row corresponds to an element. The first 4 rows comprise the table headings.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>l-tot</td>
<td>Length used in response area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hexadecimal: ≤32000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>l-elem</td>
<td>Length of the individual element or part of the response</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hexadecimal: ≤80</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>num</td>
<td>Number of elements or parts of the response in hexadecimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>notation</td>
</tr>
<tr>
<td>6</td>
<td>l-elem</td>
<td>elem1</td>
<td>Parts of the response;</td>
</tr>
<tr>
<td></td>
<td>l-elem</td>
<td>elem2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>l-elem</td>
<td>elem3</td>
<td></td>
</tr>
</tbody>
</table>

Table 80: Response area
4 Using DML statements

This chapter contains a description of the following functions of the DML statements using examples:

- “Opening logical files” on page 190
- “Closing logical files” on page 191
- “Retrieval using search” on page 192
- “Retrieval using search with join” on page 219
- “Cursor technique for search/search with join” on page 222
- “Inquiring on attribute value frequency (index browsing)” on page 233
- “Defining comparison values” on page 236
- “Retrieval using record output” on page 241
- “Retrieval using inquiry” on page 243
- “Adding new records” on page 245
- “Updating records” on page 248
- “Deleting records” on page 250
4.1 Examples

The application examples for the DML statements are based on the CALL DML tables COMPANY and SALES.

CALL DML table COMPANY

This CALL DML table was created by migrating a “diagonalized” database in a previous version.

The original COMPANY database contains the relations ARTICLE, CUSTOMER and PERSONNEL. The relations consist of the following attributes:

ARTICLE = article-number, article-name, price, stock

CUSTOMER = customer-number, c-lastname, c-firstname, c-street, c-zip, c-city, customer since, c-discount

PERSONNEL = personnel-number, p-lastname, p-firstname, p-street, p-zip, p-city, dateofbirth, department, languages, salary
<table>
<thead>
<tr>
<th>PK</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PKKEY</td>
</tr>
<tr>
<td></td>
<td>ARTICLE relation</td>
</tr>
<tr>
<td></td>
<td>CUSTOMER relation</td>
</tr>
<tr>
<td></td>
<td>PERSONNEL relation</td>
</tr>
</tbody>
</table>

Table 81: COMPANY database
### ARTICLE

<table>
<thead>
<tr>
<th>SAN</th>
<th>SAN verbal attribute name (VAN)</th>
<th>FORMAT</th>
<th>LTH</th>
<th>DP</th>
<th>MF</th>
<th>DF</th>
<th>KEY</th>
<th>I-LTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>PKEY</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KEY</td>
</tr>
<tr>
<td>AA8</td>
<td>ANAME</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td>KEY</td>
<td></td>
</tr>
<tr>
<td>AB6</td>
<td>APRICE</td>
<td>NUMERIC</td>
<td>005</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
<td>-0</td>
</tr>
<tr>
<td>AC4</td>
<td>ASTOCK</td>
<td>NUMERIC</td>
<td>004</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td>-0</td>
</tr>
</tbody>
</table>

### CUSTOMER

<table>
<thead>
<tr>
<th>SAN</th>
<th>SAN verbal attribute name (VAN)</th>
<th>FORMAT</th>
<th>LTH</th>
<th>DP</th>
<th>MF</th>
<th>DF</th>
<th>KEY</th>
<th>I-LTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>PKEY</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KEY</td>
</tr>
<tr>
<td>AD2</td>
<td>CLASTNAME</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td>KEY</td>
<td></td>
</tr>
<tr>
<td>AEZ</td>
<td>CFIRSTNAME</td>
<td>CHAR</td>
<td>012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFX</td>
<td>CSTREET</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AGV</td>
<td>CZIP</td>
<td>CHAR</td>
<td>005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AHT</td>
<td>CCITY</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AJR</td>
<td>CSINCE</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AKP</td>
<td>CDISCOUNT</td>
<td>NUMERIC</td>
<td>004</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### PERSONNEL

<table>
<thead>
<tr>
<th>SAN</th>
<th>SAN verbal attribute name (VAN)</th>
<th>FORMAT</th>
<th>LTH</th>
<th>DP</th>
<th>MF</th>
<th>DF</th>
<th>KEY</th>
<th>I-LTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>PKEY</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KEY</td>
</tr>
<tr>
<td>ALM</td>
<td>PLASTNAME</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMK</td>
<td>PFIRSTNAME</td>
<td>CHAR</td>
<td>012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANH</td>
<td>PSTREET</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF</td>
<td>PZP</td>
<td>CHAR</td>
<td>005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AOD</td>
<td>PCITY</td>
<td>CHAR</td>
<td>015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARB</td>
<td>PDOFB</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR9</td>
<td>PDEPT</td>
<td>CHAR</td>
<td>004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS7</td>
<td>PLANGS</td>
<td>CHAR</td>
<td>005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT5</td>
<td>PSALARY</td>
<td>NUMERIC</td>
<td>007</td>
<td>02</td>
<td></td>
<td>010</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

where:

- **LTH**: attribute length
- **DP**: number of decimal places
- **MF**: number of occurrences of a multiple attribute
- **DF**: default value character
- **KEY**: primary key, compound key
- **COMP**: compound key or compound key attribute
- **I-LTH**: length of index
Output of the information schema

The way in which CALL DML is used has not been changed in any way by the migration process.

The utility monitor output below contains the CALL DML table COMPANY (see the “Utility-Monitor” manual):

```plaintext
*** INF.9.3.7 INFORMATION SCHEMA, BASETABLE, COLUMNS *** - 1 -

CATALOG : CALLCOMPANY SCHEMA : COMPANYSCH
        TABLE : COMPANY

COLUMN
ASTOCK
ANAME
APRICE
CLASTNAME
CZIP
CDISCOUNT
CSINCE
CCITY
CSTREET
CFIRSTNAME
PDEPT
PLANGS
PDOFB
PSALARY
PLASTNAME
PZIP
PCITY
PSTREET
PFIRSTNAME
PKEY

*** INF.9.3.3 INFORMATION SCHEMA, BASETABLE, KEY COLUMN *** - 2 -

CATALOG : CALLCOMPANY SCHEMA : COMPANYSCH
        TABLE : COMPANY

KEY COLUMN
CONSTRAINT
POSITION

PKEY
PK9941108093845000
00001
```

The original “relations” ARTICLE, CUSTOMER and PERSONNEL are not included in the CALL DML table.
CALL DML table SALES

This is a CALL DML table created by migration from a database in a previous version.

The original SALES database contains the relation ORDER. The relation consists of the following attributes:

ORDER= order-number, quantity, customer-number, date

ORDER

<table>
<thead>
<tr>
<th>SAN</th>
<th>verbal attribute name (VAN)</th>
<th>FORMAT</th>
<th>LTH DP</th>
<th>MF</th>
<th>DF</th>
<th>KEY</th>
<th>I-LTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>CMPDKEY</td>
<td>CHAR</td>
<td>010</td>
<td></td>
<td></td>
<td>0</td>
<td>COMP</td>
</tr>
<tr>
<td>AAB</td>
<td>ORDNO</td>
<td>NUMERIC</td>
<td>004</td>
<td>00</td>
<td></td>
<td></td>
<td>COMP</td>
</tr>
<tr>
<td>AAC</td>
<td>ARTNO</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td>0</td>
<td>COMP</td>
</tr>
<tr>
<td>ABB</td>
<td>QUANTITY</td>
<td>NUMERIC</td>
<td>004</td>
<td>00</td>
<td></td>
<td></td>
<td>COMP</td>
</tr>
<tr>
<td>AB9</td>
<td>CUSTNO</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td>COMP</td>
</tr>
<tr>
<td>AC7</td>
<td>ORDDATE</td>
<td>CHAR</td>
<td>006</td>
<td></td>
<td></td>
<td></td>
<td>COMP</td>
</tr>
</tbody>
</table>

where:

- **LTH** attribute length
- **DP** number of decimal places
- **MF** number of occurrences of a multiple attribute
- **DF** default value character
- **KEY** KEY: primary key,
  COMP: compound key or compound key attribute
- **I-LTH** length of index
Using DML statements

### Output of the information schema

The CALL DML application was not changed in any way by the migration process.

The utility monitor output below contains the CALL DML table COMPANY (see the “Utility-Monitor” manual):

#### OUTPUT

<table>
<thead>
<tr>
<th>CATALOG</th>
<th>SCHEMA</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALLCOMPANY</td>
<td>SALESSCH</td>
<td>SALES</td>
</tr>
</tbody>
</table>

**COLUMN**

- ARTNO
- ORDDATE
- ORDNO
- CUSTNO
- QUANTITY

#### KEY-COLUMN

<table>
<thead>
<tr>
<th>KEY-COLUMN</th>
<th>CONSTRAINT</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTNO</td>
<td>CMPDKEY</td>
<td>00002</td>
</tr>
<tr>
<td>ORDNO</td>
<td>CMPDKEY</td>
<td>00001</td>
</tr>
</tbody>
</table>

The original “relation” ORDER is not included in the CALL DML table.
### Contents of the CALL DML tables COMPANY and SALES

**COMPANY; ARTICLE**

<table>
<thead>
<tr>
<th>AAA</th>
<th>AA8</th>
<th>AB6</th>
<th>AC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A00100</td>
<td>PENCIL</td>
<td>0.59</td>
<td>7576</td>
</tr>
<tr>
<td>A00200</td>
<td>RULER</td>
<td>1.89</td>
<td>503</td>
</tr>
<tr>
<td>A00300</td>
<td>WRITING PAD</td>
<td>1.79</td>
<td>1763</td>
</tr>
<tr>
<td>A00340</td>
<td>BALL-POINT PEN</td>
<td>2.19</td>
<td>4908</td>
</tr>
<tr>
<td>A00400</td>
<td>BRUSH</td>
<td>1.19</td>
<td>1054</td>
</tr>
<tr>
<td>A00650</td>
<td>FOLDER</td>
<td>3.95</td>
<td>107</td>
</tr>
<tr>
<td>A01400</td>
<td>ERASER</td>
<td>1.59</td>
<td>816</td>
</tr>
<tr>
<td>A01750</td>
<td>GLUE</td>
<td>3.99</td>
<td>35</td>
</tr>
<tr>
<td>A08880</td>
<td>CALENDER</td>
<td>2.56</td>
<td>84</td>
</tr>
<tr>
<td>A09000</td>
<td>DRAWING PAD</td>
<td>1.79</td>
<td>527</td>
</tr>
<tr>
<td>A09050</td>
<td>NOTEPAD</td>
<td>0.99</td>
<td>750</td>
</tr>
</tbody>
</table>

**COMPANY; CUSTOMER**

<table>
<thead>
<tr>
<th>AAA</th>
<th>AD2</th>
<th>AEZ</th>
<th>AFX</th>
<th>AGV</th>
<th>AHT</th>
<th>AJR</th>
<th>AKP</th>
</tr>
</thead>
</table>
| C01732  | HUBER | GEORG  | MAINSTRASSE    | 60311| FRANKFURT | 860114| 20,00
| C13486  | MILLER | PETER  | LEOPOLDSTRASSE | 80802| MUNICHEN | 860325| 10,00
| C17109  | KELLER | MICHAEL | GOETHESTRASSE | 80336| MUNICHEN | 860520| 15,00
| C20070  | REITER | ALEXANDER | ANNASTRASSE | 86150| AUGSBURG | 861217| 5,00
| C23979  | STEINER | WALTER | BURGSTRASSE | 90403| NUERNBERG | 870420| 10,00
| C37424  | MAYER | ROBERT  | WALDSTRASSE    | 63457| HANAU   | 870901| 5,00
| C37597  | FISCHER | OTTO  | SIEMENSSTRASSE | 91052| ERLANGER | 871118| 0,00
| C38210  | HUBER | GEORG  | LESSINGSTRASSE | 63073| OFFENBACH | 880223| 20,00
| C40013  | SCHNEIDER | WERNER | DONAUSTRASSSE | 93059| REGENSBURG | 880927| 0,00
### Using DML statements

#### Examples

**COMPANY: PERSONNEL**

<table>
<thead>
<tr>
<th>AAA</th>
<th>ALM</th>
<th>AMK</th>
<th>ANH</th>
<th>APF</th>
<th>AQP</th>
<th>ARB</th>
<th>AR9</th>
<th>AS7</th>
<th>AT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P00333</td>
<td>HUBER</td>
<td>ANDREAS</td>
<td>GOETHEPLATZ</td>
<td>80337</td>
<td>MUCNCHEN</td>
<td>430127</td>
<td>ABT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P00708</td>
<td>BINDER</td>
<td>THOMAS</td>
<td>STEINSTRASSE</td>
<td>81667</td>
<td>MUCNCHEN</td>
<td>501117</td>
<td>ABT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P01000</td>
<td>MAYER</td>
<td>DORIS</td>
<td>BAADERSTRASSE</td>
<td>80469</td>
<td>MUCNCHEN</td>
<td>470416</td>
<td>ABT3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P01140</td>
<td>STOLL</td>
<td>WERNER</td>
<td>POSTweg</td>
<td>85221</td>
<td>DACHAU</td>
<td>600301</td>
<td>ABT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P03674</td>
<td>WINKLER</td>
<td>ANDREA</td>
<td>BELGRADSTRASSE</td>
<td>80796</td>
<td>MUCNCHEN</td>
<td>520928</td>
<td>ABT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P05408</td>
<td>BAUER</td>
<td>RUDOLF</td>
<td>SEESTRASSE</td>
<td>82319</td>
<td>STARNBERG</td>
<td>640421</td>
<td>ABT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P05583</td>
<td>BOTT</td>
<td>MICHAEL</td>
<td>ROSENSTRASSE</td>
<td>60313</td>
<td>FRANKFURT</td>
<td>590603</td>
<td>ZET1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P09980</td>
<td>RICHTER</td>
<td>STEFAN</td>
<td>BRAHMSSTRASSE</td>
<td>64625</td>
<td>BENSHEIM</td>
<td>560710</td>
<td>ZET1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P11444</td>
<td>METZGER</td>
<td>KLAUS</td>
<td>BORSIGSTRASSE</td>
<td>60388</td>
<td>FRANKFURT</td>
<td>491001</td>
<td>ZET1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P11500</td>
<td>KAISER</td>
<td>FRANZ</td>
<td>KOPPSTRASSE</td>
<td>81379</td>
<td>MUCNCHEN</td>
<td>530823</td>
<td>ABT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P12691</td>
<td>RUF</td>
<td>MICHAEL</td>
<td>LUDWIGSTRASSE</td>
<td>80359</td>
<td>MUCNCHEN</td>
<td>620114</td>
<td>ABT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P13345</td>
<td>BERGER</td>
<td>FRED</td>
<td>GARTENSTRASSE</td>
<td>85354</td>
<td>FREISING</td>
<td>660420</td>
<td>ABT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P15863</td>
<td>BARTEL</td>
<td>ELVIRA</td>
<td>FELDSTRASSE</td>
<td>83022</td>
<td>ROSENHEIM</td>
<td>541010</td>
<td>ABT1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P19478</td>
<td>MAIER</td>
<td>MICHAEL</td>
<td>BAHNHOFSTRASSE</td>
<td>85435</td>
<td>ERDING</td>
<td>551130</td>
<td>ABT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P19479</td>
<td>SCHUBER</td>
<td>PETER</td>
<td>MILCHSTRASSE</td>
<td>81667</td>
<td>MUCNCHEN</td>
<td>401129</td>
<td>ABT4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P20099</td>
<td>KAISER</td>
<td>SUSANNE</td>
<td>KOPPSTRASSE</td>
<td>81379</td>
<td>MUCNCHEN</td>
<td>580208</td>
<td>ABT3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P21500</td>
<td>WEISS</td>
<td>HANS</td>
<td>ROEMERSTRASSE</td>
<td>60311</td>
<td>FRANKFURT</td>
<td>570520</td>
<td>ZET1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AS7/ 001-006**

<table>
<thead>
<tr>
<th>ENGL</th>
<th>ENGL</th>
<th>ITAL</th>
<th>PORT</th>
<th>SPAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ENGL

ENGL FRANZ

ENGL SPAN

---

U1054-J-Z125-11-76  187
### Examples

#### Using DML statements

<table>
<thead>
<tr>
<th>AT5/</th>
<th>001-010/</th>
</tr>
</thead>
<tbody>
<tr>
<td>4140.00</td>
<td>4000.00</td>
</tr>
<tr>
<td>3312.00</td>
<td>3200.00</td>
</tr>
<tr>
<td>3312.00</td>
<td>3200.00</td>
</tr>
<tr>
<td>3933.00</td>
<td>3800.00</td>
</tr>
<tr>
<td>3622.50</td>
<td>3500.00</td>
</tr>
<tr>
<td>3933.00</td>
<td>3800.00</td>
</tr>
<tr>
<td>3200.50</td>
<td>3100.00</td>
</tr>
<tr>
<td>3933.00</td>
<td>3800.00</td>
</tr>
<tr>
<td>3622.50</td>
<td>3500.00</td>
</tr>
<tr>
<td>4140.00</td>
<td>4000.00</td>
</tr>
<tr>
<td>3933.00</td>
<td>3800.00</td>
</tr>
<tr>
<td>3312.00</td>
<td>3200.00</td>
</tr>
<tr>
<td>3933.00</td>
<td>3800.00</td>
</tr>
<tr>
<td>2992.50</td>
<td>2850.00</td>
</tr>
<tr>
<td>3675.00</td>
<td>3500.00</td>
</tr>
<tr>
<td>3255.00</td>
<td>3100.00</td>
</tr>
<tr>
<td>2992.50</td>
<td>2850.00</td>
</tr>
</tbody>
</table>
### Using DML statements

#### Examples

**SALES; ORDER**

<table>
<thead>
<tr>
<th>AAA</th>
<th>AAB</th>
<th>AAC</th>
<th>ABB</th>
<th>AB9</th>
<th>AC7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1011</td>
<td>0</td>
<td>K23979</td>
<td>880930</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>A00200</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>A00650</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td>0</td>
<td>K01732</td>
<td>880930</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td>A01400</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1013</td>
<td>0</td>
<td>K38210</td>
<td>881003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1013</td>
<td>A00100</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1013</td>
<td>A01400</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1013</td>
<td>A09050</td>
<td>230</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1014</td>
<td>0</td>
<td>K20070</td>
<td>881004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1014</td>
<td>A00400</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1014</td>
<td>A09000</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.2 Opening logical files

A logical file is opened by means of the open statement (see section “Open” on page 29) in which the user defines the following variables:

- two-character file identifier for the logical file
- length of the response area
- length of the inquiry area
- access authorization for statements under this and other file identifiers

**Example**

A logical file with file identifier CO is to be opened for the CALL DML table COMPANY. For subsequent processing, response and inquiry areas each 1000 bytes in length are required. Only read access is to be allowed under file identifier CO, but direct updating is allowed under any other file identifier. This is defined by means of function code R in the statement.

Statement area:

| xxx | 2 | COMPANY: | 01000 | 01000 | R | FI | 9 |

| xxx | any three-character string, as the CALL DML table is not password-protected |
| 2 | operation code for the open statement |
| COMPANY: | CALL DML table name |
| 01000 | response area length |
| 01000 | inquiry area length |
| R | function code |
| CO | file identifier |
| 9 | end identifier |
4.3 Closing logical files

Logical files are closed by means of the close statement (see section “Close” on page 34). The statement can be used to close a single logical file, or all the logical files for a requester.

*Example 1*

Close logical file CO (file close).

Statement area:

```
xxx 8 ******************* CO 9
```

- `xxx` any three-character string instead of the password
- `8` operation code for the close statement
- `CO` file identifier of the file to be closed
- `9` end identifier

*Example 2*

Close all logical files for the requester (user close).

Statement area:

```
xxx 8 ******************* 9
```

- `xxx` any three-character string instead of the password
- `8` operation code for the close statement
- `9` end identifier
4.4 Retrieval using search

Inquiry using the search statement offers the following functions:

**Projection:** Selection of the attributes whose values are to be extracted from the CALL DML table and placed in the response area.

**Selection:** Selection of records conditionally on primary key value, record number or an attribute value.

Response records can also be sorted, and more than one response per statement can be placed in the response area (block mode).

Retrieval using search assumes that an open statement has been used to open a logical file for the CALL DML table to be searched.

**Projection**

The E subquestion used within a search defines which attributes are contained in the response record. The value of the primary key is, by default, projected into the response record.

The projection can also be updated by subquestions for selection purposes.

*Example 1*

Extract the values of the attributes ANAME and APRICE from all records in the CALL DML table COMPANY.

**Statement area:**

| xxx | 6 | 0 | 0 | E | AA8 | AB6 | 000 | 9 |

- xxx: any three-character string instead of the password
- 6: operation code for the search statement
- 0: primary key function: select all records
- 0: strategy: search table sequentially
- E: E subquestion for projecting attribute values
- AA8: symbolic attribute name of attribute ANAME
- AB6: symbolic attribute name of attribute APRICE
- 000: mandatory entry for E subquestion
Using DML statements

Retrieval using search

9  end identifier

Acknowledgment area:
CO  file identifier of the logical file

On successful execution of the statement, the DBH returns status code 00 in the acknowledgment area and writes the first response record in the response area:

```
A00100 PENCIL................ 00059
```

The remaining response records can be brought into the response area by means of the response polling statement.

Statement area:

```
xxx  7 9 9
```

- xxx any three-character string instead of the password
- 7 operation code for the response polling statement
- 9 poll next response record
- 9 end identifier

Acknowledgment area:
CO  file identifier of the logical file

Response area:

```
A00200 RULER................ 00189
```

When all response records have been output, status code 10 is returned in the acknowledgment area.

*Example 2*

Extract just the article names (AA8), without the primary key value, from the CALL DML table COMPANY.
Retrieval using search

Statement area:

```
xxx 6 0 0 EAA8000 &PSN000 9
```

- `xxx` any three-character string instead of the password
- `6` operation code for the search statement
- `0` primary key function: select all records
- `0` strategy: search table sequentially
- `EAA8000` projection of article names (symbolic attribute name AA8)
- `&PSN000` do not output primary key value in response record
- `9` end identifier

Acknowledgment area:

- `CO` file identifier of the logical file

On successful execution of the statement, the DBH returns status code 00 in the acknowledgment area and the first response record in the response area:

```
PENCIL............
```

Selection on conditions applied to primary key value

The search allows selection of records whose primary key value fulfils a particular condition. The condition is defined by means of the primary key function and one or two comparison values in the inquiry area.

Example 1

Select all records from the CALL DML table COMPANY whose primary key value falls between A00200 and A00500. Just the primary key value is to be output in the response record.

Statement area:

```
xxx 6 5 0 9
```

- `xxx` any three-character string instead of the password
- `6` operation code for the search statement
primary key function:
only those records whose primary key falls between the comparison values
in the inquiry area are selected (including limit values).

strategy:
sequential search of the table.

end identifier

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:

A00200 A00500

On successful execution of the statement, the DBH returns status code 00 in the
acknowledgment area. The primary key value of the first response record is output in the
response area:

A00200

All further responses can be retrieved by means of the response polling statement:

Statement area:

xxx 7 9 9

xxx any three-character string instead of the password
7 operation code for the response polling statement
9 poll next response record
9 end identifier

Acknowledgment area:
CO file identifier of the logical file
Retrieval using search

Response area:

A00300

Further responses, which can also be output by means of the above response polling statement, are:

A00340
A00400

Example 2

Select all records whose primary key value begins with ‘A’ from the CALL DML table COMPANY. This type of selection is called selection on primary key group value. Just the primary key value is placed in the response record.

Statement area:

xxx 6 1 0 9

xxx any three-character string instead of the password

6 operation code for the search statement

1 primary key function:
only those records whose primary key contains primary key group value ‘A’ left-justified are selected. The primary key group value must be placed in the inquiry area as the comparison value.

0 strategy:
sequential search of the table.

9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

A_____
A primary key group value for all primary key values beginning with ‘A’. The primary key group value must be right-filled with blanks to the full length of the primary key.

The first response placed in the response area by the DBH is:

A00100

Subsequent responses, which can be retrieved by means of the response polling statement (see section “Response polling” on page 117), are:

A00200
A00300
A00340
A00400
.
.
A09050

**Selection by conditions applied to the record number**

The search allows a record with a particular record number to be selected.

*Example*

Select the record with binary record number 0000000A.

**Statement area:**

```
xxx  6 8 0 9
```

- **xxx** any three-character string instead of the password
- **6** operation code for the search statement
- **8** primary key function: the record with the record number specified in the inquiry area is selected.
- **0** strategy: the table is searched sequentially.
- **9** end identifier
Acknowledgment area:
CO file identifier of the logical file

Inquiry area:
X’0000000A’ record number in binary form

The DBH returns the primary key value of record number 0000000A in the response area:
A09000

Selection by conditions applied to attribute values

The search allows records to be selected in which an attribute value satisfies one or more conditions. It is also possible to formulate conditions for the values of different attributes. The conditions can be logically ANDed and/or ORed.

Selection by a single condition

Example 1

Output from the CALL DML table COMPANY all articles with stock less than 1000. The article names (AA8) are to be projected into the response record.

Statement area:

| xxx | 6 | 1 | 1 | EAA8000 | U | AC4 | 502 | 9 |

- xxx any three-character string instead of the password
- 6 operation code for the search statement
- 1 primary key function:
  all records whose primary key value contains primary key group value ‘A’
  left-justified are selected. This selects all records for ARTICLE.
- 1 strategy:
  the DBH decides whether the table is searched sequentially or on the index.
- EAA8000 E subquestion for projecting the article names (AA8)
Using DML statements

Retrieval using search

U subquestion:
a condition is applied to the primary key value, and is ANDed with the primary key function.

AC4 symbolic attribute name of the attribute to whose value a condition is applied.

502 search condition and comparison condition:
the search selects all records whose attribute value for AC4 is less than the comparison value in the inquiry area.

9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

A______ 1000

A______ comparison value for the primary key function:
primary key group value identifying the records for ARTICLE.

1000 comparison value for the comparison condition in the U subquestion.

The comparison values must appear in the inquiry area in the same sequence as the respective attributes are referenced in the statement.

The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed and a match has been found. The first response record is then output in the response area:

A00200 RULER

The remaining responses, which can be obtained by response polling, are as follows:

| A00650 | FOLDER       |
| A01400 | ERASER       |
| A01750 | GLUE         |
| A08880 | CALENDER     |
| A09000 | DRAWING_PAD  |
| A09050 | NOTEPAD      |
Example 2

In addition to the specifications in example 1, the values of the attribute STOCK (AC4) are to be projected.

Statement area:

| xxx | 6 | 1 | EAA8000 | C | AC4 | 502 | 9 |

- xxx: any three-character string instead of the password
- 6: operation code for the search statement
- 1: primary key function:
  - all records whose primary key value contains primary key group value ‘A’ left-justified are selected. This selects all records for ARTICLE.
- 1: strategy:
  - the DBH decides whether the table is searched sequentially or on the index.
- EAA8000: E subquestion for projecting the article names (AA8)
- C: C subquestion:
  - a condition is applied to the primary key value, and is ANDed with the primary key function. The values of the referenced attribute are also projected.
- AC4: symbolic attribute name of the attribute to whose value a condition is applied.
- 502: search condition and comparison condition:
  - the search selects all records whose attribute value for AC4 is less than the comparison value in the inquiry area.
- 9: end identifier

Acknowledgment area:

CO: file identifier of the logical file
Inquiry area:

\[ \text{A} \quad 1000 \]

\[ \text{A} \quad \text{comparison value for the primary key function:} \]
\[ \text{primary key group value identifying the records for ARTICLE.} \]

1000 \quad \text{comparison value for the comparison condition in the C subquestion.}

The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed and a match has been found. The first response record is then output in the response area:

\[ \text{A00200 RULER} \quad 0503 \]

The remaining responses, which can be obtained by response polling, are as follows:

\[ \text{A00650 FOLDER} \quad 0107 \]
\[ \text{A01400 ERASER} \quad 0816 \]
\[ \text{A01750 GLUE} \quad 0035 \]
\[ \text{A08880 CALENDER} \quad 0084 \]
\[ \text{A09000 DRAWING Pad} \quad 0527 \]
\[ \text{A09050 NOTEPAD} \quad 0750 \]

**Selection by significance test**

The search allows selection of all records in which an attribute has a significant or null attribute value.

*Example*

All records are to be selected from the CALL DML table SALES where the quantity (AAB) has the null attribute value '0'. The response records are to contain the values of the attributes quantity (ABB), customer number (AB9) and order date (AC7), in addition to the compound key value.
Retrieval using search

Using DML statements

Statement area:

```
xxx 6 0 1 CABB200 EAB9AC7000 9
```

- `xxx` any three-character string instead of the password
- `6` operation code for the search statement
- `0` primary key function:
  - no selection conditions are applied to the primary key value.
- `1` strategy:
  - the DBH decides whether the table is searched sequentially or on the index.
- `C` C subquestion for selection and projection
- `ABB` symbolic attribute name of the attribute quantity, which is to be tested for non-significance.
- `200` search and comparison condition:
  - select the records where the specified attribute ABB does not have a significant value.
- `EAB9AC7000` E subquestion for projection of the attribute values of AB9 and AC7
- `9` end identifier

Acknowledgment area:

- `SA` file identifier of the logical file

Inquiry area:

- No entry, as neither the primary key function nor the C subquestion requires a comparison value.
- The DBH returns status code 00 in the acknowledgment area after successful execution of the statement. The response area contains the first response record:

```
1011 0000 C23979 880930
```

- The other responses can be output by means of the response polling statement (see section “Response polling” on page 117). They are as follows:

```
1012 0000 C01732 880930
1013 0000 C38210 881003
1014 0000 C20070 881004
```
Selection by applying several conditions to an attribute

The search allows records to be selected in which an attribute value fulfills one of a number of conditions. The individual conditions are ORed.

Example

All customers are to be selected from the CALL DML table COMPANY who obtain a discount of 15.00% or 20.00%. In addition to the customer number (AAA), the last name (AD2), city (AHT) and discount (AKP) are to be output.

Statement area:

```
xxx 6 1 1 000 EAD2AHT000 000 CAKP50101 9
```

xxx any three-character string instead of the password

6 operation code for the search statement

1 primary key function:
   all records whose primary key value contains primary key group value ‘C’ left-justified are selected. This selects all records for CUSTOMER.

1 strategy:
   the DBH decides whether the table is searched sequentially or on the index.

EAD2AHT000
   E subquestion for projecting the last names (AD2) and cities (AHT)

C
   C subquestion for selection and projection

AKP
   symbolic attribute name of the attribute discount

50101
   search condition (5) and comparison conditions (01):
   the value of attribute AKP must be equal to either the first OR the second comparison value in the inquiry area for the record to be selected.

9 end identifier

Acknowledgment area:

CO file identifier of the logical file
Retrieval using search

Inquiry area:

<table>
<thead>
<tr>
<th>C</th>
<th>1500</th>
<th>2000</th>
</tr>
</thead>
</table>

- **C** comparison value for the primary key function:
  - primary key group value identifying the records for CUSTOMER.

- **1500** first comparison value for the attribute AKP of the C subquestion (15.00 %)
- **2000** second comparison value for the attribute AKP of the C subquestion (20.00 %)

The DBH returns status code 00 in the acknowledgment area after successful execution of the statement. The response area contains the first response record:

<table>
<thead>
<tr>
<th>C01732</th>
<th>HUBER</th>
<th>FRANKFURT</th>
<th>2000</th>
</tr>
</thead>
</table>

The remaining response records, which can be output by response polling (see section “Response polling” on page 117) are as follows:

<table>
<thead>
<tr>
<th>C17109</th>
<th>KELLER</th>
<th>MUENCHEN</th>
<th>1500</th>
</tr>
</thead>
<tbody>
<tr>
<td>C38210</td>
<td>HUBER</td>
<td>OFFENBACH</td>
<td>2000</td>
</tr>
</tbody>
</table>

The customer number is the primary key value and is output automatically.

**Selection by setting the same condition for several attributes**

The search allows records to be selected where at least one of several attributes contains or does not contain a particular attribute value. The attributes must have identical attribute definitions in the attribute catalog. Several occurrences of a multiple attribute can also be tested for the same condition, as an alternative to testing several attributes. The occurrences will always have the same attribute definition.

**Example**

Those staff are to be selected from the CALL DML table COMPANY who speak Italian as their first, second or third foreign language. The personnel number (AAA), last name (ALM), first name (AMK), department (AR9) and the first three foreign languages (AS7) are to be output for each employee.
Using DML statements

Retrieval using search

Statement area:

|   |   |   | xxx | 6 | 1 | 1 | EALMAMKAR9000 | CAS7/001/AS7/002/AS7/003/501 | 9 |

xxx any three-character string instead of the password
6 operation code for the search statement
1 primary key function:
all records whose primary key value contains primary key group value ‘P’
left-justified are selected. This selects all records for PERSONNEL.
1 strategy:
the DBH decides whether the table is searched sequentially or on the index.

EALMAMKAR9000
E subquestion for projecting the last name (ALM), first name (AMK) and
department (AR9)
C C subquestion for selection and projection
AS7/001/ first occurrence of the multiple attribute foreign language
AS7/002/ second occurrence of the multiple attribute foreign language
AS7/003/ third occurrence of the multiple attribute foreign language
501 search and comparison condition:
the preceding three occurrences are tested for equality with the comparison
value in the inquiry area. A record is selected as soon as one of the three
occurrences satisfies the condition.
9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

P̂̂̂̂̂̂ ITAL̂̂̂

P̂̂̂̂ comparison value for the primary key function:
primary key group value identifying the records for PERSONNEL.
ITAL̂̂̂ comparison value for occurrences 1 to 3 of the multiple attribute AS7 in the
C subquestion.
The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed, and places the first response in the response area:

```
P00708  BINDER       THOMAS       DPT4  ENGL.  ITAL.  PORT.
```

The second and last response can be retrieved by response polling (see section “Response polling” on page 117):

```
P19479  SCHUBER      PETER       DPT4  ENGL.  FRI.   ITAL.
```

### Selection by string search

The search allows records to be selected where an attribute value contains or does not contain a particular string (string search). A string search is only possible for attributes defined as alphanumerical (data types CHAR). For a string search, the character string is enclosed in string identifiers, blank-filled to the full attribute length and placed in the inquiry area. The default string identifier is the percentage sign (%). It can be changed to another character by means of the set string identifier function (see section “Define comparison values” on page 92), for example when the character string in question contains a percentage character.

**Example**

All pads are to be retrieved from the CALL DML table COMPANY. The article number (AAA) and article name (AA8) are to be output for all articles containing the string PAD.

**Statement area:**

```
xxx  6  1  1 CAA8401  9
```

- **xxx**: any three-character string instead of the password
- **6**: operation code for the search statement
- **1**: primary key function:
  - all records whose primary key value contains primary key group value ‘A’ left-justified are selected. This selects all records for ARTICLE.
- **1**: strategy:
  - the DBH decides whether the table is searched sequentially or on the index.
- **C**: C subquestion for selection and projection
- **AA8**: symbolic attribute name of the attribute article name
- **4**: search condition for the string search
Using DML statements

Retrieval using search

01       comparison condition for equality
9        end identifier

Acknowledgment area:
CO       file identifier of the logical file

Inquiry area:

A........ %PAD%...........

A......       comparison value for the primary key function:
primary key group value identifying the records for ARTICLE.

%PAD%........
comparison value for the attribute AA8:
the character string PAD is enclosed in string identifiers (%) and blank-filled
to a length of 15 as defined in the attribute catalog.

The DBH returns status code 00 in the acknowledgment area if the statement is success-
fully executed, and places the first response in the response area:

A00300  WRITING_PAD.....

The remaining responses can be retrieved by response polling (see section “Response
polling” on page 117):

A00300  DRAWING_PAD.....
A09050  NOTEPAD........

