MaxDB
Database Administration
(Version 7.6/7.7)

Suitable for SAP and non-SAP environments

THE BEST-RUN BUSINESSES RUN SAP™

Content

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

Further Information

Learning Objectives

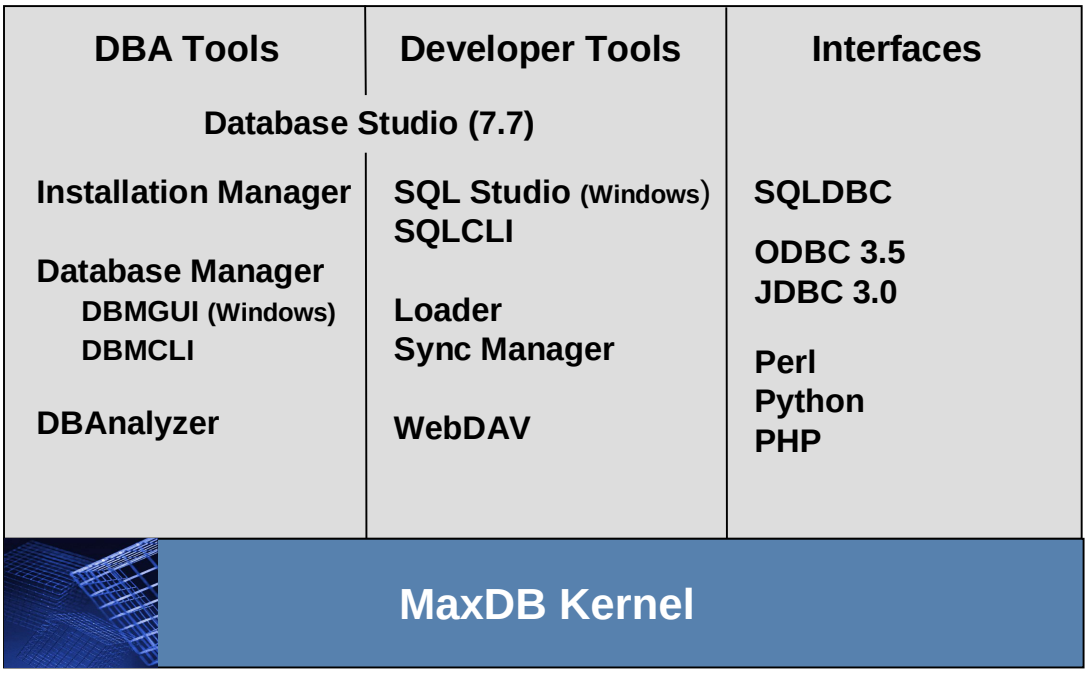
Which tools can be installed for MaxDB

Short introduction of the mainly used tools

Hard disk footprint of a MaxDB instance

Most important log files of a MaxDB instance

What Comes With MaxDB?



The Database Manager is a tool for managing MaxDB databases. You can use it to create, control, monitor, backup, and if necessary restore database instances on the local host or on remote hosts. The Database Manager consists of a server part and a client part. The following clients, providing similar functions, are available, depending on your operating system and your requirements:
Database Manager GUI for Microsoft Windows operating systems
The command line oriented Database Manager CLI
The server is called DBM Server.

