



# **MIMER**

## **System Management Handbook**

**Version 7.3**

**Copyright © 1996 Sysdeco Mimer AB**

MIMER version 7.3 System Management Handbook

November, 1996

Copyright © 1996 Sysdeco Mimer AB.

Published by Sysdeco Mimer AB,

P.O.Box 1713,

S-751 47 Uppsala, Sweden.

Tel +46(0)18-18 50 00.

Fax +46(0)18-18 51 00.

Internet: <http://www.mimer.se>

Produced by Sysdeco Mimer AB, Uppsala, Sweden.

All rights reserved under international copyright conventions.

The contents of this manual may be printed in limited quantities for use at a Mimer installation site. No parts of the manual may be reproduced for sale to a third party.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	
1.1	System management responsibilities .....	1-1
1.2	Organization of this manual.....	1-2
<b>2</b>	<b>THE MIMER INSTALLATION</b>	
2.1	Single and multi-user installations .....	2-1
<b>3</b>	<b>THE DATABASE ENVIRONMENT</b>	
3.1	The data dictionary .....	3-1
3.2	Idents.....	3-2
3.3	Databanks.....	3-2
3.3.1	System databanks .....	3-3
3.3.2	Location of system databanks.....	3-4
3.3.3	Changing databank locations .....	3-4
3.4	Other database objects .....	3-5
3.5	Database security .....	3-5
3.5.1	Idents in database security.....	3-5
3.5.2	Access rights and privileges .....	3-6
3.5.3	Recursive interactions between privileges.....	3-7
3.5.4	Restriction views .....	3-9
3.6	Data integrity.....	3-9
3.6.1	Domains .....	3-9
3.6.2	Entity integrity .....	3-9
3.6.3	Referential integrity.....	3-10
3.6.4	Table integrity .....	3-10
3.6.5	View integrity.....	3-10
<b>4</b>	<b>ESTABLISHING A MIMER SYSTEM</b>	
4.1	General information .....	4-1
4.2	Creating the central system databanks: SDBGEN .....	4-2
4.3	Creating module-specific system databanks.....	4-3
4.3.1	Upgrading existing system databanks .....	4-4
4.4	Establishing the database .....	4-4
4.5	Managing database connections .....	4-5
4.5.1	Database names.....	4-5
4.5.2	Connection names.....	4-6
<b>5</b>	<b>EXPORT/IMPORT</b>	
5.1	The main menu .....	5-1
5.2	Export options.....	5-2
5.2.1	Functions .....	5-2
5.2.2	Authorization.....	5-2
5.2.3	Exported files.....	5-3
5.3	Import options.....	5-3
5.3.1	Import – object creation .....	5-3
5.3.2	Import – data load .....	5-6

5.3.3	Authorization.....	5-7
5.4	Load and Unload functions.....	5-7
5.4.1	Data file formats.....	5-8
5.4.2	The load operation.....	5-9
5.4.3	Unload operation.....	5-9
5.4.4	Authorization.....	5-10
5.5	SYSxxGEN (generate system databanks).....	5-10
5.5.1	Functions.....	5-10
5.5.2	Authorization.....	5-10
5.6	Enter and Leave program ident.....	5-10
<b>6</b>	<b>BACKUP AND RESTORE</b>	
6.1	Background information.....	6-1
6.1.1	Database consistency.....	6-3
6.1.2	Databank backups.....	6-3
6.1.3	Databank incremental backups.....	6-4
6.1.4	Backup versus Incremental Backup.....	6-4
6.1.5	Maximizing data security.....	6-6
6.1.6	SQL system management functions and databank backups.....	6-6
6.2	Backup and restore of databanks.....	6-8
6.2.1	Making a databank backup.....	6-8
6.2.1.1	Using the SQL system management functions.....	6-8
6.2.1.2	Using the host file system.....	6-9
6.2.2	Restoring a databank.....	6-10
6.3	Backup and restore of system databanks.....	6-11
6.3.1	SYSDB.....	6-11
6.3.2	LOGDB.....	6-11
6.3.3	TRANSDB and SQLDB.....	6-12
6.3.4	Other system databanks.....	6-13
6.3.5	Re-creating TRANSDB, LOGDB and SQLDB.....	6-13
6.4	The Backup/Restore functionality.....	6-14
6.4.1	Authorization.....	6-14
6.4.2	Starting the backup and restore functionality.....	6-15
6.4.3	Backup databank.....	6-15
6.4.4	Restore databank.....	6-17
6.4.5	List backups.....	6-19
6.4.6	Drop backups.....	6-20
6.4.7	Shadows offline, online, drop log.....	6-21
6.4.8	Enter and leave a program ident.....	6-21
<b>7</b>	<b>THE READLOG FUNCTIONALITY</b>	
7.1	Functions.....	7-1
7.2	Authorization.....	7-1
7.3	Using the READLOG functionality.....	7-2
7.3.1	List definitions (output control).....	7-2
7.3.2	List restrictions.....	7-2
7.3.3	List operations.....	7-4
7.3.4	Change program id.....	7-4
7.4	Output format.....	7-5
<b>8</b>	<b>DATABASE STATISTICS</b>	
8.1	Statistical information.....	8-1
8.2	Authorization.....	8-1
8.3	The SQL statistics functions.....	8-2
8.3.1	Statistics for the entire database.....	8-2
8.3.2	Statistics for specified idents.....	8-2
8.3.3	Statistics for specified tables.....	8-2
8.4	When to use the SQL statistics functions.....	8-3

<b>9</b>	<b>SYSTEM TUNING</b>	
9.1	Bufferpool size.....	9-1
9.1.1	General considerations.....	9-1
9.1.2	Changing the bufferpool size .....	9-2
9.2	Number of threads.....	9-2
9.3	Number of shadow servers .....	9-3
9.4	The MIMSERV functionality.....	9-3
9.4.1	Functions .....	9-3
9.4.2	MIMSERV output example.....	9-8
<b>10</b>	<b>TROUBLESHOOTING</b>	
10.1	Runtime malfunctions.....	10-1
10.2	The DBC functionality .....	10-1
10.2.1	General description.....	10-1
10.2.2	Authorization.....	10-2
10.2.3	User communication.....	10-2
10.2.4	Result file contents.....	10-2
10.2.5	Example of result file.....	10-5
10.2.6	Error messages.....	10-6
<b>11</b>	<b>THE DBOPEN FUNCTIONALITY</b>	
11.1	Functions.....	11-1
11.2	Authorization.....	11-1
11.3	DBOPEN output example.....	11-2
<b>A</b>	<b>DATA DICTIONARY TABLES</b>	
	Summary of data dictionary tables .....	A-2
	ACCESS .....	A-3
	ACCCOL .....	A-4
	BACKUP .....	A-4
	CODE .....	A-4
	COLUMN.....	A-5
	COMMENT .....	A-7
	DATABANK .....	A-7
	DBFILE.....	A-7
	DBNAME .....	A-8
	DEFAULT.....	A-8
	DOMAIN .....	A-8
	FUNCTION.....	A-10
	IDENT .....	A-10
	INDEX.....	A-10
	INDEXCOL .....	A-11
	MESSAGE.....	A-11
	OBJECT.....	A-12
	PRIV .....	A-12
	RESTRICT.....	A-13
	SERVINFO .....	A-13
	SEVERITY .....	A-13
	SQLLANG .....	A-13
	SYNONYM .....	A-14
	TABLE.....	A-14
	TABLEX.....	A-15
	VIEWCOL.....	A-15
	VIEWDEP.....	A-15
	VIEWRES .....	A-16



# 1 INTRODUCTION

## 1.1 System management responsibilities

Installation of a MIMER system involves the creation of a specially privileged user called the **system administrator**, who is responsible for the installation of MIMER software and the initial creation of the database environment. In an established system, the system administrator is also responsible for the maintenance of the installation - system tuning, troubleshooting, backup/restore, and so on. Depending on the organization at a particular site, system administration responsibilities may be divided between machine administration and database administration. The specially privileged MIMER ident SYSADM, created when the system is installed, is the database system administrator. The ident name SYSADM may not be changed. In some organizations, system administrator functions may be carried out by a group of people rather than a single individual. There may however only be one SYSADM ident in a given system.

The system administrator must have a good working knowledge of the host computer operating system. In some MIMER implementations, certain system management jobs require special privileges within the operating system.

Certain MIMER functionality (particularly those for installation of the software modules) may only be run by the system administrator. Others require privileges and access rights which are initially granted only to the system administrator, but which may be passed on to other MIMER users if desired.

The system administrator has SELECT access on the internal MIMER data dictionary tables, permitting direct reading of the meta-data describing the system. In examining the contents of the data dictionary with the functionality provided, the system administrator has, by default, wider access rights than other users.

The system administrator does not, however, have general access to the contents of the database. Information stored in user-defined tables may only be accessed by other users if the creator of the table explicitly grants permission. In this context, the system administrator is treated just as any other user.

## 1.2 Organization of this manual

This manual is a general handbook for system administrators, describing the various areas of responsibility in detail. Familiarity with the principles of relational database management with MIMER is assumed. Aspects of MIMER system components and database environment which are particularly relevant to the system administrator's responsibilities, are described in Chapters 2 and 3. Chapter 4 deals with the system administrator's responsibilities in establishing a MIMER system.

The remainder of the manual describes the system management functionality:

- The EXPORT/IMPORT functionality (Chapter 5) permits base tables to be transferred between different installations. The export/import functionality also includes facilities for creating and managing system databanks (except SYSDB, TRANSDB, LOGDB and SQLDB).
- BACKUP and RESTORE commands in SQL (Chapter 6) provide facilities for full or incremental backups of the database and for restoring data in the event of a system crash. Backup functions may be complemented by database shadowing. The shadowing system is maintained with the Shadowing facility (see the *MIMER Shadowing Reference Manual*).
- The READLOG functionality (Chapter 7) reads the contents of the LOGDB databank, which can provide an audit trail or other relevant information in the event of a system failure. The contents of LOGDB can only be read with this facility.
- The STATISTICS commands in SQL maintain statistical information in the data dictionary concerning the usage of tables and indexes (Chapter 8). This information is used by the MIMER/SQL compiler in optimizing access paths for data manipulation statements.
- The MIMSERV functionality (Chapter 9) allows the system administrator to monitor a working MIMER installation, providing statistical information necessary for system tuning.
- The DBC functionality (Chapter 10) is provided to check the physical integrity of databanks. This functionality should be used if the physical file in which a databank is stored is suspected of being corrupt. The functionality can also be used to ensure that backup copies of databanks are accurate.
- The DBOPEN functionality (Chapter 11) opens all the databanks in the database. This facility can be used after a databank has not been closed properly (e.g. because of a system failure). If this is not done, the implicit databank check performed when a user first opens the databank can cause an appreciable delay.

Note: Generic names for these facilities are used throughout this manual. Refer to the machine-specific User's Guide for the actual names in your system.

Installation-dependent information for different platforms is provided separately with the specific installation. Certain details in the general documentation in this handbook may differ between different platforms.

## 2 THE MIMER INSTALLATION

### 2.1 Single and multi-user installations

All MIMER software is available for either single- or multi-user installations where permitted by the host platform.

In a **multi-user installation**, several users share the same system databanks, and both system and user databanks may be accessed by more than one user at once. The multi-user version of MIMER software is used and application programs are linked with the multi-user MIMER runtime library. A multi-user system must be started after installation. The system may be stopped for installation of new MIMER modules and other maintenance purposes, and then restarted without affecting the established database structure. A number of system-wide parameters, on some platforms including a central directory for files used in controlling and monitoring the multi-system, are defined at start-up time (see machine-dependent information).

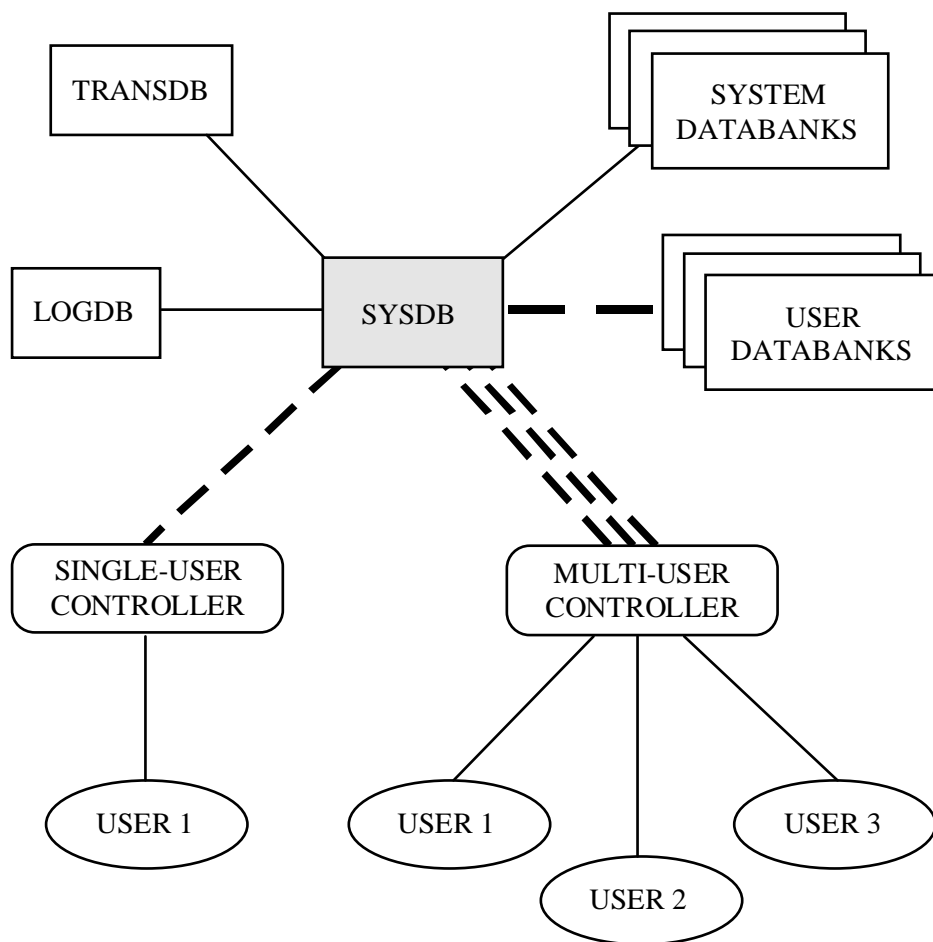
In a **single-user installation**, the user has his own system and user databanks. Only one user may access the database at one time. The single-user version of MIMER software is used, and application programs are linked with the single-user MIMER runtime library. A single-user system may be run as soon as the necessary system databanks are installed. A single-user system may include a fully-developed structure of user and program idents with restricted access rights and privileges, but only one user ident may be logged on to the system at once (regardless of which MIMER module is operative).

Most platforms may have a number of separate single-user installations running concurrently. Each will have its own local MIMER environment, and one cannot access the database owned by another. It is important to distinguish this situation from a multi-user MIMER installation, where all users share the same database environment. On some platforms it is possible to run several MIMER multi-user systems in parallel (see the machine-dependent Installation Guide).

The physical location of databank files in a MIMER system is stored in the system databank SYSDB. The correct files for a database system are thus always accessed, even in installations with several independent MIMER systems. The location of SYSDB itself is stated in the configuration of a single system or at start-up time for multi-user installations. Each databank in a MIMER system is also labeled with a system identification number, used as an additional internal check that the databank accessed 'belongs' to the currently active system.

The difference between single- and multi-user MIMER installations lies in the software programs, not in the database itself (see Figure 2.1). It is possible to access the same database from both single- and multi-user systems by specifying the same home directory (and thus the same physical SYSDB file) at system start-up, provided that the host platform permits multiple access to the same file. Any single user will, however, be denied access as long as a multi-user system is accessing the database, since single-user programs only allow one user at a time.

Single-user mode can profitably be used during the development phase of a MIMER production system. When development is complete, the database environment is transferred to a multi-user production system simply by giving the appropriate address for SYSDB at system start-up. No modifications to the database are required. Note however that application programs developed in the single-user system must be relinked with the multi-user MIMER runtime library.



**Figure 2.1** Single- and multi-user MIMER .

## 3 THE DATABASE ENVIRONMENT

### 3.1 The data dictionary

The database environment is controlled through a central **data dictionary**, stored in the system databank SYSDB and automatically maintained by MIMER. The data dictionary contains meta-data describing all the objects in the database. System access to the data dictionary tables is performed by internal routines and is transparent to the user.

Restricted facilities for examining the contents of the data dictionary are available to all users through the LIST and DESCRIBE functions in Interactive and Batch MIMER/SQL (see the respective manuals for a more complete description of these facilities). In general, a user may read data dictionary information for database objects to which he or she has access. The Interactive SQL facilities use pre-defined views on the data dictionary tables to present the information in a structured form (see the MIMER/SQL Reference Manual for documentation on the data dictionary views available to all users).

The system administrator may read the contents of the data dictionary tables directly, and may grant SELECT access on the tables to any other user. The organization of the data dictionary tables is documented in Appendix A of this manual.

No individual user, including SYSADM, may update data dictionary tables directly. All write operations in the data dictionary are performed by internally controlled routines, to ensure consistency within the dictionary.

## 3.2 Idents

An **ident** in a MIMER system is an authorized user of the system, or the collective identity of a group of users sharing common privileges. Four types of idents are recognized:

- |         |  |
|---------|--|
| USER    | idents are authorized to log on to any MIMER module. User idents are generally associated with specific physical individuals authorized to use the system.   |
| OS_USER | idents are a type of user ident with the same username and password as an operating system account. If a user logged in to the operating system is also defined as an OS_USER, the user may log in to MIMER without providing an additional username and password.   |
| PROGRAM | idents may not log on to MIMER, but may be entered from within an application program or interactive environment by using the ENTER statement. Once a program ident is entered, the privileges held by the program ident apply within the application program. Program idents are generally associated with specific functions within the system, not with physical individuals.   |
| GROUP   | idents are not as such authorized to use the system, but are collective identities for groups of user or program idents. Any privileges granted to or revoked from a group ident automatically apply to all members of the group. Any user or program ident can be a member of as many groups as is required, and a group can include an unlimited number of members. Group idents provide a facility for organizing the privilege structure in the database system. |

When MIMER is installed, the system administrator **user** ident SYSADM and the **group** ident PUBLIC are automatically created. The password for SYSADM is defined when the system is installed (see Section 4.2). All idents in the system belong to the group PUBLIC, so privileges granted to PUBLIC by any user are global to the system. Membership in the group PUBLIC cannot be revoked.

## 3.3 Databanks

The database is divided into a number of databanks, each databank is stored in a single physical file in the host system. Transaction control in application programs and a number of the system facilities (notably Backup/Restore) work at the databank level.

The division of a database into databanks is made on the basis of file handling considerations from the operating system viewpoint and transaction control considerations from the database viewpoint.

Databanks may be defined with LOG, TRANS or NULL options, which determine transaction handling and logging as follows:

<b>LOG</b>	All operations on the databank are performed under transaction control. All transactions are logged.
<b>TRANS</b>	All operations on the databank are performed under transaction control. No transactions are logged.
<b>NULL</b>	All operations on the databank are performed without transaction control (even if they are requested within a transaction), and are not logged.

Note that operations performed on databanks with the TRANS or NULL option cannot be restored in the event of system failure (see Chapter 6).

If a databank is dropped, the contents of the databank are no longer accessible from any MIMER module. The physical databank file is automatically deleted in most systems.

### 3.3.1 System databanks

A number of **system databanks** are created during installation. In general, these databanks are not used for storing user-defined information and are not accessed directly by users.

There are four system databanks created when a basic MIMER system is installed:

<b>LOGDB</b>	records all write operations performed within transactions on databanks defined with the LOG option. The contents of LOGDB is used to restore the database from a backup copy after system failure (see Chapter 6).
<b>SQLDB</b>	stores the local read sets used in transaction handling (see the MIMER/SQL Programmers Manual, Chapter 6), and is used by MIMER/SQL modules for temporary storage of result tables.
<b>SYSDB</b>	stores the contents of the data dictionary (see above).
<b>TRANSDB</b>	stores the local write sets used in transaction handling (see the MIMER/SQL Programmers Manual, Chapter 6).