Selection by search with masked comparison values

The search allows records to be selected where an attribute value does or does not contain
a particular character in a particular position. The mask character must be substituted for
the non-relevant characters in the comparison value. The default mask character is the
question mark (?). If one of the search characters is a question mark, the default mask
character can be changed to another character by means of the set mask character function
(see section “Define comparison values” on page 92).
The search with masked comparison values is only permitted for attributes of the data type
CHAR.
Example

The CALL DML table COMPANY is searched for all employees whose department name begins with the number 1. The last name (ALM), first name (AMK) and the department (AR9) are to be output in addition to the personnel number (AAA).

Statement area:

```
   xxx  6 1 1 EALMAMK000  CAR9401  9
```

xxx any three-character string instead of the password
6 operation code for the search statement
1 primary key function:
   all records whose primary key value contains primary key group value ‘P’
   left-justified are selected. This selects all records for PERSONNEL.
1 strategy:
   the DBH decides whether the table is searched sequentially or on the index.

EALMAMK000
   E subquestion for projecting the last name (ALM) and first name (AMK)
C C subquestion for selection and projection
AR9 symbolic attribute name of the attribute department
4 search condition for the search with masked comparison value
01 comparison condition for equality
9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

```
P??1
```

P??1 comparison value for the primary key function:
   primary key group value identifying the records in PERSONNEL.
??1 comparison value for attribute AR9
The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed, and places the first response in the response area:

| P00333 | HUBER_ | ANDREAS_ | DPT1 |

Subsequent response records are as follows:

| P01140 | STOLL_ | WERNER_ | DPT1 |
| P05583 | BOTT_  | MICHAEL_ | BRO1 |
| P09980 | RICHTER_ | STEFAN_ | BRO1 |
| P11444 | METZGER_ | KLAUS_ | BRO1 |
| P12921 | RUF_ | MICHAEL_ | DPT1 |
| P13345 | BERGER_ | FRED_ | DPT1 |
| P15863 | BARTEL_ | ELVIRA_ | DPT1 |
| P21500 | WEISS_ | HANS_ | BRO1 |

Selection by boundary value conditions

The R subquestion of the search allows so-called boundary value conditions to be formulated for selection. A boundary value condition may be:

- the smallest attribute value
- the largest attribute value
- the next smallest attribute value to the comparison value in the inquiry area
- the next largest attribute value to the comparison value in the inquiry area

Boundary value conditions can be applied to a maximum of 6 attributes in one search. Each R subquestion limits the set of responses to the previous R subquestion. Once an R subquestion returns just one response, the response is placed directly in the response area and further R subquestions are ignored.

Example

The SALES table is searched to find the date (AC7) on which the oldest orders that have not yet been processed were entered. The order with the largest customer number (AB9) is to be selected from these orders.
Retrieval using search

Using DML statements

Statement area:

| xxx | 6 | 0 | 1 | RAC7731 | RAB9739 | 9 |

- **xxx**: any three-character string instead of the password
- **6**: operation code for the search statement
- **0**: primary key function: select all records
- **1**: strategy: the DBH decides whether the table is searched sequentially or on the index.
- **R**: R subquestion to formulate the first boundary value condition
- **AC7**: symbolic attribute name of the attribute date
- **731**: search and comparison condition: “smallest attribute value”
- **R**: R subquestion to formulate the second boundary value condition
- **AB9**: symbolic attribute name of the attribute customer number
- **739**: search and comparison condition: “largest attribute value”
- **9**: end identifier

Acknowledgment area:

- **SA**: file identifier of the logical file

Inquiry area:

No entry as neither the primary key function nor the R subquestions require comparison values.

The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed, and places the first response in the response area:

| 1011 | 880930 | C23979 |

**Selection on complex conditions**

The search subquestions enable complex selection conditions to be formulated. The individual subquestions are connected by logical AND or OR depending on the subquestion type.
Example

All pads are to be selected from the CALL DML table COMPANY where the stock is still more than 700 items. Additionally, all articles are to be retrieved that cost less than $1. The article number (AAA), article name (AA8), price (AB6) and stock (AC4) are to be output for the articles found by the search.

Notation for the selection conditions:

\[(\text{article}=\%\text{PAD}\%) \text{ AND } (\text{stock}>700)\] OR \[(\text{price}<1,00)\]

This expression must be “multiplied out” in order to be converted to subquestions, as in SESAM/SQL logic, logical OR links more strongly than logical AND, and no parentheses are possible. To “multiply out”, each element in the first parenthesis is combined with the element in the second parenthesis. The logical operator is the operator between the two parentheses (OR).

After “multiplying out”, the following expression is obtained:

\[(\text{article}=\%\text{PAD}\%) \text{ OR } (\text{price}<1,00)\] AND \[(\text{stock}>700) \text{ OR } (\text{price}<1,00)\]

The parentheses are now superfluous in SESAM/SQL logic. The resultant structure of logical relationships is:

\[(\text{primary key}=\text{A})\]
AND (\text{article}=\%\text{PAD}\%)
OR (\text{price}<1,00)
AND (\text{stock}>700)
OR (\text{price}<1,00)

The U and C subquestions can be used to represent the AND operator, the O and L subquestions the OR operator. The C and L subquestions enclose the projection of the attribute value.

Statement area:

| xxx | 6 | 1 | 1 CAA8401 | LAB6502 | CAC4504 | OAB6502 | 9 |

xxx any three-character string instead of the password

6 operation code for the search statement

1 primary key function:
all records whose primary key value contains primary key group value ‘A’ left-justified are selected. This selects all records for ARTICLE.

1 strategy:
the DBH decides whether the table is searched sequentially or on the index.

C C subquestion for selection and projection; AND operator
Retrieval using search

Using DML statements

AA8  symbolic attribute name of the attribute article name
401  string search: “equal to comparison value in inquiry area”
L    L subquestion for selection and projection; OR operator
AB6  symbolic attribute name of the attribute price
502  search and comparison condition: “less than comparison value in inquiry area”
C    C subquestion for selection and projection; AND operator
AC4  symbolic attribute name of the attribute stock
504  search and comparison condition: “greater than comparison value in inquiry area”
O    O subquestion for selection; OR operator
AB6  symbolic attribute name of the attribute price
502  search and comparison condition: “less than comparison value in inquiry area”
9    end identifier

Acknowledgment area:

CO    file identifier of the logical file

Inquiry area:

A       %PAD%            00100 0700 00100

A       comparison value for the primary key function:
        primary key group value identifying the records for ARTICLE.

%PAD%    comparison value for subquestion CAA8401
00100    comparison value for subquestion LAB6502
0700     comparison value for subquestion CAC4504
00100    comparison value for subquestion OAB6502
Using DML statements

Retrieval using search

The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed, and places the first response in the response area:

```
A00100  PENCIL            00059  7576
```

The remaining responses are as follows:

```
A00300  WRITING_PAD       00179  1763
A09050  NOTEPAD           00099  0750
```

Selection by conditions applied to index values

The T subquestion of the search allows records to be output sorted on the values of a index attribute. A selection condition can also be applied to the index attribute.

**Example**

All employees who are not employed in department DPT4 are to be selected from the CALL DML table COMPANY. Employees are to be output sorted by department.

Statement area:

```
xxx  6  1  1  TAR9601  9
```

xxx any three-character string instead of the password

6 operation code for the search statement

1 primary key function:

all records whose primary key value contains primary key group value ‘P’ left-justified are selected. This selects all records for PERSONNEL.

1 strategy:

the DBH decides whether the table is searched sequentially or on the index.

T T subquestion for selection and projection

AR9 symbolic attribute name of the attribute department

601 search and comparison condition:

“not equal to comparison value in inquiry area”

9 end identifier
Retrieval using search

Using DML statements

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

P comparison value for the primary key function:
primary key group value identifying the records for PERSONNEL.

DPT4 Comparison value for the T subquestion

The DBH returns status code 00 in the acknowledgment area if the statement is successfully executed, and places the first response in the response area:

P00333 DPT1

Subsequent responses are as follows:

P01140 DPT1
P12921 DPT1
P13345 DPT1
P15863 DPT1
P03674 DPT2
P05408 DPT2
P19478 DPT2
P01000 DPT3
P20099 DPT3
P05583 BRO1
P09980 BRO1
P11444 BRO1
P21500 BRO1
Retrieval in block mode

The search offers the facility to process response records in blocks, in other words, each statement results in several responses being placed in the response area. The number of responses is defined in the statement, and also applies to any subsequent response polling.

Example

The CALL DML table COMPANY is to be processed and for all customers the customer number (AAA), last name (AD2), zip code (AGV) and city (AHT) output. Each response is to contain 5 response records.

Statement area:

```
xxx  6  1  1  EAD2AGVAHT000  &BLN005  9
```

- `xxx`: any three-character string instead of the password
- `6`: operation code for the search statement
- `1`: primary key function:
  - all records whose primary key value contains primary key group value ‘C’ left-justified are selected. This selects all records for CUSTOMER.
- `1`: strategy:
  - the DBH decides whether the table is searched sequentially or on the index.
- `EAD2AGVAHT000`: E subquestion for projecting the last name, zip code and city
- `&BLN005`: block mode:
  - Up to 5 response records are returned, without record numbers, in the response area.
- `9`: end identifier

Acknowledgment area:

```
CO
```

File identifier of the logical file

Inquiry area:

```
C
```


Retrieval using search

Using DML statements

Comparison value for the primary key function:
primary key group value identifying the records for CUSTOMER.

The DBH places the first 5 response records in the response area and returns status code 00 in the acknowledgment area.

<table>
<thead>
<tr>
<th>C01732</th>
<th>HUBER</th>
<th>60311</th>
<th>FRANKFURT</th>
</tr>
</thead>
<tbody>
<tr>
<td>C13486</td>
<td>MILLER</td>
<td>80802</td>
<td>MUENCHEN</td>
</tr>
<tr>
<td>C17109</td>
<td>KELLER</td>
<td>80336</td>
<td>MUENCHEN</td>
</tr>
<tr>
<td>C20070</td>
<td>REITER</td>
<td>86150</td>
<td>AUGSBURG</td>
</tr>
<tr>
<td>C23979</td>
<td>STEINER</td>
<td>90403</td>
<td>NUERNBERG</td>
</tr>
</tbody>
</table>

The remaining response records are output by means of the response polling statement (see section “Response polling” on page 117):

Statement area:

```
xxx 7 9 9
```

xxx any three-character string instead of the password

7 code for the response polling statement

7 poll the next block of response records

9 end identifier

Acknowledgment area:

CO file identifier of the logical file

The response polling statement also processes 5 response records, exactly as specified in the search. The search only produced 9 responses in total, so the response polling statement can only output 4 responses. The DBH reports status code 10, to indicate that all the response records have been output.

<table>
<thead>
<tr>
<th>C37424</th>
<th>MAYER</th>
<th>63457</th>
<th>HANAU</th>
</tr>
</thead>
<tbody>
<tr>
<td>C37597</td>
<td>FISCHER</td>
<td>91052</td>
<td>ERLANGEN</td>
</tr>
<tr>
<td>C38210</td>
<td>HUBER</td>
<td>63073</td>
<td>OFFENBACH</td>
</tr>
<tr>
<td>C40013</td>
<td>SCHNEIDER</td>
<td>93059</td>
<td>REGensburg</td>
</tr>
</tbody>
</table>
Flexible formulation of subquestions

The application program can define extensive searches, which can be adapted to suit specific requirements. This is achieved by activating or deactivating search or comparison conditions. The search condition is deactivated by overwriting it by an 8. Comparison conditions with one entry in the inquiry area are replaced by 80, those with two entries in the inquiry area by 82.

Example

The application program defines the following statement, which references PERSONNEL in the CALL DML table COMPANY:

**Statement area:**

```
xxx611 EALM000 CAPF52324 UAPF52324 9
```

**Acknowledgment area:**

CO file identifier of the logical file

**Inquiry area:**

```
PÉËËËË 80000 81999 80000 81999 80000 81999 80000 81999
```

*Case A*

- Selection of all employees who live in Munich (zip code between 80000 and 81999 inclusive)
- Projection of employee name and zip codes

**Statement area:**

```
xxx611 EALM000 CAPF52382 UAPF82324 9
```

**Acknowledgment area:**

CO file identifier of the logical file
Inquiry area:

<table>
<thead>
<tr>
<th>P</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
</tr>
</thead>
</table>

Case B

– Selection of all employees who do not live in Munich (zip code < 80000 or >81999)
– Projection of the zip code

Statement area:

<table>
<thead>
<tr>
<th>xxx611</th>
<th>EALM800</th>
<th>CAPF58224</th>
<th>UAPF82324</th>
<th>9</th>
</tr>
</thead>
</table>

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

<table>
<thead>
<tr>
<th>P</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
<th>80000</th>
<th>81999</th>
</tr>
</thead>
</table>

Case C

– Selection of all employees who do not live in Munich (zip code < 80000 or >81999)
– Projection of employee names

Statement area:

<table>
<thead>
<tr>
<th>xxx611</th>
<th>EALM000</th>
<th>CAPF82324</th>
<th>UAPF58224</th>
<th>9</th>
</tr>
</thead>
</table>

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

| P | 80000 | 81999 | 80000 | 81999 | 80000 | 81999 | 80000 | 81999 |
4.5 Retrieval using search with join

Search with join combines the records from two logical files. The values of the join attribute in one logical file are compared with the values of the join attribute in the other logical file. If they are equal, the two records are combined.

Example

The name, zip code and city are extracted from the CALL DML table COMPANY for those customers with an open order in the CALL DML table SALES. Only customers whose customer number is greater than C10000 are of interest. The following information is to appear in the response record:
- customer number (join attribute), zip code, city, order number and order date.

To perform a search with join, the full length of the two join attributes must be declared as an index, and a logical file must have been opened for both tables. The file identifier of the CALL DML table COMPANY is CO and of the SALES table SA.

Statement area:

```
xxx  6  0  1  #CO  EAD2AGVAHT000  V(  AAA  #CO  =  AB9  #SA  504  ) . . .
```

```
. . .  xxx  6  0  1  #SA  EAABAC7000  &BLN010  &PSN000  9
```

xxx       any three-character string instead of the password
6       operation code for the search statement
0       primary key function: select all records
1       strategy:
          the DBH decides whether the table is searched sequentially or on the index.

#CO  CO is the file identifier of the logical file to which the primary key function and the subsequent E subquestion refer.

EAD2AGVAHT000  E subquestion for projecting name, zip code and city from the CALL DML table CUSTOMER

V(  mandatory entry

AAA  symbolic attribute name of the join attribute of logical file CO

#CO  CO is the file identifier of the logical file opened for the CALL DML table COMPANY. It identifies the logical file containing the join attribute AAA.
Retrieval using search with join

Using DML statements

= mandatory entry
AB9 symbolic attribute name of the join attribute of logical file SA
#SA SA is the file identifier of the logical file opened for the CALL DML table SALES. It identifies the logical file containing the join attribute AB9.
504 search and comparison condition for the join attribute: “greater than the comparison value in the inquiry area”
) mandatory entry
xxx any three-character string instead of the password
6 operation code for the search statement
0 primary key function: select all records
1 strategy:
the DBH decides whether the table is searched sequentially or on the index.
#SA SA is the file identifier of the logical file to which the primary key function and the subsequent E subquestion refer.
EAABAC7000 E subquestion for projecting the order number and order date of ORDER
&BLN010 block mode:
10 response records without record numbers are output in the response area.
&PSN000 the response records are output without primary key values.
9 end identifier

Acknowledgment area:
CO file identifier of one of the two logical files

Inquiry area:

C10000 comparison value for the join attribute
The DBH returns status code 10 in the acknowledgment area to confirm that the statement has been successfully executed and that all responses have been output in the response area. The responses are:

<table>
<thead>
<tr>
<th>Projection from the SALES table</th>
<th>JOIN attribute</th>
<th>Projection from the COMPANY table</th>
</tr>
</thead>
<tbody>
<tr>
<td>REITER....................</td>
<td>86150 AUGSBURG....</td>
<td>C20070 1014 881004</td>
</tr>
<tr>
<td>STEINER....................</td>
<td>90403 NUERNBERG....</td>
<td>C23979 1011 880930</td>
</tr>
<tr>
<td>HUBER.......................</td>
<td>63073 OFFENBACH....</td>
<td>C38210 1013 881003</td>
</tr>
</tbody>
</table>

Table 82: Responses to the search with join
4.6 Cursor technique for search/search with join

Search (see section “Search” on page 37) and search with join (see section “Search with join” on page 63) statements enable a record number cursor file to be created. This cursor file contains just the record numbers of the response records. It can be restricted by a further search/search with join. The responses can be output by means of response polling statements (see section “Response polling” on page 117).

Creating and processing a cursor file

The search enables a cursor file to be created containing just the record numbers of the response records (record number cursor file). This cursor file can be processed as follows:

– Output (updated) responses using response polling statements
– Output the first (updated) response with a search and subsequent responses by response polling
– Restrict the cursor file by a search. This search only processes those records whose record numbers appear in the cursor file.

Example

Step 1

Select all customers from the CALL DML table COMPANY who were customers prior to 1.1.88. The record numbers of the relevant customer records are to be stored in a cursor file. The response records are to contain the precise date (AJR) in addition to the customer number (AAA).

Statement area:

| xxx | 6 | 1 | Z | CAJR502 | 9 |

- xxx any three-character string instead of the password
- 6 operation code for the search statement
- 1 primary key function: all records whose primary key value contains primary key group value ‘C’ left-justified are selected. This selects all records for CUSTOMER.
- Z strategy: the response records are counted, the number output in the acknowledgment area and the record numbers stored in a cursor file.
Using DML statements

Cursor technique for search/search with join

C  C subquestion for selection and projection
AJR symbolic attribute name of the attribute ‘customer since’
502 search and comparison condition: the attribute values of attribute AJR are checked to see if they are smaller than the comparison value in the inquiry area.
9 end identifier

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:

\[\text{C_\ldots 880101}\]

\[\text{C_\ldots comparison value for the primary key function: primary key group value identifying the records in CUSTOMER.}\]

\[\text{880101 comparison value for the C subquestion}\]

After successful execution of the search, the DBH returns status code 10 in the acknowledgment area together with 7, which is the number of responses. No information is output in the response area.

*Step 2*

The customer records are output in the response area.

Statement area:

\[\text{xxx 7 2 9}\]

\[\text{xxx any three-character string instead of the password}\]

\[\text{7 operation code for the response polling statement}\]

\[\text{2 output the first response}\]

\[\text{9 end identifier}\]
Cursor technique for search/search with join

Acknowledgment area:
CO file identifier identifying the cursor file

Inquiry area:
C_____ 880101

The content of the inquiry area remains unchanged, as no new conditions are applied to the primary key value.

The DBH outputs the first response record in the response area:
C01732 860114

The remaining responses must be output individually using the following response polling statement:

Statement area:
xxx 7 0 9

xxx any three-character string instead of the password
7 operation code for the response polling statement
0 output the next response
9 end identifier

Acknowledgment area:
CO file identifier identifying the cursor file

Inquiry area:
C_____ 880101

The content of the inquiry area remains unchanged.

The DBH reports status code 00 and places the response record in the response area:
C13486 860325
The remaining response records are polled in the same way until the DBH returns status code 10. The response records are as follows:

<table>
<thead>
<tr>
<th>Customer Number</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17109</td>
<td>860520</td>
</tr>
<tr>
<td>C20070</td>
<td>861217</td>
</tr>
<tr>
<td>C23979</td>
<td>870420</td>
</tr>
<tr>
<td>C37424</td>
<td>870901</td>
</tr>
<tr>
<td>C37597</td>
<td>871118</td>
</tr>
</tbody>
</table>

**Step 3**

From the customer records indicated by the record numbers in the cursor file, select those with a customer number beginning with ‘C1’ and who live in Munich (zip code starts with the ‘80’ or ‘81’). The record numbers of these customer records are to be stored in the cursor file. The information output for these customers is: customer number (AAA), zip code (AGV) and date (AJR)

**Statement area:**

| xxx | 6 | P | CAGV423 | EARJ000 | 9 |

- **xxx**: any three-character string instead of the password
- **6**: operation code for the search statement
- **1**: primary key function:
  - all records whose primary key value contains primary key group value ‘C1’ left-justified are selected.
- **P**: strategy:
  - only records whose record number appears in the cursor file are processed. The old contents of the cursor file are deleted and the new record numbers stored.
- **C**: C subquestion for selection and projection
- **AGV**: symbolic attribute name of the attribute zip code
- **423**: search and comparison condition for masked search:
  - “greater than or equal to first comparison value and less than or equal to second comparison value in inquiry area”
- **E**: E subquestion for projection
- **AJR**: symbolic attribute name of the attribute ‘customer since’
Cursor technique for search/search with join

Using DML statements

000 mandatory entry
9 end identifier

Acknowledgment area:
CO file identifier identifying the cursor file

Inquiry area:

\[
\begin{array}{c|c|c}
C1_{...} & 80??? & 81??? \\
\end{array}
\]

C1_{...} comparison value for the primary key function
80?? first masked comparison value for the C subquestion
81?? second masked comparison value for the C subquestion

After successful execution of the search, the DBH returns status code 10 in the inquiry area together with 2, which is the number of responses. No information is output in the response area.
Both responses can be output into the response area using response polling statements xxx729 and xxx709 (see “Step 2” on page 223). They are:

\[
\begin{array}{c|c|c}
C13486 & 80802 & 860325 \\
C17109 & 80336 & 860520 \\
\end{array}
\]

Sort on attribute values

The S subquestion of the search enables records to be sorted on the value of any attribute. The S subquestion can only be used in searches that create a cursor file.

Example

All employee records are to be extracted from the CALL DML table COMPANY and output sorted alphabetically on first name within last name.

Statement area:

\[
\begin{array}{c|c|c|c|c}
xxx & 6 & 1 & Z & SALMAMK734 \\
\end{array}
\]

xxx any three-character string instead of the password
6 operation code for the search statement
1 primary key function:
all records whose primary key value contains primary key group value ‘P’
left-justified are selected. This selects all records for PERSONNEL.

Z strategy:
the response records are counted, the number output in the
acknowledgment area and the record numbers stored in a cursor file.

S S subquestion for sorting and projection
ALM symbolic attribute name of the attribute last name
AMK symbolic attribute name of the attribute first name
734 search and comparison condition: “ascending sort”
9 end identifier

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:

P...... comparison value for the primary key function

In the application program, the statement must first be passed in this form to the connection
module SESMOD. On successful completion of the statement the DBH returns status code
1S and the number of response records in the acknowledgment area. The same statement
must then be passed to the SORT module SESORT, which sorts the record numbers in the
cursor file on the criteria specified in the search statement.
The response records can be output from the cursor file, e.g. by means of response polling
statements. The sorted response records are as follows:
Cursor technique for search/search with join

Using DML statements

Creating and processing a join cursor file

The search with join (see section “Search with join” on page 63) enables a so-called join cursor file to be created containing in each response record the two record numbers of the records combined by the search.

The join cursor file can be processed in the following ways:

– Output (updated) responses using response polling statements (see section “Response polling” on page 117)

– Output the first (updated) response with a further search with join and output subsequent responses by response polling

– Restrict the join cursor file by a search with join. This search only processes those records flagged by record number pairs in the join cursor file. The responses to the new search are written as record number pairs either to a new cursor file, or overwrite the old one.

Example

The CALL DML table COMPANY is to be searched for customers who have an open order in the CALL DML table SALES, and the name, zip code and city extracted. The order number and order date is to be output for each customer.

<table>
<thead>
<tr>
<th>Record Number</th>
<th>Name</th>
<th>Zip Code</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>P15863</td>
<td>BARTELB</td>
<td>ELVIRA</td>
<td></td>
</tr>
<tr>
<td>P15863</td>
<td>BAUER</td>
<td>RUDOLF</td>
<td></td>
</tr>
<tr>
<td>P11500</td>
<td>KAISER</td>
<td>FRANZ</td>
<td></td>
</tr>
<tr>
<td>P20099</td>
<td>KAISER</td>
<td>SUSANNE</td>
<td></td>
</tr>
<tr>
<td>P03674</td>
<td>WINKLER</td>
<td>ANDREA</td>
<td></td>
</tr>
</tbody>
</table>
Step 1

The join attribute is the customer number (AAA in the CALL DML table COMPANY and AB9 in the SALES table). Two logical files, C1 and C2, are opened for the CALL DML table COMPANY, and one logical file, SA, for the SALES table.

The record numbers of the response records are to be stored in a join cursor file with file identifier C1.

Statement area:

| xxx | 6 | 0 | Z | #C1 | EAD2AGVAHT000 | V( | AAA | #C1 = | AB9 | #SA | 504 | ) |
|-----|---|---|---|-----|-------------|----|-----|------|----|-----|-----|

```
EAD2AGVAHT000
```

any three-character string instead of the password

operation code for the search statement

primary key function: select all records

strategy:

the response records are counted, the number output in the acknowledgment area and the record numbers stored in a cursor file.

```
EAD2AGVAHT000
```

C1 is the file identifier of the logical file to which the primary key function and the subsequent E subquestion refer.

```
EAD2AGVAHT000
```

E subquestion for projecting name, zip code and city from CUSTOMER

```
EAD2AGVAHT000
```

mandatory entry

symbolic attribute name of the join attribute of logical file C1

```
EAD2AGVAHT000
```

C1 is the file identifier of the logical file opened for the CALL DML table COMPANY. It identifies the logical file that contains the join attribute AAA.

```
EAD2AGVAHT000
```

mandatory entry

symbolic attribute name of the join attribute of logical file SA

```
EAD2AGVAHT000
```

SA is the file identifier of the logical file opened for the SALES table. It identifies the logical file containing the join attribute AB9.

```
EAD2AGVAHT000
```

search and comparison condition for the join attribute: “greater than the comparison value in the inquiry area”

```
EAD2AGVAHT000
```

mandatory entry
Cursor technique for search/search with join

Using DML statements

XXX any three-character string instead of the password

6 operation code for the search statement

0 primary key function: select all records

Z strategy:
the response records are counted, the number output in the
acknowledgment area and the record numbers stored in a cursor file.

#SA SA is the file identifier of the logical file to which the primary key function and
the subsequent E subquestion refer.

EAABAC7000 E subquestion for projecting the order number and order date from ORDER

&PSN000 the response records are output without primary key values.

9 end identifier

Acknowledgment area:

C1 file identifier of the logical file for which the cursor file is created

Inquiry area:

C……… comparison value for the primary key function

After successful execution of the search with join, the DBH returns status code 10 in the
inquiry area together with 4, which is the number of responses. No information is output in
the response area.

Step 2

This join cursor file is to be further restricted: for those customers whose customer
number starts with ‘C2’, the name (AD2 in the table COMPANY) and the order number
(AAB in the SALES table) are to be output in addition to the customer number (the join
attribute). The record numbers of the response records are to be stored in a join cursor
file with file identifier C2. The join cursor file with file identifier C1 is to be retained.
Cursor technique for search/search with join

Using DML statements

Statement area:

```
xxx 6 1 P $C1 #C1 EAAAAD2000 V xxx 6 0 P #SA EAAB000 &PSN000 9
```

xxx any three-character string instead of the password

6 operation code for the search statement

1 primary key function:
all records flagged in join cursor file C1 whose primary key value contains
primary key group value ‘C2’ left-justified are selected.

P strategy:
only those records are processed whose record numbers are in join cursor
file C1.

$C1 C1 is the file identifier of the join cursor file to be restricted and retained.

#C1 C1 is the file identifier of the logical file to which the primary key function and
the following E subquestion refer.

EAAAAD2000
E subquestion for projecting the customer number and date from
CUSTOMER

V mandatory entry

xxx any three-character string instead of the password

6 operation code for the search statement

0 primary key function:
all COMPANY records flagged in join cursor file C1 are selected.

P strategy:
only those records are processed whose record numbers are in join cursor
file C1.

#SA SA is the file identifier of the logical file to which the primary key function and
the following E subquestion refer

EAAB000 E subquestion for projecting the order number from ORDER

&PSN000 the response records are output without primary key values

9 end identifier

Acknowledgment area:

C2 file identifier of the logical file for which the new join cursor file is to be
created. The old join cursor file C1 is retained.
Inquiry area:

C2

Comparison value for the primary key function referring to logical file C1.

After successful execution of the statement, the DBH returns status code 10 in the inquiry area together with 2, which is the number of responses whose record numbers are in the acknowledgment area.

**Step 3**

The response polling statement `xxx729` allows the first response to be polled from a join cursor file (see “Step 2” on page 223). The file identifier of the join cursor file must be placed in the acknowledgment area.

The remaining responses are polled using statement `xxx709`. Again, the file identifier of the join cursor file must be placed in the acknowledgment area.

The responses from join cursor file C1 are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Code</th>
<th>Number</th>
<th>Acknowledgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUBER</td>
<td>60311</td>
<td>FRANKFURT</td>
<td>C01732</td>
<td>1012</td>
</tr>
<tr>
<td>REITER</td>
<td>86150</td>
<td>AUGSBURG</td>
<td>C20070</td>
<td>1014</td>
</tr>
<tr>
<td>STEINER</td>
<td>90403</td>
<td>NUERNBERG</td>
<td>C23979</td>
<td>1011</td>
</tr>
<tr>
<td>HUBER</td>
<td>63073</td>
<td>OFFENBACH</td>
<td>C38210</td>
<td>1013</td>
</tr>
</tbody>
</table>

The responses from join cursor file C2 are as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Number</th>
<th>Acknowledgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C20070</td>
<td>REITER</td>
<td>1014</td>
<td></td>
</tr>
<tr>
<td>C23979</td>
<td>STEINER</td>
<td>1011</td>
<td></td>
</tr>
</tbody>
</table>
4.7 Inquiring on attribute value frequency (index browsing)

The index browsing statement enables all the values of an attribute to be listed. The frequency of each attribute value is also output. To use index browsing, the full or partial length of the attribute in question must have been declared as an index.

Example 1

Obtain from the CALL DML table SALES the number of orders for a particular article number (AAC).

Statement area:

<table>
<thead>
<tr>
<th>xxx</th>
<th>60BI</th>
<th>AAC</th>
<th>000</th>
<th>9</th>
</tr>
</thead>
</table>

xxx any three-character string instead of the password
60BI operation code for the index browsing statement
AAC symbolic attribute name of the attribute article number
000 search and comparison condition: “no conditions for the attribute value”
9 end identifier

Acknowledgment area:

SA file identifier of the logical file

Inquiry area:

No entry, as condition 000 requires no comparison value.

The DBH returns status code 00 in the acknowledgment area when the statement has been successfully executed. The response area contains the first pair of values, consisting of frequency and attribute value. The responses are sorted in ascending order of index values.

<table>
<thead>
<tr>
<th>00000004</th>
<th>~~~~~~~~~~</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000004</td>
<td>frequency in binary format</td>
</tr>
<tr>
<td>~~~~~~~~~~</td>
<td>attribute value for the article number</td>
</tr>
</tbody>
</table>
Inquiring on attribute value frequency (index browsing) Using DML statements

The remaining responses can be output by means of the response polling statement (see section “Response polling” on page 117). They are as follows:

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000001</td>
<td>A00100</td>
</tr>
<tr>
<td>00000001</td>
<td>A00200</td>
</tr>
<tr>
<td>00000001</td>
<td>A00400</td>
</tr>
<tr>
<td>00000001</td>
<td>A00650</td>
</tr>
<tr>
<td>00000002</td>
<td>A01400</td>
</tr>
<tr>
<td>00000001</td>
<td>A09000</td>
</tr>
<tr>
<td>00000001</td>
<td>A09050</td>
</tr>
</tbody>
</table>

**Example 2**

From the CALL DML table COMPANY we want to know the number of customers from Munich or Nuremberg. Customers from Munich can be identified by a zip code starting with ‘80’ or ‘81’, those from Nuremberg with ‘90’. The first two bytes of the customer zip code attribute (AGV) are inverted. Thus we need to find out the number of zip codes where the first two digits are ‘80’, ‘81’ or ‘90’.

**Statement area:**

```plaintext
xxx 60BI AGV 52301 9
```

- `xxx` any three-character string instead of the password
- `60BI` operation code for the index browsing statement
- `AGV` symbolic attribute name of the attribute customer zip code
- `523` search and comparison condition:
  - `01` The first two digits of the zip code must be equal to the comparison value in the inquiry area.
- `9` end identifier

**Acknowledgment area:**

- `CO` file identifier of the logical file
Using DML statements  Inquiring on attribute value frequency (index browsing)

Inquiry area:

80... first comparison value for the attribute customer zip code, blank-filled to the full attribute length.

81... second comparison value for the attribute customer zip code, blank-filled to the full attribute length.

90... comparison value for the attribute customer zip code, blank-filled to the full attribute length.

Following successful execution of the statement, the DBH returns status code 00 in the acknowledgment area. The response records are:

| 00000002 | 80??? |
| 00000001 | 90??? |

00000002/00000001

80???/90???

attribute value: the non-inverted positions are replaced by the current mask character.
4.8 Defining comparison values

The set mask character/string identifier function allows the default mask character “?” or the default string identifier “%” to be changed to another character. The delete mask character/string identifier function resets the current mask character or string identifier to the default character “?” or “%”.

Example 1

The @ character is to be used instead of the default mask character “?” for logical file CO of the CALL DML table COMPANY. The table is then to be searched for all employees whose department number ends in 1. For these employees, the last name (ALM), first name (AMK) and department (AR9) are to be output in addition to the personnel number (AAA). Afterwards, the mask character is to be reset to the default character “%”.

Statement area:

\[
\begin{array}{cccc}
xxx & 6 & 0 & F & 9 \\
\end{array}
\]

xxx any three-character string instead of the password
6 operation code for the define comparison values statement
0 mandatory entry
F set mask character
9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

@ new mask character:
applies until it is replaced by another character, or until logical file CO is closed.

Following successful execution of the statement, the DBH returns status code 00 in the acknowledgment area. The search is then performed to retrieve the required employee information.
Using DML statements

Defining comparison values

Statement area:

| xxx | 6 | 1 | 1 | EALMAMK000 | CAR9401 | 9 |

xxx any three-character string instead of the password
6 operation code for the search statement
1 primary key function:
all records whose primary key value contains primary key group value ‘P’
left-justified are selected. This selects all records in PERSONNEL.

1 strategy:
the DBH decides whether the table is searched sequentially or on the index.

EALMAMK000
E subquestion for projecting the last name (ALM) and first name (AMK)
C C subquestion for selection and projection
AR9 symbolic attribute name of the attribute department
4 search condition for the search with masked comparison value
01 comparison value for equality
9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

| P | @@@1 |

P comparison value for the primary key function:
primary key group value identifying the records in PERSONNEL.
@@ @ 1 comparison value for the attribute AR9

Following successful execution of the statement, the DBH returns status code 00 in the
acknowledgment area. The first response record is placed in the response area.

| P00333 | HUBER| ANDREAS | DPT1 |
Further response records can be output by means of the response polling statement (see section “Response polling” on page 117). The mask character @ is then replaced by the default mask character “?”. 

Statement area:

```
xxx 6 0 E 9
```

- `xxx`: any three-character string instead of the password
- `6`: operation code for the define comparison values statement
- `0`: mandatory entry
- `E`: delete mask character
- `9`: end identifier

Acknowledgment area:

```
CO file identifier of the logical file
```

**Example 2**

The default string identifier “%” is to be replaced by the character # for logical file CO of the CALL DML table COMPANY. All pads are then to be retrieved from the table. For every article whose name includes the string PAD, the article number (AAA) and article name (AA8) are to be output. Afterwards, the string identifier is to be reset to the default character “%”.

Statement area:

```
xxx 6 0 F S 9
```

- `xxx`: any three-character string instead of the password
- `6`: operation code for the define comparison values statement
- `0`: mandatory entry
- `F`: set string identifier
- `S`: the statement refers to the string identifier
- `9`: end identifier
Using DML statements

Defining comparison values

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:
# new string identifier:
    applies until replaced by another character or until logical file CO is closed.