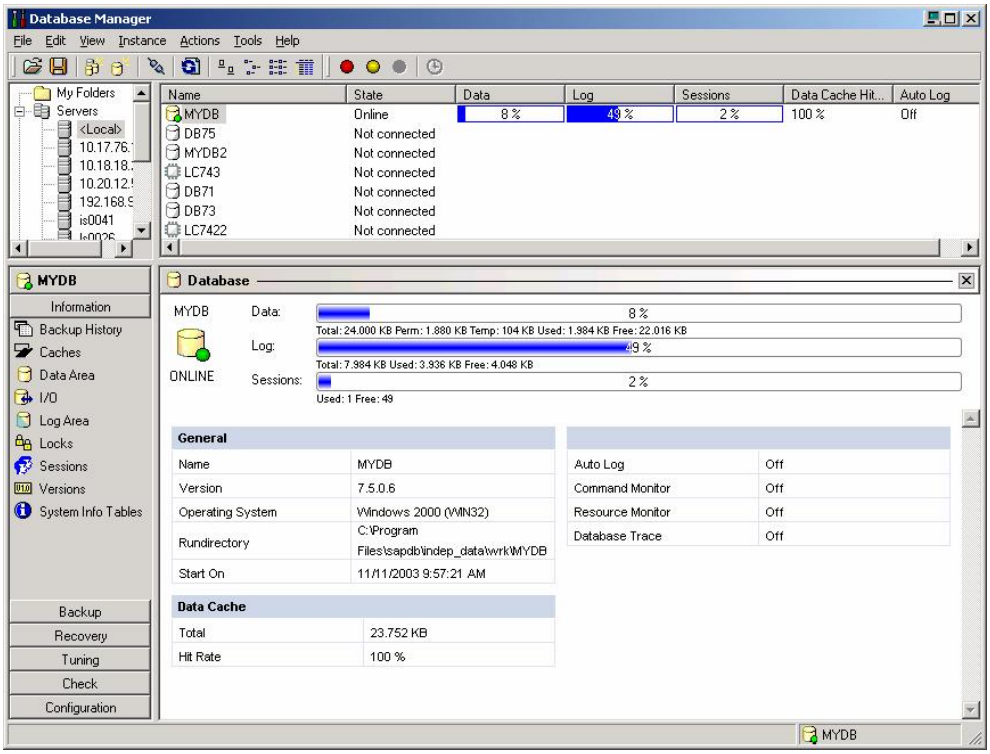
The MaxDB query tools enable easy access to application data and the database catalog of a MaxDB database instance. You can use the query tools to create, execute and manage any number of SQL statements.
SQL Studio provides a user friendly graphical interface for Microsoft Windows operating systems.
The SQLCLI is a command line oriented tool which can be used on UNIX/Linux and Windows systems.

The Database Analyzer is a database tool for analyzing the performance of MaxDB database instances. If problems occur with the database instance, you can use this tool to simplify your search for the cause of the problems.
You can use the Database Analyzer regardless of the MaxDB version. This tool can also access a database instance on a remote host.

The Loader is a database tool for unloading and loading data and for formatting (transforming) data between different data sources and data targets. The Loader can execute commands and SQL statements for these purposes.
The Loader consists of a Loader Server and a client part. A possible client is the command line-oriented Loader CLI. A script interface (such as for Perl, Python, or Java) is available. If you want to react to Loader return codes, you must use one of the script interfaces.

As of MaxDB version 7.7, Database Studio replaces the Database Manager GUI, SQL Studio and Synchronization Manager tools. It also provides new functions, such as a graphical user interface for the Loader tool. With Database Studio, you can create and configure databases, define database objects (data model), monitor databases, backup and restore databases, import and export data, and much more. Database Studio is platform independent – it can run on UNIX/Linux as well as on Windows.

Database Manager GUI



Presented above is the main screen of the Database Manager GUI.

To administer a database instance with the Database Manager GUI you have to register the server and the database instance in the Database Manager GUI.

On the upper left side you see all servers registered in the Database Manager GUI.

On the upper right side you see all registered database instances installed on the selected server.

On the left side you see all possible actions. They are grouped by command types.

On the right side you see the information selected: in this example the filling level of data and log volumes and the cache hitrates.

Database Manager CLI

Connecting to a database instance:

dbmcli -d <db_name> -n <db_server> -u <dbm_user>,<password>

-d: specifies the database instance name

-n: specifies the hostname

-u: specifies the database manager user and password

dbmcli –h shows all possible options

In a dbmcli session type help for a list of all possible commands.

The database system distinguishes between the following user classes:

Database Manager operator (DBM operator)

■DBM operators can work with the Database Manager to administer database instances.

■They cannot log on to the query tools.

■They can log on to the Database Manager more than once. E.g. they can query operating parameters while functions that take a long time are still running.

■You define the first DBM operator when you install a database instance. This operator can then create more DBM operators, which may have some or all of the authorizations of the first operator.