The contents of TRANSDB, LOGDB, and SQLDB are stored in an internal format and cannot be read by normal database access requests. The READLOG functionality (Chapter 7) allows the contents of LOGDB to be examined.

If SYSDB, TRANSDB, SQLDB, or LOGDB is damaged or missing, attempts to log on to MIMER will fail. Recovery procedures for damaged system databanks are described in Chapter 6.

Other system databanks are required by the optional MIMER modules:

<b>PROCDB</b>	is the default location for storing QL procedures, and may be used for other similar purposes, such as storing form library tables for FM.
<b>SYSFM</b>	contains screen forms and messages for the FM editor.
<b>SYSHELP</b>	contains the on-line help texts for Interactive SQL.
<b>SYSMSG</b>	contains various messages used by the MIMER modules.
<b>SYSPG</b>	contains error messages and help text for PG.
<b>SYSQF</b>	stores screen forms and application definitions for QF.
<b>SYSQL</b>	contains system information for QL.
<b>SYSRG</b>	stores all report definitions created with RG.
<b>SYSRGOUT</b>	is the default location for storing non-sequential report output from RG (see the MIMER/RG Reference Manual for further details).
<b>WORKDB</b>	is used by some modules for temporary work tables. All temporary tables in WORKDB are erased at the end of every MIMER session. (MIMER/SQL uses SQLDB, not WORKDB, for temporary tables).

### 3.3.2 Location of system databanks

In general, the location of databanks in a MIMER system is stored in the data dictionary, in the form of the file name as given when the databank is created. This is true for all system databanks *except SYSDB*.

SYSDB is stored in a file called SYSDB7. For both single-user systems and multi-user systems, the location of SYSDB7 is determined through the SQLHOSTS file (see Section 4.5). The location specified in SQLHOSTS also defines the default directory for other databank files, if no explicit directory is given when the databank is created.

### 3.3.3 Changing databank locations

User databanks may be moved by moving the file in the operating system and then changing the file name stored in the data dictionary with the ALTER DATABANK statement (see the MIMER/SQL Reference Manual for the statement syntax). An ALTER DATABANK statement may only be issued by the owner of the databank.

Facilities for relocating the system databanks except SYSDB are provided within the Backup Restore functionality (Chapter 6).

SYSDB is relocated by just moving the file and updating the SQLHOSTS file (see Section 4.5). Note that this may change the default directory for other databank files, if no explicit directory was specified when the databanks were created.

## 3.4 Other database objects

The remaining objects in the database (tables, views, domains, synonyms and indexes) are described in the MIMER/SQL Interactive Users Manual.

## 3.5 Database security

MIMER includes a sophisticated system of access rights and privileges, which permits detailed control of database security. The main components of the security system are:

- idents
- access rights and privileges
- restriction views

### 3.5.1 Idents in database security

Idents govern the right of access to the MIMER system as a whole. Careful advance planning of the hierarchical structure of idents in a system can be essential to the viability of the system. An unplanned ident structure can easily become impossible to overview and control after a relatively short period of system use.

The initial installation creates one user ident, the system administrator, with the ident name SYSADM. The system administrator has DATABANK and IDENT privileges with GRANT OPTION, and has SELECT access on all tables in the data dictionary, also with GRANT OPTION. The system administrator is ultimately responsible for the structure of the whole system.

Certain system functions may only be run by the system administrator. Note however that in other respects the system administrator is an ordinary user ident in the system. It is quite possible (and may be advisable especially in large systems) that the system administrator does not in fact have access to the actual contents of the database; the administrator's job is concerned with objects in the system, not the actual data.

The initial installation of MIMER also includes a global group ident called PUBLIC. All idents in the system are automatically members of the PUBLIC group, which may thus be used for granting global privileges.

The following general recommendations can be made for structuring the idents in a system:

- Assign functions within the system to program idents. These are not coupled to any physical individual or group of individuals, and thus have a lifetime independent of turnover of personnel. (The system administrator is such a function, but is coupled to a user ident rather than a program ident for practical purposes).

- Create user idents for physical users of the system. These may be dropped if the person concerned leaves the company. Do not grant privileges directly to user idents, other than membership of groups. Administration is simpler if privileges are granted through groups.
- Use group idents to represent logical users of the system. Grant privileges to groups rather than to individuals. This makes it easier to organize and overview privilege structure within the system, and also means that new idents can easily be granted suitable privileges by use of one or more group memberships.
- Grant the privilege to create objects (DATABANK, IDENT and TABLE privileges) to program idents only. In this way, individual user idents may be dropped with no recursive effects (see Section 3.5.3). (Creation of domains requires no special privilege and may thus be performed by any ident. Creation of views requires only SELECT access to the table on which the view is based).
- Use GRANT OPTION sparingly and try to minimize the number of levels in the ident hierarchy. This reduces the risk of recursive revocation of privileges (see Section 3.5.3).

Following these recommendations will simplify maintenance of the ident structure in the system. Access to the contents of the database is granted to relatively few group idents instead of many individual programs or users, and when a physical individual leaves the company, the corresponding user ident can be dropped with no recursive consequences.

### 3.5.2 Access rights and privileges

Each ident is given privileges within the system defining the operations the ident is allowed to perform. Privileges may be granted either directly or through membership in a group. The privileges are grouped as follows:

**System privileges** give the right to create global objects within the database. There are two system privileges:

DATABANK gives the right to create databanks  
 IDENT gives the right to create idents.

System privileges are granted to SYSADM at installation and may be passed on to other idents with or without grant option. An ident receiving a privilege with grant option may pass the privilege on to another ident.

**Object privileges** give rights over certain specified objects in the system. There are three object privileges:

TABLE gives the right to create tables in a given databank  
 EXECUTE gives the right to enter a given program ident  
 MEMBER grants membership in a specified group ident.

Object privileges are initially granted only to the creator of the object (e.g. the creator of a databank automatically has TABLE privilege on the databank). The privileges may be passed on to other idents with or without grant option.

**Access privileges** give rights of access to the contents of a specified table. There are five access privileges

SELECT	gives the right to read the table contents
INSERT	gives the right to add new rows to the table
DELETE	gives the right to remove rows from the table
UPDATE	gives the right to change the contents of existing rows in the table (this privilege may be limited to specified columns within the table)
REFERENCES	gives the right to use the primary or alternative key of the table as a foreign key from another table (this privilege may be limited to specified columns within the table). This facility applies only to tables created with SQL.

In addition to the five specific privileges listed above, ALL may be used as a shorthand method of specifying all the privileges possessed by the current ident. For example, if you have only SELECT and UPDATE privileges on a table and you grant ALL on that table to a new ident, the new ident will only be given SELECT and UPDATE.

Note: For reasons of backward compatibility, a LOAD privilege on tables is recorded in the data dictionary. LOAD privilege is not required for any explicit user operation in MIMER version 7.

Access privileges are initially granted only to the creator of the table. The privileges may be passed on to other idents with or without grant option, except UPDATE privilege on a specified column which may not be granted with grant option.

Certain operations are not controlled by explicit privileges, but may only be performed by the creator of the object involved. These operations include ALTER (with the exception of ALTER IDENT, which may be performed by either the ident himself or by the creator of the ident), DROP, and COMMENT. Similarly, privileges may only be revoked by their grantor.

### 3.5.3 Recursive interactions between privileges

Dropping an object from the database or revoking a privilege from an ident may have recursive effects on other objects and idents, depending on the way the database is organized. The keywords CASCADE and RESTRICT may be used in the DROP and REVOKE statements. When using RESTRICT the operation is inhibited if any recursive effects will occur. Otherwise (CASCADE is default) the following operations have implicit consequences:

- If an ident is dropped, all objects created by the ident are dropped and all privileges granted by the ident are revoked.
- If a databank is dropped, all tables in the databank are also dropped.
- If a table is dropped, all views based on the table are dropped.
- If a privilege with GRANT OPTION is revoked from an ident, all instances of that privilege granted to other idents under the authorization of that GRANT OPTION are also revoked.

- If SELECT privilege on a table is revoked from an ident, views created by the ident under the authorization of that SELECT privilege are dropped.

The recursive effects of revoking privileges depend on the temporal order in which the privileges are granted and revoked. An ident grants privileges, creates views and so on under the authorization of the most recent valid instance he has received of the GRANT OPTION, SELECT or appropriate privilege. The data dictionary keeps a record of the authorization under which privileges are granted. The recursive effects apply only to privileges granted or objects created under the authorization of the particular instance being revoked. This is illustrated as follows:

#### CASE 1

1. A grants with grant option to M
2. M grants to X
3. B grants with grant option to M
4. M grants to Y
5. A revokes from M X loses privilege (authorization A)  
Y keeps privilege (authorization B)

#### CASE 2

1. A grants with grant option to M
2. M grants to X
3. B grants with grant option to M
4. M grants to Y
5. B revokes from M X keeps privilege (authorization A)  
Y keeps privilege (authorization A)

#### CASE 3

1. A grants with grant option to M
2. B grants with grant option to M
3. M grants to X
4. M grants to Y
5. A revokes from M X keeps privilege (authorization B)  
Y keeps privilege (authorization B)

#### CASE 4

1. A grants with grant option to M
2. M grants to X
3. B grants **without** grant option to M
4. M grants to Y
5. A revokes from M X loses privilege (authorization A)  
Y loses privilege (authorization A)

### 3.5.4 Restriction views

Views are a powerful tool for restricting user access to defined parts of the database, and complement the system of access privileges in maintaining database security. By defining restriction views (i.e. views based on one table but restricted only to specific rows and/or columns in the table), access may be granted to a subset of a tables contents without affecting the physical database structure. In this way, the database may be designed optimally according to the relational model, while user access can be specified according to the needs of the actual data.

## 3.6 Data integrity

The following facilities are available for ensuring the integrity of a MIMER database:

- domains
- entity integrity (non-NULL primary keys)
- referential integrity (foreign keys)
- table integrity
- view integrity

### 3.6.1 Domains

Domains define sets of permissible values. By assigning a table column to a domain when the table is created, the values which the column may hold are restricted to the set defined for the domain. As many columns as are required may belong to any given domain.

Domains may also be defined with a default value, which is inserted into the column belonging to the domain when no explicit value is given. If the default value is defined outside the range of restriction values, attempts to insert the default will fail, forcing insertion of explicit data into the column.

The use of domains in table definitions is recommended, since this can provide an automatic check on the validity of data entered for the column. However, domain definitions should be carefully planned, since a domain cannot be altered once it is defined.

### 3.6.2 Entity integrity

Entity integrity refers to the requirement that every row in a table must be uniquely identified and that no row in a table may be identified by NULL (i.e. by an unknown value). According to E. F. Codd, entity integrity should be enforced in all true relational database systems.

All primary key columns in MIMER tables defined with the CREATE TABLE statement in SQL or with corresponding statements in QL and PG are defined as NOT NULL, thus ensuring entity integrity. Other columns may also be defined as NOT NULL as required by the nature of the data.

### 3.6.3 Referential integrity

Referential integrity refers to the requirement that data entered in one table in the database must already be present in another table (e.g. a component may not be entered in a parts list if it does not exist in the component list). MIMER/SQL supports referential integrity through the optional FOREIGN KEY clause in the CREATE TABLE statement. The properties of a FOREIGN KEY are as follows:

- Data inserted into the foreign key columns (by either INSERT or UPDATE operations) must either already be present in the primary key or an alternate key of the reference table or include at least one NULL column.
- The columns defined as a foreign key must correspond exactly in number and definition to the primary key or alternate key columns in the reference table.
- Rows may not be deleted from the reference table and alternate key columns in the reference table may not be updated if the values of the primary key columns are present in any foreign key.

The use of foreign keys should be carefully planned, since referential integrity constraints may not be added to or removed from existing tables. If referential integrity is to be added to an existing database structure, the tables containing the foreign keys must be dropped and redefined.

Note: Tables with foreign key definitions may not be stored in databanks defined with the NULL option.

### 3.6.4 Table integrity

Table integrity refers to the facility in MIMER/SQL of defining CHECK-clauses in table definitions, whereby the contents of one column is checked against the contents of one or more other columns in the same row of the table. Data may only be entered into the table if the CHECK-constraint is not violated.

CHECK-constraints may not be added to or removed from existing tables.

### 3.6.5 View integrity

View integrity refers to the facility in MIMER/SQL of including a WITH CHECK OPTION clause in view definitions. If a view is defined WITH CHECK OPTION, data which violates the definition of the view may not be entered into the view by INSERT or UPDATE operations.

The CHECK OPTION is inherited by views defined on other views which in turn are defined with CHECK OPTION.

## 4 ESTABLISHING A MIMER SYSTEM

### 4.1 General information

Procedures for installing MIMER software differ between the different machines, and are described in the machine-specific Installation Guide.

Once the software is installed, the database environment must be established. This is performed in three stages:

- generation of the central system databanks SYSDB, TRANSDB, LOGDB and SQLDB, and the creation of the system administrator ident SYSADM
- generation of module-specific system databanks
- establishment of the database and ident structure for the system.

Some details of the procedures for generating the system databanks are machine-dependent, and are listed in the machine-specific Installation Guide. This chapter gives a general description of the generation process.

Central system databanks are generated with program generically referred to as SDBGEN (the actual program name may differ on different platforms). This step must be performed before any other part of the database environment can be established.

Module-specific databanks are generated from the 'SYSxxGEN' option in the Export/Import functionality (see Chapter 5), run by the MIMER system administrator SYSADM.

Once the system databanks have been set up, the ident and user databank structure is established using the data definition statements in MIMER/SQL.

## 4.2 Creating the central system databanks: SDBGEN

The central system databanks SYSDB, TRANSDB, LOGDB and SQLDB are generated with the SDBGEN program (the actual program name may differ on different machines - see the machine-dependent Installation Guide).

**Note:** **SDBGEN is intended for initial databank creation only. It cannot be used directly for upgrading existing MIMER installations, since SDBGEN will not run if any system databanks already exist. If it is to be used to re-generate the system databanks, the old ones must be removed first.**

Below is an example session for SDBGEN. The general format of the functionality is the same on all machines, although the details of user-specified file names and databank sizes will vary (see the machine-specific Installation Guide).

The password given for SYSADM is not echoed on the screen when it is entered, for security reasons. It should be chosen carefully and changed at regular intervals using the MIMER/SQL ALTER IDENT statement.

The size for the databanks is given in MIMER pages. The size of a MIMER page is 2048 bytes.

**SDBGEN**

```

Initial BOOT, version 7.3.1

Filename for System Databank is  SYSDB7
Initial size for Databank          : 250
Filename for Transaction Databank  : TRANSDB
Initial size for Databank          : 250
Filename for Log Databank          : LOGDB
Initial size for Databank          : 250
Filename for SQL Work Databank     : SQLDB
Initial size for Databank          : 100

System databanks created, defining password for system user

Password for user SYSADM  : *****
Verification              : *****

Defining ODBC views
Defining MIMER system views
Defining INFORMATION_SCHEMA views

System databanks successfully created.
```

### 4.3 Creating module-specific system databanks

Module-specific system databanks are created through the SYSxxGEN option in the Export/Import functionality (this is because the system tables are loaded with data by an Import operation). The system databanks are described in Section 3.3.1. Selecting this option from the Export/Import main menu gives the following menu:

```

-- SYSxxGEN --

1. SYSMMSG
2. SYSQQL
3. SYSRGG
4. SYSQFF
5. SYSPGG
6. SYSSH
7. SYSFMM
8. SYSHELP

0. Exit

```

Selecting any one of the options presents an additional menu:

```

-- SYSxx options --

1. Generate databanks
2. Drop databank(s)
3. Change databank file name

0. Exit

```

#### Option 1 – Generate databanks

Use this option to create the databanks in a new system. The program will ask for the name of the file where the databank is to be stored, and the initial size of the databank in MIMER pages. The databank is then created and the system tables defined and loaded by the program. The file containing data to be loaded into the tables is accessed automatically, provided it is stored in the default location (see the machine-dependent Installation Guide for the location of respective files in your system).

Some modules have more than one system databank. All are created in a single operation within the respective SYSxxGEN function.

System databanks and the tables therein are owned by system program ids specific to each module. These ids are entered automatically by the SYSxxGEN functionality, and are invisible to the user.

System databanks for optional modules which are not installed appear on the menu and may be created, but are of little use since neither the data file nor the routines for using the databanks are present in the installation.

#### Option 2 – Drop databank(s)

This option deletes the module-specific system databank(s) from the system. If the module is to be used again, the system databank(s) must be re-created.

### Option 3 – Change databank file name

This option relocates the databank from the original file to a new location. It is equivalent to the ALTER DATABANK ... INTO FILE statement in MIMER/SQL. Relocation of system databanks must however be performed through use of the SYSxxGEN functionality since only the databank creator (the system program ident) is authorized to execute an ALTER DATABANK statement.

For SYSRG, an additional option '**Reallocate SYSRGOUT**' is available. This option drops the existing SYSRGOUT databank and creates a new, empty SYSRGOUT databank. The function is useful in situations where SYSRGOUT has grown as a result of a large report output which is no longer required.

#### 4.3.1 Upgrading existing system databanks

The SYSxxGEN functionality is also used for upgrading the existing system databanks following a new release of MIMER, by simply choosing to generate the databank in question.

Data in the upgraded export file for the system databank is loaded into the existing system tables. All system data rows are deleted from the existing tables *before* the new rows are inserted, so that rows in the upgraded data replace the duplicates in the existing tables. You are therefore advised to make a backup copy of the system databank before performing an upgrade. Any rows containing user data remain unchanged.

Note that although upgrading existing system databanks is an import operation, it does not follow the normal operation of the import functionality as applied to user databanks (see Chapter 5). Normally, importing duplicate table definitions into a user databank reports naming conflicts which the operator resolves by providing new table names. No such conflicts are reported when system databanks are upgraded. Similarly, duplicate rows in the imported data set are ignored in user databanks, whereas it is the duplicates in the existing tables that are ignored when system databanks are upgraded.

## 4.4 Establishing the database

Once the MIMER system has been installed, the database environment (idents, databanks, tables, and so on) should be created using SQL statements. Both interactive and batch SQL modes allow execution of a sequential file, which can then be used as permanent documentation of the CREATE statements used to establish the database (see the MIMER/SQL Interactive Users Manual). An application program employing embedded SQL can also be used, but requires more work on the part of the programmer and gives less concise documentation of the database structure.

Note: A sequential file intended for non-interactive execution through the batch SQL facility will include the username and password for the user creating the objects (often the system administrator). For security reasons, the batch file should be well protected in the operating system, and the username and password preferably edited out of any permanent copy of the file.

## 4.5 Managing database connections

This section describes how users connect to a database and how several simultaneous connections can be handled.

The following SQL statements are used for connection management (see the MIMER/SQL Reference Manual for details):

```
CONNECT
DISCONNECT
SET CONNECTION
```

### 4.5.1 Database names

Applications establish database connections with the `CONNECT` statement, which specifies the database by name. A database corresponds to a system databank and all the objects referenced therein. A database can be accessed either in single- or multi-user mode according to the system used by the application program (see Chapter 2). Several connections may be made in an application, but only one is active at any one time. The currently active connection is changed with the `SET CONNECTION` statement.

The database may be located on the same machine as the application program, or on a remote machine accessed over a network. The network connection is handled by MIMER/SQL software and is completely transparent to the application program.

#### Resolving database names

MIMER/SQL system uses a sequential file, generically referred to as 'SQLHOSTS', to resolve the database name in terms of a physical SYSDB location. This file specifies the *directory* or equivalent (not the file name) containing the respective SYSDB7 files for the accessible database systems. Refer to the machine-specific Users Guide for the name and location of the SQLHOSTS file.

The SQLHOSTS file contains three sections, specifying default, local and remote database names. Database names are case sensitive.