Following successful execution of the statement, the DBH returns status code 00 in the acknowledgment area. The search is then performed to retrieve the required article information.

Statement area:

```
xxx 6 1 1 CAA8401 9
```

xxx any three-character string instead of the password
6 operation code for the search statement
1 primary key function:
    all records whose primary key value contains primary key group value ‘A’ left-justified are selected. This selects all records in ARTICLE.
1 strategy:
    the DBH decides whether the table is searched sequentially or on the index.
C C subquestion for selection and projection
AA8 symbolic attribute name of the attribute article name
4 search condition for the string search
01 comparison condition for equality
9 end identifier

Acknowledgment area:
CO file identifier of the logical file
Defining comparison values

Inquiry area:

A________ #PAD#__________

A________ comparison value for the primary key function:
primary key group value identifying the records in ARTICLE.

#PAD#__________
comparison value for attribute AA8:
the search string PAD is enclosed in string identifiers (#) and blank-filled to
a length of 15 as defined in the attribute catalog.

Following successful execution of the statement, the DBH returns status code 00 in the
acknowledgment area. The first response record is placed in the response area.

A00300 WRITING_PAD________

The remaining response records can be output by means of the response polling statement
(see section “Response polling” on page 117).
The string identifier # is then replaced by the default string identifier “%”:

Statement area:

xxx 6 0 E S 9

xxx any three-character string instead of the password
6 operation code for the define comparison values statement
0 mandatory entry
E delete string identifier
S the statement refers to the string identifier
9 end identifier

Acknowledgment area:

CO file identifier of the logical file
4.9 Retrieval using record output

Record output enables records to be selected conditionally on their primary key values. Information can optionally be output about the attribute definitions of the attributes projected in the response record. No information at all is output for null attribute values.

Example

Output the following information from the CALL DML table SALES for the attributes quantity (ABB) and customer number (AB9):

- symbolic attribute name (SAN)
- length of attribute
- attribute value

Statement area:

| xxx | 4 | 0 | ABB | AB9 | 1 | 9 |

xxx any three-character string instead of the password
4 operation code for the record output statement
0 primary key function: select all records
ABB symbolic attribute name of the attribute quantity
AB9 symbolic attribute name of the attribute customer number
1 format identifier: output the symbolic attribute name (SAN), attribute length and attribute value
9 end identifier

Acknowledgment area:

SA file identifier of the logical file

Inquiry area:

No entry
Following successful execution of the statement, the DBH returns status code 00 in the acknowledgment area. The first response record is placed in the response area:

```
1011  AB9  05  C23979  9
```

Order number
indicates order record
Symbolic attribute name of the attribute customer number
Binary specification of attribute length -1, i.e. the attribute customer number is defined in the attribute catalog with a length of 6 bytes.
Customer number
Identifies the end of the response record

The attribute quantity (ABB) does not have a significant value and is therefore skipped. The remaining responses can be output by means of the response polling statement (see section “Response polling” on page 117).
4.10 Retrieval using inquiry

The inquiry enables records to be selected conditionally on primary key values. Each response record contains the values of the specified attributes and for each attribute, if required, the current attribute definition. The default value is output for null attribute values.

Example

The order and article number, which form the primary key, are to be retrieved from the CALL DML table SALES together with the values of attributes quantity (ABB) and customer number (AB9). The attribute definitions are not output.

Statement area:

```
xxx 5 0 0 ABB AB9 9
```

- `xxx`: any three-character string instead of the password
- `5`: operation code for the inquiry statement
- `0`: primary key function: select all records
- `0`: format identifier:
  - output attribute values only
- `ABB`: symbolic attribute name of the attribute quantity
- `AB9`: symbolic attribute name of the attribute customer number
- `9`: end identifier

Acknowledgment area:

```
SA
```

- `SA`: file identifier of the logical file

Inquiry area:

No entry as primary key function 0 does not require a comparison value.
Following successful execution of the statement, the DBH returns status code 00 in the acknowledgment area. The first response record is placed in the response area:

```
1011 0000 C23979  
```

- **1011** order number
- **0000** indicates the order record
- **C23979** default value for the attribute customer number, which does not have a significant value.

The remaining response records can be retrieved by means of the response polling statement (see section “Response polling” on page 117).
4.11 Adding new records

The addition statement (see section “Addition” on page 131) enables records to be added to a table. If several records of the same format are to be added, the first record is added using the addition statement, and subsequent records by means of the follow-up update statement (see section “Follow-up update” on page 154).

Example

Two new customers with the following data are to be added to the CALL DML table COMPANY:

<table>
<thead>
<tr>
<th>SAN</th>
<th>Value for cust1</th>
<th>Value for cust2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD2</td>
<td>WEBER-----------</td>
<td>HIRNER----------</td>
</tr>
<tr>
<td>AEZ</td>
<td>ANDREAS--------</td>
<td>CHRISTIAN-------</td>
</tr>
<tr>
<td>AFX</td>
<td>TALSTRASSE------</td>
<td>BERGSTRASSE------</td>
</tr>
<tr>
<td>AGV</td>
<td>70372</td>
<td>89081</td>
</tr>
<tr>
<td>AHT</td>
<td>STUTTGART-------</td>
<td>ULM------------</td>
</tr>
<tr>
<td>AJR</td>
<td>951019</td>
<td>951019</td>
</tr>
<tr>
<td>AAA</td>
<td>C40050</td>
<td>C40100</td>
</tr>
</tbody>
</table>

Statement area:

| xxx | 9 | C | X | N | AD2 | AEZ | AFX | AGV | AHT | AJR | AAA | 9 |

xxx any three-character string instead of the password
9 primary key code for the addition statement
C primary key function:
the primary key value can appear at any position in the input record.
X update authorization:
following the direct update, update authorization for logical file CO is cancelled.
N mandatory entry
AD2 symbolic attribute name for the attribute last name
AEZ symbolic attribute name for the attribute first name
AFX symbolic attribute name for the attribute street
AGV symbolic attribute name for the attribute zip code
AHT symbolic attribute name for the attribute city
Adding new records

Using DML statements

AJR symbolic attribute name for the attribute ‘customer since’
AAA symbolic attribute name for the attribute customer number
0 attribute update authorization:
the value in the input record (inquiry area) is adopted as the attribute value for the attribute.
9 end identifier

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:

WEBERANDREASTALSTRASSE70372

STUTTGAR951019C40050

The values must be placed in the inquiry area in the same sequence as their associated symbolic attribute names are specified in the statement. All the values must also be the same length as that specified in the attribute catalog.

When the DBH has added the new record to the table, it returns status code 00 in the acknowledgment area.
The second record is then inserted using the follow-up update statement.

Statement area:

xxx7CXN9

xxx any three-character string instead of the password
7 operation code for the follow-up update statement
C primary key function:
the primary key value can appear at any position in the input record.
X update authorization:
following the direct update, update authorization for logical file CO is cancelled.
N mandatory entry
9 end identifier
Using DML statements

**Adding new records**

Acknowledgment area:
CO file identifier of the logical file

Inquiry area:

```
HIRNER...CHRISTIAN...BERGSTRASSE...89081
```

```
ULM...951019 C40100
```

The DBH confirms the successful addition of the new record by returning status code 00 in the acknowledgment area.
4.12 Updating records

The direct update statement (see section “Update” on page 141) enables existing records to be updated.

Example

The following entries are to be made in the CALL DML table COMPANY for the employee with employee number (AAA) P11500:

Foreign language (AS7):
   Spanish to be added as third foreign language

Department (AR9):
   The new department is DPT2.

Salary (AT5):
   The salary is raised to $4300.00. It must be inserted before the current first occurrence of the multiple attribute.

Statement area:

```
x   xxx   9   C   X   A   AAA0   AS7/+01/H   AR90   AT5/001/N   9
```

<table>
<thead>
<tr>
<th>xxx</th>
<th>9</th>
<th>C</th>
<th>X</th>
<th>A</th>
<th>AAA0</th>
<th>AS7/+01/H</th>
<th>AR90</th>
<th>AT5/001/N</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>any three-character string instead of the password</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>operation code for the update statement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the primary key function: the primary key value can appear at any position in the input record.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>update authorization: following the direct update, update authorization for logical file CO is cancelled.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>record update statement: an existing record is to be updated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>symbolic attribute name of the attribute personnel number (primary key)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>attribute update function: update attribute value (ignored, however, as the primary key value cannot be updated).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an occurrence is added to the multiple attribute foreign languages (AS7).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>attribute update function: append to the last significant occurrence.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>symbolic attribute name of the attribute department</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using DML statements

Updating records

0  attribute update function: the current attribute value in the table is replaced by the value in the inquiry area.

AT5/001/  first occurrence of the attribute salary

N  attribute update function: insert an occurrence before occurrence AT5/001

9  end identifier

Acknowledgment area:

CO  file identifier of the logical file

Inquiry area:

| P11500 | SPAN- | DPT2 | 0430000 |

The values in the inquiry area must be in the sequence in which the attributes are referenced in the statement.

The DBH confirms the successful update by returning status code 00 in the acknowledgment area.
4.13 Deleting records

The deletion direct update statement (see section “Deletion” on page 150) enables complete records to be removed from the table. If several records need to be deleted, just the first record is deleted by record deletion, and all subsequent records by the follow-up update statement (see section “Follow-up update” on page 154).

Example

The two records with the primary key values C40050 and C40100 are to be deleted from the CALL DML table COMPANY.

Alternative 1

The record with the primary key value C40050 is deleted by the deletion statement.

Statement area:

| xxx | 9 | C | X | L | AAAL | 9 |

- xxx: any three-character string instead of the password
- 9: operation code for the deletion statement
- C: primary key function:
  - the primary key value can appear at any position in the input record.
- X: update authorization:
  - following the direct update, update authorization for logical file CO is cancelled.
- L: mandatory entry
- AAA: symbolic attribute name of the attribute customer number (primary key)
- L: mandatory entry
- 9: end identifier

Acknowledgment area:

CO: file identifier of the logical file
Using DML statements

Deleting records

Inquiry area:

C40050

The DBH reports status code 00 when the record has been successfully deleted. The record with the primary key value P40100 is then deleted using the follow-up update statement.

Statement area:

xxx 7 C X L 9

xxx any three-character string instead of the password
7 operation code for the follow-up update statement
C primary key function:
the primary key value can appear at any position in the input record.
X update authorization:
following the direct update, update authorization for logical file CO is cancelled.
L record update function:
delete the record
9 end identifier

Acknowledgment area:

CO file identifier of the logical file

Inquiry area

C40100
Alternative 2:
The records with the primary key values C40050 and C40100 are deleted in block mode with the deletion statement.

Statement area:

```
  xxx  9  C  X  L  AAAL  &BLN002  9
```

<table>
<thead>
<tr>
<th>xxx</th>
<th>any three-character string instead of the password</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>operation code for the deletion statement</td>
</tr>
<tr>
<td>C</td>
<td>primary key function:</td>
</tr>
<tr>
<td></td>
<td>the primary key value can appear at any position in the input record.</td>
</tr>
<tr>
<td>X</td>
<td>update authorization:</td>
</tr>
<tr>
<td></td>
<td>following the direct update, update authorization for logical file CO is cancelled.</td>
</tr>
<tr>
<td>L</td>
<td>mandatory entry</td>
</tr>
<tr>
<td>AAA</td>
<td>symbolic attribute name of the attribute customer number (primary key)</td>
</tr>
<tr>
<td>L</td>
<td>mandatory entry</td>
</tr>
<tr>
<td>&amp;BLN002</td>
<td>block mode:</td>
</tr>
<tr>
<td></td>
<td>a maximum of two records are deleted.</td>
</tr>
<tr>
<td>9</td>
<td>end identifier</td>
</tr>
</tbody>
</table>

Acknowledgment area:

CO file identifier of the logical file

Inquiry area:

```
  C40050
  C40100
```

The DBH reports status code 00 when the records have been successfully deleted.
5 Programming transactions

This chapter contains a description of

- Open and close statements and transaction boundaries
- Chained statements
- Database operations in transactions
- Resetting transactions
- Database accesses outside transaction boundaries

5.1 Open, close and transaction boundaries

In transaction programming, open and close statements can be used both inside and outside transaction boundaries. The following rules must be observed with regard to the sequence of open/close statements and transaction statements (BTA, ETA):

1. Logical files opened outside the transaction boundary must also be closed outside the boundary.

2. The end transaction statement implicitly closes all the user's logical files that were opened within the transaction.

3. If a logical file is opened outside a transaction boundary, a close statement within a transaction boundary for any logical file belonging to the user is rejected with status code 8T.
Open, close and transaction boundaries

**Example**

Of (1):

- OPEN D1
- BTA
- ETA
- CLOSE D1

Of (2):

- BTA
- OPEN GG
- OPEN XX
- OPEN YY
- OPEN W1
- OPEN HH
- ETA

Of (3):

- OPEN GG
- BTA
- OPEN JU
- OPEN RO
- CLOSE
- Status 8T

Figure 1: Example of open and close statements and transaction boundaries
5.2 Chained statements

The statements “begin transaction” (BTA) and “end transaction” (ETA) can be chained to other DML statements. This reduces the number of communication steps and thus improves processing times.

<table>
<thead>
<tr>
<th>Statement 1</th>
<th>Separator</th>
<th>Statement 2</th>
<th>Separator</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTA</td>
<td>;</td>
<td>DML statement</td>
<td>9</td>
</tr>
<tr>
<td>DML statement</td>
<td>;</td>
<td>ETA</td>
<td>9</td>
</tr>
<tr>
<td>ETA</td>
<td>;</td>
<td>Close statement</td>
<td>9</td>
</tr>
<tr>
<td>ETA</td>
<td>;</td>
<td>BTA</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 83: Structure of chained statements

Statement 2 is not executed until statement 1 has terminated with status code 00 or 10.

Example

BTA chained with search: xxx90B;xxx611...9
Follow-up update chained with end TA: xxx74XA;xxx90C9
ETA chained with begin TA: xxx90C;xxx90B9
ETA chained with close statement: xxx90C;xxx8...9
5.3 Operations in transactions

DML statements permitted in transactions

Defining and processing logical input files:
- Search
- Search with join
- Index browsing
- Inquiry
- Record output
- Response polling
- Cursor file handling

Defining and processing logical output files:
- Direct updating (addition, update, record deletion)
- Follow-up update

Information about attribute formats:
- Attribute information

Retrieval in transactions

If a retrieval statement accesses the user data in a CALL DML table, the SESAM/SQL DBH locks this record against access by other transactions until the executing transaction finishes or is reset. Depending on the open mode, either a shared or exclusive lock is set (see section “Open” on page 29).

If the retrieval statement only access the indexes in the system data, no record is locked.

SESAM/SQL also allows the following updated versions of locking:
- Read without locking
- Ignore lock

Read without locking

Retrieval statements within transactions can choose not to lock the record. The user indicates this by means of the &RNL000 option in the retrieval statement.
**Ignore lock**

If a retrieval statement with the &RNW000 option occurs in a transaction, it locks the record which it is accessing. It can, however, access a record that has been locked by another transaction. This means that the highest level of consistency is no longer guaranteed, so SESAM/SQL outputs the response to the retrieval statement with a dirty read status message (status code 9S). On receiving dirty read status, the user can decide whether to start the next transaction step or to reset the transaction.

**Read without locking and ignoring the lock**

Both modifications of the locking principle can be used simultaneously in one retrieval statement. Where this is so, the record read by the retrieval statement is not locked, while the retrieval statement can access a record that has been locked by another transaction.

The SESAM/SQL DBH treats a retrieval transaction step with the &RNL000 and &RNW000 options as being outside the scope of any transaction.

**Example**

```
BTA
  Search ...
  Search ... &RNL000 ...
  Search ... &RNW000 ...
  Search ... &RNL000&RNW000

Update ...
ETA
```
Response polling

It is advisable to process all responses from a retrieval statement within one transaction. The end of the responses to a search is indicated by status code 10.

If the response polling statement is in a different transaction from the base statement (e.g. search), this may result in the following errors:

– On a restart following system failure, the base operation is missing (search etc.).
– A further response polling statement after a restart may return all responses, including those that have already been processed within correctly closed transactions. This may result in data errors such as duplicated additions or subtractions.

Where a single continuation response is polled, e.g. primary key function “equal to primary key value”, this type of error cannot occur.

Database updates in transactions

The SESAM/SQL DBH locks any record accessed by a direct update statement. A transaction that attempts to access a locked record is placed in a wait state. The application program receives no status message. As soon as the record is free, the SESAM/SQL DBH automatically activates the waiting transaction (see also the “Core Manual”).

The following rules must be followed when programming updates in transactions:

Update authorization

X must be specified for update authorization.

Follow-up update

The follow-up update does not have to occur in the same transaction as the base statement for direct updating.

Reset transaction

When a transaction is reset, the “direct update” base statement must be repeated. Otherwise a follow-up update will result in status code 7T.
5.4 Resetting transactions

If a user discovers processing errors (e.g. after validity checks) he can reset the transaction using the DML statement reset transaction. The database administrator can also reset open transactions by means of an administration call.

5.5 Database accesses outside transaction boundaries

Database accesses can still take place outside the transaction boundary even when transaction-oriented security is in use. In this case, the SESAM/SQL DBH proceeds as follows:

- A database access outside a transaction boundary is treated by the SESAM/SQL DBH implicitly as an independent transaction with the transaction boundaries of this one step. Uncompleted update statements are reset after a system failure.
- If a retrieval is performed outside the transaction boundary, again the affected record is not locked. A record that has been locked by another transaction is returned with status code 9S (dirty read). In this case the user decides whether or not to process the record.
Database accesses outside transaction boundaries

Programming transactions

Figure 2: Skeleton of an application program

<table>
<thead>
<tr>
<th>OPEN</th>
<th>Open logical files</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT UPDATES:</td>
<td>Issue base operation</td>
</tr>
<tr>
<td>BEGIN TA; SEARCH</td>
<td></td>
</tr>
<tr>
<td>Status 10</td>
<td></td>
</tr>
<tr>
<td>Processing</td>
<td></td>
</tr>
<tr>
<td>FOLLOW-UP UPDATE</td>
<td></td>
</tr>
<tr>
<td>RESPONSE POLLING</td>
<td></td>
</tr>
<tr>
<td>END TA; BEGIN TA</td>
<td></td>
</tr>
<tr>
<td>Cycle 1</td>
<td></td>
</tr>
<tr>
<td>Processing</td>
<td></td>
</tr>
<tr>
<td>FOLLOW-UP UPDATE</td>
<td></td>
</tr>
<tr>
<td>ENDTA</td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>Close logical files</td>
</tr>
<tr>
<td>Restart routines:</td>
<td>- Start the open and base statements</td>
</tr>
<tr>
<td></td>
<td>- Reset user variables</td>
</tr>
</tbody>
</table>

Processing a set of matches

Only updates to databases
6 From compiling to starting the CALL DML programs

This chapter contains information on

– compiling an application program and entering it in a library, page 261
– linking an application program with the independent or linked-in DBH, page 262
– parameterizing the connection modules, page 264
– starting an application with the independent or linked-in DBH, page 265
– execution with the linked-in DBH, page 268
– using CALL DML in transaction mode, page 271

6.1 Compiling and entering in a library

/START-COBOL85-COMPILER *LIBRARY-ELEMENT(LIBRARY=modlib,
.ELEMENT=source),COMPILER-ACTION=MODULE-GENERATION
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-LMS
//OPEN-LIBRARY LIBRARY=modlib,MODE=UPDATE
//MODIFY-LMS-DEFAULTS WRITE-MODE=REPLACE
//ADD-ELEMENT FROM-FILE=*OMF,TO-ELEMENT=LIBRARY-ELEMENT(TYPE=R)
//END

where:

modlib Module library containing the application program, source programs and user modules

source Name of the application source program
6.2 Linking

There are several ways of using the DBH:

– independent DBH and
– linked-in DBH

6.2.1 Linking with the independent DBH

An application program must be linked as follows, regardless of the BS2000 version and the addressing mode of the DBH:

```
/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=program
//INCLUDE-MODULES LIBRARY=modlib,ELEMENT=module
//INCLUDE-MODULES LIBRARY=SYSLNK.SESAM-SQL.030,ELEMENT=SESMOD
//RESOLVE-BY-AUTOLINK LIBRARY=$TSOS.SYSLNK.CRTE
//SAVE-LLM LIBRARY=modlib,ELEMENT=program,OVERWRITE=YES
//END
```

Where:

- **program** Program name of the application program
- **module** User module
- **SYSLNK.SESAM-SQL.030** SESAM/SQL module library
- **SESMOD** Connection module

The connection module SESMOD dynamically loads the connection modules corresponding to the current version of BS2000. Note the sequence in which the module libraries are searched (see section “Starting” on page 265).

The module SESORT must be linked in if a sort operation is to be initiated with a DML statement (SESORT call).

Non-XS compatible application programs can still use the extended address space of an XS system, as the connection module automatically loads the corresponding modules into the top part of address space. Existing application programs do not need to be relinked in order to do so.
6.2.2 Linking with the linked-in DBH

SESAM/SQL-LINK is a chargeable optional product for the SESAM/SQL database system. SESAM/SQL-LINK can be used wherever one or more databases are to be processed exclusively by one program.

The module for the linked-in DBH is called SESLINK. It is linked with the user module. In BS2000, SESLINK can be located anywhere in the linkage editor structure.

The name or ENTRY of an application program cannot start with the string “SE” as this can lead to conflicts during execution of the program.

The first time it is called, the DBH core is fetched into main memory. The user must ensure that the DBH core is contained in one of the SESAM/SQL module libraries.

SESLINK assumes responsibility for obtaining the remaining storage space required.

A linked in application program must be linked with the connection module SESLINK as follows regardless of the BS2000 version:

/START-BINDER
//START-LLM-CREATION INTERNAL-NAME=program
//INCLUDE-MODULES LIBRARY=modlib,ELEMENT=module
//INCLUDE-MODULES LIBRARY=SYSLNK.SESAM-SQL.030,ELEMENT=SESLINK
//RESOLVE-BY-AUTOLINK LIBRARY=$TSOS.SYSLNK.CRTE
//SAVE-LLM LIBRARY=modlib,ELEMENT=program,OVERWRITE=YES
//END

Where:

program Program name of the application program
modlib Module library containing the user modules
module User module
SYSLNK.SESAM-SQL.030 SESAM/SQL module library
SESLINK Connection module
Parameterizing the connection modules

6.3 Parameterizing the connection modules

There is exactly one SESMOD, SESLINK, SESDCAM or SESUTMC connection module for each operating mode. Connection modules require a number of parameters in order to function. These parameters are entered in the SESMOD and SESDCAM connection modules using CALL DML calls or the configuration file (see the “Core Manual”). The parameters for the SESUTMC connection module are entered in the UTM start parameters or the configuration file (see the “Core Manual”).

Parameter entry via configuration file

The configuration file is a SAM file with a freely selectable name in which the user enters parameters with a file editor. The SET-FILE-LINK command is used to assign the configuration file to the connection module with the link name SESCONF (see the “Core Manual”, configuration file).

It is also possible to group together local configuration files for a SESAM/SQL configuration in a global configuration file (see the “Core Manual”, global configuration file). In this event, the configuration file is assigned using the CONNECT-SESAM-CONFIGURATION command.

When the application is started, the connection modules are supplied with the specified parameters (see the “Database Operation” manual), and any CALL DML statements for program control are ignored. The UNT, NOUNT and TRACE statements are an exception (see also section “Special statements” on page 340).

If no configuration file is assigned, the SESMOD connection module is parameterized with the corresponding default value “.” for the parameters CNF and NAM. Only the DBH can be addressed with the communication name “.” of the configuration “.”.

Using the configuration file gives a high degree of flexibility:

- An application program can be assigned to another SESAM/SQL DBH or configuration without altering the source or requiring a new linkage run.

- A linked-in application program can be converted to an application program executable under an independent DBH without altering the source. This only requires a linkage run (SESMOD and not SESLINK) and the exchange of the DBH configuration file for the application configuration file. Similarly, an application program executable under an independent DBH can be converted to a linked-in application program.
6.4 Starting

There are several ways of using the DBH
– independent DBH and
– linked-in DBH

6.4.1 Starting an independent DBH application

```
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=$KENN.SYSLNK.SESAM-SQL.030

/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=dbhconf
or
/CONNECT-SESAM-CONFIGURATION TO-FILE=globconf-
/,CONFIGURATION-LINK=linkname

/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=modlib-
/,ELEMENT=program-
/,PROGRAM-MODE=ANY-
/,RUN-MODE=ADVANCED(ALTERNATE-LIBRARIES=YES))
```

Where:

- **dbhconf**  DBH configuration file
- **globconf**  Global configuration file
- **modlib** Module library containing the user modules
- **program** Program name of the application program
- **linkname** Link name under which the DBH options are defined in the global configuration file
6.4.2 Starting a linked-in DBH application

/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/MODIFY-SDF-OPTIONS SYNTAX-FILE=USER(NAME=-
                         $KENN.SYSSDF.SESAM-SQL.030.USER)
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=$KENN.SYSTEM.SESAM-SQL.030
/MODIFY-MSG-ATTR TASK-LANG=D
/SET-FILE-LINK LINK-NAME=BLSLIB00,FILE-NAME=$TSOS.SYSLNK.CRT
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=$KENN.SYSLNK.SESAM-SQL.030

/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=dbhconf
or
CONNECT-SESAM-CONFIGURATION TO-FILE=globconf-
   ,CONFIGURATION-LINK=linkname

/START-PROGRAM FROM-FILE=*MODULE(LIBRARY=modlib-
        ,ELEMENT=program-
        ,PROGRAM-MODE=ANY-
        ,RUN-MODE=ADVANCED(ALTERNATE-LIBRARIES=YES))

Where:

dbhconf       DBH configuration file
globconf      Global configuration file
linkname      Link name under which the DBH options are defined in the global configuration file
modib         Module library containing the user modules
program       Program name of the application program
6.4.3 Assigning module libraries

The connection modules dynamically load modules from the following module libraries when running application programs:

1. The module library assigned by /SET-FILE-LINK LINK-NAME=SESAMOML.
2. The module library assigned by /SET-FILE-TASKLIB.
   This module library is ignored if it is the TASKLIB library of the default user ID.
3. The module library SYSLNK.SESAM-SQL.030 of the current user ID.
4. The module library SYSLNK.SESAM-SQL.030 of the default user ID.
5. The module library TASKLIB of the default user ID ($TASKLIB).

These module libraries are only searched in the specified sequence when the first module is dynamically loaded. All the other modules are loaded from the module library in which the first module was found. Therefore, all the SESAM/SQL modules that are to be dynamically loaded must be located in a single module library.

When the SESAM/SQL-DBH or a SESAM/SQL application program is started, the CRTE library must be assigned the link name BLSLIB<xx> (if possible: <xx>=00) (see the “CRTE” manual).
6.5 Execution with the linked-in DBH

- The linked-in DBH is parameterized using a DBH configuration file which has been assigned with the link name SESCONF or by means of the CONNECT-SESAM-CONFIGURATION command. If no configuration file has been assigned, the linked-in DBH requests entry of the DBH start statements and options with “//”.

- SESAM/SQL calls can be executed from within an application in 24-bit or 31-bit addressing mode. It is also possible to switch back and forth between the addressing modes from call to call.

- SESAM/SQL is administered with /SEND-MSG commands, which are identified by the string SES:

  /SEND-MSG TO=PROGRAM(JOB-ID=OWN),MSG=´SES,call´

6.5.1 Terminating the program

The linked-in DBH is normally terminated with the command

/SEND-MSG TO=PROGRAM(JOB-ID=OWN),MSG=´SES,STOP´

The application program or utility routine can then be terminated, or SESAM/SQL operation can be started again.

Under the linked-in DBH, the administration command STOP does not take effect until the application program issues another DML statement to the linked-in DBH. This DML statement is acknowledged with status code 99. If no further DML statements follow during program execution, the linked-in DBH is terminated when the application program is terminated with STXIT.
### 6.5.2 General notes

**Interrupt handling (STXIT)**

SESLINK contains a STXIT routine for handling program errors and administration commands which is activated with:

```
/SEND-MSG TO=PROGRAM(JOB-ID=OWN),MSG=´SES,...´
```

The user module may also have its own STXIT routine, which can also be activated with /SEND-MSG commands. In this case, the data in the /SEND-MSG command is passed on to the user module’s STXIT routine:

```
/SEND-MSG TO=PROGRAM(JOB-ID=OWN),MSG=´text´
```

The SESLINK STXIT routine has the following functions:

1. **Recoverable program errors**
   In the event of a program error, a program dump is executed, followed by an attempt to terminate the SESLINK session normally. The databases must be kept in a consistent state.

2. **Timer interrupt**
   Not assigned

3. **Messages to the program**
   Administration calls to the linked-in DBH are sent with the /SEND-MSG command.

4. **Administration calls are stored by SESLINK and not processed until the next SESAM/SQL CALL request. The SESAM/SQL CALL request is performed after the administration request has been processed.**

5. **Unrecoverable program errors**
   The procedure is the same as for recoverable program errors.

6. **Time runout**
   In the event of time runout, an attempt is likewise made to terminate the SESLINK session normally, i.e. to maintain the consistency of the databases.

7. **ABEND event**
   If an event of the ABEND class occurs, an attempt is made to terminate the SESLINK session normally. If this is not possible, the procedure is as for a program error except that no dump is taken.
Execution with the linked-in DBH

**Multiple loading of linked-in applications**

The linked-in DBH uses the DBH files

- SESLK[cn].CURSOR.nnnn  Cursor files
- SESLK[cn].TA-LOG1 and  Transaction log files
  SESLK[cn].TA-LOG2
- SESLK[cn].WA-LOG  Restart file
- SESLK[cn].WORK.TEMP1  Temporary work file
- SESLK[cn].WORK.TEMP2  Temporary work file

Where:

- \( c \)  Configuration identifier
- \( n \)  Communication name

It is not possible to have several DBHs with the same communication name and configuration identifier on a single computer (neither independent nor linked-in DBH, nor mixed).

**Transaction-oriented security and restart**

Transaction-oriented security is the default value:

```
SYSTEM-STRATEGIES=*PARAMETERS(TRANSACTION-SECURITY=*YES(...))
```

For restarts following system failures, the transaction log files and the restart file must be available:

- SESLK[cn].TA-LOG1
- SESLK[cn].TA-LOG2
- SESLK[cn].WA-LOG

Transaction restart is performed when a linked-in program using the same configuration and DBH names as the aborted session is started. The restart information from the corresponding transaction log file and restart file is evaluated and the affected databases recovered.
6.6 Using CALL DML in transaction mode

There are two ways of using CALL DML in transaction mode:

- openUTM
- DCAM

Comprehensive information on openUTM can be found in the “Core Manual“.

6.6.1 DCAM

The SESAM/SQL DCAM DB/DC system is a software system for processing SESAM/SQL databases in transaction mode.

Automatic restart by transaction-oriented security is not supported under DCAM.

The connection module SESDCAM manages the connection of a DCAM application program to the SESAM/SQL DBH. SESDCAM must be linked to the application program.

The connection module SESDCAM enables as many requesters to issue DML statements to the SESAM/SQL DBH in parallel in a DCAM task as specified in the REQUEST-USERS parameter in the configuration file.

Function of the SESDCAM calls

To process SESAM/SQL databases with a DCAM program, the user has available the following calls to the connection module SESDCAM:

- SESAM
- SESPUT
- SESGET
- SESGETW

In all calls, the application program must pass certain transfer areas to the connection module SESDCAM, e.g. SESAM call in a COBOL program:

CALL "SESAM" USING statement acknowledgment response inquiry identification
The calls are described below, indicating which areas are required for each call and what function the call fulfils:

**SESAM**

Transfer areas:
Statement area, acknowledgment area, response area, inquiry area and identification area

Meaning:
The application program passes a DML statement to the DBH and waits for an acknowledgment and a response.

**SESPUT**

Transfer areas:
Statement area, acknowledgment area, response area, inquiry area and identification area

Meaning:
The application program passes a DML statement to the DBH. It does not wait for an acknowledgment or a response from the DBH, but continues processing.

**SESGET**

Transfer areas:
Statement area, acknowledgment area, response area, inquiry area and identification area

Meaning:
The application program inquires whether the DBH has acknowledged a DML statement issued with a SESPUT. If it has, the application program obtains the acknowledgment and the response. Otherwise it has to repeat the SESGET call.

**SESGETW**

Transfer areas:
Statement area, acknowledgment area, response area, inquiry area and identification area

Meaning:
The application program inquires whether the DBH has already acknowledged a DML statement issued with SESPUT. If it has, the program accepts the acknowledgment and the response from the DBH. If not, it waits until the DBH supplies the acknowledgment and response.
Differences between CALL DML and TIAM

- **Identification area**

- In a task there can be as many SESPUT request open as specified in the REQUEST-USERS parameter in the configuration file.

- For SESGET/SESGETW, SESAM/SQL enters the requester identification transferred in the corresponding SESPUT in the identification area.

Transfer areas for the SESDCAM calls

The transfer areas must be defined with the correct length in the application program. The application program must assign them the correct values before the call is issued. The addresses of the transfer areas must be passed to SESDCAM when the call is issued.

The statement, acknowledgment, response and inquiry areas contain all the information required for the DML statement (see section “CALL DML calls” on page 14).

Content of the identification area

This area is used to pass a unique identification code to the SESDCAM connection module. It must consist of the processor name, application name and an extension string.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>host</td>
<td>processor name</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>application</td>
<td>application name</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>user</td>
<td>extension string</td>
</tr>
</tbody>
</table>

Table 84: Structure of the identification area

**host**

Processor name:
the same processor name must be used in all DCAM applications for a processor.

**application**

Name of the DCAM application:
the same application name must be used throughout a DCAM application.

**user**

user is a character string that makes the 24-byte string unique. The logical terminal name could, for example, be used.

The identification may only contain uppercase letters and numbers.

The application name may also be passed to SESDCAM via the configuration file (see the “Core Manual”). To do this, the configuration file must be assigned with the link name SESCONF.
Processor and application names

Processor and application names in DCAM applications can be passed to SESDCAM with the CALL DML interface by means of the identification area or the configuration file. If the configuration file is used, the processor name is determined by SESDCAM. The application name can be specified in the configuration file.

The following overview shows how processor and application names are assigned with and without a configuration file in DCAM applications.

<table>
<thead>
<tr>
<th>Case</th>
<th>Configuration file</th>
<th>Processor name</th>
<th>Application name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>with application name</td>
<td>Processor name as used by DCAM (^1)</td>
<td>as in configuration file</td>
</tr>
<tr>
<td>2</td>
<td>without application name</td>
<td>Processor name as used by DCAM (^1)</td>
<td>TSN=tsn</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>as in ID area</td>
<td>as in ID area</td>
</tr>
</tbody>
</table>

Table 85: Assigning processor and application names in DCAM applications

\(^1\) If DCAM is not available, the fixed name string HOMEPROC is used.

In cases 1 and 2 only the user name (bytes 17 - 24) is passed from the identification area. Processor and application names are ignored.

In case 3 (configuration file not assigned), the processor and application names are passed from the identification area at the first CALL DML call. These names must not subsequently be changed.

In case 2, only single task applications are possible due to TSN=tsn.

In cases 1 and 2, the user name is formed from
– the processor name for DCAM or the constant HOMEPROC and
– the application name from the configuration file and
– the last 8 bytes of the identification area

irrespective of the other entries in the identification area. Under SESGET(W), this name, which has been updated compared with the application, is returned to the application program, whilst under CALL SESAM, the identification area is not overwritten. This behavior should be noted if the first 24 positions of the user name are checked in the application program.
7 CALL DML utility routines

The following utility routines are available for the CALL DML interface:

SEDI61(L) Output retrieval responses
SEDI63(L) Test DML statements.