■The Database Manager stores the name and password of the DBM operators in uppercase characters.They can have a maximum length of 18 characters (nine characters in UNICODE). Special characters are not permitted.

Database user

■Database users can work with the query tools, for example to send SQL statements to the database.

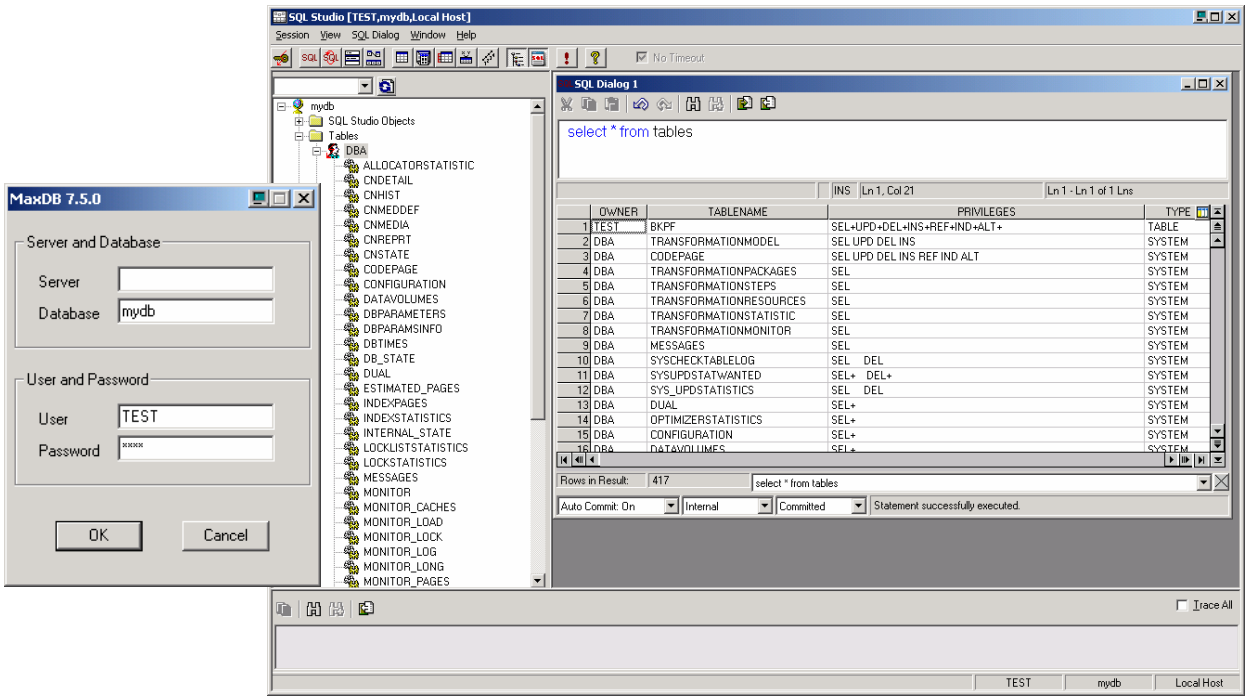
■They cannot log on to the Database Manager.

■The database system uses several different database user classes. The most important are:

- Database system administrator (SYSDBA user)
- Database administrator (DBA user)

Database System Administrator (SYSDBA user)
The SYSDBA user is the initial database user. You create this user when you install the database instance. The SYSDBA user can then use the query tools to define other database users. This user can define database objects and grant other database users privileges for these database objects.
The SYSDBA user also has the following properties:
The SYSDBA user is the owner of system tables. When system tables are uploaded, the upload tool logs on to the database instance as SYSDBA.
The SYSDBA user is the only database user who also has the authorizations of a Database Manager operator.
Database Administrator (DBA user)
A DBA user must be created by the SYSDBA user. DBA users themselves can create database users of the classes RESOURCE and STANDARD. The DBA user can also define database objects and grant other database users all or some privileges for these database objects.