An example SQLHOSTS file from a VMS environment is given below:

```

DEFAULT:
    HOTEL

LOCAL:
    HOTEL      DISK32:[HOTEL]
    TEST      TESTDB:

REMOTE:
    PRODDB    node3      tcp
    NETTEST   NODEA      decnet

```

The DEFAULT section identifies the database which is to be used as the default (i.e. the database which is addressed by the statement CONNECT TO DEFAULT and by application programs like MIMER/ISQL). Note that the default database is specified as a name, not as a physical file location. The name of the default database should be resolved in either the LOCAL or the REMOTE section.

The LOCAL section lists available databases on the local machine. Each entry specifies a database name with the physical location of the SYSDB databank for the database. The location may be given as a logical name in systems where this is supported (e.g. TESTDB: in the example above).

The REMOTE section lists available databases on remote machines. Each entry specifies a database name and the network node on which the database may be found. Additional network information may be included (see the machine-specific Users Guide for details). The physical location of the database is mapped by an entry in the LOCAL section of SQLHOSTS on the relevant remote machine. Thus in the example above, the SQLHOSTS file on node3 should contain a LOCAL entry specifying the location of the database named PRODDB.

### Alternative default databases

The SQLHOSTS file specifies one default database per machine. This may not be sufficient at times. Individual users can override the default database in the SQLHOSTS by setting the environment variable (or logical name in VMS) MIMER\_DATABASE to the required database name.

## 4.5.2 Connection names

Whenever an application connects to a database a connection name is associated with the connection. This name may be specified explicitly in the CONNECT statement. Otherwise the connection name is the same as the database name. Several connections can be made to the same database by specifying different connection names. Connection names are case-sensitive.

The connection name is used in the SET CONNECTION statement for switching between different connections, and in the DISCONNECT statement for closing connections.

When the application switches from one connection to another the previous connection becomes dormant. A dormant connection cannot be accessed until the connection is made current with SET CONNECTION statement. A dormant connection may always be closed down with a DISCONNECT statement without making it current first.

When a DISCONNECT CURRENT has been performed the application has no current connection. The only valid SQL statements in this situation are CONNECT (to establish a new connection), SET CONNECTION (to make a dormant connection active) or DISCONNECT (to close another dormant connection).



## 5 EXPORT/IMPORT

The Export/Import functionality provides facilities for:

- loading and unloading data between database tables and sequential files
- moving tables from one database system to another, either on the same platform or between different platforms
- generating system databanks for the optional MIMER modules.

The functionality is controlled through a series of menus. These are displayed sequentially at the terminal so that it may be run from a console or printing terminal.

Any MIMER ident may start the Export/Import functionality from the integrated Utilities program (see the machine-dependent Users Guide for details). Privileges required for the separate functions supported by the Utilities program are considered together with the detailed function descriptions below.

### 5.1 The main menu

Logging on to the Export/Import functionality presents the main menu shown below:

```
-- Export/Import utility --  
  
1. Export - definitions and data  
2. Export - definitions only  
3. Import - object creation  
4. Import - data load  
5. Load / Unload table  
6. SYSxxGEN  
7. Enter program ident  
8. Leave program ident  
  
0. Exit
```

The functions are described in detail below.

## 5.2 Export options

### 5.2.1 Functions

From the main menu, the user may export table definitions with or without data. The exported objects and data are stored in sequential files, which may later be imported to re-create the tables in another MIMER system, on either the same or a different platform.

If columns in tables being exported belong to domains, the appropriate domain definitions are exported together with the table definitions. Indexes defined on exported tables are also exported.

Both choices ('Export - definitions and data' or 'Export - definitions only') display a second menu giving the following options:

```
-- Export --  
  
1. Export specific tables  
2. Export all tables for ident  
3. Export all tables in databank  
  
0. Exit
```

Depending on the selection made, the user is prompted for a list of table names (the list is terminated by the first 'empty' or blank name), an ident name, or a databank name. Specific tables exported in one operation do not need to be owned by the same ident or stored in the same databank. In the case of tables selected by ident or databank, a single set of tables will be exported by each operation.

For all export functions, the user is prompted for the name of the sequential file to be used for export.

### 5.2.2 Authorization

The ident performing an export operation must have `SELECT` privilege on all the tables being exported. The 'Enter program ident' option in the main menu provides a facility for specifying an alternative ident to use while performing the export operation. The user running the Export/Import functionality must have `EXECUTE` privilege on the program ident in order to enter the program.

If export is performed for an ident or a databank, any tables to which the ident running the functionality does not have `SELECT` privilege are ignored.

### 5.2.3 Exported files

Table, domain, and index definitions are exported in the form of CREATE statements for re-creating identical objects in the import environment. The CREATE statements may be modified, if desired, with a text editor before the objects are re-created with an import operation. See the MIMER/SQL Reference Manual for the syntax and description of the relevant CREATE statements. If the export file is edited in any way, it is essential that the maximum record length of 80 bytes is not exceeded.

Note: The import function allows naming conflicts to be resolved interactively during import. Therefore, it is not necessary to edit the exported file in order to resolve naming conflicts.

Table contents are exported in sequential format. Modification of exported data with a text editor is possible but is not recommended.

In cases where exported definitions or data are to be moved between platforms which use different character representations, the file must be processed with a character set translation utility. Export files do not use any non-printable control characters except carriage-return, line-feed and end-of-file.

Export files begin with system information concerning MIMER version, machine and operating system, file record length and data storage formats, and so on. Statistical information at the end of the file records the number of data rows exported for each table.

## 5.3 Import options

The import functions use the files generated by export as input in order to re-create the exported tables in a new environment. If the tables are being moved between platforms, the export files must be transferred physically from one machine to the other and, if necessary, processed with a character set translation utility. The character set translation functionality is not provided by MIMER.

The import operation is performed in two stages, object creation and data loading. These are represented by the options, 'Import - object creation' and 'Import - data load' in the Export/Import main menu, and must be performed separately even if the table definitions and data were exported together in the same file.

### 5.3.1 Import – object creation

This phase creates the tables being imported, together with any domains used in the table definitions and indexes defined on the tables. The newly created objects are owned by the ident performing the import operation. The 'Enter program ident' option in the main menu allows imported files to be processed by a program ident.

The user is prompted for the name of the export file to be used, and also for the name of a log file which will be used if any table names are altered from those given in the export file (see below).

Before the imported tables are created, the table-databank couplings in the export file are displayed, and the user is given an opportunity to redirect the tables to new target databanks. If a target databank is specified which does not exist in the import environment, the user is prompted for the filename, size, and options and a new databank is created. If the target databank name is left blank, MIMER will place the table in the databank which is judged best for the purpose (equivalent to using the SQL statement CREATE TABLE with no IN clause).

Once the table-databank couplings have been accepted, the tables are created from the CREATE TABLE statements in the export file.

The following example shows the creation of the import objects (user input is shown in bold):

```

-- Export / Import utility --

1. Export - definitions and data
2. Export - definitions only
3. Import - object creation
4. Import - data load
5. Load / Unload table
6. SYSxxGEN
7. Enter program ident
8. Leave program ident
0. Exit

Select: 3

-- Import - object creation --

File generated by Export/Convout : BOOKEXP

Import log file : BOOKIMP

Listing of couplings between tables and target databanks

TABLE NAME          DATABANK
-----
BILL                <---->   BOOKDB
BOOKFORM            <---->   BOOKDB
BOOK_GUEST          <---->   BOOKDB
EXCHANGE_RATE       <---->   BOOKDB
FREEROOMS           <---->   BOOKDB
ROOMSTATUS          <---->   BOOKDB

Are the couplings okay <Y>?      : Y

Making definitions
Operation completed

```

### Naming conflicts

Naming conflicts will arise during import operations if the names of imported tables or domains already exist in the import environment. If this occurs, the user may choose to rename the object in question, skip creation of the particular object, or quit the object creation operation.

If any tables are renamed during the object creation, the altered names are stored in the import log file from which they are read by the data load operation in the next phase of import.

Domain conflicts need only be resolved once for any import operation. All usage of the domain in the imported table definitions are corrected according to the response to the first reported conflict.

If the user chooses to quit the object creation operation at a naming conflict, any objects created prior to the naming conflict remain in the import environment.

The following example shows how to rename an object (user input is shown in bold). Note that the duplicate table name is shown with the creator name for clarity, but that the new name may not be qualified by a creator name (the ident performing the import operation owns the created tables).

```

-- Import - object creation --

File generated by Export/Convout   : BOOKEXP

Import log file                   : BOOKIMP

Listing of couplings between tables and target databanks

TABLE NAME                        DATABANK
-----
BILL                              <--->   BOOKDB
BOOKFORM                          <--->   BOOKDB
BOOK_GUEST                        <--->   BOOKDB
EXCHANGE_RATE                    <--->   BOOKDB
FREEROOMS                        <--->   BOOKDB
ROOMSTATUS                       <--->   BOOKDB

Are the couplings okay <Y>?      : Y

Making definitions
Duplicate table name              : BOOKADM.BOOK_GUEST

Skip/Quit/Rename <S/Q/R>? R

Give new name                    : BOOKGUEST2
Operation completed

```

### Referential conflicts

Referential conflicts can arise in several ways during an import operation:

depending on the structure of the exported database and the way tables are exported, an attempt may be made during import to create a foreign key for which the reference table does not exist in the import environment

- the order in which the tables were exported may result in an attempt during import to create a foreign key before the reference table is created
- an imported table may have a foreign key which refers to a table which exists in the import environment but which is the 'wrong' table (i.e. the primary key of the reference table is not consistent with the foreign key).

It is the responsibility of the user performing the import to avoid referential conflicts in the imported tables, either by importing tables in the correct order or editing foreign key definitions in the export file before the tables are imported. The import functionality does not provide any interactive facilities for correcting conflicts of this type. If referential conflicts do arise, the table(s) affected will not be created and the import will be aborted with a message informing the user of the error.

Note that in a particularly unfortunate circumstance, a table may be imported with a foreign key clause which happens to coincide with an existing table in terms of the reference table name and column definitions in the import environment, but which the user does not actually want to use as a reference table. This situation cannot be detected by the import functionality, since the foreign key reference is syntactically valid. Care is demanded of the user to ensure that this situation does not arise (the risk is minimized by intelligent use of table and column names).

### 5.3.2 Import – data load

This phase of an import operation loads the imported tables with data from the export file. The operation must be performed separately from the object creation phase. The user is prompted for the name of the export file and for the name of the log file specified during the object creation phase (see above). It is important that data load uses the same log file that was written when the objects were created. The user is also asked if duplicates should be logged. If a duplicate is encountered, a report is written to the session log and a summary appears at the terminal.

**Note:** Data loaded into tables by the import functionality must be stored in an export file. The import functionality cannot use data files created by the Unload function (see below).

If an error occurs during the data load operation, a message appears on the terminal and the record is logged. If 100 errors occur the import operation is aborted. Tables are loaded in transactions of at most 100 rows each. The progress of the loading is reported by a message every 100th row for the first 10000 rows and thereafter every 1000th row. If the import operation is aborted because of errors, the rows which have been reported as loaded will remain in the table.

**Hint:** If there is a large amount of data to be imported it can be done faster by setting the databank option to NULL before the load operation takes place. This is because transaction handling affects the performance of the load operation. Do not forget to set the databank back to the desired option after the load is finished.

The following example shows data being loaded from the export file BOOKEXP (user input is shown in bold):

```
-- Import - data load --  
  
File generated by Export/Convout   : BOOKEXP  
Log file from 'object creation'   : BOOKIMP  
  
Log duplicates <Y>? N  
  
Loading table : BOOKADM.FREEROOMS  
    100 rows loaded  
    195 rows loaded  
  
Loading table : BOOKADM.ROOMSTATUS  
    20 rows loaded  
Operation completed
```

### 5.3.3 Authorization

The ident performing the object creation phase of the import operation must have TABLE privilege on any databank in which a table is to be created, and also DATABANK privilege if new databanks are to be created. REFERENCES privilege on the appropriate table(s) is required if any foreign keys are to be created.

The ident performing the data load phase must have INSERT privilege on tables being loaded. Normally, the data load phase is performed by the same ident as the object creation phase, in which case INSERT privilege is automatically granted (since the ident owns the tables).

The 'Enter program ident' option in the main menu allows a user to act in the capacity of a program ident. The user running the Export/Import functionality must have EXECUTE privilege on the program ident in order to enter it.

## 5.4 Load and Unload functions

The load and unload functions allow data to be moved between MIMER tables and sequential files. The functionality here is identical to that provided by the LOAD and UNLOAD commands in Interactive SQL, although the interactive screen forms differ between the two environments. (The screens used when the functionality is run from the Export/Import menu are adapted to allow load and unload to be performed from a console or printing terminal).

The term 'load' refers to moving data from a file to a table.

The term 'unload' refers to moving data from a table to a file.

Choosing the load/unload function from the main menu displays another menu offering four alternatives:

```
-- Load / Unload --

1. Load from file INTO table, default format
2. Load from file INTO table, user specified format
3. Unload to file FROM table, default format
4. Unload to file FROM table, user specified format

0. Exit
```

### 5.4.1 Data file formats

The user may choose between a default or a user-specified format for the data file. In either case one row of table data (one tuple) corresponds to one record in the sequential file.

The default format allocates space for each column according to the definition of the column in the table, with no extra space between fields in the record.

If a user specified format is chosen, the user must give the start and end positions in the record for each column of the table. Fields in the record do not need to be contiguous. If fields overlap in an unload operation, data from the earlier columns in the table definition will be overwritten in the overlapping positions by data from the later columns.

Both character and numerical data is written to the sequential file in string format. Thus the number 143.6 is unloaded as the string ' 143.6', occupying 6 bytes (the leading blank is the sign position). Files for loading tables can thus be created with any text editor, and data unloaded from tables could be edited before being reloaded.

#### NULL values

Before the load or unload operation is performed, the user is asked if a preceding NULL byte is to be used.

If data is unloaded from a table with a preceding NULL byte, an extra byte is added before each field in the sequential record. This byte is assigned an ampersand (&) if the column from which the field is derived contains NULL. Otherwise the byte is blank. At unload, if the NULL byte contains an ampersand, the rest of the field is filled with periods (...).

If data is loaded into a table using a preceding NULL byte, the first byte of each field is read as a NULL flag (an ampersand in this byte indicates NULL; any other value indicates non-NULL). Note that if the preceding NULL byte is used for the load operation, each field must be one byte longer than the corresponding column. If the NULL byte contains an ampersand, the actual data content of the field is irrelevant.

When the preceding NULL byte is used with a user-specified file format, the starting position given for each field should be the position of the NULL byte. In this case the column data starts at position+1.

### 5.4.2 The load operation

The load operation transfers data from a sequential file to a table in the order of the records in the sequential file.

If default file format is used, data from the file record is mapped into the table columns using the definition and order of the columns in the table.

If the user-specified file format is used, the user has full control over where in the record the data for each column in the table is to be taken. The same positions in the record may be used for data inserted into as many columns in one table row as is required. Values for one row in the table may not, however, be taken from more than one record. Any columns in the table omitted from the user-specified file format will be assigned the NULL indicator or a default value as appropriate.

If a string is specified which is longer than the character column width, the error message "Input character string too long" appears.

The load operation is performed, conceptually, as a series of INSERT statements, one for each row in the table. If rows with duplicate primary key values exist in the loaded data (or if a loaded row duplicates an already existing row), only the first of the duplicates is retained in the table. The number of rows loaded and the number of duplicates found is reported when the operation is complete.

**Hint:** If there is a large amount of data to be loaded it can be done faster by setting the databank option to NULL before the load operation takes place. This is because transaction handling affects the performance of the load operation. Note that the databank option may not be NULL if foreign or unique constraints are involved (an error message will be generated). Do not forget to set the databank back to the desired option after the load is finished.

Data may not be loaded into views.

### 5.4.3 Unload operation

When data is unloaded from MIMER tables, records are written to the sequential file in the ascending primary key order of the table.

All rows are unloaded from the table. If a subset of rows is required, a view can be defined containing the required rows and the data unloaded from the view. (Alternatively, the whole table can be unloaded and selected records then deleted from the sequential file). The source table remains intact after the unloading.

Selected columns may be unloaded by choosing the user-specified format and listing the required columns. Only the columns listed in the file format will be unloaded.

The sequential file is created at the beginning of the unloading process. It is not possible to append unloaded data directly to a pre-existing file.

The number of rows unloaded is reported when the operation is complete.

#### **5.4.4 Authorization**

The ident performing a load operation must have INSERT privilege on the table into which the data is being loaded.

The ident performing an unload operation must have SELECT privilege on the table or view being unloaded.

### **5.5 SYSxxGEN (generate system databanks)**

#### **5.5.1 Functions**

The Export/Import functionality is used for creating, dropping, and relocating all system databanks except SYSDB, TRANSDB, LOGDB and SQLDB. The operation of the functionality to be used for the system databanks is described in Section 4.3.

#### **5.5.2 Authorization**

SYSxxGEN may only be used by the system administrator SYSADM.

### **5.6 Enter and Leave program ident**

If a program ident is the creator of a table, the program ident may be entered in advance in order to export or unload the table.

Use 'Leave program ident' to conclude the use of the program ident.

## 6 BACKUP AND RESTORE

This chapter discusses the three ways to backup MIMER databanks:

- by backing up the databanks from the host operating system
- by using the SQL system management functions
- by using the backup and restore functionality in the integrated Utilities program.

How to restore data after a crash is also discussed. It is recommended that one person has the responsibility for taking the backup of all the databanks in a database. Refer to Section 6.2 for instructions on the recommended procedures for handling databank backup and recovery.

Data need not be restored in the event of a power failure or system shut-down that does not damage the databank files, since any transactions that were committed but not completed at the time of the failure are automatically completed when the databank involved is next accessed.

Some of the discussion in this chapter refers to Shadowing, which is an optional MIMER product that allows one or more copies of a databank to exist on different disks. Shadowing provides greater protection from disk failure and allows databanks to be backed up while the system is in use (see *MIMER Shadowing Reference Manual*). The SQL system management functions can be used to backup and restore databanks whether you are using MIMER Shadowing or not, but the functionality that applies to Shadowing can only be used if you have purchased the optional Shadowing module.

Reference is made to transaction handling at several points in this chapter. If you are not familiar with transaction handling in MIMER see Chapter 6 of the *MIMER/SQL Programmer's Manual* or Chapter 6 of the *MIMER/SQL Interactive User's Manual* for a more detailed description.

### 6.1 Background information

A MIMER database consists of a collection of databanks (each in a separate file) containing tables with data used by the applications. The SYSDB system databank contains a data dictionary describing the different objects in the database. All operations on tables located in databanks with the LOG option are logged in the system databank LOGDB. In the event of a disk crash this information can be used to achieve a full restoration of the information lost from a destroyed databank.

The databank backup facilities delete the records from LOGDB which apply to the information recorded in the databank backup after it has been taken. LOGDB therefore contains a record of all the information that has been added to a databank file since the last backup was taken for it. It is possible to take a backup of a databank without clearing the LOGDB records, however it is usual practice to clear the LOGDB after backing up a databank.

Thus, in the event of a databank being destroyed by machine failure, LOGDB can be used to restore any changes made between the last backup operation and the time when the databank was lost (assuming a proper backup copy exists for that databank). It is important to note that data handling operations that are not logged cannot be restored in the event of a hardware failure.

---

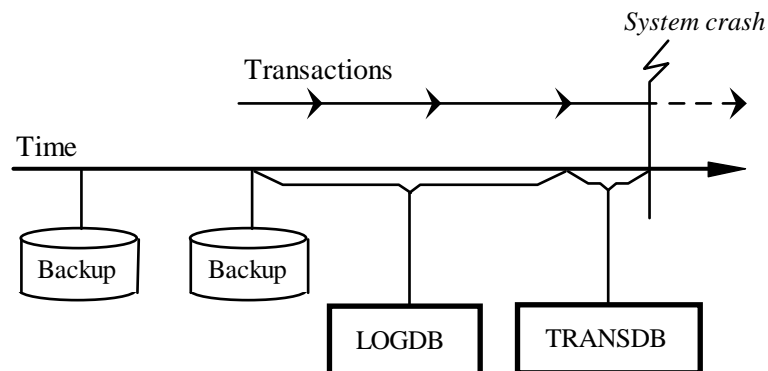
**Important!** Wherever possible, LOGDB should be stored on a different physical device from the rest of the database, to minimize the risk that a disk crash destroys both the log and the other databanks. For additional protection, LOGDB can be placed on a disk with a separate disk controller from the other databanks. This will prevent a damaged disk controller from destroying both the LOGDB and databank disks.