The variants with “L” (SESLK) are linked-in applications with the software product SESAM/SQL-LINK.

SESAM/SQL-LINK is a chargeable optional product for the SESAM/SQL database system.
Starting the CALL DML utility routines

The SESAM/SQL utility routines can be started with START-PROGRAM or with the corresponding START-SESAM... commands.

In the first case, the start command is:

```
/START-PROGRAM FROM-FILE=*MODULE(ELEMENT=SEDi nn,
    PROGRAM-MODE=ANY,
    RUN-MODE=ADVANCED(ALTERNATE-LIBRARIES=YES),
    LIBRARY=SYSLNK.SESAM-SQL.030))
```

The second case is used in all other examples in the manual:

```
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=dbhconf

or

/CONNECT-SESAM-CONFIGURATION TO-FILE=locConf

./,CONFIGURATION-LINK=linkname

/START-SESAM-RETRIEVAL-DIALOGUE

or

/START-SESAM-CDML-DIALOGUE

(1) Assigns the corresponding configuration file.

(2) Starts SEDI61 or SEDI63 with the special start command (see also “Database Operation”). The following variants are permitted:

<table>
<thead>
<tr>
<th>For SEDI61 and SEDI61L</th>
<th>For SEDI63 and SEDI63L</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>indep.</strong></td>
<td><strong>indep.</strong></td>
</tr>
<tr>
<td>START_SESAM-RETRIEVAL-DIALOGUE</td>
<td>START_SESAM-CDML-DIALOGUE</td>
</tr>
<tr>
<td>SESAM-RETRIEVAL-DIALOGUE</td>
<td>SESAM-CDML-DIALOGUE</td>
</tr>
<tr>
<td>START-SEDI61</td>
<td>START-SEDI63</td>
</tr>
<tr>
<td>SEDI61</td>
<td>SEDI63</td>
</tr>
<tr>
<td><strong>linked-in</strong></td>
<td><strong>linked-in</strong></td>
</tr>
<tr>
<td>START_SESLK-RETRIEVAL-DIALOGUE</td>
<td>START_SESLK-CDML-DIALOGUE</td>
</tr>
<tr>
<td>SESLK-RETRIEVAL-DIALOGUE</td>
<td>SESLK-CDML-DIALOGUE</td>
</tr>
<tr>
<td>START-SEDI61L</td>
<td>START-SEDI63L</td>
</tr>
<tr>
<td>SEDI61L</td>
<td>SEDI63L</td>
</tr>
</tbody>
</table>
```

The CALL DML utility routines SEDI61 or SEDI61(L) and SEDI63 or SEDI63(L) load modules dynamically from the module library SYSLNK.SESAM-SQL.030.

Before the start of a SESAM/SQL utility, the user must ensure that the following files have been assigned:

- The module library SYSLNK.SESAM-SQL.030 (see section “Linking” on page 262).
- The CRTE library under the link name BLSLIB<xx> (if possible: <xx>=00) (see the “CRTE” manual).
### 7.1 Outputting retrieval responses (SEDI61)

Two variants are available:
- For the independent DBH: SEDI61
- For the linked-in DBH: SEDI61L. Certain condition apply when working with the linked-in DBH (see section “Special statements” on page 340).

#### 7.1.1 Functions of SEDI61/SEDI61L

Under SEDI61, or SEDI61L, the responses from retrieval statements can be output to a SAM file.

SEDI61 carries out communication with the database for the following DML statements:
- Open
- Attribute information
- Search
- Record output
- Inquiry

SEDI61 performs the following operations automatically:
- Response polling
- Close
### 7.1.2 Overview of the control statements

<table>
<thead>
<tr>
<th>Control statements and operands</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[STATUSnn]</code></td>
<td>Status statement: output additional error information for status messages</td>
</tr>
<tr>
<td><code>{ ..H.</code>'filename'<code>}</code></td>
<td>Define the output file <code>..H..</code>statement: Status code 10 after a base statement causes SEDI61 to abort</td>
</tr>
<tr>
<td><code>{ ..W.</code>'filename'<code>}</code></td>
<td><code>..W..</code>statement: Status code 10 never causes SEDI61 to abort</td>
</tr>
<tr>
<td><code>{ $</code></td>
<td>Enter the DML statement open <code>$</code> statement: activate the default open statement for the CALL DML table</td>
</tr>
<tr>
<td><code>{ ..A..open,</code></td>
<td><code>..A..</code>statement: enter an open statement</td>
</tr>
<tr>
<td><code>[length-T],length-B,length-S</code></td>
<td></td>
</tr>
<tr>
<td><code>[..T..S</code></td>
<td><code>..T..</code>statements: Initiate definition of the output record structure</td>
</tr>
<tr>
<td><code>{ ..T..U,r,</code></td>
<td>Move attribute values to the output record</td>
</tr>
<tr>
<td>`{ san</td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm-nnn/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ ..T..D,r1,l,</code></td>
<td>Move constants to the output record</td>
</tr>
<tr>
<td><code>{ </code>c`</td>
<td></td>
</tr>
<tr>
<td><code>{ ..T..M,r1,</code></td>
<td>Move parts of attribute values to the output record</td>
</tr>
<tr>
<td>`{ san</td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm-nnn/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ ..T..P,r1,l1,</code></td>
<td>Move numeric attribute values to the output record in packed form</td>
</tr>
<tr>
<td>`{ san</td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ san/mmm-nnn/</code></td>
<td></td>
</tr>
<tr>
<td><code>{ ..T..P,r1,l2</code></td>
<td></td>
</tr>
<tr>
<td><code>...</code></td>
<td>Not permitted in record output</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Table 86: Overview of the control statements (part 1 of 2)
### Control statements and operands

<table>
<thead>
<tr>
<th>Control statements and operands</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{..A..statement..AC´statement´..AX´statement´}</code></td>
<td>..A..statement: Enter the statement area of a DML statement</td>
</tr>
<tr>
<td><code>{..F..inquiry area..FC´inquiry area´..FX´inquiry area´}</code></td>
<td>..F..statement: Enter comparison values</td>
</tr>
<tr>
<td><code>{..Q..nn..QC´nn´..QX´nn´}</code></td>
<td>..Q..statement: Enter the file identifier</td>
</tr>
<tr>
<td><code>{..*..$../*..END}</code></td>
<td>..* statement and $ statement: Enter the file identifier /* statement or END statement: Terminate the SEDI61 control statements and end the program</td>
</tr>
</tbody>
</table>

Table 86: Overview of the control statements (part 2 of 2)
7.1.3 Control statements and operands

**STATUS statement**

An error message to simplify diagnosis is output in addition to the status number. The error messages are in the module called STATUSnn.

<table>
<thead>
<tr>
<th>STATUSnn</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn Two-character extension to the STATUS module name; e.g. GB for STATUS module STATUSGB</td>
</tr>
<tr>
<td>XX The module STATUSXX contains standard error messages.</td>
</tr>
</tbody>
</table>

Default function

If the STATUS statement is not given, only the status number is output.

**."H." statement**

The ".H." statement causes the response output to be placed in a SAM file. After status code 10 for a retrieval statement (base statement), SEDI61 is aborted. If status code 10 occurs after a follow-up statement (automatic response polling), SEDI61 is not terminated and the next control statement is processed.

<table>
<thead>
<tr>
<th>&quot;.H.&quot;filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename Name of the SAM file in which the responses are to be output. The file is given the link name ZDAUS by SEDI61.</td>
</tr>
<tr>
<td>If another file was given the link name ZDAUS by a SET-FILE-LINK command before SEDI61 was called, the file name in the &quot;.H.&quot; statement is ignored and the file specified in the SET-FILE-LINK command is used as the output file.</td>
</tr>
</tbody>
</table>
CALL DML utility routines

Outputting retrieval responses (SEDI61)

_.W_. statement

The _.W_. statement causes the response output to be placed in a SAM file. After status code 10, the SEDI61 run is always continued and the next control statement executed.

(_.W_ `.filename`)

filename Name of the SAM file in which the responses are to be output. The file is given the link name ZDAUS by SEDI61.

If another file was given the link name ZDAUS by a SET-FILE-LINK command before SEDI61 was called, the file name in the _.W_. statement is ignored and the file specified in the SET-FILE-LINK command is used as the output file.

Use of the _.H_. and _.W_. statements

If neither an _.H_. nor a _.W_. statement is specified, SEDI61 checks that the control statements can be executed. The responses are not output.

If the output is to be to a tape file, it must be set up using the following commands:

```
/CREATE-FILE FILE-NAME=`filename`,SUPPORT=TAPE(VOLUME=`archivno`
,DEVICE-TYPE=`device-type`)    
/SET-FILE-LINK LINK-NAME=ZDAUS,FILE-NAME=`filename`
```

If the file assigned in the _.H_. or _.W_. statement does not exist, it is created by SEDI61 on the public volume in accordance to the following command:

```
/CREATE-FILE FILE-NAME=`filename`,SUPPORT=PUBLIC-DISC(SPACE=RELATIVE(PRIMARY-ALLOCATION=192,SECONDARY-ALLOCATION=24)) 
/SET-FILE-LINK LINK-NAME=ZDAUS,...
```

The _.H_. or _.W_. statement must be entered before the first open statement.
Open statement

The open statement must be entered with the following statements:

```
..A..open,[length-T],length-B,length-S
..Q..ff

{ ..* }

{ $ }
```

open Open statement (see section “Open” on page 29):

<table>
<thead>
<tr>
<th>pas</th>
<th>tabname</th>
<th>length-A</th>
<th>length-F</th>
<th>function code</th>
<th>ff</th>
<th>9</th>
</tr>
</thead>
</table>

length-T Length of the area for ..T.. statements:
length-T is calculated thus: number of ..T.. statements * 11
The statements ..T..D and ..T..S are not included.
Maximum value: 9999 bytes
Default value: 2000 bytes

length-B Block length on output to a tape file:
1 byte < length-B ≤ 2048 bytes

length-S Record length on output to a tape file:
1 byte < length-B ≤ 2048 bytes

ff File identifier of the logical file

{ ..* }

{ $ }

End of the open statement

Any further open statement can omit length-T, length-B and length-S, as they are not analyzed by SEDI61 which uses the length specifications from the first open statement. The lengths of the inquiry and response areas must be less than or equal to those in the first open statement (length-R, length-I).

If no more open statements are given, the file identifier ff can be omitted from the ..Q.. statement.

For structured output, it should be borne in mind when specifying the length of the response area for the search and inquiry statements that attribute information statements are executed internally.

The additional length of the response area is a maximum of:
number of requested attributes * 48
For structured output, SEDI61 can process a maximum of 256 attributes for the search and 145 attribute groups for the inquiry.

The block length must be a multiple of the record length or equal to it.

*Default open*

Entry of the $ statement with no preceding A and Q statement causes the following standard open statement to be executed:

```
A..XXXX2SESAM..............0204802048R619,2000,2048,256
```

The following sizes apply:

<table>
<thead>
<tr>
<th>tabname</th>
<th>SESAM..............</th>
</tr>
</thead>
<tbody>
<tr>
<td>length-R</td>
<td>2048 bytes</td>
</tr>
<tr>
<td>length-I</td>
<td>2048 bytes</td>
</tr>
<tr>
<td>length-T</td>
<td>2000 bytes</td>
</tr>
<tr>
<td>length-B</td>
<td>2048 bytes</td>
</tr>
<tr>
<td>length-S</td>
<td>256 bytes</td>
</tr>
</tbody>
</table>
.T. statements define the structure of an output record. Every sequence of .T. statements starts with the statement .T.S. Several output records can be defined within a SEDI61 run.

If no .T. statements are specified, the responses are transferred unchanged to the output file:
- with the response length in the case of attribute information
- with the response record length in the case of a search, inquiry or record output.

.T. statements are not permitted within record output.
.T. statements are ignored in attribute information.

Initiate definition of output record structure

.T.S

S Initiate definition of output record structure and, if it exists, delete the previous definition

The statement .T.S must always be the first in a series of .T. statements.

Transferring an attribute

.T.U, r, \[ \begin{align*}
\text{san} \\
\text{san/mmm/} \\
\text{san/mmm-nnn/}
\end{align*} \]

U Transfer the attribute value to the output record
r Address relative to address 0 of the output record to which the attribute value is to be transferred
san Symbolic attribute name of the attribute to be transferred
san/mmm/ Symbolic attribute name and occurrence number of a multiple attribute
san/mmm-nnn/ Symbolic attribute name and range of occurrences of a multiple attribute
Transferring constants

\[ \text{T..D},r,l,`c` \]

D Transfer a constant into the output record

r Address relative to address 0 of the output record to which the constant is to be transferred

l Length of the constant; the figure given for l must correspond to the length of c

c Constant (max. 50 characters) to be transferred to the output record

Transferring subattributes

\[ \text{T..M}, r, \left\{ \begin{array}{l} \text{san} \\ \text{san/mmm/} \\ \text{san/mmm-nnn/} \end{array} \right\}, r_1, l \]

M Transfer parts of attribute values into the output record

r Address relative to address 0 of the output record to which the subattribute value is to be transferred

san Symbolic attribute name of the attribute to be transferred

san/mmm/ Symbolic attribute name and occurrence number of a multiple attribute

san/mmm-nnn/ Symbolic attribute name and range of occurrences of a multiple attribute

r1 Address relative to displacement 0 of the attribute from which the value is to be transferred

l Length of the value to be transferred

Transferring packed attributes

\[ \text{T..P}, r, l, \left\{ \begin{array}{l} \text{san} \\ \text{san/mmm/} \\ \text{san/mmm-nnn/} \end{array} \right\}, r_1, l_1 \]

P Pack numeric attribute values or subattribute values into the output record

r Address, relative to address 0 in the output record, at which the part of the attribute value is to be inserted

l Length of the packed field in the output record

san Symbolic attribute name of the attribute to be transferred
san/mmm/ Symbolic attribute name and occurrence number of a multiple attribute
san/mmm-nnn/ Symbolic attribute name and range of occurrences of a multiple attribute
r1 Address relative to displacement 0 of the attribute from which the value is to be transferred
l1 Length of the attribute value to be transferred

The following rules apply when specifying ..T.. statements:

- The end identifiers “;” and “&” (with &BLKnnn, &BLNnnn, &RNW000 and &RNL000) in DML statements are replaced by “9”.
- Search with join is not permitted.
- In a search, only strategies 0 or 1 are permitted.
- Subquestions O and U, and the old subquestions F, H, Q, X, W and Z are not permitted.
- Primary key functions 4 and 8 are permitted, but in this case no internal response polling is performed.
- The DML statements “set mask character” and “change mask character” are permitted.
- ..T.. statements for multiple attributes:

Where different occurrences of a multiple attribute are referenced in different ..T.. statements, they cannot be referenced individually in the ..A.. statement; for example:

..T..S
..T..U,0,ABA/001/
..T..U,5,ABA/002/
..T..U,10,ABA/004/
..A..XXX500ABA/001/ABA/004/9

- Output of the attribute values takes place according to the relative addresses in the ..T.. statement. No check is made that the addresses given are compatible with the lengths of the attribute values. This may mean that attribute values in the output record are overwritten.
The \texttt{..A..} statement enables the user to specify the statement area of a DML statement.

The statements can be entered as character values (\texttt{..A..} or \texttt{..AC} statement) or in hexadecimal form (\texttt{..AX} statement):

\textbf{..A..statement}

\begin{tabular}{ll}
statement & DML statement \\
& An \texttt{..A..} statement allows a DML statement of up to 68 characters to be entered. If the DML statement is longer, it must be split over more than one \texttt{..A..} statement.
\end{tabular}

\textbf{..AC\textasciitilde statement\textasciitilde}

\begin{tabular}{ll}
statement & DML statement \\
& An \texttt{..AC} statement allows a DML statement of up to 66 characters to be entered. If the DML statement is longer, it must be split over more than one \texttt{..AC} statement.
\end{tabular}

\textbf{..AX\textasciitilde statement\textasciitilde}

\begin{tabular}{ll}
statement & DML statement in hexadecimal form \\
& An \texttt{..AX} statement allows a DML statement of up to 66 hexadecimal characters (characters 0 to 9 and A to F) to be entered. If the DML statement is longer, it must be split over more than one \texttt{..AX} statement.
\end{tabular}

The following rules apply when specifying \texttt{..A} statements:

- Chained DML statements are not permitted.
- Block mode for response output is not permitted.
**.F. statement**

The .F. statement allows the user to enter the values of the inquiry area of a DML statement.

The values can be entered as character values (.F. or .FC statement) or in hexadecimal form (.FX statement):

- **.F. inquiry area**
  
  Inquiry area entries
  
  An .F. statement allows up to 68 characters to be written to the inquiry area. Longer inquiry area entries must be split over several .F. statements.

- **.FC inquiry area**
  
  Inquiry area entries
  
  An .FC statement allows up to 66 characters to be written to the inquiry area. Longer inquiry area entries must be split over several .FC statements.

- **.FX inquiry area**
  
  Inquiry area entries, in hexadecimal form
  
  An .FX statement allows up to 66 hexadecimal characters (characters 0 to 9 and A to F) to be entered. Longer inquiry area entries must be split over several .FX statements.

*The following rules apply when specifying .F. statements:*

- Each value for the inquiry area can be in a separate .F. statement.
- Attribute values must be of the length defined for the attribute in the attribute catalog.
CALL DML utility routines

Outputting retrieval responses (SEDI61)

..Q.. statement

The ..Q.. statement is used to specify the file identifier of the open logical file. If only one logical file is open, the ..Q.. statement can be omitted.

The acknowledgment area has a fixed length of 2 bytes.

The file identifier can be entered in character (..Q.- or ..QC statement) or hexadecimal form (..QX statement).

..Q..ff

ff File identifier, two characters

..QC´ff´

ff File identifier, two characters

..QX´ff´

ff File identifier, four hexadecimal characters (0 to 9 and A to F)

..* statement and $ statement

The ..* or $ statement terminates the entry of a DML statement.

{ ..* }

{ $ }

END statement and /* statement

The END or /* statement marks the end of the SEDI61 control statements and causes all tables to be closed automatically.

{ END }

{ /* }
7.1.4 Example of SEDI61/SEDI61L

A procedure is used to extract all SEDI61 article data from the CALL DML table COMPANY and output it into the file ARTICLE.DATA (see section “Examples” on page 180).

```plaintext
/. /
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=conf
/SET-FILE-LINK LINK-NAME=ZDAUS,FILE-NAME=output-file
/START-SESAM-RETRIEVAL-DIALOGUE
STATUSXX

$A XXX2COMPANY 0100001000RFM9,2000,2048,128
$T S
T U,1,AAA
T D,7,1,:'
T U,9,AA8
T D,27,7,'Price: '
T U,34,AB6
T D,43,9,'Stock: '
T U,52,AC4
A XXX61EEABAB6AC40009
Q FM
FC 'A
$END
...
```

Where:

- `conf` DBH configuration file
- `output-file` Output file for SEDI61
Contents of the output file ARTICLE.DATA:

<table>
<thead>
<tr>
<th>Article Code</th>
<th>Description</th>
<th>Price</th>
<th>Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>A00100</td>
<td>PENCIL</td>
<td>00059</td>
<td>7576</td>
</tr>
<tr>
<td>A00200</td>
<td>RULER</td>
<td>00189</td>
<td>0503</td>
</tr>
<tr>
<td>A00300</td>
<td>WRITING PAD</td>
<td>00179</td>
<td>1763</td>
</tr>
<tr>
<td>A00340</td>
<td>BALL-POINT PEN</td>
<td>00219</td>
<td>94908</td>
</tr>
<tr>
<td>A00400</td>
<td>BRUSH</td>
<td>00119</td>
<td>1054</td>
</tr>
<tr>
<td>A00650</td>
<td>FOLDER</td>
<td>00395</td>
<td>0107</td>
</tr>
<tr>
<td>A01400</td>
<td>ERASER</td>
<td>00159</td>
<td>0816</td>
</tr>
<tr>
<td>A01750</td>
<td>GLUE</td>
<td>00399</td>
<td>03035</td>
</tr>
<tr>
<td>A08880</td>
<td>CALENDAR</td>
<td>00256</td>
<td>0084</td>
</tr>
<tr>
<td>A09000</td>
<td>DRAWING PAD</td>
<td>00179</td>
<td>0527</td>
</tr>
<tr>
<td>A09050</td>
<td>NOTEPAD</td>
<td>00099</td>
<td>0750</td>
</tr>
</tbody>
</table>
7.2 Testing DML statements (SEDI63)

There are two variants available:

- For the independent DBH: SEDI63
- For the linked-in DBH: SEDI63L.

Certain conditions apply when working with the linked-in DBH (see section “Special statements” on page 340).

7.2.1 Functions of SEDI63/SEDI63L

SEDI63, or SEDI63L, handles communication with the CALL DML table for all DML statements to be tested.

SEDI63 reads the statement, inquiry and acknowledgment areas of each DML statement from SYSDTA. SEDI63 places the responses and acknowledgment area in a runtime log which is output to SYSLST and optionally also to SYSOUT.
### 7.2.2 Overview of control statements

<table>
<thead>
<tr>
<th>Control statements and operands</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[WROUT]</td>
<td>WROUT statement: Log to SYSLST and SYSOUT</td>
</tr>
<tr>
<td>[STATUSnn]</td>
<td>STATUS statement: Output additional error information with status messages</td>
</tr>
<tr>
<td>[SEDECIMAL] [HEXADECIMAL]</td>
<td>SEDEZIMAL/HEXADECIMAL statement: Output responses edited and in hexadecimal form</td>
</tr>
<tr>
<td>[LOW{ON} [OFF]}</td>
<td>LOW statement: Differentiate between uppercase and lowercase letters</td>
</tr>
<tr>
<td>{..A..statement}</td>
<td>..A.. statement: Enter the statement area of a DML statement</td>
</tr>
<tr>
<td>{..AC´statement´}</td>
<td>..AC´statement´</td>
</tr>
<tr>
<td>{..AX´statement´}</td>
<td>..AX´statement´</td>
</tr>
<tr>
<td>{..F..inquiry area}</td>
<td>..F.. statement: Enter attribute values</td>
</tr>
<tr>
<td>{..FC´inquiry area´}</td>
<td>..FC´inquiry area´</td>
</tr>
<tr>
<td>{..FX´inquiry area´}</td>
<td>..FX´inquiry area´</td>
</tr>
<tr>
<td>{..Q..nn}</td>
<td>..Q.. statement: Enter the file identifier</td>
</tr>
<tr>
<td>{..Qc´nn´}</td>
<td>..Qc´nn´</td>
</tr>
<tr>
<td>{..QX´nn´}</td>
<td>..QX´nn´</td>
</tr>
<tr>
<td>{..*}</td>
<td>..* statement and $ statement: Finish a DML statement</td>
</tr>
<tr>
<td>{$}</td>
<td>Repeat DML statement nn times</td>
</tr>
<tr>
<td>{$nn}</td>
<td>Repeat DML statement nn times</td>
</tr>
<tr>
<td>[..A..xxx79LOOP]</td>
<td>Repeated response polling after a base statement</td>
</tr>
<tr>
<td>{/*}</td>
<td>/* statement and END statement: Terminate the SEDI63 control statements</td>
</tr>
<tr>
<td>{END}</td>
<td>and end the program</td>
</tr>
</tbody>
</table>

Table 87: Overview of the control statements

The WROUT, STATUS, HEXADECIMAL/SEDEZIMAL and LOW statements can be given in any order.

The ..A.., ..F.. and ..Q.. statements for entering a DML statement can be given in any order. The last statement must be a ..* or $ statement. This terminates entry of the DML statement and passes it to the DBH.
7.2.3 Control statements and operands

**WROUT statement**

SEDI63 logs the control statements and responses from a retrieval statement to the terminal as well as to SYSOUT.

```
WROUT
```

Default function

If WROUT is not specified, SEDI63 only outputs control statements and responses to the SYSLST.

**STATUS statement**

An error text to simplify diagnosis is output in addition to the status number. The error texts are in the module called STATUSnn.

```
STATUSnn
```

<table>
<thead>
<tr>
<th>nn</th>
<th>Two-character extension to the name of the STATUS module; e.g. GB for STATUS module STATUSGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td>The module STATUSXX contains standard error texts.</td>
</tr>
</tbody>
</table>

If no STATUS statement is given, only the status number is output.
CALL DML utility routines

Testing DML statements (SEDI63)

SEDECIMAL statement and HEXADECIMAL statement

Acknowledgment and responses are output in edited and hexadecimal form.

```plaintext
[SEDECIMAL]
[HEXADECIMAL]
```

If no SEDEZIMAL statement is given, the responses are output in edited form only. Any unprintable character is shown as X’00’, i.e. the encrypted information in these bytes is lost. The HEXADECIMAL statement has the same function as the SEDEZIMAL statement.

LOW statement

SEDI63 can accept lowercase letters as they are or can convert them to uppercase (default function).

```plaintext
[LOW

LOWON SEDI63 differentiates between uppercase and lowercase letters. SEDI63 control statements must be entered in uppercase.
LOWOFF Default function: SEDI63 interprets lowercase letters as uppercase.
..A.. statement

The ..A.. statement enables the user to enter the statement area of a DML statement.

The statements can be entered as characters (..A.. or ..AC statement) or in hexadecimal (..AX statement) form.

..A.. statement

<table>
<thead>
<tr>
<th>statement</th>
<th>DML statement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An ..A.. statement allows a DML statement of up to 253 characters to be entered. If the DML statement is longer, it must be split over more than one ..A.. statement.</td>
</tr>
</tbody>
</table>

..AC`statement`

<table>
<thead>
<tr>
<th>statement</th>
<th>DML statement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An ..AC statement allows a DML statement of up to 251 characters to be entered. If the DML statement is longer, it must be split over more than one ..AC statement.</td>
</tr>
</tbody>
</table>

..AX`statement`

<table>
<thead>
<tr>
<th>statement</th>
<th>DML statement, hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An ..AX statement allows a DML statement of up to 250 hexadecimal characters (0 to 9 and A to F) to be entered. If the DML statement is longer, it must be split over several ..AX statements.</td>
</tr>
</tbody>
</table>

Standard-Open

If a ..* or $ statement is given as the first statement for the CALL DML table without a preceding ..A.. and ..Q.. statement, the following default open statement is executed:

```
..A..XXX2SESAM................010001000X639
```

The following values are used:

- tabname: SESAM
- inquiry area: 1000 bytes
- response area: 1000 bytes
- logical file: 63
CALL DML utility routines

Testing DML statements (SEDI63)

F statement

The F statement allows the user to enter the values of the inquiry area of a DML statement.

The values can be entered as characters (F or FC statement) or in hexadecimal (FX statement) form.

F inquiry area

<table>
<thead>
<tr>
<th>inquiry area</th>
<th>Inquiry area entries</th>
</tr>
</thead>
</table>
| An F statement enables a maximum of 253 characters to be written to the inquiry area. Inquiry area entries longer than this must be split over several F statements.

FC inquiry area

<table>
<thead>
<tr>
<th>inquiry area</th>
<th>Inquiry area entries</th>
</tr>
</thead>
</table>
| An FC statement enables a maximum of 251 characters to be written to the inquiry area. Inquiry area entries longer than this must be split over several FC statements.

FX inquiry area

<table>
<thead>
<tr>
<th>inquiry area</th>
<th>Inquiry area entries, hexadecimal</th>
</tr>
</thead>
</table>
| An FX statement enables a maximum of 250 hexadecimal characters (0 to 9 and A to F) to be entered. Inquiry area entries longer than this must be split over several FX statements.

The following rules apply when specifying F statements:

- Each value in the inquiry area can be put in a separate F statement.
- Attribute values must be given the length defined for them in the attribute catalog.
..Q.. statement

The ..Q.. statement enables the file identifier of the logical file to be entered. If only one logical file is opened, the ..Q.. statement can be omitted.

The acknowledgment area has a fixed length of 16 bytes.

The file identifier can be entered as characters (..Q.. or ..QC statement) or in hexadecimal (..QX statement) form in the acknowledgment area.

..Q..ff

ff File identifier, two characters

..QC´ff´

ff File identifier, two characters

..QX´ff´

ff File identifier, four hexadecimal characters (0 to 9 and A to F)

..* statement and $ statement

The ..* or $ statement finishes the entry of a DML statement.

\{ ..*  
   $nn \}

nn Number of times the statement is to be executed: two digits with leading zeros if necessary; if the $nn statement is not preceded by an ..A.., ..F.. or ..Q.. statement, nn specifies the number of times the previous statement is to be executed.
Response polling

After a retrieval statement, the responses can be polled by means of the DML response polling statement.

SEDI63 offers a facility to retrieve all responses by means of the following statement:

.xxx79LOOP

xxx Password

The xxx79LOOP statement is designed specifically for SEDI63 and must not be used in application programs.

END statement and /* statement

The END or /* statement terminates SEDI63.
7.2.4 Examples of SEDI63/SEDI63L

The following examples show how the transfer areas (statement, acknowledgment and inquiry areas) are entered in SEDI63. The DML statements in the examples are based on the CALL DML tables COMPANY and SALES (see section “Examples” on page 180). SEDI63 logs all input and output to SYSLST.

Example 1
The statement area is to be filled by the following search:

XXX611CAAB401LAB6502CAC45049AB65029

For ease of interpretation, the subquestions of the search are entered individually in the statement area.

The statement only contains printable characters and can therefore be entered using the „A„ or „AC statement:

„A„XXX611 „A„CAAB401 „A„LAB6502 „A„CAC4504 „A„OAB6502 „A„9 „AC´XXX611´ „AC´CAAB401´ „AC´LAB6502´ „AC´CAC4504´ „AC´OAB6502´ „AC´9´

Example 2
The binary record number X´0000001B´ and the printable values ´ITAL…´ and ´SPAN…´ are to be entered in the inquiry area.

Enter the 4-byte record number:

„FX´0000001B´

Enter the printable values:

„F„ITAL„SPAN„

Example 3
Use SEDI63 to execute the following search on logical file FM:

XXX611CAAB401LAB6502CAC45049AB65029
CALL DML utility routines

Testing DML statements (SEDI63)

The comparison values to be entered in the inquiry area are as follows:

'`A`, '%BLOCK%`ABCD`EFGH', '00100', '0700' and '00100'.

```
A XXX611 CAAB8401 LAB6502 CAC4504 OAB6502
```

or:

```
A XXX611
A CAAB8401
A LAB6502
A CAC4504
A OAB6502
A 9
```

Extract from the SEDI63 runtime log:

```
% SES1201 13:27:18 OPEN   DBC-NR: 02    LD: FM
0 0         F M . . . . . . . . .
0 0 . . . . F M . . . . . . . . . A 0 0 1 0 0 P E N C I L                   0 0
0 5 9 7 5 7 6
0 1 7 9 1 7 6 3
0 0 . . . . F M . . . . . . . . . A 0 9 0 5 0 N O T E   P A D               0 0
```

...
8 DML statements for old data types

Before writing an application program, the user must know whether the CALL DML tables to be processed only contain the new data types CHAR, NUMERIC, DECIMAL, INTEGER and SMALLINT, or whether they also contain old data types from SESAM versions < V13.1.

Chapter 3 on page 27 describes the DML statements for tables containing only new data types.

Chapter 9 summarizes the special points that must be taken into account where old data types are being used.

There are two types of old data types:

- Old data types that can be interpreted as new data types: these attributes can be processed without restriction by the DML statements described in chapters 3 and 9.
- Old data types that cannot be interpreted as new data types: the DML statements and conditions described in chapter 9 apply here.

8.1 Differences when working with old data types

The following points must be observed when working with old data types:

The DML statement index browsing (see section “Index browsing” on page 85) can only be used for attributes that have been declared in full or in part as an index.

All data types are allowed for attributes declared in full as an index.

Attributes that are partially inverted must have the following data type:

Alignment: left-justified

Storage format: character with/without filler bytes
8.2 Record output

Record output comprises the following functions:

- Selection of records conditionally based on the primary key value
- Projection of attribute sequences; null attribute values are not output
- Optional output of the attribute definition of the projected attributes

Record output produces variable length responses, as null attribute values are suppressed from response output. If the response area defined in the open statement was not large enough, remainder response polling (see section “Response polling” on page 117) can be performed.

Record output can be used, for example, to retrieve continuous text from a table.

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:

Acknowledgment area: The application program supplies the file identifier, and the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values for the primary key value, they must be made available in the inquiry area.
## Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/-</td>
<td>-/1</td>
</tr>
</tbody>
</table>