SQL Studio


SQL Studio [TEST,mydb,Local Host]
Session View SQL Dialog Window Help
No Timeout
mydb
SQL Studio Objects
Tables
DBA
ALLOCATORSTATISTIC
CNDETAIL
CNHIST
CNMEDDEF
CNMEDIA
CNREPRT
CNSTATE
CODEPAGE
CONFIGURATION
DATAVOLUMES
DBPARAMETERS
DBPARAMSINFO
DBTIMES
DB_STATE
DUAL
ESTIMATED_PAGES
INDEXPAGES
INDEXSTATISTICS
INTERNAL_STATE
LOCKLISTSTATISTICS
LOCKSTATISTICS
MESSAGES
MONITOR
MONITOR_CACHES
MONITOR_LOAD
MONITOR_LOCK
MONITOR_LOG
MONITOR_LONG
MONITOR_PAGES
SQL Dialog 1
select * from tables
INS Ln 1, Col 21
Ln 1 - Ln 1 of 1 Lns
OWNER TABLENAME PRIVILEGES TYPE
1 TEST BKPF SEL+UPD+DEL+INS+REF+IND+ALT+ TABLE
2 DBA TRANSFORMATIONMODEL SEL UPD DEL INS SYSTEM
3 DBA CODEPAGE SEL UPD DEL INS REF IND ALT SYSTEM
4 DBA TRANSFORMATIONPACKAGES SEL SYSTEM
5 DBA TRANSFORMATIONSTEPS SEL SYSTEM
6 DBA TRANSFORMATIONRESOURCES SEL SYSTEM
7 DBA TRANSFORMATIONSTATISTIC SEL SYSTEM
8 DBA TRANSFORMATIONMONITOR SEL SYSTEM
9 DBA MESSAGES SEL SYSTEM
10 DBA SYSCHECKTABLELOG SEL DEL SYSTEM
11 DBA SYSUPDSTATWANTED SEL+ DEL+ SYSTEM
12 DBA SYS_UPDSTATISTICS SEL DEL SYSTEM
13 DBA DUAL SEL+ SYSTEM
14 DBA OPTIMIZERSTATISTICS SEL+ SYSTEM
15 DBA CONFIGURATION SEL+ SYSTEM
Rows in Result: 417
select * from tables
Auto Commit: On
Internal
Committed
Statement successfully executed.
Trace All
TEST
mydb
Local Host
MaxDB 7.5.0
Server and Database
Server
Database
mydb
User and Password
User
TEST
Password
OK
Cancel

SQLCLI

Connecting to a database instance:

sqlcli -d <db_name> -n <db_server> -u <sql_user>,<password>

-d: specifies the database instance name

-n: specifies the hostname

-u: specifies the sql user and password

Essential SQLCLI Commands

connecting to a database:

\c[onnect] -n <database_server>[:<port>]
-d <database_name>
-u <user_name,password>

print out currently used
host, database, user etc.: \s[tatus]

list columns: \dc [PATTERN]

list procedures: \dp [PATTERN]