---

If a MIMER system is stopped (either deliberately or by a system failure) during the commitment of a transaction, information is retained in the TRANSDB system databank and used to complete the transaction on the next occasion the databanks involved are opened. This is only true for databanks with the TRANS or LOG option. Once information has been successfully written to both LOGDB and the databank file, it is removed from TRANSDB.

LOGDB and TRANSDB should be stored on different devices, since data may be lost if both TRANSDB and LOGDB are destroyed while a committed transaction is being processed.

The following diagram illustrates the relative functions of LOGDB and TRANSDB in crash protection.



### 6.1.1 Database consistency

Database consistency is handled on two levels: physical and logical.

*Physical consistency* means that the tables are readable by MIMER/DB. This is ensured as long as the databank file is not physically damaged.

*Logical consistency* means that the tables contain correct data and no transactions are incomplete. This is ensured by MIMER's transaction handling. All transactions are saved in the TRANSDB databank during build-up and are performed on the databanks when they are committed. To use transaction handling the databank must be created with the TRANS or LOG option.

Transaction handling makes it possible to ensure that a user's operations do not read data that is being updated by another user. If a transaction is successfully committed then all operations in the transaction are performed. If the transaction is aborted due to a conflict, none of the operations in the transaction are performed.

The tables may be logically inconsistent if MIMER is stopped before all operations in a committed transaction have been performed. After the system is restarted all uncompleted transactions will be read from TRANSDB and automatically performed to get the system into a consistent state again. This happens on a per-databank basis, when a databank is first accessed following the restart. A transaction is fully completed when all the involved databanks have been opened. Transactions that were not committed before the stop are aborted.

The DBOPEN facility (see Chapter 11) can be used to open all databanks in one operation and thus achieve transaction consistency quickly.

Backup copies of a databank should always be taken when the databank is in a logically consistent state, that is, no uncompleted transactions should exist. If a transaction updates two databanks that depend on each other, both databanks should be in a logically consistent state when the backups are taken. This is best accomplished by stopping the MIMER multi-user system and then running the single-user version of DBOPEN before making a backup.

### 6.1.2 Databank backups

A databank **backup** is a complete copy of a databank file.

It is the starting point for any restore operation, and should be stored in a safe place separate from the working databank files (copied to a different disk or preferably written to a backup tape or floppy disks and removed from the machine). A databank **backup** covers all the updates made to the databank, including all those updates recorded in the LOGDB databank file at the time the backup is taken.

This type of backup can be taken either by using the SQL system management functions or by using the host file system.

After a **backup** is taken, the updates logged for the databank in question should be cleared from LOGDB. This will be done automatically when the SQL system management functions are used to take the backup. If the host file system is used, the databank (or the whole database) must be set offline before the backup is taken and then set online again afterwards with the **reset log** option to clear LOGDB.

Refer to Section 6.1.6 for more information on use of the SQL system management functions.

### 6.1.3 Databank incremental backups

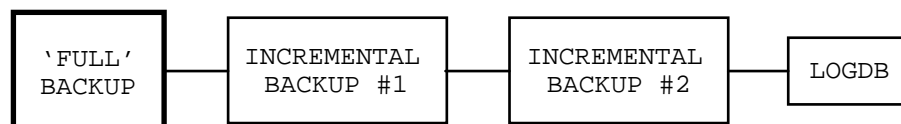
A databank **incremental backup** is a copy of all the updates recorded for it in the LOGDB system databank file.

This type of backup cannot be performed using the host file system, the SQL system management functions must be used.

An **incremental backup** is a record of all the updates made to the tables in a databank since the last **backup** or **incremental backup** was taken (or more precisely, since the updates logged in LOGDB for the databank were last cleared). Note that an **incremental backup** is not a copy of the databank file, the information comes from the updates logged in LOGDB, not from the databank file.

**Incremental backups** are economical in both time and storage space, but it is essential for the backup function that the chain of **incremental backups** is intact from the most recent **backup** of the databank.

A databank may thus be protected by a sequential chain of **incremental backups**, originating from a **backup** of the databank and ending with the updates currently recorded in LOGDB for the databank:



**Backups** taken by the host file system and those taken by using the SQL system management functions are equally valid as the origin for a chain of **incremental backups**.

The **incremental backups** can only be taken using the SQL system management functions.

### 6.1.4 Backup versus Incremental Backup

Any sequence of databank backups must start from a **backup**, which is a complete copy of the databank file. After that the choice about whether to take an **incremental backup** or a **backup** comes down to a trade-off between the time it takes to restore the database in the event of a failure, versus the risk that the contents of a databank file may have been recorded after it got into an invalid state.

A **backup** is a complete copy of the contents of a databank file. Everything in the databank will be copied, including any data corruption that may exist. Therefore, there is some risk that this type of backup may record data that cannot be used to restore a useable databank. Taking this kind of backup is likely to be slower than taking an **incremental backup** because a much greater volume of information is typically involved. Restoring a databank from it, however, will be the fastest choice because the databank will be almost completely restored in one operation rather than the many operations involved in a restore from **incremental backups**.

An **incremental backup** is a record of updates made to the tables in a databank, which have been logged in LOGDB. When this type of backup is used to restore a databank, the updates are re-applied sequentially to the tables in the databank. Therefore, provided the restore of **incremental backups** starts from a valid backup copy of a databank file, this type of backup can be used to recover from a data corruption situation. Taking an **incremental backup** is likely to be faster because it only records the table updates made since the last **backup** was taken, instead of the entire contents of the databank file. It will, however, take longer to recover a databank by restoring it from a series of **incremental backups** because the table updates must be re-applied one at a time.

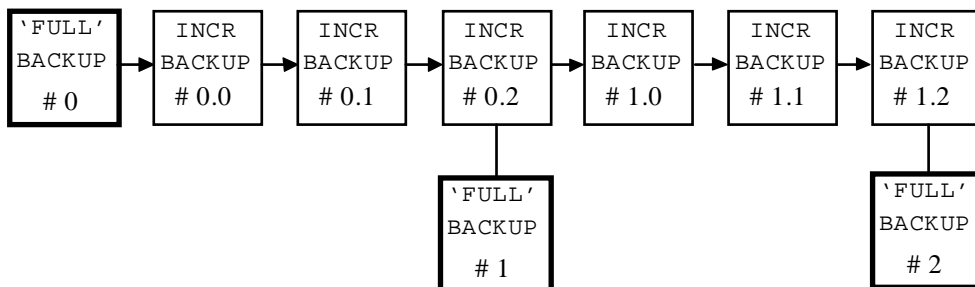
### 6.1.5 Maximizing data security

The scheme that offers the best security, whilst retaining the possibility of a short recovery time, is one that involves taking an **incremental backup** at regular intervals (e.g. daily), with a **backup** being taken regularly at some slightly longer interval (e.g. weekly) at the same time as an **incremental backup**.

The suggested backup scheme is illustrated below.

Note that the **incremental backups** are not replaced by a **backup** at the 'full' backup interval, but a **backup** is taken in addition to the incremental backup. In this way the **incremental backup chain** remains unbroken.

The **backups** serve as potential starting points for a restore operation. If the most recent **backup** proves to be unusable because it recorded the databank file after it got into an invalid state, an earlier **backup**, plus the intervening **incremental backups** (which protect against data corruption), can be used to restore the databank to the same point in time.



### 6.1.6 SQL system management functions and databank backups

Refer to the *MIMER/SQL Reference Manual* for a detailed description, and syntax definition, of the SQL system management functions. A brief description of the purpose of each function appears here.

The SQL system management functions which can be used to take backups are:

**CREATE BACKUP** used to take a **backup** with an optional **incremental backup** being taken at the same time.

**CREATE INCREMENTAL BACKUP** used to take an **incremental backup**.

The user using these functions to take a databank backup must either be the creator of the databank, or have EXECUTE privilege on MIMER\_BR.

When the SQL functions are used to take a **backup** and/or an **incremental backup**, the entire process of taking a databank backup is handled automatically.

If a databank **shadow** is available and accessible, the databank backup functions will use this in preference to the databank itself.

The process used by the databank backup functions is as follows: the databank, or the shadow if this is being used, is set **offline**, then the **backup** and/or **incremental backup** information is copied to the specified backup file(s) and finally the databank, or the shadow if this is being used, is set **online** again, with the **reset log** option.

Note that the databank backup process will always automatically clear the LOGDB records when the databank, or its shadow, are set online again. If you want to take a **backup** and an **incremental backup** at the same time (as recommended in Section 6.1.5), you **must** use the CREATE BACKUP function to take both at once.

The SQL system management functions (typically used when taking databank backups using the host file system) which can be used set a databank, shadow or the whole database **online** or **offline** are:

SET DATABANK OFFLINE	sets a databank <b>offline</b> , making it unavailable.
SET DATABANK ONLINE	sets a databank <b>online</b> , making it available, optionally clearing records from LOGDB.
SET DATABASE OFFLINE	sets the entire database <b>offline</b> , making it unavailable.
SET DATABASE ONLINE	sets the entire database <b>online</b> , making it available, optionally clearing records from LOGDB.
SET SHADOW OFFLINE	sets a list of shadows <b>offline</b> , making them unavailable.
SET SHADOW ONLINE	sets a list of shadows <b>online</b> , making it available, optionally clearing records from LOGDB.

The user using these functions to set a databank or a shadow of it online or offline must either be the creator of the databank or have EXECUTE privilege on MIMER\_BR.

A user setting the database online or offline, must have EXECUTE privilege on MIMER\_BR and must be the only user accessing the database.

The SQL system management function which can be used to recover a databank in the event of it being damaged or destroyed is:

ALTER DATABANK RESTORE	used to restore a databank from databank backups and/or from information in LOGDB.
------------------------	--

A user using this function to restore a databank must be the creator of a databank or have EXECUTE privilege on MIMER\_BR.

## 6.2 Backup and restore of databanks

This section describes procedures for taking databank backups and procedures for restoring a databank in the event of it being damaged or destroyed. The procedures described here do not apply to the system databanks, these are dealt with in Section 6.3.

Refer to the *MIMER/SQL Reference Manual* for a full description of the SQL system management functions.

### 6.2.1 Making a databank backup

It is possible to take a databank backup using either the SQL system management functions or by using the host file system.

#### 6.2.1.1 Using the SQL system management functions

The procedure for taking a databank backup of a single databank using the SQL system management functions is as follows:

To take a **backup** of a databank, use:

```
CREATE BACKUP IN 'file-name'
FOR DATABANK databank-name
```

This will copy the entire contents of the specified databank file to the specified file-name. If there is a shadow available, this will be copied in preference to the databank itself. During the backup process, the databank (or the shadow if that was used) will be automatically set offline. At the end of the backup process, the databank (or shadow if that was used) will be automatically set online again and records in LOGDB, which apply to the databank, will be cleared.

To take an **incremental backup** of a databank, use:

```
CREATE INCREMENTAL BACKUP IN 'file-name'
FOR DATABANK databank-name
```

This will copy the information contained in the LOGDB records for the databank to the specified file. At the end of the backup, the LOGDB records which have been copied will be cleared from LOGDB.

To take both a **backup** and an **incremental backup** at the same time, to maximize the security of the databank data (see Section 6.1.5), use:

```
CREATE BACKUP IN 'file-name-1'
INCREMENTAL BACKUP IN 'file-name-2'
FOR DATABANK databank-name
```

This will copy the entire contents of the databank file to the file specified by file-name-1 and will also copy the information contained in all the LOGDB records for the databank to the file specified by file-name-2. During the backup process, the databank will be automatically set offline. At the end of the backup process, the databank (or shadow if that was used) will be automatically set online again, and LOGDB records which apply to the databank will be cleared from LOGDB.

Note: The file names specified in connection with these functions are recorded in the database (in MIMER.BACKUP) for use by the backup and restore functionality in the Utilities program. This behavior may, however, be changed in the future so it should not be relied on. The person taking the backup is responsible for keeping a record of the backup file names.

Databank backup file names are subject to the same restrictions that apply to the SQL function CREATE DATABANK - see the *MIMER/SQL Reference Manual*.

### 6.2.1.2 Using the host file system

You can use the host file system to take a **backup** of a specific databank, or to take a **backup** of all the databanks in the database (see Section 6.3 for specific information regarding backing up the system databanks).

Note: it is recommended that the SYSDB system databank be included in a backup of all the databanks, therefore you must stop the multi-user system to close the SYSDB databank file. Note also that setting SYSDB offline does **not** close the databank file.

You can **not** use the host file system to take an **incremental backup**, you can only make copies of entire databank files in the same way as taking a **backup** using the SQL system management functions.

To take a **backup** of the whole database and clear LOGDB entirely, do the following:

- set the database **offline** using the following command:  

```
SET DATABASE OFFLINE
```
- stop the multi-user system if you wish to include SYSDB in the backup
- make backup copies of the databank files (on disk or tape). Normally, all databank files should be backed up at the same time
- start the multi-user system again, if it was stopped in an earlier step
- set the database **online** again using the following command:

```
SET DATABASE ONLINE RESET LOG
```

Note that if the backup fails the **preserve log** option should be used when setting the databank online to leave LOGDB unaltered.

To take a **backup** of a specific databank, do the following:

- set the databank **offline** using the following command:
- take a copy of the databank file using the host file system commands
- set the databank **online** again using the following command:

```
SET DATABANK databank-name OFFLINE
```

```
SET DATABANK databank-name ONLINE RESET LOG
```

It is important to use the **reset log** option when setting the databank online again, provided the backup did not fail, so that the LOGDB records which apply to the databank you have just backed-up are correctly cleared from LOGDB (as they would have been if the **backup** had been taken using the SQL system management functions). If the backup fails, the **preserve log** option must be used.

---

**Note.** The **reset log** option removes all records written to LOGDB since the last backup. This is essential to maintain consistency between the log and the backup.

---

If two databanks depend on each other it is important that they are backed up at the same time, otherwise logical inconsistency between the backups may cause problems if LOGDB is lost. It is essential that **all** databanks that are dependent on each other are not used while the backup is being taken. It is also important that transaction consistency in the databanks be achieved before taking the backup.

The purpose of setting the database offline is to ensure that none of the databanks are used. (Additionally, stopping the multi-user system ensures the SYSDB databank is also not used.) Setting the database offline (using the SQL function SET DATABASE OFFLINE) also achieves transaction consistency because the function will not return until the database has been brought into a consistent state, prior to going offline. In particular, setting the database offline will ensure that TRANSDB is flushed and that all shadow updating has been completed.

### 6.2.2 Restoring a databank

Restoring a databank after it has been damaged or destroyed will typically involve both the host file system and SQL functions.

Any databank restore operation must start with a copy of the databank file that is not damaged or corrupt. This is generally the copy made during the last **backup**, whether taken by the host operating system or by using the SQL system management functions.

Restoring this file is done using the host file system to copy it from backup disk or other media. The file may be copied into the correct location for the databank file (as recorded in SYSDB), or may be placed in an alternative location.

If the file is placed in an alternative location, the databank must be altered to point to the new file location, using the following command:

```
ALTER DATABANK databank-name INTO 'new-file-name'
```

The next step is to restore any **incremental backup** information. For each incremental backup file, the updates recorded in it should be applied to the databank using the following command:

```
ALTER DATABANK databank-name RESTORE  
USING 'file-name'
```

Finally, the updates made since the last backup was taken, which are recorded in LOGDB, should be applied to the databank, using the following command:

```
ALTER DATABANK databank-name RESTORE  
USING LOG
```

This command also automatically sets the databank **online**, making it available for use.

## 6.3 Backup and restore of system databanks

System databanks require special procedures for backup and restore, as described below. Note that backup protection for SYSDB is particularly important for protecting the database, since SYSDB contains all information describing the database structure. If SYSDB is lost, the system must be rebuilt from scratch.

### 6.3.1 SYSDB

The system databank SYSDB cannot be protected by **incremental backups** as this is not supported. SYSDB must therefore always be backed up by taking a 'full' **backup**.

If SYSDB is **not** shadowed (see the *MIMER Shadowing Reference Manual*), the host operating system **must** be used when taking backups of it. The multi-user system must always **be stopped** before a backup copy of SYSDB is made. This ensures that no operations are performed between taking a copy of the databank and dropping the log.

SYSDB can only be restored using the backup and restore functionality in the Utilities program.

### 6.3.2 LOGDB

If a databank is lost (e.g. due to a disk crash), LOGDB can be used to restore all changes made since the last backup. LOGDB cannot be backed up using the SQL backup functions, host system backup facilities **must** be used.

Backups of LOGDB are only recommended if you are using host system backups for your whole database.

---

**CAUTION!** If the LOGDB databank is lost for any reason, no problems will be encountered immediately. All changes will have been properly recorded in the application databanks. A new, empty, LOGDB can simply replace the log that was lost. However, a **backup** of the entire database **must** be taken immediately. The new LOGDB will be empty, and therefore in a state consistent with a **backup** having just been taken and all LOGDB records cleared. If a **backup** is not taken immediately, a later attempt to restore a databank may fail because the restore operation will expect to find information in LOGDB that was lost when it was destroyed.

---

Whenever a backup operation has been performed, the contents of the log must be dropped for the particular databank being backed up. This is performed automatically if the backup operation is performed using the SQL system management functions, but must be done 'manually' if a databank file is copied by the host backup system.

### 6.3.3 TRANSDB and SQLDB

The contents of TRANSDB and SQLDB are normally transient, so that these databanks do not need backup protection.

However, backing up TRANSDB can be advisable if there are unfinished transactions (due to a system crash, etc.) and TRANSDB is in danger of being lost before the MIMER system is restarted. TRANSDB cannot be backed up with the SQL system management functions.

The following scenarios consider the possible effects of losing TRANSDB:

- If TRANSDB is lost while a transaction is being built but before it is committed, the uncommitted transaction is lost. You can simply create a new TRANSDB and start the build over again. You do not need to restore any files.
- If TRANSDB is lost while a committed transaction is in progress but before it is completed, some of the changes requested in the transaction may have been made while others are unfinished. As a result, the database as a whole is not internally consistent (but it is not physically corrupt). If you suspect this to be the case, restore the most recent backup of your databanks and use LOGDB to bring the backup copies to the state they were in before the transaction was started.
- If both TRANSDB and LOGDB are lost while a committed transaction is in progress but before it is completed, the restoration described above cannot be accomplished. In such a case, the best solution is to repair the inconsistency immediately after restarting the multi-user system using a tool such as ISQL or BSQL. This is usually possible if the user who initiated the interrupted transaction can be identified and contacted. (Many applications maintain a parallel transaction log file for tracking purposes which can be used as a basis for repair work.)

The alternative solution is to start over from the last previous backup of your databanks and reprocess all transactions since that time. This may be a very costly operation.

Keeping TRANSDB and LOGDB on separate disks under separate disk controllers will minimize the risk that both databanks are lost at the same time.

### 6.3.4 Other system databanks

Other system databanks (SYSRG, SYSQF, etc.) should be treated as normal databanks with respect to backup protection.

### 6.3.5 Re-creating TRANSDB, LOGDB and SQLDB

No MIMER modules can be run if TRANSDB, LOGDB or SQLDB is missing. In this event, an attempt to start the Backup and Restore functionality in the Utilities program (you must be SYSADM, see Section 6.4) will give you an opportunity to re-create the missing databanks with the same file names as the lost databanks, or to alter the filenames in the case where the files were moved. *A complete backup of the system should always be taken before LOGDB is re-created.* Otherwise the old backups will be inconsistent with the new, empty LOGDB.

Some data retrieval requests in MIMER/SQL may require large work areas or transaction handling areas for intermediate processing of the data (for instance, requests to sort or group large result sets will require large work tables in SQLDB). This is particularly relevant to Interactive SQL, where ad-hoc queries may be submitted with little thought for the processing requirements or performance of the query. In systems where files expand automatically, the files for TRANSDB and SQLDB can become very large as the result of one badly-planned query. If the work session is terminated normally and there are no unfinished transactions in the system, the TRANSDB and SQLDB files can be deleted and then re-created using the Backup and Restore functionality in the Utilities program.