```
pas 4 0 san1 ...
1 san1/mmm/ san2 0
2 san2/nnn/ 1
```

### Block mode

```
&BLKnnn
&PSN000
&BLNnnn
```

### Lock mode

```
&RL000
&RNW000
```
Key

PK (primary key) 0  All data
    function 1  Equal to primary key group value
              2  Range of primary key group values
              3  Greater than primary key group value
              4  Equal to primary key value
              5  Range of primary key values
              6  Greater than primary key value
              8  Equal to record number

Format identifier 0  Full attribute definition and attribute values for old data types
                   1  SAN, attribute length and attribute values
                   2  Only attribute values
                   3  Full attribute definition and attribute values

End identifier     9  End of statement
                   ;  Chain statement

Password (0/3)

pas  Password for a protected CALL DML table,
     any three-character string for an unprotected CALL DML table.

Operation code (3/1)

4  Operation code for the record output statement

Primary key function (PK function) (4/1)

The primary key function enables records to be selected conditionally based on primary key value or record number. The comparison values used for selection are entered in the inquiry area (see “Inquiry area” on page 316). A primary key group value can be used for selection as an alternative to the primary key value:
The primary key group value designates a group of records in which the primary key group value is contained, left-justified, in the primary key. The comparison value in the inquiry area must be blank-filled to the full length of the primary key.

With a compound key, the primary key must be a value of compound key attribute AAB or of several compound key attributes (beginning AAB and ascending AAC etc.). Blanks must be entered in the inquiry area for the remaining compound key attributes.
0 All data:
All records are selected.

1 Equal to primary key group value:
A primary key group value must be entered in the inquiry area as the comparison value.
All records whose primary key value contains the primary key group value left-justified are selected.

2 Range of primary key group values:
Two primary key group values defining a range of primary key group values must be entered in the inquiry area as comparison values.
All records with a primary key value greater than or equal to the first comparison value and less than or equal to the second comparison value are selected.
The first comparison value must not be larger than the second.

3 Greater than primary key group value:
A primary key group value must be entered in the inquiry area as the comparison value.
All records whose primary key value is greater than the comparison value are selected.

4 Equal to primary key value:
A primary key value must be entered in the inquiry area as a comparison value. The record whose primary key value is equal to the comparison value is selected.

5 Range of primary key values:
Two primary key values defining a range of primary key values must be entered as comparison values in the inquiry area.
All records whose primary key value is greater than or equal to the first comparison value and less than or equal to the second comparison value are selected.
The first comparison value must not be greater than the second.

6 Greater than primary key value:
A primary key value must be entered as the comparison value in the inquiry area.
All records whose primary key value is greater than the comparison value are selected.

8 Equal to record number:
A record number must be entered as the comparison value in the inquiry area. The record with the specified record number is selected.
**Attribute selection (5/-)**

Record output enables attribute sequences to be projected into the output record. An attribute sequence is defined by means of a start and end attribute. The start and end attributes must be specified in ascending order. Any number of attribute sequences may be specified. They must, however, be given in ascending sequence and must not overlap. An attribute sequence can also begin or end with an occurrence of a multiple attribute. The start and end attributes are equal if just one attribute needs to be referenced.

- **San1** Symbolic attribute name of the start attribute
- **San1/mmm/** Occurrence of a multiple attribute with which the attribute sequence begins
- **San2** Symbolic attribute name of the end attribute
- **San2/nnn/** Occurrence of a multiple attribute with which the attribute sequence ends

The end attribute can be omitted from the last attribute sequence in record output if the last attribute defined in the attribute catalog is to be used as the end attribute.

**Format identifier (-/1)**

The format identifier defines what information is to be output for the projected attributes:

- **0** Only for old data types:
  The full attribute definition and the attribute values are output. Attributes with a new data type are treated as non-existent attributes (status code 42).

- **1** Symbolic attribute name, attribute length and attribute values are output.

- **2** Only the attribute values or values of occurrences of a multiple attribute are output.

- **3** The full attribute definition and the attribute values are output.

If no format identifier is specified, format identifier 0 is used as the default.
Attribute definition and attribute value are only output if the attribute has a significant value.
Block mode (-/7)

The user can specify how many of the responses that have been found are to be placed in the response area.

&BLKnnn
  nnn responses are to be placed in the response area. The record number of each response is output.

&BLNnnn
  nnn responses are to be placed in the response area. The record numbers of the responses are not output.

If neither &BLKnnn nor &BLNnnn is specified, just one response record is placed in the response area without a record number.

Lock mode (-/7)

&PSN000
  The response records are output without the primary key value.

&RNL000
  The record being accessed by a record output within a transaction is not locked.

&RNW000
  The record output can access a record that has been locked by another transaction (dirty read). The statement is acknowledged with status code 9S. Following a dirty read in block mode, no further responses are placed in the response area. If &RNW000 is omitted, a transaction attempting to access a locked record is placed in a wait state until the record becomes free.

End identifier (-/1)

9  Indicates the end of the statement

;  End of statement. The statement is chained to a subsequent end transaction statement.
Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(00) (10)</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>length</td>
<td>Response length</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 88: Acknowledgment area on a response

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(4A) (9S)</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>length</td>
<td>Response length</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 89: Acknowledgment area on error

Status code 4A is reported if the response to record output is longer than the response area length defined in the open statement. The symbolic attribute name of the first attribute that cannot be fitted into the response area is output together with the length of the response placed in the response area.

The remainder of the response can be retrieved by response polling statement xxx739 (see section “Response polling” on page 117).
Status code 9S (read locked record) may also contain the cause of the error giving status code 4A. In this case, the acknowledgment area contains the same information as for status code 4A.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(40</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>42</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>47</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>70</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>98</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9S</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 90: Acknowledgment area on error
**Response area**

In the response area, SESAM/SQL returns the attribute values and possibly also the full attribute definition, or parts of it, depending on the format identifier.

Information is only output for attributes that have a significant value. Null attributes are ignored.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>[pkv]</td>
<td>Primary key value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The primary key value of each response is output with the length specified in the attribute catalog. If &amp;PSN000 is specified, no primary key value is output.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´00´ to X´FF´</td>
<td>Attribute length:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 to 256 bytes</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´00´ to X´0F´</td>
<td>Number of decimal places:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 to 15 decimal places</td>
</tr>
</tbody>
</table>

- Bit 2⁷ 0  Alignment: - right-justified  1  left-justified
- Bit 2⁶ 0  Attribute type: - normal attribute  1  multiple attribute
- Bit 2³ 0  Index lock:  1  index not locked  1  index locked

Table 91: Response area for format identifier 0 (part 1 of 2)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>Search strategy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– search via primary data</td>
</tr>
<tr>
<td>Bit 2⁷</td>
<td></td>
<td></td>
<td>– search via index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Null as attribute value:</td>
</tr>
<tr>
<td>Bit 2⁶</td>
<td></td>
<td>0</td>
<td>– not permitted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– permitted</td>
</tr>
<tr>
<td>Bit 2⁵</td>
<td></td>
<td>0</td>
<td>Result of calculation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– no</td>
</tr>
<tr>
<td>Bit 2⁴</td>
<td></td>
<td></td>
<td>– yes, possible</td>
</tr>
<tr>
<td>Bit 2³,</td>
<td>00</td>
<td></td>
<td>Index information:</td>
</tr>
<tr>
<td>2²</td>
<td>01</td>
<td></td>
<td>– index non-existent</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>– index available</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td>Storage format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– packed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– character without filler bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– character with filler bytes</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X’01´ to X´FF´</td>
<td>Number of occurrences of a multiple attribute as defined in the attribute catalog:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 to 255 occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L(AC)</td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>atv</td>
<td>The attribute value is output with the length given in the attribute catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With multiple attributes, the attribute definition and value are output for each occurrence.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[...]</td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[...]</td>
<td>In block mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>responses 2 - nnn</td>
</tr>
</tbody>
</table>

Table 91: Response area for format identifier 0 (part 2 of 2)
### Record output

**Table 92: Response area for format identifier 1 or 2**

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
</tbody>
</table>
| 4      | L(AC)  | [pkv]  | Primary key value  
The primary key value of each response is output with the length specified in the attribute catalog. If &PSN000 is specified, no primary key value is output. |
| -      | 3      | [san]  | Symbolic attribute name, omitted for format identifier 2 |
| -      | 1      | [X‘00´ to X´FF´ ] | Attribute length: 1 to 256 bytes; omitted for format identifier 2 |
| -      | L(AC)  | atv    | Attribute value  
The attribute value is output with the length given in the attribute catalog. With multiple attributes, the attribute definition and value are output for each occurrence. |
| -      | [...]  |        | Attribute values 2 - n of the response |
| -      | 1      | 9      | End of response record: end identifier that terminates the last attribute value |
| -      | [...]  |        | In block mode: responses 2 - nnn |

### Table 93: Response area for format identifier 3 (part 1 of 3)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
</tbody>
</table>
| 4      | L(AC)  | [pkv]  | Primary key value  
The primary key value of each response is output with the length specified in the attribute catalog. If &PSN000 is specified, no primary key value is output. |
| -      | 3      | san    | Symbolic attribute name |
| -      | 2      | X´0001´ to X´0100´ | Attribute length: 1 to 256 bytes |

Table 93: Response area for format identifier 3 (part 1 of 3)
### DML statements for old data types

#### Record output

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>X’00´ to X’0F´</td>
<td>Number of decimal places: 0 to 15 decimal places</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’11´</td>
<td>Data type: CHAR</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’21´</td>
<td>NUMERIC or corresponding old, interpretable data type for INTEGER</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’22´</td>
<td>DECIMAL Index specification X´10´</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’24´</td>
<td>INTEGER Index specification X´10´</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’28´</td>
<td>SMALLINT Uninterpretable old data type</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>i-length</td>
<td>Index length (binary)</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’01´ to X’0F´</td>
<td>Number of occurrences of a multiple attribute as defined in attribute catalog:</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’00´</td>
<td>Not a multiple attribute</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>dfc</td>
<td>Default value character</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’80´</td>
<td>Compound key details:</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’40´</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>X’00´</td>
<td>Not a compound key</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
</tbody>
</table>

Table 93: Response area for format identifier 3 (part 2 of 3)
### Inquiry area

PK function 0 does not require an entry in the inquiry area.

Comparison values must be entered for PK functions 1 to 6 and 8:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record: end identifier that terminates the last attribute value</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>In block mode: responses 2 - nnn</td>
</tr>
</tbody>
</table>

#### Table 93: Response area for format identifier 3 (part 3 of 3)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AC)</td>
<td>pkv1</td>
<td>For PK functions 1 to 6: primary key (group) value with the length of the primary key as defined in the attribute catalog</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rno</td>
<td>For PK function 8: record number</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>pkv2</td>
<td>For PK functions 2 and 5: Primary key (group) value with the length of the primary key as defined in the attribute catalog</td>
</tr>
</tbody>
</table>

#### Table 94: Inquiry area

Numeric comparison values entered in the inquiry area must be of the correct data type (see under Search, “Inquiry area” on page 59).
8.3 Inquiry

The inquiry offers the following functions:

- Selection of records conditionally based on primary key value
- Projection of attribute sequences, where the default value is output for null attribute values
- Optional output of the attribute definition of the projected attributes

Unlike record output (see section “Record output” on page 304), the responses are always of fixed length as the default value is output for null attribute values.

Up to 1024 attributes can be specified in any one statement. Every occurrence of a multiple attribute counts as one attribute.

The inquiry can be used when the user wishes to select records conditionally on primary key value and not on the values of other attributes. An inquiry allows the record structure required by the application program to be created from the original record structure.

Contents of the transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier, and the DBH returns the acknowledgment to the statement.

Response area: The SESAM/SQL DBH supplies the first response or, in block mode, the first group of responses.

Inquiry area: If the statement requires comparison values for the primary key value, the application program must make them available in the inquiry area.
Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
<td>6/-</td>
</tr>
</tbody>
</table>

Block mode  Lock mode  End id.
-/-7         -/-7         -/-7         

...  

&BLKnnn  &PSN000  &RNL000  &RNW000  

&BLNnnn  

;
DML statements for old data types

Inquiry

Key

<table>
<thead>
<tr>
<th>PK (primary key) function</th>
<th>0</th>
<th>All data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Equal to primary key group value</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Range of primary key group values</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Greater than primary key group value</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Equal to primary key value</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Range of primary key values</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Greater than primary key value</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Equal to record number</td>
</tr>
</tbody>
</table>

Format identifier

<table>
<thead>
<tr>
<th>0</th>
<th>Attribute values only</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Full attribute definition and attribute values for old data types</td>
</tr>
<tr>
<td>3</td>
<td>Full attribute definition and attribute values</td>
</tr>
</tbody>
</table>

End identifier

<table>
<thead>
<tr>
<th>9</th>
<th>End of statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Chain statement</td>
</tr>
</tbody>
</table>

Password (0/3)

pas Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

Operation code (3/1)

5  Operation code for the inquiry statement

Primary key function (PK function) (4/1)

The primary key function enables records to be selected conditionally based on primary key value or record number. The comparison values used for selection are entered in the inquiry area (see “Inquiry area” on page 328). A primary key group value can be used for selection as an alternative to the primary key value:
The primary key group value designates a group of records in which the primary key group value is contained, left-justified, in the primary key. The comparison value in the inquiry area must be blank-filled to the full length of the primary key.

With a compound key, the primary key must be a value of compound key attribute AAB or of several compound key attributes (beginning AAB and ascending AAC etc.). Blanks must be entered in the inquiry area for the remaining compound key attributes.
0 All data:
   All records are selected.

1 Equal to primary key group value:
   A primary key group value must be entered in the inquiry area as the comparison
   value.
   All records whose primary key value contains the primary key group value left-
   justified are selected.

2 Range of primary key group values:
   Two primary key group values defining a range of primary key group values must be
   entered in the inquiry area as comparison values.
   All records with a primary key value greater than or equal to the first comparison
   value and less than or equal to the second comparison value are selected.
   The first comparison value must not be larger than the second.

3 Greater than primary key group value:
   A primary key group value must be entered in the inquiry area as the comparison
   value.
   All records whose primary key value is greater than the comparison value are
   selected.

4 Equal to primary key value:
   A primary key value must be entered in the inquiry area as a comparison value.
   The record whose primary key value is equal to the comparison value is selected.

5 Range of primary key values:
   Two primary key values defining a range of primary key values must be entered as
   comparison values in the inquiry area.
   All records whose primary key value is greater than or equal to the first comparison
   value and less than or equal to the second comparison value are selected.
   The first comparison value must not be greater than the second.

6 Greater than primary key value:
   A primary key value must be entered as the comparison value in the inquiry area.
   All records whose primary key value is greater than the comparison value are
   selected.

8 Equal to record number:
   A record number must be entered as the comparison value in the inquiry area.
   The record with the specified record number is selected.

**Format identifier (5/1)**

The format identifier defines what information is to be output for the projected attributes:

0 Only the attribute values or values of occurrences of a multiple attribute are output.
DML statements for old data types

**Inquiry**

1. Only for old data types:
   The full attribute definition and the attribute values are output.

3. The full attribute definition and the attribute values are output.

If an attribute does not have a significant value, the default value is output instead.

**Attribute selection (6/-)**

The inquiry enables attribute sequences to be projected.

An attribute sequence is defined by means of a start and end attribute. The start and end attributes must be specified in ascending order. Any number of attribute sequences may be specified in any order. The sequences may also overlap.

An attribute sequence can also begin or end with an occurrence of a multiple attribute. The start and end attributes are equal if just one attribute needs to be referenced.

- **san1** Symbolic attribute name of the start attribute
  - san1/mmm/
    - Occurrence of a multiple attribute with which the attribute sequence begins

- **san2** Symbolic attribute name of the end attribute
  - san2/nnn/
    - Occurrence of a multiple attribute with which the attribute sequence ends

With format identifier 0, the first and last attribute in each group must be present and able to be read. Any attributes in a sequence that do not have read authorization are ignored.

**Block mode (-/7)**

The user can specify how many of the responses that have been found are to be placed in the response area.

- **&BLKnnn**
  - nnn responses are to be placed in the response area. The record number of each response is output.

- **&BLNnnn**
  - nnn responses are to be placed in the response area. The record numbers of the responses are not output.

If neither &BLKnnn nor &BLNnnn is specified, just one response record is placed in the response area without a record number.
Inquiry

DML statements for old data types

Lock mode (-/7)

&PSN000
The response records are output without the primary key value.

&RNL000
The record being accessed by a record output within a transaction is not locked.

&RNW000
The inquiry can access a record that has been locked by another transaction (dirty read). The statement is acknowledged with status code 9S. Following a dirty read in block mode, no further responses are placed in the response area. If &RNW000 is omitted, a transaction attempting to access a locked record is placed in a wait state until the record becomes free.

End identifier (-/1)

9 Indicates the end of the statement

; End of statement. The statement is chained to a subsequent end transaction statement.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00\10</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 95: Acknowledgment area on response output
### Inquiry

#### Displ. Length Entry Meaning

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(5\overline{A} ) (5\overline{C} ) (9\overline{S})</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>[san]</td>
<td>Symbolic attribute name (omitted with status code 9S)</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>t-length</td>
<td>Total length of responses</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>e-length</td>
<td>Length of each response</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>rno</td>
<td>Record number; in block mode, this is the record number of the last response record placed in the response area</td>
</tr>
</tbody>
</table>

Table 96: Acknowledgment area on error with response output

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>(5\overline{0} ) (5\overline{1} ) (5\overline{3} ) (5\overline{4} ) (5\overline{D}) (5\overline{M} ) (5\overline{Z} ) (7\overline{0} ) (7\overline{D})</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>[san]</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>ss</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 97: Acknowledgment area on error without response output
Response area

In the response area, SESAM/SQL returns the attribute values and possibly also the full attribute definition, depending on the format identifier.

The default value is output for attributes that do not have a significant value.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
</tbody>
</table>
| 4      | L(AC)  | [pkv] | Primary key value  
The primary key value of each response is output with the length specified in the attribute catalog.  
If &PSN000 is specified, no primary key value is output. |
|        | L(AC)  | atv   | Attribute value  
The attribute value is output with the length specified in the attribute catalog. |
|        | [...   |       | Attribute values 2 to n of the response |
|        | [...   |       | In block mode:  
responses 2 to nnn |

Table 98: Response area for format identifier 0

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>[rno]</td>
<td>Record number, only for block mode &amp;BLKnnn</td>
</tr>
</tbody>
</table>
| 4      | L(AC)  | [pkv] | Primary key value  
The primary key value of each response is output with the length specified in the attribute catalog.  
If &PSN000 is specified, no primary key value is output. |
|        | 3      | san   | Symbolic attribute name |
|        | 1      | X’00´ to X´FF´ | Attribute length:  
1 to 256 bytes |
|        | 1      | X’00´ to X´0F´ | Number of decimal places:  
0 to 15 decimal places |

Table 99: Response area for format identifier 1 (part 1 of 3)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The attribute value is output with the length given in the attribute catalog.</td>
</tr>
<tr>
<td>-</td>
<td>[...]</td>
<td></td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>end identifier that terminates the last attribute value</td>
</tr>
</tbody>
</table>

Table 99: Response area for format identifier 1 (part 2 of 3)
### Table 99: Response area for format identifier 1 (part 3 of 3)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td></td>
<td>In block mode: responses 2 - nnn</td>
</tr>
<tr>
<td>4</td>
<td>L(AC)</td>
<td>[pkv]</td>
<td>Primary key value&lt;br&gt;The primary key value of each response is output with the length specified in the attribute catalog. If &amp;PSN000 is specified, no primary key value is output.</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>X´0001´ to X´0100´</td>
<td>Attribute length: 1 to 256 bytes</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´00´ to X´0F´</td>
<td>Number of decimal places: 0 to 15 decimal places</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´11´, X´21´, X´22´, X´24´, X´28´, X´00´</td>
<td>Data type: CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT&lt;br&gt;Uninterpretable old data type</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´00´, X´02´, X´04´, X´08´, X´10´, X´20´</td>
<td>Index information: Index non-existent, Index locked, Index available, Index required, Old data type&lt;br&gt;Null not permitted as attribute value (old data type)</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>i-length</td>
<td>Index length (binary)</td>
</tr>
</tbody>
</table>
| -      | 1      | X´01´ to X´0F´, X´00´ | Number of occurrences of a multiple attribute as defined in attribute catalog: 1 to 255 occurrences
Not a multiple attribute |

### Table 100: Response area for format identifier 3 (part 1 of 2)
### DML statements for old data types

#### Inquiry

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>dfc</td>
<td>Default value character</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X'80'</td>
<td>Compound key details:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X'40'</td>
<td>Compound key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X'00'</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not a compound key</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>atv</td>
<td>Attribute value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Output with the length specified in the attribute catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With multiple attributes, the attribute definition and attribute value are output for every occurrence.</td>
</tr>
<tr>
<td>-</td>
<td>[...]</td>
<td></td>
<td>Attribute values 2 - n of the response</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>9</td>
<td>End of response record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>end identifier that terminates the last attribute value</td>
</tr>
<tr>
<td>-</td>
<td>[...]</td>
<td></td>
<td>In block mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>responses 2 - nnn</td>
</tr>
</tbody>
</table>

Table 100: Response area for format identifier 3 (part 2 of 2)
Inquiry area

PK function 0 does not require an entry in the inquiry area.

Comparison values must be entered for PK functions 1 to 6 and 8:

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L(AK)</td>
<td>pkv1</td>
<td>For PK functions 1 to 6: primary key (group) value with the length of the primary key as defined in the attribute catalog</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>rno</td>
<td>For PK function 8: record number</td>
</tr>
<tr>
<td>-</td>
<td>L(AC)</td>
<td>[pkv2]</td>
<td>For PK functions 2 and 5: primary key (group) value with the length of the primary key as defined in the attribute catalog</td>
</tr>
</tbody>
</table>

Table 101: Inquiry area

Numeric comparison values entered in the inquiry area must be of the correct data type (see under Search, “Inquiry area” on page 59).
8.4 Setting and deleting the delete identifier

The value of an attribute can be deleted in an update or a follow-up update by entering the delete identifier instead of the attribute value in the inquiry area. This can avoid the need for interrupting a follow-up update and issuing another statement.

The delete identifier is assigned by means of the set delete identifier statement and cancelled by means of the delete delete identifier statement.

The delete identifier applies
– until a new delete identifier is defined,
– until it is cancelled by a delete delete identifier statement, or
– until the logical file is closed.

The delete identifier cannot be used to delete a record by deleting the primary key value.

Content of the transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Inquiry area: The application program supplies the character to be used as the delete identifier.

The response area and, on deletion of the delete identifier, the inquiry area, are not evaluated.
Setting and deleting the delete identifier

Statement area

<table>
<thead>
<tr>
<th>Pass-</th>
<th>Op.</th>
<th>Func-</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>code</td>
<td>tion</td>
<td>id.</td>
</tr>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1</td>
</tr>
</tbody>
</table>

Password (0/3)

pas Password for a protected CALL DML table,
any three-character string for an unprotected CALL DML table.

Operation code (3/1)

9 Operation code for the set/delete delete identifier statement

Function (5/1)

F Set delete identifier.
The new delete identifier must be placed in the inquiry area (see “Inquiry area” on page 331).

E Delete current delete identifier.

End identifier (6/1)

9 Indicates end of statement

; The statement can be chained to a following end transaction statement.
### Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>[00 \01 \0A \0F]</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[ss]</td>
<td>Status subnumber</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 102: Acknowledgment area

### Inquiry area

The inquiry area only need be given a value when setting the delete identifier.

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>Character to which the function of the delete identifier is assigned.</td>
</tr>
</tbody>
</table>

Table 103: Inquiry area
8.5 Attribute information

The attribute information statement provides information about the definition of one or more attributes. The following information can be obtained:

- symbolic and/or verbal attribute name
- data type
- attribute length and number of decimal places
- index information
- number of occurrences of a multiple attribute
- default value character
- compound key details

The attribute information statement differs from the record output (see section “Record output” on page 96) and inquiry (see section “Inquiry” on page 107) statements in that only the above information is output. The attribute values cannot be obtained.

Contents of the transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area:

- The application program supplies the file identifier; the DBH returns the acknowledgment to the statement.

Response area: The DBH returns the required information.

The inquiry area is not evaluated.
Statement area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/1</td>
<td>4/1</td>
<td>5/1 6/-</td>
<td>-/1</td>
</tr>
</tbody>
</table>

Key

Information function

1  SAN (old and new data types)
2  VAN (old and new data types)
3  SAN and VAN (old and new data types)
4  Attribute definition (old data types)
5  SAN and attribute definition (old data types)
6  VAN and attribute definition (old data types)
7  SAN, VAN and attribute definition (old data types)
A  SAN, VAN and attribute definition (old data types)

End identifier

9  End of statement

;  Chain statement
Password (0/3)

pas Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

Operation code (3/1)

3 Operation code for the attribute information statement

Information function (4/1)

The information function specifies what information is to be output about an attribute.

For old and new data types:

1 Output symbolic attribute name
2 Output verbal attribute name
3 Output symbolic and verbal attribute names

Only for old data types:

4 Output attribute definition
5 Output symbolic attribute name and attribute definition
6 Output verbal attribute name and attribute definition
7 Output full attribute definition including symbolic and verbal attribute names

For new and old interpretable data types:

A Output full attribute definition including symbolic and verbal attribute names
Attribute selection (5/-)

Attribute selection requires a code number indicating the type of attribute name to follow, and the attribute name itself.

*id.* attribute name

| 1 | san | Symbolic attribute name |
| 3 | san1 | Symbolic attribute name of the start attribute in an attribute sequence |
|    | san2 | Symbolic attribute name of the end attribute in an attribute sequence |
| 2 | van | Verbal attribute name: verbal attribute names less than 31 characters long must be blank-filled on the right to the full length of 31 characters. |
| 4 | vgn | Group name for verbal attribute names: common part of verbal attribute names all beginning with the same character string, blank-filled on the right to the full length of 31 characters if necessary. |

The maximum number of attributes is limited by the capacity of the response area (max. 32000 bytes).

End identifier (-/1)

9 Indicate the end of the statement;

End of statement. The statement is chained to a following end transaction statement.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
<td>Status</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>ff</td>
<td>File identifier</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>r-length</td>
<td>Response length (binary)</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 104: Acknowledgment area for response
Attribute information

### Displ. | Length | Entry | Meaning |
|--------|--------|--------|---------|
| 0      | 2      | [30
31
33
3V
3Z] | Status |
| 2      | 4      | [van
san_] | Verbal attribute name for status codes 31, 3V
Symbolic attribute name for status codes 31, 33 |
| 6      | 2      | ff     | File identifier |
| 8      | 2      | r-length | Response length (binary) |
| 10     | 6      | -      | - |

Table 105: Acknowledgment area on error

---

**Response area**

### Displ. | Length | Entry | Meaning | Info. function |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>[san]</td>
<td>Symbolic attribute name</td>
<td>1, 3, 5, 7</td>
</tr>
<tr>
<td>-</td>
<td>31</td>
<td>[van]</td>
<td>Verbal attribute name</td>
<td>2, 3, 6, 7</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X’00´ to X´FF´</td>
<td>Attribute length: 1 to 256 bytes</td>
<td>4, 5, 6, 7</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>X´00´ to X´0F´</td>
<td>Number of decimal places: 0 to 15 decimal places</td>
<td>4, 5, 6, 7</td>
</tr>
<tr>
<td>- Bit 2^7</td>
<td>1</td>
<td>0</td>
<td>Alignment: right-justified</td>
<td></td>
</tr>
<tr>
<td>Bit 2^6</td>
<td></td>
<td>1</td>
<td>left-justified</td>
<td></td>
</tr>
<tr>
<td>Bit 2^3</td>
<td></td>
<td>0</td>
<td>normal attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>multiple attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>Index lock: index not locked</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>index locked</td>
<td></td>
</tr>
</tbody>
</table>

Table 106: Response area for information functions 1, 2, 3, 4, 5, 6, and 7 (part 1 of 2)
### DML statements for old data types

#### Attribute information

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
<th>Info. function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>Search strategy:</td>
<td>4, 5, 6, 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– search via primary data</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– search via index</td>
<td></td>
</tr>
<tr>
<td>Bit 2⁷</td>
<td>0</td>
<td>1</td>
<td>Null as attribute value:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>– not permitted</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>– permitted</td>
<td></td>
</tr>
<tr>
<td>Bit 2⁶</td>
<td>1</td>
<td></td>
<td>Result of calculation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td>– no</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>– yes, possible</td>
<td></td>
</tr>
<tr>
<td>Bit 2⁵</td>
<td>1</td>
<td></td>
<td>Index information:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td>– Index non-existent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>– Index available</td>
<td></td>
</tr>
<tr>
<td>Bit 2⁴</td>
<td>00</td>
<td></td>
<td>Storage format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01</td>
<td></td>
<td>– binary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>– packed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td>– character without filler bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– character with filler bytes</td>
<td></td>
</tr>
<tr>
<td>Bit 2³, ²²</td>
<td>X’01´ to X´0F´</td>
<td>1</td>
<td>No. of occurrences of a multiple attribute as defined in attribute catalog:</td>
<td>4, 5, 6, 7</td>
</tr>
<tr>
<td></td>
<td>X’FF´</td>
<td></td>
<td>1 to 255 occurrences</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X’00´</td>
<td></td>
<td>Not a multiple attribute</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>[...]</td>
<td>Responses for attributes 2 to n</td>
<td>1 to 7</td>
</tr>
</tbody>
</table>

Table 106: Response area for information functions 1, 2, 3, 4, 5, 6, and 7 (part 2 of 2)

### Table 107: Response area for information function A (part 1 of 2)

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>san</td>
<td>Symbolic attribute name</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>van</td>
<td>Verbal attribute name</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>X´0001´ to X´0100´</td>
<td>Attribute length: 1 to 256 bytes</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>X´00´ to X´0F´</td>
<td>Number of decimal places: 0 to 15 decimal places</td>
</tr>
</tbody>
</table>

Table 107: Response area for information function A (part 1 of 2)
<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>1</td>
<td>X′11´</td>
<td>Data type: CHAR or corresponding old, interpretable data type for INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′21´</td>
<td>NUMERIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′22´</td>
<td>DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′24´</td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′28´</td>
<td>SMALLINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′00´</td>
<td>Uninterpretable old data type</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>X′00´</td>
<td>Index information: Index non-existent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′02´</td>
<td>Index locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′04´</td>
<td>Index available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′08´</td>
<td>Index required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′10´</td>
<td>Old data type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X′20´</td>
<td>Null not permitted as attribute value (old data type)</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>i-length</td>
<td>Index length (binary)</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>X´01´ to X´FF´</td>
<td>Number of occurrences of a multiple attribute as defined in attribute catalog: 1 to 255 occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>Not a multiple attribute</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>dfc</td>
<td>Default value character</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>X´80´</td>
<td>Compound key details: Compound key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´40´</td>
<td>Compound key attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X´00´</td>
<td>Not a compound key</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>ck-displ</td>
<td>Displacement of a compound key attribute from the start of the compound key</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>[...]</td>
</tr>
</tbody>
</table>

Table 107: Response area for information function A (part 2 of 2)
9 Appendix

In this chapter you will find
– special statements
– the table of operation codes
– examples of database accesses
9.1 Special statements

The following statements can be entered in the application program:

- **NAM**: Selection of the Database Handler
- **NOTYPE**: Suppress messages to SYSOUT
- **UNT**: Search interrupt is indicated to the application program.
- **NOUNT**: Search interrupt is not indicated to the application program.
- **TRACE**: Activate/deactivate trace

The parameters NAM, NOTYPE and TRACE are normally specified in the configuration file (see the “Core Manual”). The configuration must be assigned with the link name SESCONF or by means of the CONNECT-SESAM-CONFIGURATION command.

The following statements meet with differing responses in the independent DBH and linked-in DBH:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Independent DBH</th>
<th>Linked-in DBH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Response</td>
<td>Response</td>
</tr>
<tr>
<td></td>
<td>Reply in acknowledged area</td>
<td>Reply in acknowledged area</td>
</tr>
<tr>
<td>NAM</td>
<td>Evaluated</td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Status 00MOD.£</td>
<td>Status 00LINK</td>
</tr>
<tr>
<td>NOTYPE</td>
<td>Evaluated</td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Status 00</td>
<td>Status 00LINK</td>
</tr>
</tbody>
</table>

Table 108: Differing responses in the two SESAM/SQL variants

The user program may contain any one of these statements, and knows from the reply in the acknowledgment area which DBH variant is involved. It can (and should) be fully tested on the independent DBH before it is linked via SESLINK to the linked-in application.
9.1.1 NAM statement

The NAM statement is used to select the DBH with which the application program is to operate.

The format of the NAM statement is different for TIAM and DCAM operation. The NAM statement must be given before the first statement that results in contact with a DBH. Otherwise, SESAM/SQL returns the status code 90.

Contents of transfer areas:
Statement area: The application program supplies the statement.
Acknowledgment area: The DBH returns the acknowledgment to the statement.
The inquiry and response areas are not used.

Statement area

<table>
<thead>
<tr>
<th>Pass-word</th>
<th>Op. code</th>
<th>DBH name</th>
<th>Buffer size</th>
<th>Start indic.</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/4</td>
<td>7/1</td>
<td>8/5</td>
<td>13/5</td>
<td>18/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19/5</td>
<td>-/1</td>
</tr>
<tr>
<td>pas</td>
<td>NAM= x</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>;PUF=</td>
<td>buff-lgth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>;</td>
<td>START</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key

<table>
<thead>
<tr>
<th>DBH name</th>
<th>x</th>
<th>Name of the SESAM/SQL DBH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer size</td>
<td>buff-lgth</td>
<td>DCAM: maximum size of send and receive buffer</td>
</tr>
<tr>
<td>Start indic.</td>
<td>START</td>
<td>DCAM: delete all reserved resources</td>
</tr>
</tbody>
</table>
Password (0/3)

pas Password for a protected CALL DML table,
any three-character string for an unprotected CALL DML table.

Operation code (3/4)

NAM Selects the DBH with which the application program is going to operate.

DBH name (7/1)

x DBH name of the SESAM/SQL DBH with which the application program is going to operate.
The communication name is also called the NAM identifier.

End identifier (-/1)

9 Indicates the end of the statement
;
End of statement. The statement is chained to a following ETA, RTA or BTA statement.

Only for DCAM application programs:

Buffer size (13/5)

buff-lgth Defines maximum size of send and receive buffers for a DCAM application.
The send and receive buffers are required by the communication modules for communication with the SESAM/SQL DBH.
Maximum value: 32000
Minimum value: Maximum length of response and inquiry areas as defined in the open statement.
Default value: 4096

Start indicator (19/5)

START The start indicator refers to the DCAM application whose identification is passed in bytes 0 to 15 of the identification area in the NAM statement. The start statement causes the communication module and the SESAM/SQL DBH to delete all resources used by the DCAM application. The start indicator must be given in the primary task each time a cold start is performed for a DCAM application. It must not be given in the secondary task, otherwise it will cause an unauthorized release of all resources reserved for the DCAM application.
### Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{00 \ 90}</td>
<td>Status code</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>{MOD_\ LINK}</td>
<td>reported by</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– SESMOD,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– SESLINK</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 109: Acknowledgment area
9.1.2 NOTYPE statement

The NOTYPE statement causes messages to be suppressed that would otherwise be output to SYSOUT by the connection module SESMOD.

For DCAM application programs, values must be placed in the identification area (see section “DCAM” on page 271).

Contents of transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The DBH returns the acknowledgment to the statement.

The inquiry and response areas are not used, but must be made available.

Statement area

<table>
<thead>
<tr>
<th>Password (0/3)</th>
<th>Op. Code (3/6)</th>
<th>End Identifier (9/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pas</td>
<td>NOTYPE</td>
<td>9</td>
</tr>
</tbody>
</table>

Password (0/3)

pas Password for a protected CALL DML table, any three-character string for an unprotected CALL DML table.

Operation code (3/6)

NOTYPE SYSOUT messages from SESMOD are suppressed.

End identifier (9/1)

9 Indicates the end of the statement
Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0      | 2      | 00
      |         | 90      | Status code                  |
| 2      | 4      | MOD,
      |         | LINK    | reported by                  |
|        |        |         | – SESMOD,                     |
|        |        |         | – SESLINK                      |
| 6      | 10     | -       | -                            |

Table 110: Acknowledgment area
9.1.3 UNT/NOUNT statement

With the RETRIEVAL-CONTROL option, a search can be interrupted after a specified number of accesses.

The UNT statement causes the search interrupt to be indicated to the application program with status code 16.
The application program can continue the search with a follow-up statement, or terminate it.

The NOUNT statement (default function) cancels the effect of the UNT statement, i.e. status code 16 is not passed to the application program.

Contents of transfer areas:
Statement area: The application program supplies the statement.
Acknowledgment area: The DBH returns the acknowledgment to the statement.
The inquiry and response areas are not used, but must be made available.

**Statement area**

<table>
<thead>
<tr>
<th>Password</th>
<th>Operation code</th>
<th>End id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø/3</td>
<td>3/-</td>
<td>-/1</td>
</tr>
<tr>
<td>pas</td>
<td>UNT</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>NOUNT</td>
<td></td>
</tr>
</tbody>
</table>

**Password (0/3)**

pas Any three-character string

**Operation code (3/-)**

UNT If a search is interrupted, status code 16 is passed to the application program.

NOUNT If a search is interrupted, status code 16 is *not* passed to the application program.
End identifier (/1)

9 Indicates the end of the statement

If no UNT statement is given, NOUNT applies by default.

Acknowledgment area

<table>
<thead>
<tr>
<th>Displ.</th>
<th>Length</th>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>{00} {90}</td>
<td>Status code</td>
</tr>
</tbody>
</table>
| 2      | 4      | \{MOD, \}
\{LINK \} | reported by
– SESMOD,
– SESLINK |
| 6      | 10     | -      | -             |

Table 111: Acknowledgment area
9.1.4 TRACE statement

The TRACE statement can be used in the current session to activate and deactivate the trace.

– The statements passed to SESAM/SQL and the corresponding responses are logged, or
– the message transferred from the user task to the SESAM/SQL DBH task and the corresponding response are logged.

Contents of the transfer areas:

Statement area: The application program supplies the statement.

Acknowledgment area: The DBH returns the acknowledgment to the statement.

The inquiry and response areas are not used, but must be made available.

Statement area

<table>
<thead>
<tr>
<th>Password</th>
<th>Operations code</th>
<th>Type of trace</th>
<th>Output medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/3</td>
<td>3/-</td>
<td>-/-</td>
<td>-/-</td>
</tr>
<tr>
<td>adm</td>
<td>TRACE=OFF</td>
<td>TYPE=CALL</td>
<td>OUTPUT=SYSLST</td>
</tr>
<tr>
<td></td>
<td>TRACE=ON</td>
<td>TYPE=MSG</td>
<td>OUTPUT=SYSOUT</td>
</tr>
<tr>
<td></td>
<td>TYPE=(CALL,MSG)</td>
<td></td>
<td>OUTPUT=(SYSOUT,SYSLST)</td>
</tr>
</tbody>
</table>

Password (0/3)
adm Administrator password; this password must be the password defined for the DBH option ADMINISTRATOR when the DBH was started.

Operation code (3/-)
TRACE=ON activates the trace function
TRACE=OFF deactivates the trace function

The other parameters can be omitted for OFF. The trace function is deactivated.
Type of trace (-/-)

TYPE=CALL statements used in the CALL-DML interface are logged.
TYPE=MSG messages exchanged between the user task and the SESAM/SQL DBH are logged.

Output medium (-/-)

OUTPUT=SYSOUT
logging is performed to SYSOUT

OUTPUT=SYSLST
logging is performed to SYSLST

Default value is SYSLST.
## 9.2 Table of operation codes

<table>
<thead>
<tr>
<th>Op. code</th>
<th>DML statement</th>
<th>Function</th>
<th>see section</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td>Administration statement for the DBH</td>
<td>Issue administration commands to the SESAM/SQL DBH</td>
<td>3.20 on page 172</td>
</tr>
<tr>
<td>020</td>
<td>Administration statement for SESDCN</td>
<td>Issue administration commands to SESAM/SQL-DCN</td>
<td>3.21 on page 174</td>
</tr>
<tr>
<td>1</td>
<td>Cursor file handling</td>
<td>Process cursor files, e.g. after a search</td>
<td>3.12 on page 125</td>
</tr>
<tr>
<td>2</td>
<td>– Open</td>
<td>Open a logical file</td>
<td>3.2 on page 29</td>
</tr>
<tr>
<td></td>
<td>– Administrator open</td>
<td>Open communication between the administrator program and the DBH</td>
<td>3.19 on page 169</td>
</tr>
<tr>
<td>3</td>
<td>Attribute information</td>
<td>Inquire on the attribute definitions of a table; attribute information for tables with old data formats</td>
<td>3.17 on page 160</td>
</tr>
<tr>
<td>4</td>
<td>Record output</td>
<td>Define a logical input file by projection and selection; output the significant attribute values, optional output of the attribute definitions</td>
<td>3.9 on page 96</td>
</tr>
<tr>
<td>5</td>
<td>Inquiry</td>
<td>Define a logical input file by projection and selection; output the significant and null attribute values, optional output of the attribute definitions</td>
<td>3.10 on page 107</td>
</tr>
<tr>
<td>6</td>
<td>– Search</td>
<td>Define a logical input file by projection and selection</td>
<td>3.4 on page 37</td>
</tr>
<tr>
<td></td>
<td>– Search with join</td>
<td>Define a logical input file by projection, selection and join</td>
<td>3.5 on page 63</td>
</tr>
<tr>
<td></td>
<td>– Restrict a join cursor file</td>
<td>Select responses after a search with join that created a cursor file</td>
<td>3.6 on page 76</td>
</tr>
<tr>
<td></td>
<td>– Define comparison values</td>
<td>Replace the default mask character/string identifier for a masked or string search</td>
<td>3.8 on page 92</td>
</tr>
<tr>
<td>60BI</td>
<td>Index browsing</td>
<td>Determine frequency of attribute values</td>
<td>3.7 on page 85</td>
</tr>
</tbody>
</table>

Table 112: Operation codes (part 1 of 2)
<table>
<thead>
<tr>
<th>Op. code</th>
<th>DML statement</th>
<th>Function</th>
<th>see section</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>- Response polling</td>
<td>Process a logical input file after a search, record output or inquiry</td>
<td>3.11 on page 117</td>
</tr>
<tr>
<td></td>
<td>- Follow-up update (direct update)</td>
<td>Process a logical output file after a direct update</td>
<td>3.16 on page 154</td>
</tr>
<tr>
<td>8</td>
<td>Close</td>
<td>Close a logical file</td>
<td>3.3 on page 34</td>
</tr>
<tr>
<td>9</td>
<td>- Addition (direct update)</td>
<td>Add records</td>
<td>3.13 on page 131</td>
</tr>
<tr>
<td></td>
<td>- Update (direct update)</td>
<td>Insert, update or delete attribute values or occurrences of a multiple attribute</td>
<td>3.14 on page 141</td>
</tr>
<tr>
<td></td>
<td>- Deletion (direct update)</td>
<td>Delete records</td>
<td>3.15 on page 150</td>
</tr>
<tr>
<td>90B</td>
<td>Begin transaction</td>
<td>Initiate a transaction</td>
<td>3.18 on page 166</td>
</tr>
<tr>
<td>90C</td>
<td>End transaction</td>
<td>Terminate a transaction</td>
<td>3.18 on page 166</td>
</tr>
<tr>
<td>90R</td>
<td>Reset transaction</td>
<td>Reset a transaction</td>
<td>3.18 on page 166</td>
</tr>
<tr>
<td>90F</td>
<td>Set the deletion identifier</td>
<td>Define the deletion identifier for deleting an attribute value (for old data formats only);</td>
<td>8.4 on page 329</td>
</tr>
<tr>
<td>90E</td>
<td>Delete the current deletion identifier</td>
<td>Release the deletion identifier (for old data formats only);</td>
<td>8.4 on page 329</td>
</tr>
</tbody>
</table>

Table 112: Operation codes (part 2 of 2)
9.3 Examples of database accesses

The following section contains examples of database accesses by means of application programs:

– Assembler program
– COBOL program
– FORTRAN program
– PL/I program
Example of an Assembler program

PROG1    START 0
BALR  3,0
USING *,3,4
LA    4,1
LA    4,4095(4,3)
B     ANF

* THIS PROGRAM OPERATES WITH THE INDEPENDENT DBH SESAMC.(NAM=C)
* IT OPENS THE FOLLOWING LOGICAL FILES:
* LOGICAL FILE A1 = DB FOR DIRECT UPDATES AND SEARCHES
* LOGICAL FILE B1 = DB FOR SEARCH WITH JOIN 1
* LOGICAL FILE C1 = DB FOR SEARCH WITH JOIN 2
*
* THIS ASSEMBLER PROGRAM PERFORMS THESE FUNCTIONS:
* - SELECTS THE REQUIRED DIRECT UPDATES (ADDITION, UPDATE, DELETION),
* OR SEARCHES, ALL WITH TRANSACTION-ORIENTED SECURITY
* - CALLS THE APPROPRIATE INQUIRY AREA
* (THE USER READS IN THE CURRENT VALUES FROM THE TERMINAL INTERACTIVELY)
* - OUTPUTS THE RESPONSES AT THE TERMINAL
*
* IN THE EVENT OF ERRORS STATUS CODE NOT '00', THE LOGICAL FILE IS
* CLOSED. PRIOR TO THIS THE STATUS CODE IS DISPLAYED AT THE TERMINAL.
***********************************************************************
* A.RABOESE                                                        *
***********************************************************************
*
* STATEMENT AREA FOR OPEN
*
LANWOP   DC    H'39'
DC    X'4040'
ANWOP    DC    C'XXX'                         PASSWORD
DC    C'2'                           OP CODE FOR OPEN
DBNAME   DC    CL17' '                        SPACES FOR DB NAME
DC    C'01000'                       MAXIMUM LENGTH OF
*                                             RESPONSE AREA
DC    C'01000'                       MAXIMUM LENGTH OF
*                                             INQUIRY AREA
DC    C'X'                           READ/WRITE ALLOWED FOR
*                                             ALL USERS
KZOP     DS    CL2                            FILE IDENTIFIER
DC    C'9'                           END OF STATEMENT
*
* STATEMENT AREA FOR NAM
*
LANWNAM  DC    H'13'
DC    X'4040'
ANWNAM  DC  C'XXXNAM=C9'

SELECT DBH SESAM

* ACKNOWLEDGMENT AREA FOR ALL OPERATIONS
*
QUIT    DS  OCL16
STATUS   DS  CL2
DS  CL4
LOGKZ   DS  CL2
DS  CL4

* STATEMENT AREA FOR CLOSE WITH TRANSACTION
*
LANWCLO DC  H'33'
DC  X'4040'
ANWCLO DC  C'XXX90C;XXX8 9'

* STATEMENT AREA FOR SEARCH
*
LANWSUC DC  H'35'
DC  X'4040'
ANWSUC DC  C'XXX90B;XXX601EAC7AB9000UAB95019'

* RESPONSE AREA FOR SEARCH
*
ANTWSUC DC  CL12' '

* INQUIRY AREA FOR SEARCH
*
LFRASUC DC  H'10'
DC  X'4040'
FRASUC DC  C'K23450'

* STATEMENT AREA FOR RESPONSE POLLING
*
LANWANT DC  H'10'
DC  X'4040'
ANWANT DC  C'XXX799'

* STATEMENT AREA FOR SEARCH WITH JOIN 1
*
LAWJOIN1 DC  H'80'
DC  X'4040'
ANWJOIN1 DC  C'XXX90B;XXX601#A1EAB9AC7000UAB9501UAC7501V(AAB#A1=''
DC  C'AAB#B1)XXX601#B1EAAACABB0009'

* RESPONSE AREA FOR SEARCH WITH JOIN 1
*
ANTJOIN1 DC  CL32' '
* INQUIRY AREA FOR SEARCH WITH JOIN 1
* LFRJOIN1 DC H'16'
   DC X'4040'
FRAJOIN1 DC C'K23450840401'
* STATEMENT AREA FOR SEARCH WITH JOIN 2
* LAWJOIN2 DC H'73'
   DC X'4040'
ANWJOIN2 DC C'XXX90B;XXX601#A1EAB9000UAC7501V(AB9#A1=AAA#C1)'
   DC C'XXX601#C1EAD2AGVAFX0009'
* RESPONSE AREA FOR SEARCH WITH JOIN 2
* ANTJOIN2 DC CL50' '
* INQUIRY AREA FOR SEARCH WITH JOIN 2
* LFRJOIN2 DC H'10'
   DC X'4040'
FRAJOIN2 DC C'840401'
* STATEMENT AREA FOR DIRECT UPDATING - ADDITION1
* LAWDIRN1 DC H'35'
   DC X'4040'
ANWDIRN1 DC C'XXX90B;XXX9CXNAAB#AB90AC70AAC09'
* RESPONSE AREA FOR ADDITION1
* ANTDIRN1 DC CL50' '
* INQUIRY AREA FOR ADDITION1
* LFRDIRN1 DC H'26'
   DC X'4040'
FRADIRN1 DC CL22' '
* STATEMENT AREA FOR DIRECT UPDATING - ADDITION2
* LAWDIRN2 DC H'39'
   DC X'4040'
ANWDIRN2 DC C'XXX90B;XXX9CXNAAB0AAC0ABB0&&BLN0039'
* RESPONSE AREA FOR ADDITION2
*
ANTDIRN2 DC CL50' '
*   * INQUIRY AREA FOR ADDITION2
*   
LFRDIN2 DC H'46'
   DC X'4040'
FRADIRN2 DC CL42' '
*   * STATEMENT AREA FOR DIRECT UPDATING – UPDATE
*   
LANWDIRA DC H'31'
   DC X'4040'
ANWDIRA DC C'XXX90B;XXX9CXAAAB0AAC0ABB09'
*   * RESPONSE AREA FOR UPDATE
*   
ANTDIRA DC CL50' '
*   * INQUIRY AREA FOR UPDATE
*   
LFRADIRA DC H'18'
   DC X'4040'
FRADIRA DC CL14' '
*   * STATEMENT AREA FOR DIRECT UPDATING – DELETION1
*   
LAWDIRL1 DC H'35'
   DC X'4040'
ANWDIRL1 DC C'XXX90B;XXX9CXLAABLAACL&BLN0029'
*   * RESPONSE AREA FOR DELETION1
*   
ANTDIRL1 DC CL50' '
*   * INQUIRY AREA FOR DELETION1
*   
LFRDIRL1 DC H'14'
   DC X'4040'
FRADIRL1 DC CL10' '
*   * STATEMENT AREA FOR DIRECT UPDATING – DELETION2
*   
LAWDIRL2 DC H'35'
   DC X'4040'
ANWDIRL2 DC C'XXX90B;XXX9CXLAABLACL&BLN9999'
*   * RESPONSE AREA FOR DELETION2
*
ANTDIRL2 DC CL50' '  
*  
* INQUIRY AREA FOR DELETION2  
*  
LFRDIRL2 DC H'26'  
DC X'4040'  
FRADIRL2 DC CL22' '  
*  
* AUXILIARY FIELDS, SWITCHES, PARAMAREAS  
*  
TERMOUT DC H'128'  
DC X'404040'  
ASATZ DS CL30  
DIRANTW1 DC CL12' '  
DIRANTW2 DC CL32' '  
DIRANTW3 DC CL49' '  
*  
TERMFEHL DC H'42'  
DC X'404040'  
STAT DS CL2  
DC C' '  
AUSG DS CL20  
SICHER DS 4F  
SAFE1 DS F  
SAFE2 DS F  
*  
MKARTE1 DC H'77'  
DC X'404040'  
ASATZ DS CL20  
DC C'DIREKTAENDERUNG: '  
DC C'NEUAUFNAHME =N/ '  
DC C'AENDERUNG =A/ '  
DC C'LOESCHUNG =L/ '  
DC C'SUCHFRAGE =S: '  
*  
MKARTE2 DC H'143'  
DC X'404040'  
DC C'WELCHE SUCHFRAGE ? '  
DC C'UEBER KDNR ZUM AUFTRAG (1) '  
DC C'UEBER KDNR UND DATUM ZUM ARTIKEL UND ZUR MENGE (2) '  
DC C'UEBER DATUM ZUR ANSCHRIFT DES KUNDEN (3) '  
*  
FNEU DC H'32'  
DC X'404040'  
FNEU1 DC H'19'  
DC X'404040'  
DC C'KUNDENNUMMER: '
FNEU2    DC    H'12'
        DC    X'404040'
        DC    C'DATUM: '
*
FNEU3    DC    H'32'                        ADDITION2 INQUIRY AREA
        DC    X'404040'
        DC    C'FRAGEBEREICH NEUAUFNAHME2: '
FNEU4    DC    H'22'
        DC    X'404040'
        DC    C'ARTIKELNUMMER 1: '
FNEU5    DC    H'14'
        DC    X'404040'
        DC    C'MENGE 1: '
FNEU6    DC    H'22'
        DC    X'404040'
        DC    C'ARTIKELNUMMER 2: '
FNEU7    DC    H'14'
        DC    X'404040'
        DC    C'MENGE 2: '
FNEU8    DC    H'22'
        DC    X'404040'
        DC    C'ARTIKELNUMMER 3: '
FNEU9    DC    H'14'
        DC    X'404040'
        DC    C'MENGE 3: '
*
FAEND    DC    H'29'                        UPDATE INQUIRY AREA
        DC    X'404040'
        DC    C'FRAGEBEREICH AENDERUNG: '
FAEND1   DC    H'21'
        DC    X'404040'
        DC    C'AUFTRAGSNUMMER: '
FAEND2   DC    H'20'
        DC    X'404040'
        DC    C'ARTIKELNUMMER: '
FAEND3   DC    H'12'
        DC    X'404040'
        DC    C'MENGE: '
*
FLOESCH  DC    H'30'                       DELETION1 INQUIRY AREA
        DC    X'404040'
        DC    C'FRAGEBEREICH LOESCHUNG1: '
FLOESCH1 DC    H'21'
        DC    X'404040'
        DC    C'AUFTRAGSNUMMER: '
FLOESCH2 DC    H'20'
        DC    X'404040'
        DC    C'ARTIKELNUMMER: '
* FLOESCH3 DC $'30' DELETION2 INQUIRY AREA
   DC $'404040' FUNCTION CALLS
   DC C'FRAGEBEREICH LOESCHUNG2: '
FLOESCH4 DC $'21'
   DC $'404040'
   DC C'AUFTRAGSNUMMER: '
*
FSUCH DC $'50'
   DC $'404040'
   DC C'WANN HAT KUNDE K23450 EINEN AUFTRAG GEGEBEN ?'
*
FJOIN1 DC $'80'
   DC $'404040'
   DC C'WELCHE ARTIKEL UND WELCHE MENGE HAT KUNDE K23450 '
   DC C'AM DATUM 840401 BESTELLT ?'
*
FJOIN2 DC $'73'
   DC $'404040'
   DC C'WELCHE ANSCHRIFT (NAME,WOHNORT,STRASSE) HAT KUNDE '
   DC C'MIT DATUM 840401 ?'
*
FBNEU1 DS CL10
FBNEU2 DS CL10
FBNEU4 DS CL10
FBNEU5 DS CL8
FBNEU6 DS CL10
FBNEU7 DS CL8
FBNEU8 DS CL10
FBNEU9 DS CL8
FBAEND1 DS CL8
FBAEND2 DS CL10
FBAEND3 DS CL8
FBLOE1 DS CL8
FBLOE2 DS CL10
FBLOE4 DS CL8
BUCHST1 DS CL5
BUCHST2 DS CL5
SATZ DS CL30
ANTDIR DC CL50''
ANTDIR1 DC CL12''
ANTDIR2 DC CL32''
ANTDIR3 DC CL49''
*
* ADDRESS AREAS FOR ALL OPERATIONS
*
POpen DC A(ANWOP)
   DC A(QUIT)
Assembler program