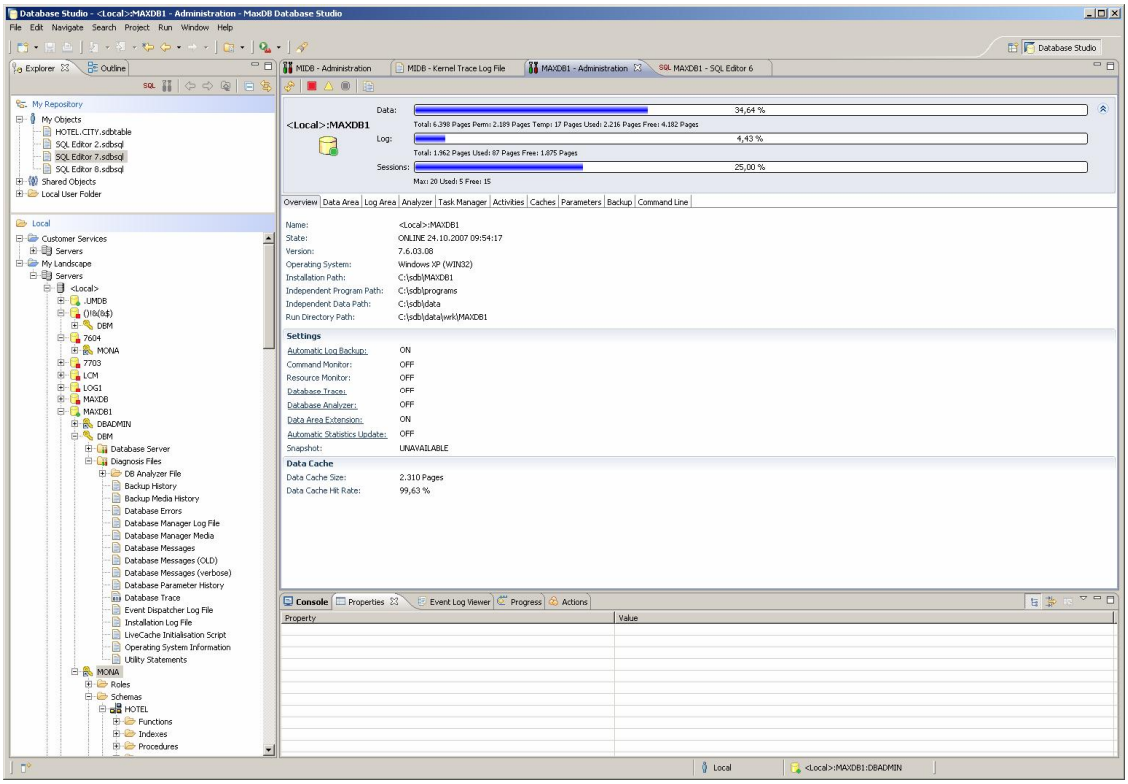
list tables: \dt [PATTERN]

list users: \du [NAME]

list views: \dv [PATTERN]

PATTERN = [OWNER.][OBJECT NAME]

Database Studio



Presented above is the Database Studio.

To administer a database instance with the Database Studio you have to register the server and the database instance in the Database Studio.

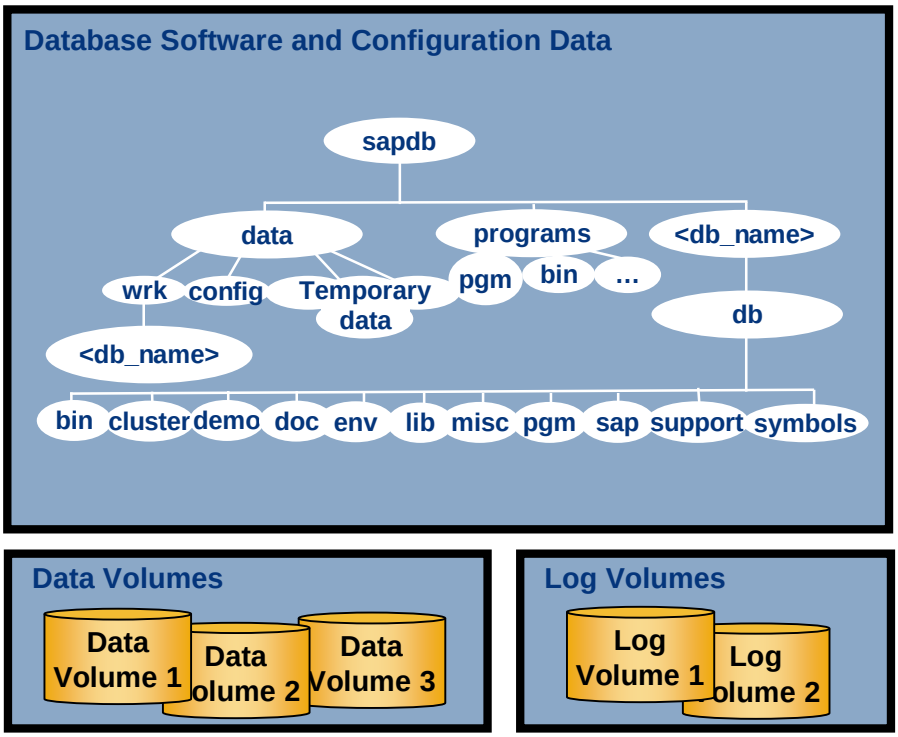
You can order your systems in different landscapes.