---

**WARNING!** If a new TRANSDB is created, uncompleted transactions in the old TRANSDB are lost. Information in TRANSDB is used to complete committed transactions. This means that if a new TRANSDB is created when the transactions are not yet completed, inconsistency will occur. Therefore, make sure that the transactions are finished (the best way to achieve this is to stop the MIMER multi-user system and then run the single-user version of DBOPEN) before TRANSDB is recreated.

---

The following example shows how to re-create TRANSDB (user input is shown in bold):

```

M I M E R / U T

Username: SYSADM
Password:

MIMER/DB fatal error -16142 in function CONNECT
  Cannot open databank TRANSDB,
  file transdb7 not found

-- Redefinition of system databank --

-- Description of databank name and file --

DATABANK FILENAME
=====
TRANSDB  transdb7
Re-definition of TRANSDB <Y>?   Y
CREATE new file or ALTER filename for TRANSDB <C/A>? C
Size                                     : 1000
Databank TRANSDB redefined

```

## 6.4 The Backup/Restore functionality

The Backup/Restore functionality in the integrated Utilities program, provided in previous versions of MIMER to provide a mechanism for recovery of data in the event of a disk crash or other system failure resulting in corruption of the database, is still supported for backward compatibility. The backup and restore functionality is a menu-driven program available for both single and multi-user systems. Note that a databank cannot be accessed while backup copies are being taken using Backup/Restore functionality.

Backup and restore functions operate on one databank at a time.

The SQL system management functions documented earlier in this chapter, and in the *MIMER/SQL Reference Manual*, supersede this facility. We recommend using the SQL system management functions in preference to the old backup and restore functionality, for all backup and restore operations.

The Backup/Restore functionality described here remains available, for backward compatibility, and the underlying logic now uses the SQL system management functions.

### 6.4.1 Authorization

To use the Backup/Restore functionality to take a backup copy or incremental backup of a databank you must either have EXECUTE privilege on the system program ident MIMER\_BR or be the owner of the databank. The system administrator SYSADM holds EXECUTE privilege on MIMER\_BR with grant option, and may thus take responsibility for backup schedules for the whole system or delegate the responsibility to selected ident(s).

The main menu includes options for entering and leaving program idents, so that backup responsibilities may be assigned to a program ident which is entered from the main menu.

Note that the user with backup privilege (i.e. EXECUTE privilege on MIMER\_BR) is not necessarily permitted to read the contents of the databank. Backup procedures are independent of the normal access privileges to the contents of tables.

### 6.4.2 Starting the backup and restore functionality

The main menu offers the following options:

```
-- Backup/restore utility --

    1. Backup databank
    2. Restore databank
    3. List backups
    4. Drop backups
    5. Shadows offline,online,drop log
    6. Enter program ident
    7. Leave program ident

    0. Exit
```

### 6.4.3 Backup databank

The first option presents the following menu:

```
-- Backup databank --

    1. Create full backup and drop log
    2. Create backup databank and drop log
    3. Create incremental backup and drop log

    0. Exit
```

All choices from this menu (except Exit) prompt for the name of the databank to be handled. Backup operations must open the databank with exclusive access, meaning that no other user may gain access to the databank during the backup operation. If the databank is already in use, an open failure occurs and an error message is displayed.

#### **Create full backup and drop log**

---

**NOTE:** The term "full backup" used here, in connection with the Backup/Restore functionality, refers to a combination of both a **backup** and an **incremental backup** being taken together.

---

This option creates both an incremental backup linked to the previous backup and a new backup copy. The option is recommended as the standard one to use. The incremental backup serves to keep the backup chain intact to an older copy in case the backup copy is lost or damaged. For this reason, it may be advisable to store the backup copy and the incremental backup separately.

This example shows how to make a MIMER backup using the Backup/Restore functionality (user input is shown in bold):

```

M I M E R / U T
Username: SYSADM
Password:

-- MIMER UTILITIES --
1. Backup/restore
2. Export/import, SYSxxGEN
3. SQL statistics
4. Readlog
5. Shadowing
0. Exit

Select: 1

-- Backup/restore utility --
1. Backup databank
2. Restore databank
3. List backups
4. Drop backups
5. Shadows offline,online,drop log
6. Enter program ident
7. Leave program ident

0. Exit

Select: 1

-- Backup databank --
1. Create full backup and drop log
2. Create backup databank and drop log
3. Create incremental backup and drop log
0. Exit

Select: 1

-- Create full backup --
Databank name          : HOTELDB
Databank HOTELDB opened
Incremental backup, file name : HOTELDBUNIT1
Size                   : 100
3829 records copied from log to incremental backup HOTELUNIT
Backup databank, file name   : HOTELDBCOPY
Backup databank HOTELDBCOPY created
3829 records dropped from log for databank HOTELDB

```

### Create backup databank and drop log

This option creates a backup copy but no incremental backup. The option should only be used for the first backup copy of a databank or if incremental backups are not used.

### Create incremental backup and drop log

This option creates only a incremental backup. It is meaningful only as one of a sequence of incremental backups started by a backup copy of the databank. Note however that the functionality does **not** check that the incremental backup is part of a chain. The user must ensure that backups are consistent.

The facility prompts for the file name for the incremental backup. This name is stored in the data dictionary and is used for automatic access to the backup file in a restore operation.

For all three options, the information in LOGDB relating to the databank being backed up is dropped when the backup has been taken.

The number of records written to incremental backups and dropped from the log is displayed on the terminal.

#### 6.4.4 Restore databank

The restore operation involves restoring a databank file (from the backup copy) and execution of the operations in LOGDB, so it may take some time. The time it takes to copy a databank depends on its size, and the time it takes to restore depends on the number of operations in LOGDB (this depends on how long it has been since the last backup).

The restore operation prompts first for the name of the databank to be restored. If more than one databank is damaged, each databank must be restored as a separate operation.

You are then asked if the databank is to be restored from a MIMER copy (taken with the backup and restore functionality) or a *host system* copy (see Section 6.2.1.2).

In the case of a MIMER backup, the databank is restored automatically from the most recent backup copy, the chain of incremental backups if any, and the current contents of LOGDB. File names and the sequence of incremental backups are fetched as required from the data dictionary. If an error occurs in accessing the databank copy, you are prompted for an alternative file name. A file containing an earlier copy may be used provided there is an intact chain of incremental backups from this copy. If an error occurs in accessing a databank incremental backup, you are prompted once for an alternative file name. If the correct incremental backup cannot be accessed at this second attempt, the restoration process is aborted.

In the case of a host system backup, you are prompted for the backup file.

If the backup file has not been copied to the original databank file location, the databank is automatically altered to reference the backup file. Normally, all entries in LOGDB are used to restore the databank from the backup copy. However, the functionality checks the timestamps recorded in LOGDB against those in the backup copy to ensure that the restoration is correct. If there are timestamps in LOGDB earlier than those in the backup copy, this indicates that the log was not dropped when the copy was taken. In that case, the functionality allows the operator to select restoration from a specified date and time.

Restoration from a specified date and time must be used with some care, since it is important that the date and time specified coincide with the time that the host databank copy was made. It is recommended that the log be dropped as part of the routine of making host databank copies, rather than relying on being able to restore from the correct time.

An example of restoring a databank from a MIMER backup is shown below (user input is shown in bold):

```

-- Backup/restore utility --
  1. Backup databank
  2. Restore databank
  3. List backups
  4. Drop backups
  5. Shadows offline,online,drop log
  6. Enter program ident
  7. Leave program ident
    0. Exit

Select: 2

-- Restore databank --
Databank name      : DB1
Restoration from MIMER backup (M) or SYSTEM backup (S) or QUIT (Q)?
(M/S/Q): M
Databank DB1 created and copied from backup databank DB1COPY
Databank DB1 restored from log
2384 records restored

```

### Restoring SYSDB

If SYSDB is lost, a backup copy of SYSDB must be restored to allow MIMER to start again. No MIMER application may be used before this is done.

If SYSDB is lost or corrupted, the host system backup copy should be copied to the same file location as the original SYSDB. The contents of SYSDB may then be brought completely up to date by restoring the information from LOGDB.

Start the program and login as SYSADM. A message is displayed saying that you have an old version of SYSDB that must be restored. Answer Y to the question "Restore SYSDB" to restore the copy of SYSDB. Since SYSDB always has the LOG option, this will restore SYSDB to the state it had before it was lost.

Example, assuming a backup of SYSDB has been copied to the original location (user input is shown in bold):

```
MIMER/DB fatal error -16159 in function CONNECT
      Old version of the databank SYSDB cannot be accessed without
      restoring the databank with the backup and restore utility
      -- Restore databank --

Restore SYSDB (Y/N)? Y
Databank SYSDB restored from log
2384 records restored

-- MIMER UTILITIES --

1. Backup/restore
2. Export/import, SYSxxGEN
3. SQL statistics
4. Readlog
5. Shadowing
   0. Exit

Select: 0
```

### 6.4.5 List backups

The third option in the main menu allows backup operations on a specified databank to be listed. The data displayed is fetched from the data dictionary, so that any host system backup copies are not included.

The menu for listing backups offers the following choices:

```
-- List backups --

1. List backups of databank
2. List latest backup of databank
3. List backups of databank before date
4. List incremental backups of databank
5. List drop log of databank

0. Exit
```

Wildcard characters based on the SQL standard can be used in or for databank names when listing backups. Use the percent sign (%) by itself to designate all databanks or in combination with other characters to indicate similar names (that is, DB% means every databank name beginning with DB). To indicate a single character, use the underscore (\_) character. Thus, DB\_ specifies all databank names that consist of the letters DB followed by one (and only one) alphanumeric character or a space character.

**List backups of databank**

The date and time when the backups were taken and the files where the backups are stored are listed for all backups recorded in the data dictionary (i.e. for all backups taken with the MIMER backup and restore functionality).

**List latest backup of databank**

The date and time when the most recent backup recorded in the data dictionary was taken and the file where the backup is stored are listed (i.e. for the most recent backup taken with the MIMER backup and restore functionality).

**List backups of databank before date**

The date and time when the backups were taken and the files where the backups are stored are listed for all backups before a given date and time and recorded in the data dictionary (i.e. backups taken with the MIMER Backup/Restore functionality).

The date is given as a number with the format YYYYMMDD, and the time as a number with the format HHMMSS based on a 24-hour clock. Seconds and minutes may be omitted from the time specification. Examples of valid dates and times are

```
Date   : 19960110
Time   : 153000      - before 3:30 pm, 10th January 1996

Date   : 19960901
Time   : 00          - before September 1996.
```

**List incremental backups of databank**

The date and time when the incremental backups were taken and the files where the incremental backups are stored are listed for all incremental backups taken on the specified databank.

**List drop log of databank**

This option lists all occasions when the log of a specified databank was dropped, whether automatically during a MIMER backup operation or explicitly with the drop log function. The dates and times when the log was dropped are displayed.

A user may list backup information concerning his own databanks, regardless of who performed the backup operations. A user with EXECUTE privilege on MIMER\_BR may list information for any databank.

**6.4.6 Drop backups**

This option allows backups of a databank to be dropped from the data dictionary and thus be eliminated from the automatic sequence of restore operations. The option prompts for date and time (given in the format described for 'List backups' above), and removes all records of backups taken before the specified time.

Use this option with care. Make sure that a backup being dropped is not required as part of a sequence of incremental backups.

Note that dropping backups deletes the entries in the data dictionary but does not remove the corresponding physical backup files. Files should be deleted with the appropriate function in the host operating system.

### 6.4.7 Shadows offline, online, drop log

This option should be used, whenever a databank copy has been made with the host system backup facility, to ensure that the contents of the log are consistent with the contents of the backup copies.

If a system backup copy is made without dropping the relevant records from the log, any subsequent restoration of the databank concerned must be performed with the TIME option, from the time when the system backup copy was taken. Otherwise the restoration will attempt to repeat operations recorded in the log from the time before the backup copy was taken.

The 'Shadows offline,online,drop log' option presents four possibilities:

```
-- Shadows offline,online,drop log --  
  
1. Set shadows offline  
2. Set shadows online and drop log  
3. Drop entire log  
4. List shadowing information  
  
0. Exit
```

Options 1 and 4 only concern Shadowing and are described in the *MIMER Shadowing Reference Manual*. The second option prompts for a list of the databanks for which shadows are to be set online and records are to be dropped from the log. An example of drop log is shown in Section 6.3.1.

The third option drops all records for all databanks from LOGDB. This option may only be performed by an ident with EXECUTE privilege on MIMER\_BR. The option should be used with great care in production environments since backup protection will be incomplete unless a copy of every databank in the system has just been made. The option may however be useful in development environments where backup protection of the database contents may be less important.

Note: During operations which drop either the entire log or log records for SYSDB, LOGDB may not be accessed for any other purposes. This means that no user transaction may be logged (or executed, if databanks with LOG option are involved) while the entire log or SYSDB log records are being dropped. The restriction does not apply to dropping selected log records for databanks other than SYSDB.

### 6.4.8 Enter and leave a program ident

These options allow the backup responsibilities to be assigned to a program ident rather than a user. The program ident is granted EXECUTE privilege on MIMER\_BR by SYSADM. Users are then authorized to perform backup functions by being granted EXECUTE privilege on the program ident.



## **7 THE READLOG FUNCTIONALITY**

### **7.1 Functions**

The READLOG functionality allows the contents of LOGDB to be read, so that logged operations performed on the database since the last backup copy or incremental backup was taken may be checked. This facility may be used as an audit trail or in the event of a system failure to determine which databanks need to be restored (i.e. which databanks have been altered since the last backup). In cases where system failure destroys TRANSDB, reading the most recent contents of LOGDB can reveal whether there are any uncompleted transactions which must be corrected manually.

The READLOG functionality can only read the contents of the current LOGDB, not the contents of incremental backups.

The functionality allows information to be selected from LOGDB on the basis of time interval, ident performing the operation, and specified databanks or tables. This is particularly useful in production systems where LOGDB can contain a large number of entries.

### **7.2 Authorization**

Authorization restrictions apply to the list operations (options 6-8 in the menu).

To list the log for selected tables or tables in a databank, the user must have SELECT access on the tables in question.

To list the log for the whole database, the user must have EXECUTE privilege on the system program ident MIMER\_BR.

## 7.3 Using the READLOG functionality

The READLOG functionality is controlled from a menu from which different functions may be activated before performing the read operation. The menu is redisplayed after each choice until the listing is activated by selecting one of the options under 'List operations'.

List definitions	List restrictions	List operations	Change program id
-----	-----	-----	-----
1. Log list file	3. Time interval	6. Specified tables	9. Enter program
2. Log list width	4. Ident	7. Tables in databank	10. Leave program
	5. Databank	8. All (no data)	
0. EXIT			

### 7.3.1 List definitions (output control)

#### Log list file

Choosing this option allows the operator to specify the name of a sequential file into which the listing is to be placed. In systems where the terminal may be addressed by a logical file name, this may be given to display the listing on the terminal. If this option is not selected, a sequential file with the default name RDLOGL will be used.

```
Example - Select: 1
          Log list file: READLOG.DAT
```

#### Log list width

This option allows the width of the page to be set for the listing output. The default value is 80. The value given must lie between 72 and 132.

```
Example - Select: 2
          Log list width: 120
```

### 7.3.2 List restrictions

#### Time interval

This option allows the listing to be restricted to a given time interval, specified as a starting time and a finishing time. Times are given as a single parameter representing year, month, day, hour, minute and second in the format YYYYMMDDHHMMSS. If an incomplete time specification is given (truncated from the right), the remaining parameters are taken as low for the starting time and high for the finishing time. Thus giving 199304 as both starting and finishing time lists the log from the beginning to the end of April 1993.

Default values for starting and finishing times are the beginning and end of the log respectively. The default applies if no time interval is selected, or may be chosen for starting or finishing time by specifying a 'blank' time. If the default is chosen for both starting and finishing times, the following message is displayed.

```
'** No time restriction'
```

A selected time interval applies for all subsequent list operations in the current session until the time interval is reset. A time interval of two months has been selected in the following example.

```
Example - Select: 3
          Format  : YYYYMMDDHHMMSS
          Starttime: 199604
          Endtime  : 199605
```

### Ident

Selecting an ident restricts the listing to operations performed by that ident. Only one ident may be selected for a given listing.

The default setting lists operations performed by all idents. The default applies if no ident restriction is selected, or may be chosen by specifying a blank ident. If the default is chosen, the following message is displayed.

```
'** No ident restriction'
```

A selected ident applies for all subsequent list operations in the current session until the ident is reset.

```
Example - Select: 4
          Identname: HOTELADM
```

### Databank

Selecting a databank restricts the listing to operations performed on that databank. This option must be specified if the list operation 7 (Tables in databank) is to be used. Only one databank may be selected for a given listing.

The default condition lists operations performed on all databanks. The default applies if no databank restriction is selected, or may be chosen by specifying a blank databank. If the default is chosen, the following message is displayed.

```
'** No databank restriction'
```

A selected databank applies for all subsequent list operations in the current session until the databank is reset.

```
Example - Select: 5
          Databank: HOTELDB
```

### 7.3.3 List operations

#### Specified tables

This option activates listing of the log for selected tables in the database. The operator may specify as many tables as required, with the table name qualified if necessary by the creator of the table. If no creator is given, the current ident is assumed.

Databank restrictions selected with option 5 are ignored if specified tables are selected. Ident and time restrictions selected with options 3 and 4 are however applied.

The ident running the READLOG functionality must have SELECT access on the requested tables, otherwise the message '\*\* No select access on table' is displayed for the table in question. If a non-existent table is requested, the message '\*\* No such table' is displayed. Errors of this type do not abort the listing if valid and invalid requests are mixed in the same operation.

The list operation is activated by giving a blank response to the prompt for a table name when all the required tables have been specified.

```
Example - Select: 6
          Table: HOTELADM.EMPLOYEE
          Table: HOTELADM.STAFF
          Table: HOTELADM.SALARY
          Table:
```

#### Tables in databank

Operations on all tables in the databank specified under option 5 are listed. If no databank has been selected, the following message is displayed and the user must select a new option.

```
'** Databank not entered'
```

Time or ident restrictions selected with options 3 or 4 are applied.

Data is listed only for those tables to which the ident running the READLOG functionality has SELECT access. Tables to which access is denied are indicated by the following message in the log list file:

```
'Table <creator.table-name> - No select access'
```

#### All (no data)

This option lists logged operations without details of data records (see below). The ident running the READLOG functionality must have EXECUTE privilege on the system program ident MIMER\_BR. If the privilege is not held by the current ident the error message: '\*\* AUTHORIZATION FAILURE' is given.

### 7.3.4 Change program id

#### Enter, leave program ident

If a program ident is the creator of a databank, that ident may be entered to read the log for that databank.

## 7.4 Output format

The output from the READLOG functionality is divided into transactions, showing the date and time, the ident performing the transaction (with entered program idents where appropriate) and the number of database records read during the transaction. Note that the output does not contain statements for reconstructing the logged operations - it is simply a documentary record of the transactions performed on the database.

If list operations 6 or 7 (select by table or databank) are selected, the contents of the affected rows in the table are displayed. Insert and delete operations are listed as a single row. Update operations are recorded as the state of the row before and after the update.

If the list operation 8 is selected (no data), the operations are listed without the data records.