```
DC    A(ANTWSUC)
DC    A(FRASUC)
*
PNAM  DC    A(ANWNAM)
DC    A(QUIT)
DC    A(ANTWSUC)
DC    A(FRASUC)
*
PCLOSE DC    A(ANWCL0)
DC    A(QUIT)
DC    A(ANTWSUC)
DC    A(FRASUC)
*
PANT   DC    A(ANWANT)
DC    A(QUIT)
DC    A(ANTJOIN2)
DC    A(FRAJOIN2)
*
PSUCHEN DC    A(ANWSUC)
DC    A(QUIT)
DC    A(ANTWSUC)
DC    A(FRASUC)
*
PJOIN1  DC    A(ANWJOIN1)
DC    A(QUIT)
DC    A(ANTJOIN1)
DC    A(FRAJOIN1)
*
PJOIN2  DC    A(ANWJOIN2)
DC    A(QUIT)
DC    A(ANTJOIN2)
DC    A(FRAJOIN2)
*
PDIRN1  DC    A(ANWDIRN1)
DC    A(QUIT)
DC    A(ANTDIRN1)
DC    A(FRADIRN1)
*
PDIRN2  DC    A(ANWDIRN2)
DC    A(QUIT)
DC    A(ANTDIRN2)
DC    A(FRADIRN2)
*
PDIRA   DC    A(ANWDIRA)
DC    A(QUIT)
DC    A(ANTDIRA)
DC    A(FRADIRA)
*
```
Appendix

Assembler program

PDIRL1 DC A(ANWDIRL1)
DC A(QUIT)
DC A(ANTDIRL1)
DC A(FRADIRL1)
*
PDIRL2 DC A(ANWDIRL2)
DC A(QUIT)
DC A(ANTDIRL2)
DC A(FRADIRL2)
*
*
ANF EQU *
COMMUNICATION NAME FOR SESMOD

START OF ASSEMBLER PROGRAM

STM 14,1,SICHER
LA 1,PNAM
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
CLC STATUS,=C'00'
BNE ENDE

OPEN LOGICAL FILES A1,B1 AND C1

MVC DBNAME,=C'VERTRIEB'
MVC KZOP,=C'A1'
MVC LOGKZ,=C'A1'
BAL 5,SESOP
CLC STATUS,=C'00'
BNE ENDE

MVC DBNAME,=C'VERTRIEB'
MVC KZOP,=C'B1'
MVC LOGKZ,=C'B1'
BAL 5,SESOP
CLC STATUS,=C'00'
BNE ENDE

MVC DBNAME,=C'FIRMA'
MVC KZOP,=C'C1'
MVC LOGKZ,=C'C1'
BAL 5,SESOP
CLC STATUS,=C'00'
BNE ENDE

MENU SELECTION1

WRTRD MKARTE1,,BUCHST1,,,FEHL
CLI BUCHST1+4,C'N'
BE NEUAUF1
CLI BUCHST1+4,C'A'
BE AENDERN
CLI BUCHST1+4,C'L'
BE LOESCHE1
CLI BUCHST1+4,C'S'
Assembler program

BNE FEHL

* MENU SELECTION2
WRTRD MKARTE2,,BUCHST2...FEHL
CLI BUCHST2+4,C'1'
BE SUCHEN
CLI BUCHST2+4,C'2'
BE JOIN1
CLI BUCHST2+4,C'3'
BE JOIN2
B FEHL

* NEUAUF1 EQU * DIRECT UPDATE SUBPROGRAM
WROUT FNEU,FEHL ADDITION
WRTRD FNEU1,,FBNEU1...FEHL
MVC FRADIRN1.=C'bbbb'
MVC FRADIRN1+4(L'FRADIRN1-4),FBNEU1+4
WRTRD FNEU2,,FBNEU2...FEHL
MVC FRADIRN1+10(L'FRADIRN1-10),FBNEU2+4
MVC LOGKZ.=C'A1'
STM 14,1,SICHER
LA 1,PDIRN1
L 15.=V(SESAM)
BALR 14,15
LM 14,1,SICHER
CLC STATUS.=C'00'
BE NEUAUF2
MVC STAT,STATUS
MVC AUSG,=C'SATZNEUAUFNAHME1'
WRROUT TERMFEHL,FEHL
B END1

NEUAUF2 EQU *
MVC SATZ.=C'NEUAUFNAHME1 DURCHGEFUEHRT ! '
BAL 6,TERMINAL
WRROUT FNEU3,FEHL
WRTRD FNEU4,,FBNEU4...FEHL
MVC FRADIRN2(4),ANTDIRN1
MVC FRADIRN2+4(L'FRADIRN2-4),FBNEU4+4
WRTRD FNEU5,,FBNEU5...FEHL
MVC FRADIRN2+10(L'FRADIRN2-10),FBNEU5+4
MVC FRADIRN2+14(L'FRADIRN2-14),ANTDIRN1
WRTRD FNEU6,,FBNEU6...FEHL
MVC FRADIRN2+18(L'FRADIRN2-18),FBNEU6+4
WRTRD FNEU7,,FBNEU7...FEHL
MVC FRADIRN2+24(L'FRADIRN2-24),FBNEU7+4
MVC FRADIRN2+28(L'FRADIRN2-28),ANTDIRN1
WRTRD FNEU8,,FBNEU8...FEHL
MVC FRADIRN2+32(L'FRADIRN2-32),FBNEU8+4
WRTRD FNEU9,,FBNEU9...FEHL
Appendix

Assembler program

MVC FRADIRN2+38(L'FRADIRN2-38),FNEU9+4
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PDIN2
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
MVC SATZ,=C'SATZ AUFGENOMMEN !
CLC STATUS,=C'00'
BE END1
MVC STAT,STATUS
MVC AUSG,=C'SATZNEUAUFNAHME2'
WROUT TERMFEHL,FEHL
B END2

* AENDERN  EQU   *  DIRECT UPDATE SUBPROGRAM
WROUT FAEND,FEHL  UPDATE
WRTRD FAEND1,,FBAEND1,,,FEHL
WRTRD FAEND2,,FBAEND2,,,FEHL
MVC FRADIRA,FBAEND1+4
WRTRD FAEND3,,FBAEND3,,,FEHL
MVC FRADIRA+4(L'FRADIRA-4),FBAEND2+4
WRTRD FAEND4,,FBAEND4,,,FEHL
MVC FRADIRA+10(L'FRADIRA-10),FBAEND3+4
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PDIN1
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
MVC SATZ,=C'SATZ GEAENDERT !
CLC STATUS,=C'00'
BE END1
MVC STAT,STATUS
MVC AUSG,=C'SATZAENDERUNG'
WROUT TERMFEHL,FEHL
B END2

* LOESCHE1  EQU   *  DIRECT UPDATE SUBPROGRAM
WROUT FLOESCH,FEHL  DELETION
WRTRD FLOESCH1,,FBLOE1,,,FEHL
MVC FRADIRL1,FBLOE1+4
WRTRD FLOESCH2,,FBLOE2,,,FEHL
MVC FRADIRL1+4(L'FRADIRL1-4),FBLOE2+4
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PDIRL1
L 15,=V(SESAM)
BALR 14,15
Assembler program

LOESCHE2 EQU *
MVC SATZ,=C'SATZLOESCHUNG1 DURCHGEFUEHRT !'
BAL 6,TERMINAL
MVC FRADIRL2,FBLOE1+4
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PDIRL2
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
MVC SATZ,=C'SATZ GELOESCHT !'
CLC STATUS,=C'00'
BE END1
MVC STAT,STATUS
MVC AUSG,=C'SATZLOESCHUNG2'
WROUT TERMFEHL,FEHL
B END2

*SUCHEN EQU *
SEARCH SUBPROGRAM
WROUT FSUCH,FEHL
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PSUCHEN
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
MVC SATZ,=C'GESUCHTES DATUM: '
MVC ANTDIR1,ANTWSUC+10
CLC STATUS,=C'00'
BE ANTWORT1
MVC STAT,STATUS
MVC AUSG,=C'SATZSUche'
WROUT TERMFEHL,FEHL
B END2

*ANTWORT1 EQU *
BAL 6,TERMINAL
MVC LOGKZ,=C'A1'
STM 14,1,SICHER
LA 1,PANT
L 15,=V(SESAM)
Appendix

Assembler program

BALR 14,15
LM 14,1,SICHER
CLC STATUS,=C'00'
BNE END2
MVC SATZ,=C'
MVC ANTDIR1,ANTWSUC+10
B ANTWORT1

JOIN1 EQU * SEARCH WITH JOIN 1
WROUT FJOIN1,FEHL SUBPROGRAM
MVC LOGKZ,=C'B1'
STM 14,1,SICHER
LA 1,PJOIN1
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
MVC SATZ,=C'GESUCHTER AUFTRAG:
MVC ANTDIR2,ANTJOIN1+10
MVC ANTDIR2+16(L'ANTDIR2-16),ANTJOIN1+30
CLC STATUS,=C'00'
BE ANTWORT2
MVC STAT,STATUS
MVC AUSG,=C'SATZSUCHE MIT JOIN '
WROUT TERMFEHL,FEHL
B END2

ANTWORT2 EQU *
BAL 6,TERMINAL
MVC LOGKZ,=C'B1'
STM 14,1,SICHER
LA 1,PANT
L 15,=V(SESAM)
BALR 14,15
LM 14,1,SICHER
CLC STATUS,=C'00'
BNE END2
MVC SATZ,=C'
MVC ANTDIR2,ANTJOIN1+10
MVC ANTDIR2+16(L'ANTDIR2-16),ANTJOIN1+30
B ANTWORT2

JOIN2 EQU * SEARCH WITH JOIN 2
WROUT FJOIN2,FEHL SUBPROGRAM
MVC LOGKZ,=C'C1'
STM 14,1,SICHER
LA 1,PJOIN2
L 15,=V(SESAM)
BALR 14,15
LM  14.1,SICHER
MVC  SATZ,=C'GESUCHTER SATZ:              '
MVC  ANTDIR3,ANTJOIN2+22
CLC  STATUS,=C'00'
BE   END1
MVC  STAT,STATUS
MVC  AUSG,=C'SATZSUCHE MIT JOIN '
WROUT TERMFEHL,FEHL
B    END2

* ANTWORT3 EQU *
BAL   6,TERTMINAL
MVC   LOGKZ,=C'A1'
STM   14.1,SICHER
LA    1,PANT
L    15.=V(SESAM)
BALR  14.15
LM   14.1,SICHER
CLC   STATUS,=C'00'
BNE   END2
MVC   SATZ,=C'
MVC   ANTDIR3,ANTJOIN2+22
B    ANTWORT3

* FEHL EQU *
B    END1
END1 EQU *
BAL   6,TERTMINAL
END2 EQU *
STM   14.1,SICHER
LA    1,PCLOSE
L    15.=V(SESAM)
BALR  14.15
LM   14.1,SICHER
B    ENDE
ENDE EQU *
B    STOP
SESOP EQU *
ST    5,SAFE2
ST    6,SAFE1
STM   14.1,SICHER
LA    1,POPN
L    15.=V(SESAM)
BALR  14.15
LM   14.1,SICHER
L    5,SAFE2
BR    5
L    6,SAFE1
BR 6

* TERMINAL EQU *
ST 6,SAFE1
BAL 5,BILD
L 6,SAFE1
BR 6
B END1

BILD EQU * OUTPUT TO TERMINAL
ST 5,SAFE2
MVC ASATZ,SATZ
CLI BUCHST2+4,C'1'
BE BILD1
CLI BUCHST2+4,C'2'
BE BILD2
CLI BUCHST2+4,C'3'
BE BILD3

BILD1 MVC DIRANTW1,ANTDIR1
WROUT TERMOUT,FEHL
B BILD5

BILD2 MVC DIRANTW2,ANTDIR2
WROUT TERMOUT,FEHL
B BILD5

BILD3 MVC DIRANTW3,ANTDIR3
WROUT TERMOUT,FEHL

BILD5 L 5,SAFE2
BR 5

STOP EQU *
TERM END OF ASSEMBLER
END PROG1 PROGRAM
Example of a COBOL program

This program opens three logical files:

- logical file A1 = CALL DML table SALES for direct updates and searches
- logical file B1 = CALL DML table SALES for B-search
- logical file C1 = CALL DML table COMPANY for C-search

This program offers users facilities for processing the CALL DML table SALES to suit their own requirements in the following ways:

- adding orders
- deleting orders
- amending article quantities
- inquiring on the CALL DML table SALES by means of three searches

Each of these processes is split into a statement, inquiry, response and acknowledgment area. It is then immediately checked for validity by means of a status comparison (statu= “00”). If an error has occurred (statu≠ “00”), the complete acknowledgment area is displayed for the user at the terminal so that the error condition can be ascertained.

The program continues running until terminated by the user, i.e. the user can carry out several operations in succession.

The statements open, close and TA-security are executed automatically by the program.

An order in the CALL DML table SALES is created as follows (see B-search):

<table>
<thead>
<tr>
<th>ORDER NO.</th>
<th>CU-NO.</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated by the DBH itself when an order is added</td>
<td>Entered by the user when adding an order</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARTICL E 1:</th>
<th>QUANT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTICL E 2:</td>
<td>QUANT 2</td>
</tr>
<tr>
<td>ARTICL E 3:</td>
<td>QUANT 3</td>
</tr>
</tbody>
</table>

Thus the user is able to enter a maximum of three articles per order. If only one or two articles are required, the user responds to the redundant requests for article and quantity by pressing the DUE key without making any further entries.
All lines beginning with an asterisk are comments lines clarifying the action of the program.

ID DIVISION.
  PROGRAM-ID.    COBSES.
REMARKS.       PROGRAM FOR DATABASE PROCESSING
                 BY THE USER.
ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SPECIAL-NAMES.
    TERMINAL IS T.
    DECIMAL-POINT IS COMMA.
DATA DIVISION.
  WORKING-STORAGE SECTION.
  *
  * SWITCH FOR THE PROCESSING REQUIREMENT ENTERED BY THE
  * TERMINAL USER.
  *
  01 SCHALTER.
    02 ARBEIT       PIC X.
    02 WIEDERHOLUNG PIC X.
    02 SUCHE        PIC X.
  *
  /
  ******************************************************
  * STANDARD AREAS FOR ALL SESAM CALLS   *
  ******************************************************
  01 ANWEISUNG.
    02 ANW-LAENGE  PIC 9(04)   COMP.
    02 FILLER      PIC XX     VALUE SPACES.
    02 ANW        PIC X(76).
  01 FRAGEBEREICH.
    02 FRA-LAENGE  PIC 9(04)   COMP.
    02 FILLER      PIC XX     VALUE SPACES.
    02 FRAGE      PIC X(95).
  01 QUITTUNG.
    02 STATU      PIC XX.
    02 FILLER     PIC X(04).
    02 Q-LOG-DAT  PIC XX.
    02 FILLER     PIC X(08).
  *
  * RESPONSE AREA FOR ALL DIRECT UPDATES AND THE A-SEARCH.
  *
  01 ANTWORT.
    02 ANTW.
  03 ANT        PIC X(04).
  03 FILLER     PIC X(06).
    02 S-DATUM    PIC X(06).
    02 ANT-KNR    PIC X(06).
02 REST PIC X(132).

* RESPONSE AREA FOR THE B-SEARCH.
*
01 B-ANTWORT.
  02 B-ANT-AUFTR-NR PIC X(04).
  02 FILLER PIC X(32).
  02 B-ANT-ART PIC X(06).
  02 B-ANT-MENGE PIC X(04).
  02 FILLER PIC X(108).

* RESPONSE AREA FOR THE C-SEARCH.
*
01 C-ANTWORT.
  02 C-ANT-AUFTR-NR PIC X(04).
  02 FILLER PIC X(06).
  02 C-ANT-KNR PIC X(06).
  02 FILLER PIC X(12).
  02 C-ANT-NAME PIC X(15).
  02 C-ANT-STADT PIC X(15).
  02 C-ANT-STRASSE PIC X(15).
  02 C-ANT-PLZ PIC 9(05).
  02 FILLER PIC X(77).

/ ******************************************
* OPEN STATEMENT.
*
01 ANWOP.
  02 OP-LAENGE PIC 9(04) COMP VALUE 39.
  02 FILLER PIC XX VALUE SPACES.
  02 FILLER PIC X(04) VALUE "XXX2".
  02 TAB-NAME PIC X(17).
  02 FILLER PIC X(11) VALUE "0100001000X".
  02 OP-LOG-DAT PIC XX.
  02 FILLER PIC X VALUE "9".
  02 FILLER PIC X(41) VALUE SPACES.

* CLOSE STATEMENT (WITH TRANSACTION-ORIENTED SECURITY END) FOR ALL DIRECT UPDATES.
*
01 ANWCL.
  02 CL-LAENGE PIC 9(04) COMP VALUE 33.
  02 FILLER PIC XX VALUE SPACES.
  02 FILLER PIC X(07) VALUE "XXX90C;".
  02 FILLER PIC X(04) VALUE "XXX8".
  02 FILLER PIC X(17) VALUE SPACES.
02 FILLER PIC X VALUE "9".
02 FILLER PIC X(47) VALUE SPACES.

* CLOSE STATEMENT (WITHOUT TRANSACTION-ORIENTED SECURITY END) FOR ALL SEARCHES.
*
01 SUCH-CL.
   02 SUCH-CL-LENGE PIC 9(04) COMP VALUE 26.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(04) VALUE "XXX8".
   02 FILLER PIC X(17) VALUE SPACES.
   02 FILLER PIC X VALUE "9".
   02 FILLER PIC X(54) VALUE SPACES.
   */
   ******************************************************************
   * STATEMENT AREA FOR ADDITION (STEP 1)
   * (WITH TRANSACTION-ORIENTED SECURITY START).
   *
   01 A1-NEU.
   02 A1-NEU-LENGE PIC 9(04) COMP VALUE 35.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(07) VALUE "XXX90B;".
   02 FILLER PIC X(19) VALUE "XXX9CXNAAB#AB90AC70".
   02 FILLER PIC X(05) VALUE "AAC09".
   02 FILLER PIC X(45) VALUE SPACES.
   *
   * INQUIRY AREA FOR ADDITION (STEP 1).
   *
   01 F1-NEU.
   02 F1-NEU-LENGE PIC 9(04) COMP VALUE 26.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(04) VALUE "BBBB".
   02 K-NR PIC X(06).
   02 DATUM PIC X(06).
   02 ARTIKEL PIC X(06) VALUE SPACES.
   *
   * STATEMENT AREA FOR ADDITION (STEP 2).
   *
   01 A2-NEU.
   02 A2-NEU-LENGE PIC 9(04) COMP VALUE 31.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(19) VALUE "XXX9CXNAAB0AAC0ABB0".
   02 FILLER PIC X(08) VALUE "&BLN0039".
   02 FILLER PIC X(49) VALUE SPACES.
   *
   * INQUIRY AREA FOR ADDITION (STEP 2).
*  
01 F2-NEU.  
  02 F2-NEU-LAENGE PIC 9(04) COMP VALUE 46.  
  02 FILLER PIC XX VALUE SPACES.  
  02 AUF1-NEU PIC 9(04).  
  02 ART1-NEU PIC X(06).  
  02 M1-NEU PIC X(04).  
  02 AUF2-NEU PIC 9(04).  
  02 ART2-NEU PIC X(06).  
  02 M2-NEU PIC X(04).  
  02 AUF3-NEU PIC 9(04).  
  02 ART3-NEU PIC X(06).  
  02 M3-NEU PIC X(04).  
*  
* STATEMENT AREA FOR UPDATE.  
*  
01 A-AENDERUNG.  
  02 A-AEND-LAENGE PIC 9(04) COMP VALUE 31.  
  02 FILLER PIC XX VALUE SPACES.  
  02 FILLER PIC X(07) VALUE "XXX90B;".  
  02 FILLER PIC X(15) VALUE "XXX9CXAAAB0AAC0".  
  02 FILLER PIC X(05) VALUE "AB909".  
  02 FILLER PIC X(49) VALUE SPACES.  
*  
* INQUIRY AREA FOR UPDATE.  
*  
01 F-AENDERUNG.  
  02 F-AEND-LAENGE PIC 9(04) COMP VALUE 18.  
  02 FILLER PIC XX VALUE SPACES.  
  02 AUFTR-NR PIC X(04).  
  02 ART-NR PIC X(06).  
  02 MENGE PIC X(04).  
*  
/  
*******************************************************************************  
* STATEMENT AREA FOR DELETION (STEP 1)  
* (WITH TRANSACTION-ORIENTED SECURITY START).  
*  
01 A1-LOESCHEN.  
  02 A1-LOE-LAENGE PIC 9(04) COMP VALUE 35.  
  02 FILLER PIC XX VALUE SPACES.  
  02 FILLER PIC X(07) VALUE "XXX90B;".  
  02 FILLER PIC X(15) VALUE "XXX9CXAAABLAACL".  
  02 FILLER PIC X(09) VALUE "AB9LAC7L9".  

Appendix

COBOL program

02 FILLER PIC X(45) VALUE SPACES.

* INQUIRY AREA FOR DELETION (STEP 1).

01 F1-LOESCHEN.
  02 F1-LOE-LAENGE PIC 9(04) COMP VALUE 26.
  02 FILLER PIC XX VALUE SPACES.
  02 F1-AUFTR PIC X(04).
  02 FILLER PIC X(06) VALUE SPACES.
  02 FILLER PIC X(12) VALUE "BBBBBBBBBBBBB".

* STATEMENT AREA FOR DELETION (STEP 2)

01 A2-LOESCHEN.
  02 A2-LOE-LAENGE PIC 9(04) COMP VALUE 31.
  02 FILLER PIC XX VALUE SPACES.
  02 FILLER PIC X(19) VALUE "XXX9CXLAAABLABL".
  02 FILLER PIC X(08) VALUE "$BLN0039".
  02 FILLER PIC X(49) VALUE SPACES.

* INQUIRY AREA FOR DELETION (STEP 2).

01 F2-LOESCHEN.
  02 F2-LOE-LAENGE PIC 9(04) COMP VALUE 46.
  02 FILLER PIC XX VALUE SPACES.
  02 AUF1-LO PIC X(04).
  02 ART1-LO PIC X(06).
  02 M1-LO PIC 9(04) VALUE 0.
  02 AUF2-LO PIC X(04).
  02 ART2-LO PIC X(06).
  02 M2-LO PIC 9(04) VALUE 0.
  02 AUF3-LO PIC X(04).
  02 ART3-LO PIC X(06).
  02 M3-LO PIC 9(04) VALUE 0.

******************************************************************************

* LOOP STATEMENT FOR AUTOMATICALLY REPEATING THE
* PRECEDING DML STATEMENT (RESPONSE POLLING).

*]

01 SCHLEIFE.
  02 SCHLEIFE-LAENGE PIC 9(04) COMP VALUE 10.
  02 FILLER PIC XX VALUE SPACES.
  02 ANW-SCHLEIFE PIC X(06) VALUE "XXX799".
  02 FILLER PIC X(70) VALUE SPACES.

******************************************************************************
* STATEMENT AREA FOR A-SEARCH.
*
01 ANW-A-SUCHE.
   02 AAS-LAENGE PIC 9(04) COMP VALUE 28.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(13) VALUE "XXX601UAB9501".
   02 FILLER PIC X(11) VALUE "EAC7AB90009".
   02 FILLER PIC X(52) VALUE SPACES.
*
* INQUIRY AREA FOR A-SEARCH.
*
01 FRA-A-SUCHE.
   02 FAS-LAENGE PIC 9(04) COMP VALUE 10.
   02 FILLER PIC XX VALUE SPACES.
   02 FAS-KNR PIC X(06).
  
******************************************************************
*
STATEMENT AREA FOR B-SEARCH.
*
01 ANW-B-SUCHE.
   02 ABS-LAENGE PIC 9(04) COMP VALUE 73.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(19) VALUE "XXX601#A1EAB9AC7000".
   02 FILLER PIC X(19) VALUE "UAB9501UAC7501V(AAB".
   02 FILLER PIC X(19) VALUE ">#A1=AAB#B1)XXX601#B".
   02 FILLER PIC X(12) VALUE "1EAACABB0009".
   02 FILLER PIC X(07) VALUE SPACES.
*
* INQUIRY AREA FOR B-SEARCH.
*
01 FRA-B-SUCHE.
   02 FBS-LAENGE PIC 9(04) COMP VALUE 16.
   02 FILLER PIC XX VALUE SPACES.
   02 FBS-KUNDE PIC X(06).
   02 FBS-DATUM PIC X(06).
  
******************************************************************
*
STATEMENT AREA FOR C-SEARCH.
*
01 ANW-C-SUCHE.
   02 ACS-LAENGE PIC 9(04) COMP VALUE 69.
   02 FILLER PIC XX VALUE SPACES.
   02 FILLER PIC X(16) VALUE "XXX601#A1EAB9000".
   02 FILLER PIC X(16) VALUE "UAC7501V(AB9#A1=".
   02 FILLER PIC X(16) VALUE "#AAA#C1)XXX601#C1".
   02 FILLER PIC X(17) VALUE "EAD2AGVAFXAFY0009".
Appendix

COBOL program