On the lower left side you see all servers registered in the Database Studio.

The opened applications are displayed in the upper right part – e.g. an administration window, the SQL editor, a log file, …

The context menu (right mouse click on an item) is essential for the usage of Database Studio.

Hard Disk Footprint



The term volume means a whole physical disk or one part of a physical disk.
A database instance uses three disk areas:

Data volumes

Log volumes and

Database software and configuration data.

The data volumes contain the user data (tables, indexes), the SQL catalog and the converter pages. Because of the database internal striping the data of each table is distributed on all data volumes evenly.

In the log volumes all changes of data are stored in the form of redo log entries, which are used in case of restoring to redo all changes not being part of a full database backup.
To assure savety the disks of the log volumes should be mirrored physically or by the operating system.

If it is not possible to mirror the log volumes physically or by the operating system, they can be mirrored by the database instance.

Redo log entries only contain the changes of the transactions, i.e. the after images. The undo log entries are stored separately in the data area.

With the database software executables, sources und utilities are delivered which allow the creation of database processes and working with the database instance. The software is installed in a fixed directory with some subdirectories. During database work additional log and status files are created which are stored in the data directory.

The Independent Directories 1/2

The independent data directory contains the configuration data and rundirectories of MaxDB database instances and MaxDB applications.

The independent programs directory contains the programs and libraries shared by the MaxDB database instances and MaxDB applications. These programs are downwards compatible.

The default location of the independent data directory is

- /sapdb/data/ on UNIX/Linux
- C:\sapdb\data on Windows

The default location of the independent programs directory is

- /sapdb/programs/ on UNIX/Linux
- C:\sapdb\programs on Windows

The Independent Directories 2/2

The location of these directories is specified during the first installation of MaxDB software

If you don’t know the locations of these directories you can determine them with:

- dbmcli dbm_getpath indepprogpath
- dbmcli dbm_getpath indepdatapath

The Dependent Directory

Contains the server software that depends on the database version (e.g. kernel)

Several dependent directories can exist alongside each other

Every database instance should be assigned to its own dependent directory – that is two database instances should not share one dependent directory

The location of this directory is specified during the installation of MaxDB software, the default is:

- /sapdb/<db_name>/db (on UNIX/Linux)
- C:\sapdb\<db_name>\db (on WINDOWS)

If you don't know the location of this directory you can determine it with:

- dbmcli inst_enum

The Rundirectory

By default most log and status files are stored in the rundirectory

Every database instance has its own rundirectory

The default location of the rundirectory is

- /sapdb/data/wrk/<db_name>/ on UNIX/Linux
- C:\sapdb\data\wrk\<db_name> on Windows

The rundirectory location is specified by the database parameter RUNDIRECTORY

If you don't know the location of the rundirectory you can determine it with:

- dbmcli –d <db_name> -u <dbm_user>,<pwd> param_directget RUNDIRECTORY

The Data/Log Volumes

The location of the data volumes is specified by the database parameters DATA_VOLUME_NAME_<#>

The location of the log volumes is specified by the database parameters LOG_VOLUME_NAME_<#>

The default locations for SAP installations are:
C:\sapdb\<db_name>\data\DISKD0001
C:\sapdb\<db_name>\log\DISKL0001

The data/log volumes contain binary data

The data/log volumes are exclusively managed by the database kernel

Client programs don’t access the data/log volumes directly

The client programs establish a connection to the database kernel and then send requests as SQL queries

Status and Log Files

By default all status and log files are located in the rundirectory

Files that are written by the database kernel:

- Version 7.6: knldiag, knldiag.err, dbm.utl, dbm.knl
- Version 7.7: KnlMsg, KnlMsg.old, KnlMsgArchive, dbm.knl

Files that are written by the Database Manager:

- dbm.prt, dbm.ebp, dbm.ebl

All log files of version 7.6 are files which can be read with any text editor.

As of version 7.7 the files written by the database kernel are written in pseudo XML. The dbmserver converts them into readable files.