The following example uses the READLOG functionality to read all the transactions that have been committed after 1995-05-23 10:55 and before 1995-05-23 12:00 and shows extracts from the output for All (no data) and for table BOOK\_GUEST (user input is shown in bold):

```

-- Read log --
List definitions  List restrictions  List operations  Change program id
-----
1. Log list file  3. Time interval  6. Specified tables  9. Enter program
2. Log list width 4. Ident          7. Tables in databank 10. Leave program
0. EXIT          5. Databank      8. All (no data)
Select: 1
Log list file: LOGFILE

-- Read log --
List definitions  List restrictions  List operations  Change program id
-----
...

Select: 3
Format   : YYYYMMDDHHMMSS
Starttime: 199605231055
Endtime  : 199605231200

-- Read log --
List definitions  List restrictions  List operations  Change program id
-----
...

Select: 8

```

## Extract from output All (no data):

```

Transno Function
-----
106 Update before in HOTELADM.FREEROOMS
106 Update after in HOTELADM.FREEROOMS
106 Delete from HOTELADM.MAINTENANCE
106 Delete from foreign key
106 Commit 1996-06-23 11:06:59:74 by HOTELADM/CHARLIE Read 6
-----
107 Insert into HOTELADM.CHARGES
107 Update before in HOTELADM.BOOK_GUEST
107 Update after in HOTELADM.BOOK_GUEST
107 Commit 1996-06-23 11:08:00:45 by HOTELADM/GEORGE Read 5
-----

-- Read log --
List definitions List restrictions List operations Change program id
-----
1. Log list file 3. Time interval 6. Specified tables 9. Enter program
2. Log list width 4. Ident 7. Tables in databank 10. Leave program
5. Databank 8. All (no data)

0. EXIT
Select: 6
Table: BOOK_GUEST
Table:

```

## Extract from output for table BOOK\_GUEST:

```

Table HOTELADM.BOOK_GUEST
Transno Function RESERVATION, BOOKING_DATE, HOTELCODE, ROOMTYPE,
RESERVED_BY, TELEPHONE, RESERVED_FOR, ARRIVE, LEAVE,
GUEST, ADDRESS, CHECKIN, CHECKOUT, ROOMNO, PAYMENT
-----
92 Insert 1382, 19960523, 'WINS', 'SGLB', 'JULIO PEREZ',
'6-6549868', 'JULIO SANCHEZ', 19960528, 19960531,
'JULIO SANCHEZ', 'CARLOTA 7, MADRID, SPAIN', <NULL>,
<NULL>, 'WINS301', <NULL>
92 Commit 1996-05-23 10:57:56:49 by HOTELMANAGER Read 3
-----
99 Delete 1353, 19960516, 'SKY', 'DBLB', 'HOT SERVICE LIMITED',
'08-135709', 'BASIL FAWCETT', 19960602, 19960612,
'ALFRED SMITH', '23 BACK NELLY VIEW, ACKWORTH',
<NULL>, <NULL>, 'SKY124', 'CASH'
99 Commit 1996-05-23 10:58:59:74 by HOTELADM/RICHARD Read 1
-----
107 Update before 1348, 19960505, 'SKY', 'SGLS', 'SYSDECO MIMER AB',
'018-185000', 'BENGT PETTERSON', 19960520, 19960523,
'BENGT PETTERSSON', 'MIMERGATAN 64, UPPSALA',
19960520, <NULL>, 'SKY101', 'EUROCARD'
107 Update after 1348, 19960505, 'SKY', 'SGLS', 'SYSDECO MIMER AB',
'018-185000', 'BENGT PETTERSON', 19960520, 19960523,
'BENGT PETTERSSON', 'MIMERGATAN 64, UPPSALA',
19960520, 19960523, 'SKY101', 'EUROCARD'
107 Commit 1996-05-23 11:08:00:45 by HOTELADM/GEORGE Read 5
-----

```

## 8 DATABASE STATISTICS

### 8.1 Statistical information

The SQL statistics functions collect statistical information about table and index data in the database and store this information in the data dictionary. The information is used by the SQL compiler in optimizing access paths for SQL data manipulation statements.

The statistical information includes:

- the total number of rows in each base table, stored in the column `CARD` in the table's row in `MIMER.TABLEX`
- the number of distinct values in each column of a table, stored in the column `COLCARD` in the column's row in `MIMER.COLUMN`
- the number of non-NULL values in each column of a table, stored in the column `COLCNOTN` in the column's row in `MIMER.COLUMN`
- the lowest and highest values in each column of a table, stored in the columns `LOW` and `HIGH` in the column's row in `MIMER.COLUMN`
- the date and time for the last time statistics were gathered for a table is stored in `MIMER.OBJECT` with the type 'TABSTAT'. This can be read by the view `MIMER.OBJECTS`.

Statistics may be collected for the entire database (i.e. all tables in all databanks recorded in the same `SYSDB`), for tables owned by specified ident, or for specific tables.

### 8.2 Authorization

The user running the SQL statistics functions must either have `EXECUTE` privilege on the system program ident `MIMER_SC` or be the owner of the table(s) or ident(s) for which statistics are being collected. The system administrator `SYSADM` holds `EXECUTE` privilege on `MIMER_SC` with grant option, and may thus take responsibility for maintaining statistics for the whole system or delegate the responsibility to selected ident.

Note that a user with statistics privilege (i.e. EXECUTE privilege on MIMER\_SC) is not necessarily permitted to read the contents of the databank using data manipulation statements, this privilege only permits access for the collection of statistics.

## 8.3 The SQL statistics functions

The SQL statistics functions may be used to collect statistical information in the areas described below. Also refer to the *MIMER/SQL Reference Manual* for details on the SQL statistics functions.

### 8.3.1 Statistics for the entire database

To collect statistical data for all tables in the database, use the following function:

```
UPDATE STATISTICS
```

The user must have EXECUTE privilege on MIMER\_SC.

Even in a database of only moderate size, collecting statistical data for all tables is time-consuming. It is recommended that this option in particular is run at off-peak working times, and preferably in batch mode.

### 8.3.2 Statistics for specified idents

To collect statistics for all base tables owned by a list of specified idents, use the following function:

```
UPDATE STATISTICS FOR IDENT list-of-idents
```

A user requesting statistics for tables owned by an ident other than himself must have EXECUTE privilege on MIMER\_SC.

To collect statistics for SYSDB, the pseudo-ident MIMER may be specified.

### 8.3.3 Statistics for specified tables

To collect statistics for a list of specified tables, use the following function:

```
UPDATE STATISTICS FOR TABLE list-of-tables
```

The user requesting statistics for the tables specified in the list must either be the owner of them or have EXECUTE privilege on MIMER\_SC.

## 8.4 When to use the SQL statistics functions

The SQL statistics functions should be used whenever truly **optimal performance** is an issue of significant importance.

MIMER collects approximate statistics for each table whenever the table is opened. These statistics may suffice for maintaining high performance in many situations. If optimal performance is required for an application, the SQL statistics functions should be used to supplement and refine the automatically collected information.

When this is the case, statistics should be typically updated in the following situations:

- when the size of a table has altered
- when the maximum/minimum limits on values in a table have altered significantly.

The statistics information in the data dictionary is used only by the MIMER/SQL compiler.

Note that only the performance, not the result, of an SQL statement is affected by the statistical information.

The statistics functionality in the integrated Utilities program, provided in previous versions of MIMER to provide a mechanism for statistics collection, is still supported for backward compatibility.



## 9 SYSTEM TUNING

In a MIMER installation there are a number of system-wide parameters that have a major impact on the overall system performance. The most important parameters are the bufferpool size, the number of threads ('Level 1' tasks) in multi-threaded systems and the number of background servers. The MIMSERV program provides facilities for monitoring performance parameters in an active MIMER system.

### 9.1 Bufferpool size

#### 9.1.1 General considerations

The bufferpool is the main primary memory cache used by the basic data access routines in MIMER database management. The storage unit in the bufferpool is the MIMER page, the size of which is installation-dependent. It is generally possible to allocate 2048 byte, 16384 byte and 65536 byte pages. Whenever the bufferpool is full and access to a new page is required, space is released in the bufferpool by deleting the least-recently-used resident page.

Frequent page replacement operations detract from the overall system performance since access to disk is relatively slow. The best MIMER performance is thus obtained by having as large a bufferpool as possible. In practice, it is always necessary to find a suitable compromise between allocation of memory to the MIMER bufferpool and keeping memory available for user applications and operating system tasks. The compromise will be influenced by a wide range of factors specific to each system (and often varying with time in any one system), such as a dominant type of MIMER or other applications, number of simultaneous users, operating system demands, etc.

Some general guidelines for bufferpool tuning are:

- Initial choice of bufferpool size is made from a rough knowledge of the available memory space and the requirements for MIMER applications in a new installation. Fine tuning of the bufferpool is performed when the MIMER system is fully installed and functional, and should be repeated whenever there is a significant change in the computer workload distribution. Since the MIMER bufferpool size affects the performance of both MIMER and other applications (by reserving memory to MIMER functions), it is advisable to perform regular routine checks on the bufferpool statistics in an established system using MIMSERV.

- Whenever main memory is available, it should be allocated, if possible, to the bufferpool.
- Ensure that the bufferpool is not subject to system paging or swapping, since the paging algorithms used by MIMER and the operating system usually differ, and forced cooperation between the two will often detract considerably from MIMER's performance.
- If pages in the bufferpool are never used for storing data, the pool is too large. This can be detected by the MIMSERV functionality (see below). In this situation, MIMER database management performance is never limited by bufferpool paging, but the allocation of unnecessarily large amounts of memory to the bufferpool may detract from the performance of other aspects of application programs (for example operating system I/O).
- If more than about 5% of all MIMER page requests result in a page fault, the bufferpool is too small. Statistics for page requests and faults are provided by the MIMSERV functionality (see below).
- It is important to take note of the page fault statistics **for each region** in the bufferpool to ensure that the most appropriate allocation has been made in each. The MIMER system decides which page size is most appropriate for each task to be performed. For example, 16K pages are currently used for write sets (this may change in the future) and therefore allocating too few 16K pages may currently adversely affect performance even though generous allocations have been made in the other bufferpool regions.

### 9.1.2 Changing the bufferpool size

The routines for setting the bufferpool size in multi- and single-user systems depend on the hardware platform, and are described in the machine-specific Installation Guide.

## 9.2 Number of threads

In most multi-user implementations of MIMER, there are a number of separate MIMER system processes or threads (so-called 'Level 1' tasks) running simultaneously under the operating system. Throughput in multi-user systems is dependent on the number of available threads. Increasing the number of these threads can improve performance.

The number of threads does not usually correspond to the number of users in the system. The ratio between active MIMER users and Level 1 tasks is platform-dependent. However, on some machines, the MIMER Level 1 process may be included in the user task. Refer to the installation-dependent information for further details.

The number of threads in a multi-user system is defined at system start-up. A change to the number of threads on some machines requires that the system be stopped and re-started.

The maximum number of concurrent MIMER Level 1 processes is limited by the size of the bufferpool.

## 9.3 Number of shadow servers

Note: The term “shadow servers” refers to background processes responsible for performing certain tasks in multi-user MIMER systems.

The shadow servers in the MIMER multi-user system perform tasks such as:

- Writing transaction changes to disk.
- Recording transactions in LOGDB.
- Updating shadow databanks.
- Freeing write-sets for use by other transactions.

There should be enough shadow servers to perform these tasks as quickly as possible. For most installations, two shadow servers (the minimum number) are sufficient.

If an installation relies extensively on database shadowing, the number of shadow servers may have to be increased. Start the system with more servers (e.g. four), then check with MIMSERV to see how much each server is being used. Alternatively check the CPU usage for each server process in the operating system. If some shadow servers are not used, reduce the number of servers and check with MIMSERV again.

## 9.4 The MIMSERV functionality

### 9.4.1 Functions

The MIMSERV functionality can be used by the system administrator to monitor performance parameters during MIMER use. A similar facility, called MIMDUMP, is used by MIMER agents for trouble-shooting when multi-user system problems are reported by customers. When assistance from the MIMER agent is needed, it is important that a MIMDUMP output is generated and included with the error report.

Execution of the MIMSERV program requires special operating system privileges in some systems (see machine-dependent information). No log-on procedure is required by MIMSERV itself: all security restrictions are at the operating system level.

The MIMSERV functionality provides five kinds of statistical information which may be useful for system tuning (statistics for page management, transactions, shadow servers, databank and table usage). The information is updated continuously, so that values change as system usage fluctuates.

Note: When MIMSERV is used as an aid to system tuning, it is important that the program is run when the system is in full use. The output from several executions over a period of a few hours or days can provide valuable indications of fluctuations in system usage.

Output from the MIMSERV functionality is displayed on the terminal. Hard copy of the output is obtained by re-directing terminal output to a file or printer.

The output contains the following information:

## **General statistics**

### **Current date and time**

### **Current hardware and operating system**

### **Current MIMER/DB version**

### **Name of database**

### **Starting date and time**

### **System status**

If system status is in an error state MIMDUMP should be run and the output saved. The multi-user system can then be restarted. The multi-user log file should be inspected to find the cause of the failure.

### **Error count**

Number of errors that have been written to the multi-user system log file (the log file is described in the MIMER machine-dependent Installation Guide). This value should normally be zero.

### **No. of MIMER kernel processes**

Number of multi-user system threads (see Section 9.2).

### **No. of shadow server processes**

Number of shadow servers started (see Section 9.3).

### **No. of I/O servers**

Number of I/O threads. Under VMS the single I/O process is multi-threaded.

### **No. of users active**

Number of users logged in to the multi-user system.

### **No. of denied users**

Number of login attempts that have been denied because the maximum number of concurrent users are already logged in.

### **Max number of users**

Maximum number of users that can log in to this system. This number is specified when the multi-user system is started. The maximum number of users for all multi-user systems started on the machine may not exceed the number given in the MRS license.

## **Page Management statistics**

### **No. of page buffers per sorter**

Total number of MIMER pages that a sorting thread may utilize.

### **No. of remaining sorters**

The initial value specifies the number of concurrent sort/merge steps that are allowed.

### **No. of pages written to disk**

An indication of the frequency of disk update operations.

**No. of file extend operations**

The total number of times databank files have been dynamically extended since the latest startup. The value should preferably be as low as possible for performance reasons. It is possible to check databank size usage with the DESCRIBE command in ISQL or BSQL. A databank can be extended by using the commands ALTER DATABANK ADD... or ALTER SHADOW ADD....

**Buffer size 2k (16k, 64k)**

The bufferpool is divided into a region with 2k buffers, one region with 16k buffers, and one region with 64k buffers. The following information is given for each region:

**No. of page buffers**

This is the number of page buffers allocated to this bufferpool region.

**No. of page partitions**

The bufferpool is divided into separate partitions. Each partition can be accessed concurrently by the MIMER kernel threads. In tightly coupled multi-processor systems it is desirable, for performance reasons, to have at least as many partitions as there are CPUs. The number of partitions may be increased by increasing the bufferpool size.

**No. of page requests**

Total number of access operations to pages in the bufferpool since latest system start-up.

**No. of page faults**

Total number of page access requests that resulted in disk read operations. If this value is more than 5-10% of the total number of page requests, performance may be improved significantly by increasing the bufferpool size.

**No. of pages swapped out**

Total number of pages which were written to disk when they were swapped out of the bufferpool.

**Transaction management statistics****No. of transaction commits**

Total number of successful read/write transaction commits since latest system start-up.

**No. of read commits**

Total number of successful read-only transactions since latest system start-up.

**No. of transaction checks**

A high proportion of transaction checks in relation to the total number of transactions may indicate ill-designed application programs, with long transactions that are more likely to give rise to transaction conflicts.

**No. of transaction aborts**

Total number of transactions aborted by the optimistic concurrency protocol since latest system start-up. User requested transaction aborts are not counted.

**No. of pending restarts**

This is an indication of how much information is stored in TRANSDB. Number of restarts is counted for each databank used in a transaction. This figure grows larger when shadows are set offline.

If all databanks have been accessed and there are no offline shadows there should not be any pending restarts.

**Shadow servers****SWA**

Shadow server identifier.

**State**

State of the shadow server. If the shadow server is currently working with a transaction, 'active' is displayed. If the shadow server is not doing anything, 'inactive' is displayed. 'I/O processing' means the shadow server is flushing one or more transactions to disk and 'unused' means that no shadow process is using that slot.

**Trans-no**

The number of the transaction currently being processed.

**Transaction-count**

The number of transactions processed by the shadow server. This gives an indication of how many shadow servers are needed in the system. A background server which is never used indicates that the number of shadows servers started can be reduced.

**Pending shadow server requests**

This indicates how many transactions have not yet been processed by the shadow servers. If there are too few shadow servers this value will grow.

**Application waiting for trans-no**

For certain operations (set databank offline, for example) the application has to wait for the shadow servers to complete their operations. If there are too few shadow servers it may take some time before this operation is complete. By comparing this trans-no with the trans-no being handled by the shadow servers it is possible to see how many transactions are left before the operation is completed. (Note that the following example does not show a figure for this because the application is not waiting for transactions to complete.)

## Databank statistics

### Name

The name of the databank or shadow.

### DBANKID, SEQNO

Databank identification. These two values correspond to the columns with the same names in the data dictionary table MIMER.DBFILE.

### Type

The databank option NULL, TRANS, or LOG. See Chapter 6 of the MIMER/SQL Interactive Users Manual for a description of these options.

The SQLDB databank has the type WORK, and shadows have the type SHADOW.

### Users

Current user count. This is the number of applications currently using the databank, not the number of different MIMER users.

### Access

Access mode by which the databank was opened. The possible values are: 'Read', 'Write', 'Shared' and 'Exclusive'. If the databank is not open 'None' is displayed. If a shadow is offline, it is also indicated here.

### No. of databanks currently open

A count of both databanks and shadows opened in the system.

### Max number of databanks open concurrently

This number is implementation-dependent.

### Databank verification count

Databank verification is automatically performed when a databank is re-opened without having been correctly closed. Each time this happens a log entry is written to the multi-user log file. When a databank is verified the databank verification count is incremented. The count is cleared when the system is started, and a databank is only verified once per session.

### Databank verification is only done on index pages...

### Databank verification is performed on all pages...

These information messages indicate the installation-dependent databank verification mode.

## Table statistics

### No. of tables currently open

This shows the number of tables open in both master and shadow databanks. Also included are the read and write sets used by each user.

### Max no. of tables open concurrently

This number is implementation-dependent.

### 9.4.2 MIMSERV output example

The following example output is from a MIMER system which is not being limited by the bufferpool size (page faults, in 2K region, only about 1% of page requests). The number of transaction checks is low, indicating either that concurrent user update requests are infrequent or that transaction handling in application programs is well-designed.

The 16K region has been used very little, and the 64K region has not been used at all. It is possible to reduce the sizes of these regions. However, this will reduce the number of Level 1 processes that can be started, so it may be appropriate to keep the current sizes of these regions.

The error count has a value of 4 which indicates that 4 errors have been written to the multi-user log file.

The databank SYMDB has an error status. The multi-user log file (described in the MIMER machine-dependent Administration Guide) will contain further information. The explanation of the error code can be found in the dictionary table MIMER.MESSAGE.

The shadow CASE5\_S1 has been set offline as indicated at the end of the row describing the shadow.

The system is running with 3 shadow servers. The third server has a low transaction count and might be unnecessary.

```
*** MIMER/DB runtime statistics 1996-09-23 16:07:58 ***
```

```
Hardware, operating system ALPHA/VMS
Current MIMER/DB version is 7.3.1
Database name: DB731
System started at 1996-09-17 16:35:27
System status is up
```

```
Error count                :           4
No. of kernel servers      :           2
No. of shadow servers      :           3
No. of I/O servers         :           1
No. of users active        :           5
No. of users denied access :           0
Max number of users        :          10
```

```
Page management statistics
=====
No. of page buffers per sorter :          11
No. of remaining sorters      :           2
No. of pages written to disk  :          1774
No. of file extend operations :           0
Synchronization collisions, MDR:          12
```

```
Buffer size 2K
-----
No. of page buffers          :          512
No. of page partitions       :           3
No. of page requests         :          40859
No. of page faults           :           432      1 %
No. of pages swapped out     :           0
```

```

Buffer size 16K
-----
No. of page buffers           :           42
No. of page partitions       :             1
No. of page requests         :          44817
No. of page faults           :             7      0 %
No. of pages swapped out     :             0

Buffer size 64K
-----
No. of page buffers           :           42
No. of page partitions       :             1
No. of page requests         :             0
No. of page faults           :             0
No. of pages swapped out     :             0

Transaction management statistics
=====
No. of transaction commits    :           1336
No. of read commits          :             0
No. of transaction checks    :           1436
No. of transaction aborts    :            103
No. of pending restarts     :             18

Shadow servers
=====
      SWA State           Trans-no Trans-count
175 Inactive              0          11985
189 Inactive              0           1382
203 Inactive              0             12

Pending shadow server requests :             0

Databank statistics
=====
Name                DBANKID SEQNO Type      Users Access
TESTDBM1            310      0 TRANS        5 Shared
TESTDBM2            311      0 TRANS        5 Shared
SYSDB                1       0 LOG         6 Shared
TRANSDB              2       0 TRANS        5 Shared
LOGDB                3       0 LOG         5 Shared
SQLDB                4       0 WORK        5 Shared
CASE5                279     0 LOG         0 Shared
CASE5_S1            279     1 SHADOW       1 Shared  Offline
SYMDB                285     0 TRANS        1 Shared

Databank error status:      -16142

No. of databanks currently open      :           6
Max number of databanks open concurrently :          100
Databank verification count          :             0
Databank verification is performed on all pages

Table statistics
=====
No. of tables currently open          :           36
Max number of tables open concurrently :          4000

```



## 10 TROUBLESHOOTING

### 10.1 Runtime malfunctions

MIMER modules have, in general, comprehensive module-specific error handling facilities, which are documented in the respective reference manuals.