```cobol
02 FILLER PIC X(11) VALUE SPACES.
*
* INQUIRY AREA FOR C-SEARCH.
*
01 FRA-C-SUCHE.
  02 FCS-LAENGE PIC 9(04) COMP VALUE 10.
  02 FILLER PIC XX VALUE SPACES.
  02 FCS-DATUM PIC X(06).
******************************************************************
*/
******************************************************************
*               M A I N   P R O G R A M                          *
*        ON AN ERROR IN "DATABASE OPEN"                          *
*        THE PROGRAM IS IMMEDIATELY TERMINATED WITH AN           *
*        APPROPRIATE ERROR MESSAGE (ACKNOWLEDGMENT AREA).        *
*                                                                *
******************************************************************
* PARAGRAPH ST93 GIVES THE USER THE OPTION OF                 *
* CONTINUING DATABASE PROCESSING OR TERMINATING                *
* THE PROGRAM.                                                 *

PROCEDURE DIVISION.
STEUER SECTION.
ST15.
  PERFORM DATENBANK-OEFFNEN.
  IF STATU NOT EQUAL '00' THEN PERFORM FEHLER
  GO TO ST95.
ST20.
  PERFORM BEARBEITUNGSART.
  IF ARBEIT = "S" THEN PERFORM SUCHFRAGE
  GO TO ST93
  ELSE IF ARBEIT = "N" THEN PERFORM NEUAUFNAHME
  ELSE IF ARBEIT = "A" THEN PERFORM AENDERN
  ELSE IF ARBEIT = "L" THEN PERFORM LOESCHEN
  ELSE DISPLAY "FEHLERHAFTE EINGABE" UPON T
  GO TO ST20.
ST90.
  PERFORM DATENBANK-SCHLIESSEN.
  IF STATU NOT EQUAL '00' THEN PERFORM FEHLER.
ST93.
  PERFORM WIEDERHOLUNGEN.
  IF WIEDERHOLUNG = "Y" THEN GO TO ST15
  ELSE IF WIEDERHOLUNG = "N" THEN GO TO ST95
  ELSE DISPLAY "FEHLERHAFTE EINGABE!" UPON T
  GO TO ST93.
ST95.
  PERFORM PROG-BEENDEN.
ST99.
STOP RUN.
```
RUECKSETZEN SECTION.
*                                                *
*      STATEMENT, INQUIRY AND RESPONSE AREA        *
*      ARE SET TO BLANKS.                         *
*      ( TO ENSURE THAT ONLY THE REQUIRED INFORMATION *
*      SUBSEQUENTLY APPEARS IN EACH AREA. )         *
*                                                *
RU10.
    MOVE SPACES TO ANW.
    MOVE SPACES TO FRAGE.
    MOVE SPACES TO ANTWORT.
RU20.
    EXIT.
******************************************************************
SESAMU SECTION.
*                                                *
*  SESAM CALL FOR ALL OPEN STATEMENTS, ALL       *
*     DIRECT UPDATES, BOTH CLOSE STATEMENTS       *
*     AND THE A-SEARCH.                          *
*                                                *
SE10.
    CALL "SESAM" USING ANW QUITTUNG ANTWORT FRAGE.
SE20.
    EXIT.
******************************************************************
SESAMB SECTION.
*                                                *
*  SESAM CALL FOR B-SEARCH.                      *
*                                                *
SB10.
    CALL "SESAM" USING ANW QUITTUNG B-ANTWORT FRAGE.
SB20.
    EXIT.
******************************************************************
SESAMC SECTION.
*                                                *
*  SESAM CALL FOR C-SEARCH.                      *
*                                                *
SC10.
    CALL "SESAM" USING ANW QUITTUNG C-ANTWORT FRAGE.
SC20.
    EXIT.
******************************************************************
DATENBANK-OEFFNEN SECTION.
*                                                *
*  ON ERROR, THE ACKNOWLEDGMENT AREA IS DISPLAYED *
*  AND THE USER IS ALSO INFORMED WHICH "OPEN" CAUSED *
* THE ERROR.
* DA10.
  PERFORM RUECKSETZEN.
  MOVE "VERTRIEB" TO TAB-NAME.
  MOVE "A1" TO OP-LOG-DAT Q-LOG-DAT.
  MOVE ANWOP TO ANWEISUNG.
  PERFORM SESAMU.
  IF STATU NOT EQUAL "00" THEN DISPLAY "VERTRIEB (1)"
  UPON T
  GO TO DA99.
DA20.
  PERFORM RUECKSETZEN.
  MOVE "VERTRIEB" TO TAB-NAME.
  MOVE "B1" TO OP-LOG-DAT Q-LOG-DAT.
  MOVE ANWOP TO ANWEISUNG.
  PERFORM SESAMU.
  IF STATU NOT EQUAL "00" THEN DISPLAY "VERTRIEB (1)"
  UPON T
  GO TO DA99.
DA30.
  PERFORM RUECKSETZEN.
  MOVE "FIRMA" TO TAB-NAME.
  MOVE "C1" TO OP-LOG-DAT Q-LOG-DAT.
  MOVE ANWOP TO ANWEISUNG.
  PERFORM SESAMU.
  IF STATU NOT EQUAL "00" THEN DISPLAY "FIRMA"
  UPON T
DA99.
  EXIT.
******************************************************************
BEARBEITUNGSART SECTION.
BE10.
  DISPLAY "WELCHE DATENBANKBEARBEITUNG WOLLEN SIE VORNEHMEN ?"
  UPON T.
  DISPLAY "(SUCHFRAGE=S,NEUAUFNAHME=N,AENDERN=A,LOESCHEN=L)"
  UPON T.
  DISPLAY "BITTE GROSSBUCHSTABEN VERWENDEN !"  UPON T.
  ACCEPT ARBEIT FROM T.
BE20.
  EXIT.
******************************************************************
WIEDERHOLUNGEN SECTION.
WI10.
  DISPLAY "IST EINE WEITERE DATENBANKBEARBEITUNG ERWUENSCHT ?" 
  "(JA=Y,NEIN=N):"  UPON T.
  DISPLAY "BITTE GROSSBUCHSTABEN VERWENDEN !"  UPON T.
  ACCEPT WIEDERHOLUNG FROM T.
WI20.
EXIT.
******************************************************************
FEHLER SECTION.
FE10.
   DISPLAY "QUITTUNGSBEREICH: " UPON T.
   DISPLAY QUITTUNG UPON T.
FE20.
EXIT.
******************************************************************
DATENBANK-SCHLIESSEN SECTION.
*                   FOR ALL DIRECT UPDATES.                      *
DS10.
   PERFORM RUECKSETZEN.
   MOVE ANWCL TO ANWEISUNG.
   PERFORM SESAMU.
DS20.
EXIT.
******************************************************************
PROG-BEENDEN SECTION.
PR10.
   DISPLAY "DAS PROGRAMM IST BEENDET. AUF WIEDERSEHEN!"
      UPON T.
PR20.
EXIT.
******************************************************************
NEUAUFNAHME SECTION.
NU10.
   PERFORM RUECKSETZEN.
NU20.
   A D D I T I O N   ( STEP 1 )
   DISPLAY "KUNDENNUMMER (6-STELLIG) : " UPON T.
   ACCEPT K-NR FROM T.
   DISPLAY "DATUM (IN DER FORM JJMMTT) : " UPON T.
   ACCEPT DATUM FROM T.
NU25.
   MOVE "A1" TO Q-LOG-DAT.
*          THE FILE IDENTIFIER IS WRITTEN    *
*    INTO THE ACKNOWLEDGMENT AREA TO    *
*          ENSURE THAT THE DBH 'SESAMI' *
*    SEARCHES THE CORRECT DATABASE    *
*    (AND THE CORRECT ATTRIBUTE CATALOG)*
*          FOR THE SPECIFIED ATTRIBUTES. *
   MOVE A1-NEU TO ANWEISUNG.
   MOVE F1-NEU TO FRAGEBEREICH.
NU30.
   PERFORM SESAMU.
   IF STATU NOT EQUAL '00' THEN PERFORM FEHLER
GO TO NU99.
NU40.
*                NEU AUFNAHME (2.SCHRITT)
MOVE ANT TO AUF1-NEU AUF2-NEU AUF3-NEU.
DISPLAY "ARTIKEL 1 (6-STELLIG) : " UPON T.
ACCEPT ART1-NEU FROM T.
DISPLAY "MENGE 1 (4STELLIG) : " UPON T.
ACCEPT M1-NEU FROM T.
DISPLAY "ARTIKEL 2 (6-STELLIG) : " UPON T.
ACCEPT ART2-NEU FROM T.
DISPLAY "MENGE 2 (4STELLIG) : " UPON T.
ACCEPT M2-NEU FROM T.
DISPLAY "ARTIKEL 3 (6-STELLIG) : " UPON T.
ACCEPT ART3-NEU FROM T.
DISPLAY "MENGE 3 (4STELLIG) : " UPON T.
ACCEPT M3-NEU FROM T.
NU50.
PERFORM RUECKSETZEN.
MOVE "A1" TO Q-LOG-DAT.
MOVE A2-NEU TO ANWEISUNG.
MOVE F2-NEU TO FRAGEBEREICH.
NU60.
PERFORM SESAMU.
IF STATU = "9D" THEN GO TO NU70
*          ERROR 9D (SUBNO.01) CAN BE IGNORED FOR AN ADDITION
*          IF THE ORDER CONTAINS LESS THAN THREE (DIFFERENT)
*          ORDER ITEMS.
ELSE IF STATU NOT EQUAL "00" THEN PERFORM FEHLER
          GO TO NU80.
NU70.
*          TELLS THE USER THAT THE ORDER HAS BEEN SUCCESSFULLY
*          ADDED.
DISPLAY "SATZ NEUAUFGENOMMEN." UPON T.
NU80.
*          BLANK LINE
DISPLAY " " UPON T.
NU99.
EXIT.
******************************************************************
AENDERN SECTION.
*          THE ORDER QUANTITY FOR THE ARTICLE
*          IS AMENDED.
*          THE ORDER NUMBER AND ARTICLE NUMBER
*          NEED TO BE SPECIFIED.
AE10.
PERFORM RUECKSETZEN.
AE20.
DISPLAY "AUFTRAGSNUMMER (4-STELLIG) : " UPON T.
ACCEPT AUFRTR-NR FROM T.
DISPLAY "ARTIKELNUMMER (6-STELLIG)" UPON T.
ACCEPT ART-NR FROM T.
DISPLAY "MENGE (4-STELLIG)" UPON T.
ACCEPT MENGE FROM T.
AE30.
   MOVE "A1" TO Q-LOG-DAT.
   MOVE A-AENDERUNG TO ANWEISUNG.
   MOVE F-AENDERUNG TO FRAGEBEREICH.
AE40.
   PERFORM SESAMU.
   IF STATU NOT EQUAL '00' THEN PERFORM FEHLER
   GO TO AE60.
AE50.
   * TELLS THE USER THAT THE UPDATE HAS BEEN SUCCESSFULLY
   * PERFORMED.
   DISPLAY "AENDERUNG DURCHGEFUEHRT." UPON T.
AE60.
   DISPLAY " " UPON T.
AE99.
   EXIT.
******************************************************************
LOESCHEN SECTION.
* IF ANY ERROR OTHER THAN "95" OR "9G" OCCURS
* THE USER IS GIVEN BOTH AN ERROR MESSAGE
* (ACKNOWLEDGMENT AREA) AND AN INDICATION OF THE
* DELETION STEP (DEL 1,DEL 2) IN WHICH IT OCCURRED*
* - 95 AND 9G ERRORS CAN BE IGNORED FOR
* A DELETION IF THE ORDER CONTAINS LESS THAN THREE (DIFFERENT) ORDER ITEMS
* *
LO10.
   PERFORM RUECKSETZEN.
LO20.
   DISPLAY "AUFTRAGSNUMMER (4-STELLIG)" UPON T.
   ACCEPT AUF1-LO FROM T.
   MOVE AUF1-LO TO AUF2-LO AUF3-LO F1-AUFTR.
LO25.
   DISPLAY "ARTIKEL 1 ? (GROSSBUCHSTaben!)" UPON T.
   ACCEPT ART1-LO FROM T.
   DISPLAY "ARTIKEL 2 ? (GROSSBUCHSTaben!)" UPON T.
   ACCEPT ART2-LO FROM T.
   DISPLAY "ARTIKEL 3 ? (GROSSBUCHSTaben!)" UPON T.
   ACCEPT ART3-LO FROM T.
LO30.
   * D E L E T I O N ( STEP 1 )
   MOVE "A1" TO Q-LOG-DAT.
   MOVE A1-LOESCHEN TO ANWEISUNG.
MOVE F1-LOESCHEN TO FRAGEBEREICH.

LO40.
    PERFORM SESAMU.
    IF STATU NOT EQUAL "00" THEN PERFORM FEHLER
    DISPLAY "LO 1" UPON T
    GO TO LO80.

LO50.
    *  D E L E T I O N  ( S T E P  2  )
    PERFORM RUECKSETZEN.
    MOVE "A1" TO Q-LOG-DAT.
    MOVE A2-LOESCHEN TO ANWEISUNG.
    MOVE F2-LOESCHEN TO FRAGEBEREICH.

LO60.
    PERFORM SESAMU.
    IF STATU = "95" OR STATU = "9G" THEN GO TO LO70
    ELSE IF STATU NOT EQUAL "00" THEN PERFORM FEHLER
    DISPLAY "LO 2" UPON T
    GO TO LO80.

LO70.
    *  T E L L S  T H E  U S E R  T H A T  T H E  O R D E R  H A S
    *  B E E N  S U C C E S S F U L L Y  D E L E T E D.
    DISPLAY "SATZ GELOESCHT!" UPON T.

LO80.
    *  B L A N K  L I N E .
    DISPLAY " " UPON T.

LO99.
    EXIT.

******************************************************************
SUCHFRAGE SECTION.
SU10.
    PERFORM SU-AUSWAHL.

SU20.
    IF    SUCHE = "A" THEN PERFORM A-SUCHE
    ELSE IF SUCHE = "B" THEN PERFORM B-SUCHE
    ELSE IF SUCHE = "C" THEN PERFORM C-SUCHE
    ELSE    DISPLAY " FEHLERHAFTE EINGABE!" UPON T
            GO TO SU10.

SU30.
    PERFORM RUECKSETZEN.
    MOVE SUCH-CL TO ANWEISUNG.

SU40.
    PERFORM SESAMU.
    IF STATU NOT EQUAL "00" THEN PERFORM FEHLER.

SU99.
    EXIT.

******************************************************************
SU-AUSWAHL SECTION.
DISPLAY "WELCHE SUCHFRAGE ?" UPON T.
DISPLAY "-WANN HAT KUNDE X EINEN AUFTRAG BESTELLT ?"
"(=A)" UPON T.
DISPLAY "-WELCHEN AUFTRAG(AUFTR-NR,ART-NR,MENGE) HAT "
"KUNDE X AM TAG X BESTELLT ?" UPON T.
DISPLAY "(=B)" UPON T.
DISPLAY "-WIE LAUTET DIE ANSCHRIFT(NAME,STADT,STRASSE)"
" DER KUNDEN,DIE AM TAG X " UPON T.
DISPLAY "-EINEN AUFTRAG BESTELLT HABEN? (=C)"
UPON T.

ACCEPT SUCHE FROM T.
SA99.
EXIT.
******************************************************************
A-SUCHE SECTION.

THE CUSTOMER NUMBER IS USED TO FIND OUT ALL DATES ON WHICH THE CUSTOMER PLACED AN ORDER.
ERROR STATUS "10" INDICATES THAT NO SUCH RECORD WAS FOUND.

DISPLAY "KUNDENNUMMER (6-STELLIG) : " UPON T.
ACCEPT FAS-KNR FROM T.

PERFORM RUECKSETZEN.
MOVE "A1" TO Q-LOG-DAT.
MOVE ANW-A-SUCHE TO ANWEISUNG.
MOVE FRA-A-SUCHE TO FRAGEBEREICH.

PERFORM SESAMU.
IF STATU = "10" THEN DISPLAY "ES LIEGT KEIN AUFTRAG MIT DIESER KUNDENUMMER VOR!" UPON T
DISPLAY " " UPON T
GO TO AC99
ELSE IF STATU NOT EQUAL "00" THEN PERFORM FEHLER GO TO AC99
ELSE DISPLAY "GESUCHTES DATUM: " S-DATUM " " ANT-KNR UPON T.

L O O P FOR OUTPUTTING FURTHER RESPONSES (DATES)
MOVE "A1" TO Q-LOG-DAT.
MOVE SCHLEIFE TO ANWEISUNG.

PERFORM SESAMU.
IF STATU = "10" THEN GO TO AC50
ELSE IF STATU = "00" THEN
   DISPLAY "GESUCHTES DATUM: " S-DATUM " " ANT-KNR
   UPON T
   GO TO AC40
ELSE PERFORM FEHLER.
AC50.
   DISPLAY " " UPON T.
AC99.
EXIT.
******************************************************************
B-SUCHE SECTION.
* * *
* THE CUSTOMER NUMBER AND DATE ARE USED TO *
* INQUIRE ON THE FULL CONTENTS OF THE ORDER *
* THE CUSTOMER PLACED ON THIS DATE. *
* *
BS10.
   DISPLAY "KUNDENNUMMER (6STELLIG): " UPON T.
   ACCEPT FBS-KUNDE FROM T.
   DISPLAY "DATUM (JJMMTT): " UPON T.
   ACCEPT FBS-DATUM FROM T.
BS20.
   PERFORM RUECKSETZEN.
   MOVE "A1" TO Q-LOG-DAT.
   MOVE ANW-B-SUCHE TO ANWEISUNG.
   MOVE FRA-B-SUCHE TO FRAGEBEREICH.
   PERFORM SESAMB.
   IF STATU = "00"
      * HEADER LINE FOR ORDER OUTPUT.
      THEN DISPLAY "KUNDENNUMMER: " FBS-KUNDE " DATUM: "
      FBS-DATUM " AUFTR.NR: " B-ANT-AUFTR-NR UPON T
      DISPLAY " " UPON T
   ELSE IF STATU = "10" THEN DISPLAY "KUNDE HAT AN DIESEM "
      "TAG KEINEN AUFTRAG GEGEBEN!" UPON T
      DISPLAY " " UPON T
   GO TO BS99
ELSE PERFORM FEHLER
   GO TO BS99.
BS30.
   * LOOP FOR OUTPUTTING THE ORDER ITEMS
   * FOR AN ORDER.
   MOVE "A1" TO Q-LOG-DAT.
   MOVE SCHLEIFE TO ANWEISUNG.
   PERFORM SESAMB.
   IF STATU = "10" THEN GO TO BS99
ELSE IF STATU = "00"
   THEN DISPLAY "ARTIKEL: " B-ANT-ART " MENGE: "
**COBOL program**

---

**Appendix**

B-ANT-MENGE UPON T
DISPLAY " " UPON T
GO TO BS30
ELSE PERFORM FEHLER.
BS99.
EXIT.

******************************************************************
C-SUCHE SECTION.
* * *
* THE DATE IS USED TO INQUIRE ON WHICH CUSTOMERS * *
* (WITH ADDRESSES) PLACED AN ORDER ON THIS DATE. * *
* * *
* * *
CS10.
DISPLAY "DATUM : " UPON T.
ACCEPT FCS-DATUM FROM T.
CS20.
PERFORM RUECKSETZEN.
MOVE "A1" TO Q-LOG-DAT.
MOVE ANW-C-SUCHE TO ANWEISUNG.
MOVE FRA-C-SUCHE TO FRAGEBEREICH.
PERFORM SESAMC.
CS30.
* H E A D E R L I N E FOR CUSTOMER LISTING.
IF STATU = "00" THEN DISPLAY "AUFTR-NR : "
"K-NR : "
"NAME : "
"STADT : "
"STRASSE : " UPON T
* C U S T O M E R 1.
DISPLAY " " C-ANT-AUFTR-NR
" " C-ANT-KNR
" " C-ANT-NAME
" " C-ANT-PLZ " " C-ANT-STADT
" " C-ANT-STRASSE
UPON T
DISPLAY " " UPON T
ELSE IF STATU = "10" THEN DISPLAY "AN DIESEM TAG WURDE "
"KEIN AUFTRAG BESTELLT ! "
UPON T
DISPLAY " " UPON T
GO TO CS99
ELSE PERFORM FEHLER
GO TO CS99.
CS40.
* L O O P FOR OUTPUTTING FURTHER RESPONSES (CUSTOMERS).
MOVE "A1" TO Q-LOG-DAT.
MOVE SCHLEIFE TO ANWEISUNG.
PERFORM SESAMC.
   IF STATU = "10" THEN GO TO CS99
ELSE IF STATU = "00" THEN DISPLAY " " C-ANT-AUFTR-NR
   " " C-ANT-KNR
   " " C-ANT-NAME
   " " C-ANT-PLZ " " C-ANT-STADT
   " " C-ANT-STRASSE
   UPON T
   DISPLAY " " UPON T
   GO TO CS40
ELSE PERFORM FEHLER.
CS99.
   EXIT.
*******************************************************************************
Example of a FORTRAN program

C.................................................................................................
C   PROGRAM SESDML
C   FORTRAN PROGRAM WITH AN INTERFACE FOR SESAM DML CALLS
C
C   TO MAINTAIN A SIMPLE USER INTERFACE, THE DML STATEMENTS TO THE
C   SESAM DBH ARE CALLED FROM A SUBROUTINE
C
IMPLICIT COMPLEX (A-Z)
CHARACTER*(80,V) ANWEISUNG /' '/
CHARACTER*(80,V) FRAGE      /' '/
CHARACTER*(80,V) ANTWORT   /' '/
CHARACTER*(16,V) QUITTUNG  /' '/
C...
INTEGER*4 SYSDTA /1/
INTEGER*4 SYSOUT /2/
INTEGER*4 MODE /0/
C...
CHARACTER*3 KENNWORT /'XXX'/
CHARACTER*2 LOGDAT /'HH'/
C...
10000 IF (MODE .LT. 5) THEN
   WRITE (SYSOUT, *) '    NAM-ANWEISUNG       : (1)   '
   WRITE (SYSOUT, *) '   OPEN-ANWEISUNG       : (2)   '
   WRITE (SYSOUT, *) '   SUCH-ANWEISUNG       : (3)   '
   WRITE (SYSOUT, *) '  CLOSE-ANWEISUNG       : (4)   '
   WRITE (SYSOUT, *) '              END       : (5)   '
C...
   READ  (SYSDTA, '(I1)') MODE
   QUITTUNG(7:8) = LOGDAT
C...
   IF (MODE .EQ. 1) THEN
      WRITE (SYSOUT, *) '  NAM-ANWEISUNG      NAM = I'
      ANWEISUNG(5:13) = KENNWORT//'NAM=I9'
   ELSE IF (MODE .EQ. 2) THEN
      ANWEISUNG(5:39) = KENNWORT//'2FIRMA       '//'0102401024X'//LOGDAT//'9'
      WRITE (SYSOUT, *) ' OPEN DATENBANK FIRMA LOG.DATEI HH'
   ELSE IF (MODE .EQ. 3) THEN
      ANWEISUNG(5:19) = KENNWORT//'601EAA80009'
      FRAGE(5:80) = ''
      WRITE (SYSOUT, *) ' SUCHEN IN DER DATENBANK FIRMA'
   ELSE IF (MODE .EQ. 4) THEN
      ANWEISUNG(5:39) = KENNWORT//'8                 9'
WRITE (SYSOUT, *) 'CLOSE LOG. DATEI HH'
END IF
C...
CALL SESSUB (ANWEISUNG, QUITTUNG, ANTWORT, FRAGE)
C...
WRITE (SYSOUT, *) 'SES-STATUS :', QUITTUNG(1:16)
C...
IF ((MODE .EQ. 3) .AND. (QUITTUNG(1:2) .EQ. '00')) THEN
   WRITE (SYSOUT, *) 'SES-ANTWORT :', ANTWORT(1:80)
END IF
C...
END IF
GOTO 10000
END
C.......................................................................
C   SUBROUTINE SESSUB (COMMAND, QUIT, ANSWER, QUESTION)
C
C   THE SESSUB SUBROUTINE PROVIDES THE LENGTH FIELDS FOR THE STATEMENT
C   AND INQUIRY AREAS.
C
C   BYTES 1-2 : HEXADECIMAL LENGTH OF VARIABLE AREA
C   BYTES 3-4 : BLANK FIELDS (X'40')
C
C   DESCRIPTORS OF CHARACTER VARIABLES ARE NOT TRANSFERRED IN A
C   SUBROUTINE CALL. THUS THE FORMAL TRANSFER PARAMETER MUST BE
C   EXPPLICITLY ASSIGNED TO ANOTHER VARIABLE IN THE SUBROUTINE.
C
SUBROUTINE SESSUB (COMMAND, QUIT, ANSWER, QUESTION)
IMPLICIT COMPLEX (A-Z)
C...
CHARACTER*(*) COMMAND
CHARACTER*(*) QUIT
CHARACTER*(*) ANSWER
CHARACTER*(*) QUESTION
C...
CHARACTER*(80) NEWCOM
CHARACTER*(80) NEWQUEST
C...
CHARACTER*4 HEXKOPF
C...
DATA HEXKOPF /Z00504040/
C...
COMMAND(1:4) = HEXKOPF
QUESTION(1:4) = HEXKOPF
NEWCOM(1:80) = COMMAND(1:80)
NEWQUEST(1:80) = NEWQUEST(1:80)
C...
CALL SESAM (NEWCOM(5:80), QUIT, ANSWER, NEWQUEST(5:80))
C...
RETURN
END
Example of a PL/I program

PLIISES: PROC OPTIONS(MAIN):
/*********************************************
* STANDARD PL/I PROGRAM FOR SESAM CALLS  *
*                                          *
* THE PROGRAM READS STATEMENT AND INQUIRY  *
* AREAS FROM THE SCREEN AND PASSES THE     *
* CALL DML STATEMENT TO THE SESAM DBH.     *
*********************************************
/*                                                                        */
DCL SESAM ENTRY OPTIONS (ASSEMBLER):
DCL SESAREA AREA(6000):
/*                                                                        */
DCL 1 ANW BASED (ZANW),       /* Statement area              */
   5 LAENGE BIN FIXED(15),
   5 LEER   CHAR(2) INIT(' '),
   5 ANWEISUNG,
   10 KENNWORT CHAR(3),
   10 OPCODE  CHAR(1),
   10 TEXT   CHAR(LANW-4):
/*                                                                        */
DCL 1 QUITTUNG,               /* Acknowledgment area         */
   5 STATUS  CHAR(2) INIT(' '),
   5 RES1    CHAR(4) INIT(' '),
   5 DATEIKZ CHAR(2) INIT(' '),
   5 RES2    CHAR(2) INIT(' '),
   5 ZUSINFO CHAR(2) INIT(' '),
   5 RES3    CHAR(4) INIT(' ');
/*                                                                        */
DCL ANTWORT CHAR(LANT) BASED (ZANT);  /* Response area              */
/*                                                                        */
DCL 1 FRA     BASED (ZFRA),     /* Inquiry area                */
   5 LAENGE BIN FIXED(15),
   5 LEER   CHAR(2) INIT(' '),
   5 FRAGE  CHAR(LFRA):
/*                                                                        */
DCL ANWEISUNGSBEREICH CHAR(2044) VAR;  /* Input buffer for sta.      */
DCL FRAGEBEREICH  CHAR(2044) VAR;  /* Input buffer for inq.      */
DCL (LANW,LANT,LFRA) BIN FIXED(15);  /* Length fields               */
DCL (ZANW,ZANT,ZFRA) POINTER INIT (NULL);  /* Pointer to AREA          */
DCL (SUBSTR,NULL,ONSOURCE) BUILTIN;
/*                                                                        */
ON CONVERSION BEGIN;          /* OPEN error handling        */
   DISPLAY ('FALSCHE EINGABE FUER MAX. ANTWORTBEREICHS-GROESSE: '••
   ONSOURCE());
   DISPLAY ('BITTE EINGABE WIEDERHOLEN ODER E(=ENDE)')
REPLY (ANWEISUNGSBEREICH);
GOTO ANF;
END;
/*
DISPLAY ('BITTE SESAM-ANWEISUNG ODER E(=ENDE) EINGEBEN: ')
REPLY (ANWEISUNGSBEREICH):
/*
ANF:
DO WHILE (ANWEISUNGSBEREICH ?= 'E');
/*
LANW = LENGTH (ANWEISUNGSBEREICH); /* Set up statement area */
ALLOCATE ANW IN (SESAREA);
ANW.LAENGE = LANW+4; /* Set statement area to */
ANW.KENNWORT = SUBSTR(ANWEISUNGSBEREICH,1,3); /* current statement */
ANW.OPCODE = SUBSTR(ANWEISUNGSBEREICH,4,1);
ANW.TEXT = SUBSTR(ANWEISUNGSBEREICH,5);
/*
IF ANW.OPCODE = '2' THEN DO; /* For OPEN statement:
LFRA = 0; /* set up inquiry area */
ALLOCATE FRA IN (SESAREA);
FRA.LAENGE = LFRA + 4;
LANT = SUBSTR(ANW.TEXT,18,5); /* Read response length */
ALLOCATE ANTWORT IN (SESAREA); /* from OPEN statement */
ANTWORT = ' '; /* Set up response area */
END;
ELSE DO; /* Other than OPEN sta.: */
DISPLAY ('BITTE SESAM-ATTRIBUTE FUER FRAGEBEREICH EINGEBEN: ')
REPLY (FRAGEBEREICH);
LFRA = LENGTH (FRAGEBEREICH); /* Determine inq. area */
ALLOCATE FRA IN (SESAREA); /* length and set up and */
FRA.LAENGE = LFRA + 4; /* put values in inq.area */
FRA.FRAGE = FRAGEBEREICH;
END;
/*
CALL SESAM (ANWEISUNG,QUITTUNG,ANTWORT,FRAGE); /* CALL DML call */
/*
IF STATUS = '00' THEN DO; /* SESAM statement o.k.: */
IF ANTWORT = '' THEN: /* Output response area */
ELSE DO;
DISPLAY (ANTWORT);
ANTWORT = ''; /* if not empty */
END;
ELSE DO; /* Status handling: */
DISPLAY ('STATUS:'QUITTUNG.STATUS); /* Display ack. area */
DISPLAY ('ZUSATZ-INFO:'QUITTUNG.ZUSINFO);
END;
/*
FREE ANW IN (SESAREA):                          /* Release statement area */
FREE FRA IN (SESAREA):                          /* Release inquiry area */
ANWEISUNGSBEREICH,FRAGEBEREICH = ' ';           /* Delete input buffers */
/*
DISPLAY ('BITTE SESAM-ANWEISUNG ODER E(=ENDE) EINGEBEN:')
   REPLY (ANWEISUNGSBEREICH);
END:                                            /* End WHILE loop */
/*
END PLIISES:
Related publications

Ordering manuals

Please apply to your local office for ordering the manuals.

**SESAM/SQL-Server** (BS2000/OSD)
Core Manual
User Guide

*Target group*
The manual is intended for all users and to anyone seeking information on SESAM/SQL.

*Contents*
The manual gives an overview of the database system. It describes the basic concepts. It is the foundation for understanding the other SESAM/SQL manuals.

**SESAM/SQL-Server** (BS2000/OSD)
User Guide

*Target group*
The manual is intended for all users who wish to process an SESAM/SQL database by means of SESAM/SQL statements.

*Contents*
The manual describes how to embed SQL statements in COBOL, and the SQL language constructs. The entire set of SQL statements is listed in an alphabetical directory.

**SESAM/SQL-Server** (BS2000/OSD)
User Guide

*Target group*
The manual is intended for all users responsible for SESAM/SQL database administration.

*Contents*
An alphabetical directory of all utility statements, i.e. statements in SQL syntax implementing the SESAM/SQL utility functions.
**Related publications**

**SESAM/SQL-Server (BS2000/OSD)**
Database Operation
User Guide

*Target group*
The manual is intended for SESAM/SQL system administrators.

*Contents*
The manual covers the options available to the system administrator for controlling and monitoring database operation.

**SESAM/SQL-Server (BS2000/OSD)**
Utility Monitor
User Guide

*Target group*
The manual is intended for SESAM/SQL-Server database and system administrators.

*Contents*
The manual describes the utility monitor. The utility monitor can be used to administer the database and the system. One aspect covered is its interactive menu interface.

**SESAM/SQL-Server (BS2000/OSD)**
Glossary and Master Index
User Guide

*Target group*
This manual is addressed to anyone who uses or wishes to find out about SESAM/SQL.

*Contents*
The manual contains all technical terms and keywords relevant for all SESAM/SQL manuals, as well as the keywords for ESQL, UTM, DRIVE and DBA.

**SESAM/SQL-Server (BS2000/OSD)**
Messages
User Guide

*Target group*
All users of SESAM/SQL.

*Contents*
All SESAM/SQL messages, sorted by message number.
Related publications

**SESAM/SQL-Server** (BS2000/OSD)
Migrating SESAM Databases and Applications to SESAM/SQL-Server
User Guide

*Target group*
Users of SESAM/SQL-Server.

*Contents*
This manual gives an overview of the new concepts and functions. Its primary subject is, however, the difference between the previous and the new SESAM/SQL version(s). It contains all the information a user may require to migrate to SESAM/SQL-Server V2.0.

**SESAM/SQL-Server** (BS2000/OSD)
Performance
User Guide

*Target group*
Experienced users of SESAM/SQL.

*Contents*
The manual covers how to recognize bottlenecks in the behavior of SESAM/SQL and how to remedy this behavior.

**ESQL-COBOL** (BS2000/OSD)
ESQL-COBOL for SESAM/SQL-Server
User Guide

*Target Group*
COBOL programmers wishing to work with SESAM/SQL databases using SQL statements.

*Contents*
The manual describes the structure of an ESQL-COBOL program, how to embed SQL in COBOL, and how to compile, link and start such a program.

**openUTM** (BS2000/OSD)
Generating and Handling Applications
User Guide

*Target group*
This manual is intended for application planners, technical programmers, administrators and users of UTM applications.

*Contents*
The manual describes the generation of UTM applications with distributed processing, the tools available with openUTM for this purpose, and the UTM objects created in the course of generation. It also contains all the information necessary for structuring, operating and monitoring a productive UTM application.
Related publications

**openUTM**
Concepts and Functions
User Guide

*Target group*
Anyone who wants information about the functionality and performance capability of openUTM.

*Contents*
The manual contains a general description of all the functions and features of openUTM, plus introductory information designed to help first-time users of openUTM.

**FHS (TRANSDATA)**
User Guide

*Target group*
Programmers

*Contents*
Program interfaces of FHS for TIAM, DCAM and UTM applications. Generation, application and management of formats.

**CRTE (BS2000/Osd)**
Common RunTime Environment
User Guide

*Target group*
This manual addresses all programmers and system administrators in a BS2000 environment.

*Contents*
It describes the common runtime environment for COBOL85, COBOL2000, C and C++ objects and for "language mixes":
- CRTE components
- ILCS program communication interface
- linkage examples
BS2000/OSD
Softbooks English

Target group
BS2000/OSD users

Contents
The CD-ROM "BS2000/OSD SoftBooks English" contains almost all of the English manuals and README files for the BS2000 system software of the latest BS2000/OSD version and also of the previous versions, including the manuals listed here. These Softbooks can also be found in the Internet on our manual server. You can browse in any of these manuals or download the entire manual.

Order number
U26175-J8-Z125-1-76

Internet address
http://manuals.mchp.siemens.de
Index

Im Stichwortverzeichnis verweisen **halbfette** Seitenzahlen auf die Hauptfundstellen von Stichwörtern und **kursive** Seitenzahlen auf Beispiele.