It should be noted that DBC and DBOPEN return an error status to the operating system when an error is encountered. This may be useful when running them from scripts or in batch mode.

Some multi-user errors are recorded in a MIMER log file (see the machine-specific Administration Guide for more details). In addition, some operating systems maintain a log file of all errors and other system events. This file, or files, should be investigated in the event of unexpected system interrupts. Contact your MIMER representative if the errors appear to reflect a problem with MIMER.

### 10.2 The DBC functionality

#### 10.2.1 General description

The DBC functionality allows investigation of databank files to ensure that the physical structure is not damaged. The functionality may also be used to examine the organization of databank files and display statistical information on file usage.

Note: The functionality checks only the **physical** condition of the databank. Informational errors such as data inconsistency, invalid data format, and data outside the validation limits of domains are not detected by DBC.

It may be advisable to run DBC routinely immediately before each backup operation, to make sure that the backup copy will be error-free. If errors are detected, the databank can be restored from the previous backup copy before the backup operation is performed. This practice is particularly recommended for databanks which are not used intensively in applications, where the chance of detecting errors during normal use is relatively low.

Note that the only satisfactory way to reconstruct a databank damaged by physical storage errors is to perform a databank restore operation from a back-up copy (except when MIMER Shadowing is used). **Direct editing of the databank file must never be attempted.**

DBC produces detailed diagnostic information when databank errors are detected. The diagnostic output should always be included with any information you send to MIMER support regarding databank errors.

### 10.2.2 Authorization

The DBC program operates directly against the databank file, with no reference to the MIMER database manager. The program may not be run on a file which is currently held open by another user (an error message is displayed in such a case). The system administrator should arrange for exclusive access to the databank file during DBC operations.

### 10.2.3 User communication

The program prompts for the name of the databank file and a name for the result file, which is a sequential file created by the DBC functionality and used for output. If an error occurs when opening the databank file (e.g. file not found or file locked by another user), or while creating the result file, an appropriate error message is displayed.

If no errors are detected in the databank file, the message 'No errors found' is displayed. The result file then contains statistics describing the physical databank organization. Otherwise, error descriptions (see below) are written to the result file, and the message 'Errors logged in result file' is shown. The result file should be examined to investigate the nature of the errors.

The message 'Too many bitmap pages', displayed during the DBC operation, does not reflect an error in the databank file. Instead, it indicates that a complete bitmap check has not been performed, due to a restriction imposed on the program during installation at your site. Contact your MIMER agent if this situation arises.

### 10.2.4 Result file contents

The result file contains the databank file name and the time when the DBC operation was performed. This is followed by header information read from the databank, such as the version number under which the databank was created. If this is not the same as the current version there will also be a "converted to" message. Also shown are the backup timestamp, the structure level, the system identification number of the databank, the number of blocks for the bitmap pages (starting at 0) and for the root pages (starting at 1), and the total number of allocated and used MIMER pages. If the databank file was created as a shadow copy the sequence number is also given. If the databank was not properly closed when MIMER was stopped, there may be an additional message saying "No bitmap checking against databank!".

An identification record is given for each table in the databank. The following information is shown:

<b>Tabid</b>	The system identification number of the table. This is the number used to identify the table in the data dictionary. The table name is not stored directly in the databank file.
<b>Startp</b>	The block number of the start page for the highest index level. If the number of levels is 1, this is the only data block. If the start page is 0, the table is empty.
<b>Levels</b>	The number of levels in the table storage structure.
<b>Keylen</b>	The length of the primary key in bytes.
<b>Reclen</b>	The record length (row length) of the table.
<b>Type of table</b>	'Base table', 'Base table (VL)', 'Secondary index table' or 'Secondary index table (VL)'. (VL) indicates that data pages have variable-length records (i.e. are compressed)
<b>Status of table</b>	'Resident' or 'Marked for delete'.
<b>Index page blocksize</b>	Size of the index page in bytes.
<b>Data page blocksize</b>	Size of the data page in bytes.
<b>Number of index pages</b>	Indicates how many index pages were checked.
<b>Number of data pages</b>	Indicates how many data pages were checked.
<b>Required/Allocated datapages</b>	Gives an approximate measure of the space used in the datapages. This is expressed as a percentage, which may be >100% for data pages having variable-length (i.e. compressed) records because the required number of pages is calculated based on uncompressed record sizes.
<b>Reached no of records</b>	If no errors are reported, the number of records reached is equal to the total number of records (rows) stored for the table.

For LOGDB, TRANSDB and SQLDB the identification record contains the following information:

<b>Tabid</b>	Ordered identification.
<b>Startpage</b>	First page in the sequential file.
<b>Endpage</b>	Last page in the sequential file.
<b>Type of table</b>	'Sequential table' or 'Sequential table (VL)'. (VL) indicates that pages have variable-length records (i.e. are compressed).
<b>Status of table</b>	'Resident' or 'Marked for delete'.
<b>Page blocksize</b>	Size of the page in bytes.
<b>No. of pages read</b>	Number of pages checked.
<b>No. of records read</b>	Number of records checked.

Any errors detected in the databank file are written to the result file directly after the identification record for the table affected. Tables with no error indications can be read by MIMER, and copied to sequential files using the UNLOAD function in Interactive SQL. In this way information from intact tables in a damaged databank file can be saved, providing partial security in case the backup routines for the databank are inadequate.

### 10.2.5 Example of result file

The following example illustrates the result of running DBC on the SYSHELP databank file. No errors were detected.

```

MIMER Databank Check Utility
  Version 7.3.1

Databank file:  SYSHELP

Time: 1996-09-23 16:29:09

Version created:      731  Backup timestamp:   1
Structure level:     6   System ident.:      269

Bitmap pages:        0

Root  pages:         1

No. of pages allocated:      160           No. of pages used:           115

Tabid:      270  Startp:          3  Levels:  2  Keylen:   4  Reclen:  25
Base table, Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         1             Number of data pages:    4
Required/Allocated datapages: 100%          Reached no of records:  255

Tabid:      271  Startp:          5  Levels:  2  Keylen:   8  Reclen:  12
Base table, Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         1             Number of data pages:    6
Required/Allocated datapages:  83%          Reached no of records:  700

Tabid:      272  Startp:          15  Levels:  2  Keylen:   8  Reclen:  89
Base table (VL), Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         1             Number of data pages:   92
Required/Allocated datapages: 183%          Reached no of records: 3707

Tabid:      273  Startp:         108  Levels:  2  Keylen:   8  Reclen:  12
Base table, Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         1             Number of data pages:    2
Required/Allocated datapages: 100%          Reached no of records:  222

Tabid:      274  Startp:           4  Levels:  1  Keylen:   4  Reclen:   8
Base table, Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         0             Number of data pages:    1
Required/Allocated datapages: 100%          Reached no of records:  166

Tabid:      275  Startp:         112  Levels:  2  Keylen:   4  Reclen:  89
Base table (VL), Resident
Index page blocksize          2048           Data page blocksize      2048
Number of index pages:         1             Number of data pages:    3
Required/Allocated datapages: 266%          Reached no of records:  166
    
```

### 10.2.6 Error messages

The following error situations are described by explicit messages in the result file:

#### Bitmap errors

Illegal number of free bits in bitmap

The number of free bits in the bitmap is marked as a negative number or as a number greater than the permissible value.

Illegal pointer to first word with free bit in bitmap

The pointer to the first word is either negative or greater than the number of words per page, or points outside the number of allocated blocks.

#### Root page errors

Illegal record length in root page

The record length is not valid for the current version or for the site.

Blockno. outside databank: x

Blockno. not marked as used: x

The reference to the next block number (x) is invalid. (Applies only where there is more than one root page.)

#### Sequential structure errors

Pointer to previous page invalid

Error in block linkage.

Invalid record length

The record length is either not the same as in the previous block or outside the block limits.

Record crosses page boundary

A record stretches over the block limits.

#### Table structure errors

Error in root record

The value for start page, levels, key length or record length is outside the legal values for the site.

Block has illegal record length: x

The record length (x) given in the block is not the same as that given for the table.

Block has illegal last-record pointer: x

The pointer (x) to data within a block is outside the page limits. The limits are the values of bytes/header and bytes/page.

Block has records in wrong order. Pos: x

The records in the block are not correctly sorted. The value of x is the byte position within the block for the start of the wrong record.

Block has last-record key > index key. Pos: x

The records within a block include key values greater than those in the index level above. The value of x is the byte position within the block for the start of the last record.

Blockno. outside databank: x

Reference is made to a block number (x) higher than the highest allocated block.

Blockno. referenced twice

There are at least two references to the same block number (x). One of these references should also give another error, or an error should have been notified earlier in the file.

Blockno. not marked as used: x

Bitmap blockno:      Word:      Bit:

A block that is used is illegally marked as free. Continued insertion of data in the databank will result in a double referenced block.

All table structure errors are followed by a listing of the block numbers passed in the B\*-tree structure on the way to the error:

B\*-tree

Block no: x1 x2 x3 .... xn

Byte pos: y1 y2 y3 .... yn

where yn indicates the byte position in the block xn where the record holding the reference to the next level starts.

If the error occurs at an index level, the additional message 'Branch is interrupted' is given, and no checks are made at lower levels.



## 11 THE DBOPEN FUNCTIONALITY

The DBOPEN functionality is used to make sure that the users can open all databanks quickly. It can take a long time to open a databank if it is a large databank that was closed abnormally (for example if the machine crashed).

The DBOPEN functionality should normally be run as soon as the multi-user system has been started.

### 11.1 Functions

The DBOPEN functionality opens all available databanks in a system. Each opened databank is closed before the next one is opened. As each databank is opened, the integrity is checked and any transactions that were interrupted by the abnormal close are completed. The integrity check is analogous to running DBC and it is automatically performed when opening a databank that has not been closed normally.

The checks performed when an abnormally closed databank is opened may take some time, particularly if the databank is large. Running DBOPEN means that the checks are performed at a time determined by the system administrator, rather than a time determined by application programs. As a result, users will always have fast access to all databanks.

The databanks are opened in a randomly determined order. Running several DBOPEN sessions in parallel may speed up the checking process for the database as a whole.

### 11.2 Authorization

Any user who has access to the data dictionary table MIMER.DATABANK can run DBOPEN.

### 11.3 DBOPEN output example

The following example output is from a MIMER system where two databanks were deleted before the multi-user system was started.

#### DBOPEN

```
MIMER Databank Open Utility

Username: SYSADM
Password:

Opening databank SYSMMSG
Opening databank SYSQSQL
Opening databank SYSRSG
Opening databank SYSRSGOUT
Opening databank SYSQF
Opening databank SYSPG
Opening databank CMD
Opening databank SYSSH
Opening databank SYSFM
Opening databank SYSHELP
Opening databank WORKDB
Opening databank HOTELDB
Opening databank BOOKDB
Opening databank ROOMSDB
Opening databank WORKDB
Opening databank L2DB3

MIMER/DB fatal error -16142
      Cannot open databank L2DB3,
      file L2DB3 not found

Opening databank L2DB1

MIMER/DB fatal error -16142
      Cannot open databank L2DB1,
      file L2DB1 not found
Opening databank RECORDS

      2 databanks were not possible to open
```

## A DATA DICTIONARY TABLES

This appendix documents the organization of the data dictionary tables, which are stored in the databank SYSDB.

The tables are created when the system is first installed, and are owned by a system ident called MIMER. The system administrator SYSADM is granted SELECT access on the dictionary tables with grant option. No other user may read the data dictionary base tables unless authorized to do so by SYSADM.

A set of system views is defined on the data dictionary tables when the system is installed (see MIMER/SQL Reference Manual, Appendix C, for documentation of these views). The global group ident PUBLIC is granted SELECT access on these views, so that any user may read the (in many cases restricted) dictionary information presented in the views. Many of the view definitions restrict the information presented to descriptions of objects accessible to the current user. Note that if SYSADM reads the contents of such a view, the result shows only the objects to which SYSADM has access. In order to gain information on inaccessible objects, SYSADM must read the contents of the dictionary base tables directly.

The system administrator may define installation-specific views on the data dictionary tables to supplement the system-defined views. Such views may be tailor-made for the installation or system in use, and SELECT access on the views may be granted to limited user groups if desired.

All maintenance of the data dictionary is performed by internal routines and is invisible to the user. No user, including SYSADM, may alter the contents of the data dictionary directly.

Note: MIMER reserves the right to change the internal organization of the data dictionary, without maintaining backward compatibility with user-written application programs which read the data dictionary tables directly.

### Summary of data dictionary tables

Table name	Description
ACCESS	access privileges on tables and views
ACCCOL	access privileges on specific columns
BACKUP	backup operations performed with backup and restore functionality
CODE	conditions in table, view and domain definitions
COLUMN	columns in tables and views
COMMENT	comments on database objects
DATABANK	databanks in the database
DBFILE	databank file names
DBNAME	databank copies by name
DEFAULT	default value strings for domains
DOMAIN	domains in the database
FUNCTION	translation of id to function or module name
IDENT	idents in the database
INDEX	secondary indexes in the database
INDEXCOL	columns in indexes (primary, secondary and foreign)
MESSAGE	message codes and texts
OBJECT	objects in the database
PRIV	system and object privileges
RESTRICT	domain restrictions valid in level 2
SERVINFO	attributes of current database system or server
SEVERITY	error code severity
SQLLANG	contains one row for each SQL standard and SQL dialect supported
SYNONYM	synonyms in the database
TABLE	tables and views in the database
TABLEX	base, secondary index, and foreign key tables
VIEWCOL	columns in views valid in level 2
VIEWDEP	view dependencies
VIEWRES	view restrictions valid in level 2

**ACCESS****Records privileges held by users on tables and views.**

Column name	Data type	Description
TABLEID	INT(10)	System identifier for table or view
GRANTEE	CHAR(18)	Name of the user who holds the privilege
GRANTOR	CHAR(18)	Name of the user who granted the privilege
TIMENO	INT(10)	Internal system time stamp
DEL	CHAR(1)	Privilege to delete rows 'N' = not held 'Y' = held without grant option 'G' = held with grant option
INS	CHAR(1)	Privilege to insert rows 'N' = not held 'Y' = held without grant option 'G' = held with grant option
LOA	CHAR(1)	Privilege to load rows 'N' = not held 'y' = held without grant option 'g' = held with grant option
SEL	CHAR(1)	Privilege to select rows 'N' = not held 'Y' = held without grant option 'G' = held with grant option
REF	CHAR(1)	Privilege to use the primary key of the table as a foreign key reference 'N' = not held 'Y' = held without grant option 'G' = held with grant option
UPD	CHAR(1)	Privilege to update rows 'N' = not held 'Y' = held without grant option 'G' = held with grant option
UPDCOL	CHAR(1)	Restricted update privilege 'N' = privilege applies to all columns (except primary key columns) 'Y' = privilege applies only to columns with matching rows in ACCCOL
REFCOL	CHAR(1)	Restricted reference privilege 'N' = privilege applies to all columns 'Y' = privilege applies only to columns with matching rows in ACCCOL
SELCOL	CHAR(1)	Restricted select privilege 'N' = privilege applies to all columns 'Y' = privilege applies only to columns with matching rows in ACCCOL
INSCOL	CHAR(1)	Restricted insert privilege 'N' = privilege applies to all columns 'Y' = privilege applies only to columns with matching rows in ACCCOL
Primary key: TABLEID, GRANTEE, GRANTOR, TIMENO		
Secondary index: GRANTEE		

**ACCCOL****Records access privileges held by users on individual columns of a table or view.**

Column name	Data type	Description
TABLEID	INT(10)	System identifier for the table or view
COLNO	INT(3)	Column no to which the update applies
GRANTEE	CHAR(18)	Name of the user who holds the privilege
GRANTOR	CHAR(18)	Name of the user who granted the privilege
TIMENO	INT(10)	Internal system time stamp
ACCTYPE	CHAR(3)	Privilege type 'INS' = insert 'REF' = reference 'UPD' = update
Primary key: TABLEID, COLNO, GRANTEE, GRANTOR, TIMENO, ACCTYPE		
Secondary index: GRANTEE		

**BACKUP****Contains one row for each backup operation.**

Column name	Data type	Description
DATABANK	CHAR(18)	Name of the databank
DATEB	CHAR(8)	Date of operation
TIMEB	CHAR(6)	Time of operation
TYPE	CHAR(2)	Function type 'BC' = backup copy 'BU' = incremental backup 'DL' = drop log
FILENO	INT(3)	File number
IDENT	CHAR(18)	Name of the user who invoked the operation
FILE	CHAR(64)	Name of the file in the host file management system
TIMENO	INT(10)	Time order number
Primary key: DATABANK, DATEB, TIMEB, TYPE, FILENO		

**CODE****Contains one or more rows for each condition for table, view or domain.**

Column name	Data type	Description
CODEID	INT(10)	System identifier for table, view or domain
TYPE	CHAR(1)	Indicates the type of code 'T' = text 'A' = abstract syntax tree
VERNO	INT(3)	Version number for AST generation
SEQNO	INT(3)	Sequential order number
LENGTH	INT(5)	Length of code null if SEQNO not equal to 1
DATA	CHAR(200)	Code field
Primary key: CODEID, TYPE, VERNO, SEQNO		

**COLUMN**

**Contains one row for every column in each table or view.**

Column name	Data type	Description
TABLEID	INT(10)	System identifier of the table or view which contains the column
COLUMN	CHAR(18)	Name of the column
COLNO	INT(3)	Ordinal number of the column in the table or view
COLOFF	INT(5)	Offset in row for the column
TYPE	CHAR(1)	Type of column 'C' = character 'D' = decimal 'F' = float 'I' = integer 'P' = period (interval) 'T' = time (datetime) 'V' = varchar
SIZE	INT(5)	The length attribute of the column: for character types this is the maximum number of chars for numeric types this is the precision for datetime types this is the length of the datetime string for interval types this is the leading precision
SCALE	INT(2)	For decimal types this is the number of digits after the decimal point. For integer or character types this is 0. For floating point types this is -1. For datetime or interval types this is the number of digits in the fractional seconds component.
LENGTH	INT(5)	The internal length of the column
NULLS	CHAR(1)	Indicates whether the column may contain null values 'N' = no 'Y' = yes
DEFAULT	CHAR(1)	Indicates whether column is associated with a default value 'N' = no 'Y' = yes
DOMAINID	INT(10)	System identifier for domain null if no domain exists
UPDATES	CHAR(1)	Indicates whether the column is updatable 'N' = no 'Y' = yes (The value is 'N' if the column is part of a primary key or is derived from a function or arithmetic expression)
COLCARD	INT(10)	Number of distinct values in the column (null if statistics not collected)
COLCNOTN	INT(10)	Number of non-null values in the column (null if statistics not collected)
LOW	CHAR(16)	Lowest value of the column (null if statistics not collected)
HIGH	CHAR(16)	Highest value of the column (null if statistics not collected)

DATATYPE	CHAR(18)	Data type of column: 'CHARACTER' 'CHARACTER VARYING' 'SMALLINT' 'INTEGER' 'INTEGER(p)' 'DECIMAL' 'NUMERIC' 'REAL' 'FLOAT' 'FLOAT(p)' 'DOUBLE PRECISION' 'DATE' 'TIME' 'TIMESTAMP' 'YEAR' 'MONTH' 'YEAR TO MONTH' 'DAY' 'HOUR' 'MINUTE' 'SECOND' 'DAY TO HOUR' 'DAY TO MINUTE' 'DAY TO SECOND' 'HOUR TO MINUTE' 'HOUR TO SECOND' 'MINUTE TO SECOND'
Primary key: TABLEID, COLUMN		
Secondary index: TABLEID, COLNO		
Secondary index: TABLEID, COLOFF		

**COMMENT**

Contains comments for an object.

Column name	Data type	Description
OBJECTID	INT(10)	System identifier for the object
COLNO	INT(3)	Zero or column number
SEQNO	INT(3)	Sequential order number
LENGTH	INT(5)	Length of comment (null if SEQNO not equal to 1)
COMMENT	CHAR(64)	Comment on the object
Primary key: OBJECTID, COLNO, SEQNO		

**DATABANK**

Contains one row for each databank (including system, transaction and log databank).

Column name	Data type	Description
DBANKID	INT(10)	System identifier for the databank
SEQNO	INT(3)	Sequence number
DATABANK	CHAR(18)	Name of the databank
CREATOR	CHAR(18)	Name of the creator of the databank
TYPE	CHAR(1)	Indicates type of transaction handling 'L' = transaction handling with logging 'T' = transaction handling without logging 'N' = transaction handling not used 'W' = work databank (SQLDB)
Primary key: DBANKID		

**DBFILE**

Contains one row for each databank file.

Column name	Data type	Description
DBANKID	INT(10)	system identifier for the databank
SEQNO	INT(3)	sequence number
FILENO	INT(3)	file number
SHADOW	CHAR(18)	shadow name
SUSPEND	CHAR(1)	databank/shadow offline status 'Y' = offline, may not be accessed 'N' = online, may be accessed
FILE	CHAR(64)	shadow file name
Primary key: DBANKID, SEQNO, FILENO		

**DBNAME**

Contains one row for each copy of a databank.

Column name	Data type	Description
DBNAME	CHAR(18)	Databank or shadow name
DBANKID	INT(10)	System identifier for the databank
SEQNO	INT(3)	Sequence number
Primary key: DBNAME		

**DEFAULT**

Contains default value string connected to a domain or column.

Column name	Data type	Description
DOMAINID	INT(10)	System identifier for domain or table with column default value
COLNO	INT(3)	Column number (0 if domain)
SEQNO	INT(3)	Sequence number
LENGTH	INT(5)	Length of default value -2 indicates USER -3 indicates CURRENT_DATE -4 indicates CURRENT_TIME -5 indicates CURRENT_TIMESTAMP (NULL if SEQNO not equal to 1)
DEFVALUE	INT(32)	Default value ( if USER the USER elseif current date/time then CURRENT_DATE or CURRENT_TIME(p) or CURRENT_TIMESTAMP(p) )
Primary key: DOMAINID, COLNO, SEQNO		

**DOMAIN**

Contains one row for each domain.

Column name	Data type	Description
CREATOR	CHAR(18)	Name of the creator of the domain
DOMAIN	CHAR(18)	Name of the domain
DOMAINID	INT(10)	System identifier for the domain
TYPE	CHAR(1)	Type of column 'C' = character 'D' = decimal 'F' = float 'I' = integer 'P' = period (interval) 'T' = time (datetime) 'V' = varchar
SIZE	INT(5)	The length attribute of the column: for character types this is the maximum number of chars for numeric types this is the precision for datetime types this is the length of the datetime string for interval types this is the leading precision

SCALE	INT(2)	For decimal types this is the number of digits after the decimal point. For integer or character types this is 0. For floating point types this is -1. For datetime or interval types this is the number of digits in the fractional seconds component.
LENGTH	INT(5)	The internal length of the column
CODE	CHAR(1)	Indicates whether check option exists 'N' = no 'Y' = yes
DEFAULT	CHAR(1)	Indicates whether default value exists 'N' = no 'Y' = yes
LEVELS	CHAR(1)	Indicates whether legal to use for DB level 2 'N' = no 'Y' = yes
DATATYPE	CHAR(18)	Data type of column: 'CHARACTER' 'CHARACTER VARYING' 'SMALLINT' 'INTEGER' 'INTEGER(p)' 'DECIMAL' 'NUMERIC' 'REAL' 'FLOAT' 'FLOAT(p)' 'DOUBLE PRECISION' 'DATE' 'TIME' 'TIMESTAMP' 'YEAR' 'MONTH' 'YEAR TO MONTH' 'DAY' 'HOUR' 'MINUTE' 'SECOND' 'DAY TO HOUR' 'DAY TO MINUTE' 'DAY TO SECOND' 'HOUR TO MINUTE' 'HOUR TO SECOND' 'MINUTE TO SECOND'
Primary key: CREATOR, DOMAIN		
Secondary index: DOMAINID		

**FUNCTION****Translation of id to function or module name.**

Column name	Data type	Description
MODULEID	INT(3)	Module identification
FUNCTION	INT(5)	Function identification
TEXT	CHAR(40)	Module name or routine name
Primary key: MODULEID, FUNCTION		

**IDENT****Contains one row for each ident.**

Column name	Data type	Description
IDENT	CHAR(18)	Name of the ident
CREATOR	CHAR(18)	Name of the creator of the ident
TYPE	CHAR(1)	Indicates the type of the ident 'U' = user 'G' = group 'O' = OS-user 'P' = program
PASSWORD	CHAR(18)	Enciphered password (null for OS-user without password)
IDENTID	INT(10)	System identifier for the ident
Primary key: IDENT		

**INDEX****Contains one row for each secondary index.**

Column name	Data type	Description
CREATOR	CHAR(18)	Name of the creator of the index
INDEX	CHAR(18)	Name of the index
INDEXID	INT(10)	System identifier for the index
TABLEID	INT(10)	System identifier for the table with the index
Primary key: CREATOR, INDEX		

**INDEXCOL****Contains one row for every column of an index (primary, secondary or foreign).**

Column name	Data type	Description
TABLEID	INT(10)	System identifier for the base table (the referenced table if IXTYPE = 'X')
INDEXID	INT(10)	System identifier for the index table (the base table if IXTYPE = 'P' or 'G' secondary index table if IXTYPE = 'S' or 'Q' foreign key table if IXTYPE = 'F' or 'X' alternate key table if IXTYPE = 'A')
IXTYPE	CHAR(1)	Indicates the type of the index 'A' = alternate (unique) key 'F' = foreign key 'G' = generated primary key 'P' = primary 'Q' = unique secondary index 'S' = secondary 'X' = cross reference for foreign
INDEXNO	INT(3)	Ordinal number of the column in the index table
COLNO	INT(3)	Ordinal number of the column in the base table
COLOFF	INT(5)	Offset in row for the column in the base table
LENGTH	INT(5)	Internal length of the column
SORTDIR	CHAR(1)	Sort direction for column in index 'A' = ascending (used if not secondary index) 'D' = descending
RTABLEID	INT(10)	System identifier for referenced table (base table if IXTYPE = 'X', null if IXTYPE = 'A', 'G', 'P', 'S' or 'Q')
RINDEXID	INT(10)	System identifier for referenced index table null if IXTYPE = 'A', 'G', 'P', 'S' or 'Q')
RCOLNO	INT(3)	Ordinal number of column in referenced table (null if IXTYPE = 'A', 'G', 'P', 'S' or 'Q')
RCOLOFF	INT(5)	Offset in row for column in referenced table (null if IXTYPE = 'A', 'G', 'P', 'S' or 'Q')
Primary key: TABLEID, INDEXID, IXTYPE, INDEXNO		

**MESSAGE****Connection between message codes and their messages.**

Column name	Data type	Description
MODULEID	INT(3)	Module identification
MESSAGID	INT(5)	Message identification
LANGUAGE	INT(3)	Message text language identification
LINENO	INT(3)	Zero = short message, + = long message
MESSAGE	CHAR(80)	Message text in specified language
Primary key: MODULEID, MESSAGID, LANGUAGE, LINENO		

**OBJECT**

Contains one row for each created object,  
one special record for updates of system identifiers and  
one special record for updates of system timestamp.

Column name	Data type	Description
CREATOR	CHAR(18)	Name of the creator of the object
OBJECT	CHAR(18)	Name of the object
COLUMN	CHAR(18)	Blank or name of the column
TYPE	CHAR(8)	Indicates the type of the object: 'COLUMN' 'DATABANK' 'DOMAIN' 'IDENT' 'INDEX' 'SHADOW' 'SYNONYM' 'TABLE' 'TABSTAT' 'VIEW'
DATEC	CHAR(8)	Date of creation in form YYYYMMDD
TIMEC	CHAR(6)	Time of creation in form HHMMSS
OBJECTID	INT(10)	System identifier for the object
COLNO	INT(3)	Zero or column number
Primary key: CREATOR, OBJECT, COLUMN, TYPE		

**PRIV**

Records system and object privileges held by users.

Column name	Data type	Description
PRIV	CHAR(1)	Privilege: 'I' = Ident 'D' = Databank 'T' = Table 'M' = Member 'E' = Execute
OBJECTID	INT(10)	Zero if privilege is Ident or Databank system identifier for databank if privilege is Table system identifier for ident if privilege is Member or Execute
GRANTEE	CHAR(18)	Name of the user who holds the privilege
GRANTOR	CHAR(18)	Name of the user who granted the privilege
TIMENO	INT(10)	Internal system time stamp
GRANTOPT	CHAR(1)	Indicates whether the privilege is held with grant option 'N' = no 'Y' = yes
Primary key: PRIV, OBJECTID, GRANTEE, GRANTOR, TIMENO		

**RESTRICT****Restriction for domains legal for use by DB level 2 (internal information).**

Column name	Data type	Description
DOMAINID	INT(10)	System identifier for domain with restrictions
SEQNO	INT(3)	Restriction order number
LOGOP	CHAR(3)	Logical operator('AND', 'OR')
RELOP	CHAR(2)	Relational operator('EQ', ...)
LENGTH	INT(5)	Length of value
RESVALUE	CHAR(64)	Restriction value
Primary key: DOMAINID, SEQNO		

**SERVINFO****Attributes of current database system or server.**

Column name	Data type	Description
SERVATTR	CHAR(18)	Server attribute
ATTRVAL	CHAR(18)	Attribute value
Primary key: SERVATTR		

**SEVERITY****Error code severity level and optional module.**

Column name	Data type	Description
MODULEID	INT(3)	Module identification
MESSAGID	INT(5)	Zero or message identification
SEVERITY	INT(1)	Message, warning, error, fatal, internal
MODULE	CHAR(20)	Module where error was encountered
Primary key: MODULEID, MESSAGID		

**SQLLANG****Contains one row for each SQL standard and SQL dialect supported.**

Column name	Data type	Description
SOURCE	CHAR(18)	Organization that defined this SQL version
YEAR	CHAR(18)	Year when the relevant source document was approved
BINDSTYL	CHAR(18)	To envisage future adoption of binding styles
PROGLANG	CHAR(18)	Host language for which the binding style is supported
CONFORM	CHAR(18)	Conformance level to the relevant document, the implementation claims
INTEGRIT	CHAR(18)	Indication of whether the implementation supports integrity enhancement feature
IMPLEMEN	CHAR(18)	Identification of SQL product
Primary key: SOURCE, YEAR, BINDSTYL, PROGLANG		

**SYNONYM**

Contains one row for each synonym of a table or view.

Column name	Data type	Description
CREATOR	CHAR(18)	Name of the creator of the synonym
SYNONYM	CHAR(18)	Synonym name for the table or view
SYNONID	INT(10)	System identifier for the synonym
BCREATOR	CHAR(18)	Name of the creator of the table or view
BTABLE	CHAR(18)	Name of the table or view
Primary key: CREATOR, SYNONYM		

**TABLE**

Contains one row for each table or view.

Column name	Data type	Description
CREATOR	CHAR(18)	Name of the creator of the table or view
TABLE	CHAR(18)	Name of the table or view or synonym
TYPE	CHAR(2)	Indicates whether table or view (and synonym): 'T' = table 'V' = view 'TS' = synonym for table 'VS' = synonym for view
TABLEID	INT(10)	System identifier for table or view (if synonym then identifier for referenced table or view)
CODE	CHAR(1)	Indicates whether check clause for table, subselect clause for view 'N' = no 'Y' = yes
UPDATES	CHAR(1)	Indicates whether the view is updatable, null if not view 'N' = no 'Y' = yes
CHECK	CHAR(1)	Indicates whether check option specified for view, null if not view 'N' = no 'Y' = yes
LEVELS	CHAR(1)	Indicates whether legal to use for DB level 2 'N' = no 'Y' = yes
Primary key: CREATOR, TABLE		
Secondary index: TABLEID		

**TABLEX**

Contains one row for each base table, secondary index table or foreign key table.

Column name	Data type	Description
TABLEID	INT(10)	System identifier for the table
TYPE	CHAR(1)	Indicates the type of table: 'A' = alternate key table 'F' = foreign key table 'Q' = unique secondary index table 'S' = secondary index table 'T' = base table
NOKEYCOL	INT(3)	Number of key columns
KEYLEN	INT(5)	Primary key length
NOCOL	INT(3)	Number of columns
RECLEN	INT(5)	Record length
DBANKID	INT(10)	System identifier for the databank containing the table
CARD	INT(10)	Total number of rows in table null if statistics not gathered
Primary key: TABLEID		
Secondary index: DBANKID		

**VIEWCOL**

Contains one row for every column of a view acceptable by DB level 2 (internal information).

Column name	Data type	Description
VTABID	INT(10)	View table identifier
VCOLNO	INT(3)	View table column number
BTABID	INT(10)	Base table identifier
BCOLNO	INT(3)	Base table column number
Primary key: VTABID, VCOLNO		

**VIEWDEP**

Records the dependencies of views on tables or other views.

Column name	Data type	Description
TABLEID	INT(10)	System identifier of the table or view the view is dependent on
TYPE	CHAR(1)	Indicates whether the view is dependent on another view or on a table 'T' = table 'V' = view
DEPTABID	INT(10)	System identifier for the dependent view
Primary key: TABLEID, TYPE, DEPTABID		

**VIEWRES****Records restrictions for DB level 2 (internal information).**

Column name	Data type	Description
VTABID	INT(10)	View table identifier
SEQNO	INT(3)	Restriction order number
LOGOP	CHAR(3)	Logical operator('AND', 'OR')
BTABID	INT(10)	Base table identifier
BCOLNO	INT(3)	Base table column number
RELOP	CHAR(2)	Relational operator('EQ', ...)
LENGTH	INT(5)	Length of value
RESVALUE	CHAR(64)	Restriction value
Primary key: VTABID, SEQNO		

# INDEX

## A

- access privileges 3-7
- authorization
  - backup and restore 6-14
  - database statistics functionality 8-1
  - DBC functionality 10-2
  - DBOPEN 11-1
  - export functions 5-2
  - generating system databanks 5-10
  - import functions 5-7
  - load and unload 5-10
  - MIMSERV functionality 9-3
  - READLOG 7-1

## B

- backup and restore
  - SQL functions 6-8
  - UTIL functionality 6-14
- bufferpool 9-1

## C

- CHECK constraints in tables 3-10
- CHECK OPTION in views 3-10
- connecting to the database 4-5
- connection names 4-6

## D

- data dictionary 3-1
- data files
  - for load and unload 5-8
- data integrity 3-9
- DATABANK privilege 3-6
- databanks 3-2
  - backup and restore 6-1
  - backups 6-3
  - incremental backups 6-4
  - moving between systems 5-1
  - options 3-3
  - physical integrity 10-1
  - restoring from backup 6-10
- database
  - administration 1-1
  - connection 4-5
  - names 4-5
  - security 3-5

- DBC functionality 10-1
  - authorization 10-2
  - error messages 10-6
  - output format 10-2
- DBOPEN 6-3, 6-13
  - functionality 11-1
- default database 4-6
- DELETE privilege 3-7
- domains 3-9
  - default value 3-9
- drop log 6-8, 6-16
- dropping objects
  - recursive effects 3-7
- duplicate rows
  - load operation 5-6, 5-9
  - upgrading system databanks 4-4
- E**
- entity integrity 3-9
- EXECUTE privilege 3-6
- export 5-3
  - authorization 5-2
  - modifying exported definitions 5-3
- F**
- foreign keys
  - in import 5-5
- G**
- group idents 3-2
- H**
- host system backups 6-9
- I**
- IDENT privilege 3-6
- idents 3-2
  - GROUP 3-2
  - in database security 3-5
  - OS\_USER 3-2
  - PROGRAM 3-2
  - recommended organization 3-5
  - USER 3-2
- import 5-3
  - authorization 5-7
  - data loading 5-6
  - naming conflicts 5-4
  - object creation 5-3
  - referential conflicts 5-5
- incremental backups 6-4
- INSERT privilege 3-7
- integrity 3-9
  - domains 3-9
  - entity 3-9
  - in view definitions 3-10
  - of databank files 10-1
  - referential 3-10
  - within tables 3-10

**L**

- Level 1 processes 9-2
- LOAD privilege 3-7
- loading data into tables 5-9
  - authorization 5-10
- LOG databank option 3-3
- LOGDB 3-3, 6-1, 6-3
  - initial creation 4-2
  - reading contents 7-1
  - replacing after loss 6-11

**M**

- machine administration 1-1
- MEMBER privilege 3-6
- MIMSERV functionality 9-3
- module-specific system databanks
  - initial creation 4-3
- multi user systems 2-1
  - creating system databanks 4-2

**N**

- NULL databank option 3-3
- NULL indicators
  - in data files 5-8

**O**

- object privileges 3-6
- optimization 8-3
- os\_user idents 3-2

**P**

- performance
  - optimization 8-3
  - overall system 9-1
- primary key 3-9
- program idents 3-2
  - in backup and restore 6-6, 6-14
  - in import/export 5-7, 5-10
  - in statistics functionality 8-1
- PUBLIC group ident 3-2

**R**

- READLOG functionality 7-1
  - authorization 7-1
  - listing restrictions 7-2
  - output destination 7-2
  - output format 7-4
- recursive effects 3-7
- REFERENCES privilege 3-7
- referential integrity 3-10
- relocating
  - system databanks 3-4, 4-4
  - user databanks 3-4
- reset log 6-8
- revoking privileges
  - recursive effects 3-7

**S**

- SDBGEN 4-2
- SELECT privilege 3-7
- shadowing 6-1
- single user systems 2-1
  - creating system databanks 4-2
- size (block size) 4-2
- SQL compiler 8-1
- SQLDB 3-3
  - initial creation 4-2
  - re-creating 6-12
- sqlhosts 4-5
- statistics functionality 8-1
  - authorization 8-1
  - recommended usage 8-3
- SYSADM 1-1
  - access rights 1-1
- SYSDB 3-3
  - backup protection 6-9
  - initial creation 4-2
- system databanks
  - changing location 3-4, 4-4
  - dropping 4-3
  - file location 3-4
  - upgrading 4-4
- system management functions (SQL) 6-6
- system privileges 3-6
- SYSxxGEN 4-3, 5-10

**T**

- table integrity 3-10
- TABLE privilege 3-6
- threads 9-2
- TRANS databank option 3-3
- transaction control 3-2
- TRANSDB 3-3
  - backup protection 6-12
  - initial creation 4-2
  - replacing after loss 6-12
- tuning 9-1
  - bufferpool size 9-2
  - number of threads 9-2

**U**

- unload data from table 5-9
  - authorization 5-10
- UPDATE privilege 3-7
- upgrading system databanks 4-4
- user databanks
  - changing location 3-4
- user idents 3-2

**V**

- view integrity 3-10
- views
  - in database security 3-9