```plaintext
* statement
  SEDI61 289
  SEDI63 298
_A statement
  SEDI61 287
  SEDI63 296
_F statement
  SEDI61 288
  SEDI63 297
_H statement
  SEDI61 280
_Q statement
  SEDI61 289
  SEDI63 298
_T statement
  SEDI61 284
_W statement
  SEDI61 281
$ statement
  SEDI61 289
  SEDI63 298
/* statement
  SEDI61 289
  SEDI63 299

A
ABEND event 269
acknowledgment area 16
  adding new records 246
add
  attribute value 141
  occurrence 141
  record 131
  adding new records 245
  addition 131, 154
```

addition 136
administration statement 173
attribute information 162, 335
cursor file handling 129
define comparison values 94, 236
deleting records 250, 252
deletion 152
follow-up update 157, 247, 251
index browsing 89, 233, 234
inquiry 112, 243, 322
NAM 343
NOTYPE 345
record output 102, 310
response polling 121, 193, 195, 224, 226
restricting a join cursor file 80
retrieval using record output 241
search 55, 193, 194, 195, 196
search with join 71, 220
SESDCN administration 176
setting and deleting the delete character 331
transaction-oriented security 168
UNT/NOUNT 347
update 146
updating records 249

U1054-J-Z125-11-76 399
acknowledgment area 136
inquiry area 138
response area 138
statement area 132
addressing mode 262, 268
administration
by application program 172
administration call 173, 259
send 172
administration command
SESDCN administration 176
administration statement 13, 172
acknowledgment area 173
response area 173
SESDCN 174
statement area 172
administrator open 169
acknowledgment area 171
statement area 169
administrator password 175
ALGOL program
CALL DML call 24
append
occurrence 141
appendix 339
application name 273, 274
Assembler program
CALL DML call 21
attribute 43, 50, 303
index browsing 87
inverted 87
number of 46
partially inverted 91
symbolic 160, 162
transfer 284
verbal 160
attribute catalog 84, 91
attribute definition 162
attribute information 160, 277
acknowledgment area 162, 335
old data types 332
response area 164, 336
statement area 161, 333
attribute length 51, 160, 332
attribute name 335
symbolic 53, 308, 332, 335
attribute selection
addition 134
attribute information 162, 335
deletion 152
inquiry 111, 321
record output 100, 308
search 53
update 144
attribute sequence 100, 111
attribute update function
addition 135
deletion 152
update 145
attribute value 76
add 141
delete 141
determine frequency of 85
inquiring on frequency 233
null 13, 148, 317
sort on 61, 226
update 141
authentication 2
automatic restart 271

B
base statement 154
begin transaction 166, 167, 255
block mode 58, 72, 123
addition 135
deletion 152
follow-up update 157
index browsing 89
inquiry 111, 321
record output 101, 309
search 44, 215
update 146
blocking factor 58, 72
BLSLIB 267, 276
boundary condition 52, 54
formulate 55
BSI (Bundesamt für Sicherheit in der Informationstechnik) 2
BTA see begin transaction
buffer size
  NAM 342
Bundesamt für Sicherheit in der Informationstechnik 2

C
C subquestion 50
C2 2
CALL DML Applications (manual) 6
CALL DML call 14
  in a FORTRAN program 22
  in a PASCAL program 22
  in an ALGOL program 24
  in an Assembler program 21
  in the PL/I program 23
CALL DML interface 13
CALL DML mode 19
CALL DML program
  compile 261
  execution with linked-in DBH 268
  link 262
  start 265
CALL DML statements see DML statements
CALL DML table 13, 19, 29, 303
  COMPANY 180
  SALES 184
CALL DML utility routine
  start 276
chaining
  DML statements 255
  open statements 32
client/server architecture 1
close 34, 253, 277
  logical file 34
  statement area 35, 191
  user 34
CNF 264
COBOL program 20
code number 162, 335
combinations
  in search with join 69
  communication name 264, 270
COMPANY

CALL DML table 180
comparison condition 54
  index browsing 88
  turn off 54
comparison value 41, 92
  length 75
  numeric 60
  sequence 59, 75
compile
  CALL DML program 261
compiling, linking and starting CALL DML programs 261
compound key 54, 306
compound key attribute 63, 306
compound key details 160, 332
configuration 264
configuration entry
  SESDCN administration 175
configuration file 264, 273, 274
configuration identifier 270
connection module 262, 271
constant transfer 285
constants 11
control statements and operands
  SEDI61 280
  SEDI63 294
Conversion of ... (manual) 7
count field 131
crte library 267, 276
cursor file 76
  create 222
  paging in 119, 123
  process 222
  record number 125
  response polling 117
  sort 125
  user 125
cursor file handling 125
  acknowledgment area 129
  inquiry area 130
  response area 130
  statement area 126
cursor technique 222
Index

D

data format see data type

data retrieval
  index browsing 85
  inquiry 107
  output to a SAM file 277
  record output 96
  response polling 117
  restricting a join cursor file 76
  search 37
  search with join 63

data type 303, 332
  new 303
  old 303

database access 14
  examples 352

database administrator 259

Database Operation (manual) 6

DB/DC system 2

DBCON see connection module

DBH
  select 341

DBH configuration file 268

DBH files 270

DBH name
  NAM 342

DCAM 271

DCAM application program 342

DCN 2
  see SESDCN

DCN administration command
  SESDCN administration 176

DCN entry
  SESDCN administration 175
decimal places
  number of 51
default open 283
  SEDI63 296
default value 51, 52, 111, 317
default value character 151, 160, 332
define comparison values
  acknowledgment area 94
  inquiry area 95
  statement area 93

delete
  attribute value 141
  delete identifier 329
  inquiry area 251–252
  mask character 92
  occurrence 141
  record 150
  statement area 252
  string identifier 92
deleting records 250
deletion 154
  acknowledgment area 152
  inquiry area 153
  statement area 150

Department of Defense 2
diagnostic documentation
  output 172, 174
differences to TIAM 273
direct update 133
  addition 131
  follow-up update 154
  record deletion 150
  statement 154
direct updating
  in a transaction 258
  update 141
distributed databases 2

DML statement 13, 255
  acknowledgment area for 16
  chaining of 255
  data types < V13.1 303
  inquiry area for 17
  response area for 17
  statement area for 16
  testing 292
dynamic loading
  modules 267

E

E subquestion 51
end identifier 173
  addition 135
  administrator open 171
  attribute information 162, 335
close 35
cursor file handling 129
delete comparison values 94
deletion 152
follow-up update 157
index browsing 89
inquiry 112, 322
NAM 342
NOTYPE 344
open 32
record output 101, 309
response polling 120
search 45
SESDCN administration 176
setting and deleting the delete identifier 330
transaction-oriented security 167
UNT/NOTUNT 347
update 146
END statement
SEDI63 299
end transaction 166, 167, 255
acknowledgment area 168
statement area 167
error messages see messages
ESQL program 6
ESQL-COBOL 19
ETA see end transaction
evidence of operations 2
example 180
Assembler program 353
COBOL program 368
database accesses 352
FORTRAN program 386
PL/I program 389
SEDI63 300
execute linked-in DBH
CALL DML program 268
extended address space 262

F
F2/Q3 2
file
logical 29, 190, 253
file close 34, 167, 191

file identifier 190
administrator open 171
open 31
restricting a join cursor file 79
search with join 70
follow-up update 154, 258
acknowledgment area 157
inquiry area 159
response area 159
statement area 154
format
statement area 17
format identifier
inquiry 111, 320
record input 308
record output 100
format inquiry area 17
FORTRAN program
CALL DML call 22
function
cursor file handling 127
define comparison values 94
setting and deleting the delete identifier 330
function code
administrator open 170
open 31
functions
SEDI61 277
SEDI63 292

G
general notes
linked-in application 269
glossary 7

H
HEXADECIMAL statement
SEDI63 295

I
identification area 273
ignore lock 257
independent DBH 262, 340
index 43, 303
### Index

| Details of | 160 |
| Search via | 43 |
| Index browsing | 85, 233 |
| Acknowledgment area | 89 |
| Inquiry area | 91 |
| Response area | 91 |
| Index values | 52 |
| Sort | 60 |
| Information about an attribute definition | 160 |
| Information function | 162, 334 |
| Input file | 37 |
| Logical | 37 |
| Inquiry | 107, 119, 243, 277, 317 |
| Acknowledgment area | 112, 322 |
| Inquiry area | 116 |
| Response area | 114, 324 |
| Statement area | 108, 318 |
| Inquiry area actual length | 18 |
| Define comparison values | 95, 236 |
| Delete | 251–252 |
| Deletion | 153 |
| Follow-up update | 159, 247, 251 |
| Format | 17 |
| Index browsing | 91, 233, 235 |
| Inquiry | 116, 243 |
| Length | 31, 170 |
| Maximum length | 31 |
| Record output | 316 |
| Response polling | 122, 224 |
| Restricting a join cursor file | 83 |
| Retrieval using record output | 241 |
| Search | 59, 195, 196 |
| Search with join | 74, 220 |
| Setting and deleting the delete character | 331 |
| Update | 147 |
| Updating records | 249 |
| Insert occurrence | 141 |
| Interrupt handling | STXIT 269 |
| Inverted attribute | 87 |

| J |
| Join | 63 |
| Search with | 63 |
| Join attribute | 63 |
| Search with join | 70 |
| Join cursor file | 63, 128 |
| Create | 228 |
| Process | 228 |
| Restricting | 76 |

| K |
| Key subquestion | 50 |

| L |
| L subquestion | 51 |
| Length of response area | 175 |
| SESDCN administration | 175 |
| Link CALL DML program | 262 |
| Linked-in application | 275 |
| General notes | 269 |
| Terminate | 268 |
| Linked-in DBH | 263, 340 |
| Multiple loading | 270 |
| Lock | 166 |
| Ignore | 257 |
| Lock mode inquiry | 112, 322 |
| Record output | 101, 309 |
| Response polling | 120 |
| Search | 45 |
| Logical file | 29, 190, 191, 253 |
| Open | 29 |
| Logical input file | 37 |
| Logical subquestion relationships | 46 |
| LOW statement | SEDI63 295 |

| M |
| Manuals (SESAM/SQL) | 6 |
| Mask character | 92 |
| Change | 92 |
| Delete | 92 |
mask search 54, 88, 92
messages
   suppress 344
Messages (manual) 7
metacharacters 11
migration see conversion
migration, see “Migrating …” (manual)
mixed operation 19
module
dynamic loading of 267
module library 262, 267
MS-DOS 1
MS-Windows 1
multi-db processing 2
multiple attribute 332
multiple loading
   linked-in applications 270
multi-user operation 2

N
NAM 264, 341
   acknowledgment area 343
   statement area 341
network 2
NOTYPE
   acknowledgment area 345
   statement 344
   statement area 344
NOTOUNT 264, 346
null attribute value 13, 148, 304, 317
number
   attributes 46
numeric comparison value 60

O
O subquestion 51
occurrence 332
   add 141
   append 141
   delete 141
   insert 141
   update 141
old data types
   DML statements 303
online application 2
open 29, 190, 253, 277
   administrator 169
   logical file 29
   statement area 29, 190
open statement
   chaining 32
   SEDI61 282
operation code
   020 SESDCN administration 175
   addition 133
   administration statements for the DBH 173
   administrator open 169
   attribute information 161, 334
   close 35
   cursor file handling 127
   define comparison values 93
deletion 151
   follow-up update 155
   index browsing 87
   inquiry 109, 319
   NAM 342
   NOTYPE 344
   open 30
   record output 98, 306
   response polling 118
   search 41
   setting and deleting the delete identifier 330
   table 350
   TRACE 348
   transaction-oriented security 167
   UNT/OUNT 346
   update 143
Orange Book 2
output
   diagnostic documentation 172
   retrieval responses 277
to a SAM file 277
output medium
   TRACE 349
output record structure
   initiate definition 284
output retrieval responses 277
overview of control statements
Performance (manual) 7
PK function 41
see primary key function
PL/I program
CALL DML call 23
polling condition 119
previous version 7
primary key 41, 63, 134

primary key function 41
addition 133
deletion 151
follow-up update 155
inquiry 109, 319
record output 96, 306
restricting a join cursor file 78
search with join 68
update 143

administrator open 169
attribute information 161, 334
close 35
cursor file handling 127
define comparison values 93
deletion 151
follow-up update 155
index browsing 87
inquiry 109, 319
NAM 342
NOTYPE 344
open 30
record output 96, 306
response polling 118
search 41
setting and deleting the delete identifier 330
TRACE 348
transaction-oriented security 167
UNT/NOTOUND 346
update 143

Q
Q3 2

R
R subquestion 52
read without locking 256, 257
README file 7
record
delete 150
update 141
record deletion 150
acknowledgment area 152
record number 70, 306
cursor file 127
record numbers function
restricting a join cursor file 79
search with join 70
record output 96, 119, 241, 277, 304
acknowledgment area 102, 310
inquiry area 106, 316
response area 103, 312

packed attribute
transfer 285
paging
in a cursor file 119, 123
parallel 2
parameter entry 264
partially inverted attribute 91
PASCAL program
CALL DML call 22
password 175
addition 133
administration statements for the DBH 172
administrator open 169
attribute information 161, 334
close 35
cursor file handling 127
define comparison values 93
deletion 151
follow-up update 155
index browsing 87
inquiry 109, 319
NAM 342
NOTYPE 344
open 30
record output 96, 306
response polling 118
search 41
setting and deleting the delete identifier 330
TRACE 348
transaction-oriented security 167
UNT/NOTOUND 346
update 143

Q
Q3 2

R
R subquestion 52
read without locking 256, 257
README file 7
record
delete 150
update 141
record deletion 150
acknowledgment area 152
record number 70, 306
cursor file 127
record numbers function
restricting a join cursor file 79
search with join 70
record output 96, 119, 241, 277, 304
acknowledgment area 102, 310
inquiry area 106, 316
response area 103, 312
Index

statement area 97, 305
record update
  deletion 151
record update function
  follow-up update 156
  for update 144
references 4
reliable operation, see operational reliability
requesters 271
REQUEST-USERS 271
reset transaction 166, 167, 258, 259
  acknowledgment area 168
  statement area 167
resources 19
response
  independent DBH 340
  linked-in DBH 340
response area 17
  administration statement 173
  attribute information 164, 336
  follow-up update 159
  index browsing 91
  inquiry 114, 324
  length 30, 170, 175
  maximum length 30
  record output 312
  response polling 122, 193, 196, 224
  restricting a join cursor file 81
  search 58, 195, 197
  search with join 72
  SESDCON administration 177
response polling 117, 258, 277, 304
  acknowledgment area 121
  inquiry area 122
  response area 122
SEDI63 299
  statement area 118
response record
  sort 60
restart 270
  with SESAM-LINK 270
restricting a join cursor file 76
  acknowledgment area 80
  inquiry area 83
response area 81
  statement area 76
retrieval
  in a transaction 256
  search 215
  statement 13
  using search with join 219
RTA see reset transaction
S
S subquestion 52, 61
SALES
  CALL-DML table 184
SAN see symbolic attribute name
SAT 2
save data 166
search 37, 119, 277
  acknowledgment area 55, 193
  inquiry area 59
  interrupt 346
  sequential 43
  statement area 38
  subquestion of 44, 45
  via index 43
search condition 54
search 53
  statement area 47
search response area 58
search with join 219
  acknowledgment area 71
  inquiry area 74
  response area 72
  statement area 64
Security Audit Trail function 2
security criteria 2
SEDECIMAL statement
SEDI63 295
SEDI61 275, 277, 277
$ statement 289
/* statement 289
_* statement 289
_A_statement 287
_F_statement 288
_H_statement 280
Index

_Q_statement 289
_T_statement 284
_W_statement 281
control statements 280
END statement 289
functions 277
open statement 282
operands 280
overview of control statements 278
SEDI61L 277
SEDI61L see SEDI61
SEDI63 275, 292
$ statement 298
/* statement 299
* statement 298
_A_statement 296
_F_statement 297
_Q_statement 298
control statements 294
default open 296
END statement 289
example 300
functions 292
HEXADECIMAL statement 295
LOW statement 295
operands 294
overview of control statements 293
response polling 299
SEDECIMAL statement 295
STATUS statement 294
WROUT statement 294
SEDI63L see SEDI63
selection 37, 41–42, 96, 304, 317
separator 255
sequential search 43
server 1
SESAM call
  in DCAM 272
SESAM operation
  control of 172, 174
  information 172
SESAM subquestion logic 46
SESAM/DCAM DB/DC system 271
SESAM/SQL DBH 172
SESAM/SQL DCAM application 271
SESAM/SQL manuals 6
SESAM/SQL-DCN 2
  operation information 174
SESAM/SQL-DCN operation
  control 174
SESAM/SQL-LINK 263, 275
SESAM/SQL-Server 1
SESAMOML 267
SESCONF 264, 273
SESDCN administration 174
  acknowledgment area 176
  response area 177
  statement area 174
SESGET call 15
  in DCAM 272
SESGETW call 15
  DCAM 272
SESORT 52
SESPUT 19
SESPUT call 15
  in DCAM 272
session 270
set
  delete identifier 329
  mask character 92
  string identifier 92
setting and deleting the delete character
  acknowledgment area 331
  inquiry area 331
setting and deleting the delete identifier
  statement area 330
significant value 53, 87
SINIX 1
sort
  attribute values 61
  cursor file 125
  index values 60
  primary key values 60
  response record 60
set criteria 54
  formulate 55
sort cursor file
  compress 128
sort sequence 52
special statements 340
SQL mode 19
SQL Reference Manual 6
SQL server 1
start
CALL DML program 265
start command 276
start indicator
   NAM 342
START-SESLK-RETRIEVAL-DIALOGUE 276
statement area 16
   actual length 18
   addition 132
   administration statement 172
   administrator open 169
   attribute information 161, 333
   close 35, 191
   cursor file handling 126
   define comparison values 93, 236
   delete 252
   deleting records 250
   follow-up update 154, 246, 251
   format 17
   index browsing 233, 234
   inquiry 108, 243, 318
   NAM 341
   NOTYPE 344
   open 29, 190
   record output 97, 305
   response polling 118, 193, 195, 216, 224, 225
   restricting a join cursor file 76
   retrieval using record output 241
   retrieval using search with join 219
   search 38, 192, 194
   search condition 47
   search with join 64
   SESDCN administration 174
   setting and deleting the delete identifier 330
   subquestion 47
   TRACE 348
   transaction-oriented security 167
   UNT/NOUNT 346
   update 142
   updating records 248
   statement element
      restricting a join cursor file 78
      search with join 68
   status code 16, 19
   STATUS statement
      SEDI63 294
   status subnumber 16
   strategy 117, 119
      cursor file handling 127
      restricting a join cursor file 79
      search 42
      search with join 69
   string
      SES 268
   string identifier 53
      change 92
      delete 92
   string search 53, 88, 92
   structure
      chained statements 255
   STXIT
      interrupt handling 269
   subattribute
      transfer 285
   subquestion 45, 46, 50, 53
      E 51
      flexible construction of 62
      L 51
      logical relationship 46
      O 51
      search 44
      statement area 47
      T 52
      U 51
      Z 51
   subquestion type 50
      old 46
   symbolic attribute 160, 162
   symbolic attribute name 53, 308, 332, 335
   syntax diagram 10
      addition 132
      administration statements for the DBH 172
Index

administrator open 169
attribute information 161, 333
close 35
cursor file handling 126
define comparison values 93
deletion 150
follow-up update 154
index browsing 86
inquiry 108, 318
NAM 341
NOTYPE 344
open 29
record output 97, 305
restricting the join cursor file 76–77
search 38
search with join 64–65
SESDCN administration 174
subquestion 47–49
TRACE 348
transaction-oriented security 167
UNT 346
update 142
syntax element 11
SYS.MOD 276

T
T subquestion 52
table 131, 134
table name
  open 30
table of operation codes 350
TASKLIB 267
temporary work files 270
terminate
  linked-in application 268
testing
  DML statements 292
TIAM
difference to 273
time runout 269
timer interrupt 269
TRACE 264
  statement 348
  statement area 348
trace
  activate/deactivate 348
transaction
  begin 167
  data retrieval in a 256
  database update in 258
  direct updating in a 258
  end of 166–167
  operations in 256
  reset 166, 167, 259
  start of 166
transaction boundary 135, 146, 152, 166, 253
transaction mode 271
  using CALL-DML 271
  with DCAM 271
transaction monitor 2
transaction processing 2
transaction-oriented security 135, 146, 152, 270, 271
  acknowledgment area 168
  statement area 167
  statements for 166
transfer
  attribute 284
  constant 285
  packed attribute 285
  subattribute 285
transfer area
  format of 16
  meaning of 16
transparent access 2
trusted communication, see trusted operation
trusted operation 2
type of character
  define comparison values 94
type of trace
  TRACE 349

U
U subquestion 51
Universal Transaction Monitor, see universal trans-
action monitor
UNIX 1
UNT 264, 346
Index

UNT/NOTUNT
  acknowledgment area 347
  statement 346
  statement area 346
update 141, 154
  acknowledgment area 146
  attribute value 141
  inquiry area 147
  occurrence 141
  record 134, 141
  statement 13
  statement area 142
update authorization 258
  addition 133
  deletion 151
  follow-up update 156
  update 144
updating 248
user close 34
user cursor file 125, 128
user module 263, 269
using CALL DML
  transaction mode 271
Utilities (manual) 6
Utility Monitor (manual) 6

V
value
  significant 87
variable 11
verbal attribute 160
verbal attribute name 162, 332
view
  see logical input file

W
work files
  temporary 270
WROUT statement
  SEDI63 294

X
XS system 262
Contents

1  Preface .............................................................. 1
   1.1 Brief product description .................................................. 1
   1.2 Target group ........................................................... 3
   1.3 Summary of the contents of the manuals .............................. 4
   1.3.1 Summary of the contents of this manual ............................... 5
   1.3.2 Guide to the SESAM/SQL manuals ...................................... 6
   1.4 README file .......................................................... 7
   1.5 Changes in V3.0 made since V2.2 .................................... 8
   1.6 Notational conventions ................................................. 10

2  CALL DML interface ................................................... 13
   2.1 CALL DML calls ....................................................... 14
   2.2 Format and meaning of the transfer areas .............................. 16
   2.3 Mixed operation of SQL and CALL DML interfaces .................... 19
   2.4 Examples of the various programming languages ...................... 20

3  DML statements ........................................................ 27
   3.1 Overview of DML statements ............................................. 27
   3.2 Open ............................................................... 29
   3.3 Close ............................................................... 34
   3.4 Search .............................................................. 37
   3.5 Search with join ....................................................... 63
   3.6 Restricting a join cursor file .......................................... 76
   3.7 Index browsing ......................................................... 85
   3.8 Define comparison values .............................................. 92
   3.9 Record output ......................................................... 96
   3.10 Inquiry ............................................................. 107
   3.11 Response polling .................................................. 117
   3.12 Paging in a cursor file in block mode ............................... 123
### Contents

3.12 Cursor file handling .............................................................. 125
3.13 Addition .................................................................................. 131
  Automatic count field in a compound key ......................................... 139
3.14 Update ...................................................................................... 141
  Deletion by changing to the null attribute value ................................ 148
3.15 Deletion ...................................................................................... 150
3.16 Follow-up update ........................................................................ 154
3.17 Attribute information ................................................................... 160
3.18 Statements for transaction-oriented security .................................... 166
3.19 Administrator Open ...................................................................... 169
3.20 Administration statements for the DBH ........................................ 172
3.21 Administration statement for SESDCN .......................................... 174

4 Using DML statements .................................................................... 179
4.1 Examples ..................................................................................... 180
  CALL DML table COMPANY .......................................................... 180
  CALL DML table SALES ............................................................... 184
  Contents of the CALL DML tables COMPANY and SALES .................. 186
4.2 Opening logical files ...................................................................... 190
4.3 Closing logical files ....................................................................... 191
4.4 Retrieval using search .................................................................... 192
  Projection ......................................................................................... 192
  Selection on conditions applied to primary key value .......................... 194
  Selection by conditions applied to the record number ......................... 197
  Selection by conditions applied to attribute values ............................. 198
  Retrieval in block mode .................................................................... 215
  Flexible formulation of subquestions ................................................. 217
4.5 Retrieval using search with join .................................................. 219
4.6 Cursor technique for search/search with join .................................. 222
  Creating and processing a cursor file ............................................... 222
  Sort on attribute values ..................................................................... 226
  Creating and processing a join cursor file .......................................... 228
4.7 Inquiring on attribute value frequency (index browsing) .................. 233
4.8 Defining comparison values .......................................................... 236
4.9 Retrieval using record output ....................................................... 241
4.10 Retrieval using inquiry ............................................................... 243
4.11 Adding new records ..................................................................... 245
4.12 Updating records .......................................................................... 248
4.13 Deleting records .......................................................................... 250

5 Programming transactions ............................................................. 253
5.1 Open, close and transaction boundaries ........................................ 253
5.2 Chained statements ...................................................................... 255
5.3 Operations in transactions ........................................................... 256
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Resetting transactions</td>
<td>259</td>
</tr>
<tr>
<td>5.5</td>
<td>Database accesses outside transaction boundaries</td>
<td>259</td>
</tr>
<tr>
<td>6</td>
<td>From compiling to starting the CALL DML programs</td>
<td>261</td>
</tr>
<tr>
<td>6.1</td>
<td>Compiling and entering in a library</td>
<td>261</td>
</tr>
<tr>
<td>6.2</td>
<td>Linking</td>
<td>262</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Linking with the independent DBH</td>
<td>262</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Linking with the linked-in DBH</td>
<td>263</td>
</tr>
<tr>
<td>6.3</td>
<td>Parameterizing the connection modules</td>
<td>264</td>
</tr>
<tr>
<td>6.4</td>
<td>Starting</td>
<td>265</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Starting an independent DBH application</td>
<td>265</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Starting a linked-in DBH application</td>
<td>266</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Assigning module libraries</td>
<td>267</td>
</tr>
<tr>
<td>6.5</td>
<td>Execution with the linked-in DBH</td>
<td>268</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Terminating the program</td>
<td>268</td>
</tr>
<tr>
<td>6.5.2</td>
<td>General notes</td>
<td>269</td>
</tr>
<tr>
<td>6.6</td>
<td>Using CALL DML in transaction mode</td>
<td>271</td>
</tr>
<tr>
<td>6.6.1</td>
<td>DCAM</td>
<td>271</td>
</tr>
<tr>
<td>7</td>
<td>CALL DML utility routines</td>
<td>275</td>
</tr>
<tr>
<td>7.1</td>
<td>Outputting retrieval responses (SEDI61)</td>
<td>277</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Functions of SEDI61/SEDI61L</td>
<td>277</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Overview of the control statements</td>
<td>278</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Control statements and operands</td>
<td>280</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Example of SEDI61/SEDI61L</td>
<td>290</td>
</tr>
<tr>
<td>7.2</td>
<td>Testing DML statements (SEDI63)</td>
<td>292</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Functions of SEDI63/SEDI63L</td>
<td>292</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Overview of control statements</td>
<td>293</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Control statements and operands</td>
<td>294</td>
</tr>
<tr>
<td>7.2.4</td>
<td>Examples of SEDI63/SEDI63L</td>
<td>300</td>
</tr>
<tr>
<td>8</td>
<td>DML statements for old data types</td>
<td>303</td>
</tr>
<tr>
<td>8.1</td>
<td>Differences when working with old data types</td>
<td>303</td>
</tr>
<tr>
<td>8.2</td>
<td>Record output</td>
<td>304</td>
</tr>
<tr>
<td>8.3</td>
<td>Inquiry</td>
<td>317</td>
</tr>
<tr>
<td>8.4</td>
<td>Setting and deleting the delete identifier</td>
<td>329</td>
</tr>
<tr>
<td>8.5</td>
<td>Attribute information</td>
<td>332</td>
</tr>
<tr>
<td>9</td>
<td>Appendix</td>
<td>339</td>
</tr>
<tr>
<td>9.1</td>
<td>Special statements</td>
<td>340</td>
</tr>
<tr>
<td>9.1.1</td>
<td>NAM statement</td>
<td>341</td>
</tr>
<tr>
<td>9.1.2</td>
<td>NOTYPE statement</td>
<td>344</td>
</tr>
<tr>
<td>9.1.3</td>
<td>UNT/NOTUNT statement</td>
<td>346</td>
</tr>
<tr>
<td>9.1.4</td>
<td>TRACE statement</td>
<td>348</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>9.2</td>
<td>Table of operation codes</td>
<td>350</td>
</tr>
<tr>
<td>9.3</td>
<td>Examples of database accesses</td>
<td>352</td>
</tr>
<tr>
<td></td>
<td>Example of an Assembler program</td>
<td>353</td>
</tr>
<tr>
<td></td>
<td>Example of a COBOL program</td>
<td>368</td>
</tr>
<tr>
<td></td>
<td>Example of a FORTRAN program</td>
<td>386</td>
</tr>
<tr>
<td></td>
<td>Example of a PL/I program</td>
<td>389</td>
</tr>
<tr>
<td></td>
<td><strong>Related publications</strong></td>
<td>393</td>
</tr>
<tr>
<td></td>
<td><strong>Index</strong></td>
<td>399</td>
</tr>
</tbody>
</table>
SESAM/SQL-Server V3.0A (BS2000/OSD)

CALL DML Applications

Target group
SESAM application programmers

Contents
– CALL DML statements for processing SESAM databases using application programs
– Transaction mode with UTM and DCAM
– Utility routines SEDI61 and SEDI63 for data retrieval and direct updating
– Notes on using both CALL DML and SQL modes

Edition: January 2000

File: SES_DML.PDF

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.
All hardware and software names used are trademarks of their respective manufacturers.
Comments on SESAM/SQL-Server V3.0A
CALL DML Applications

Submitted by
Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format …@ts.fujitsu.com.

The Internet pages of Fujitsu Technology Solutions are available at http://ts.fujitsu.com/...


Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument


Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.


Copyright Fujitsu Technology Solutions, 2009