Database Message Log – Version 7.6

Most important status file: knldiag

Contains status and error messages

Has a fixed size (database parameter KERNELDIAGSIZE) and is written cyclically

- A header with the startup messages is persistent

Is always created during startup

- The previous knldiag content is copied to knldiag.old

Database Message Log – Version 7.7

Most important status file: KnlMsg

Replaces file knldiag, contains status and error messages

Consists of three parts:

- startup messages
- runtime information
- shutdown/crash messages

The biggest part (runtime information) is written cyclically and has a fixed size (database parameter KERNELDIAGSIZE)

Is always created during startup

The previous KnlMsg content is copied to KnlMsg.old

- In directory DIAGHISTORY\History_KnlMsg up to KNLMSG_HISTORY_NUM copies of file KnlMsg
  (called KnlMsg_<date>_<time>) are kept

File KnlMsg can be displayed with dbmcli:

dbmcli -d <dbname> -u <dbmusr>,<pwd> -nohold file_getfirst KNLMSG

To view the files in directory History_KnlMsg use:

dbmcli -d <dbname> -u <dbmusr>,<pwd> -nohold file_getfirst DIAGDIR#History_KnlMsg/KnlMsg_<date>_<time>

The Error Log – Version 7.6

knldiag.err

All error messages occuring during operation are also written to the error log

As the entries in this file are not overwritten, this file is important for extended error analysis

When the database state changes from OFFLINE to ADMIN the message ’Starting’ is written to this file

The Kernel Administration Log – Version 7.6

dbm.utl

Contains administrative commands sent to the database kernel (e.g. SHUTDOWN, BACKUP, CHECK DATA) including their return code(s)

Has a fixed size and is written cyclically

The Error Log – Version 7.7

KnlMsgArchive

Replaces files knldiag.err and dbm.utl

Contains

- all error messages written into file KnlMsg

and messages concerning

- all configuration changes (add and drop volume)
- all initializations and restore operations
- all consistency checks

Is not overwritten cyclically, but can be truncated by dbmserver (contains maximal the data of one year)

File KnlMsgArchive can be displayed with dbmcli:

dbmcli -d <dbname> -u <dbmusr>,<pwd> -nohold file_getfirst KNLMSGARC

The Database Manager Log File

dbm.prt

Contains all commands that are sent to the dbmserver (administrative commands)

The Backup Log Files

Backup history: dbm.knl

Contains information about all backups with label, date, time, size, returncode

External backup log: dbm.ebp, dbm.ebl

dbm.ebp contains information about backups created using external backup tools (like Networker, ADSM, Backint,...)

dbm.ebp is overwritten when a new request is sent to an external backup tool (using a new dbmserver)

dbm.ebl contains a history of dbm.ebp files – the size depends on the dbmserver parameter DBM_EBLSIZE

The X-Server Log File

X-Server log file: xserver_<hostname>.prt

Contains error messages concenring remote communication.

If network problems occur, error messages are logged in this file.

The first part contains information about operating system settings and the user envronment in which the x_server was started (e.g. limits concerning heap usage or number of open files).

This file is stored in directory <indep_data_path>\wrk.

The Kernel Dump

Kernel Dump File: knldump

Contains the global memory, e.g. :

- lock list, Data Cache, Catalog Cache, ...

This file is created:

- during a database crash
- using the tool x_diagnose: by a DBA user
- using dbmcli: db_stop –dump

This file can get very large. It is written in binary format and can only be evaluated using x_diagnose.

This file is mainly needed in case of database crashes. The developer will request access to this file if necessary.

UNIX: If the database crashes because of UNIX signal, no knldump file is written.

File ***knldump*** is created in the rundirectory. If necessary the location and filename can be changed using database parameter _KERNELDUMPFILE – e.g. if more space is needed to store this file.

The RTE Dump

RTE Dump File: rtedump

Contains the status of the runtime environment in case of a crash

- x_cons <SID> show all
- helps to identify active tasks
- contains detail information of the tasks
- contains information about regions, suspend reasons, counter statistics

Used in addition to the KnlMsg file to analyze a database crash

This file is created in the rundirectory

Other Dump Files

Dump Files: *.bad, *.cor

Dump of corrupted pages

- check sum error: *.bad
- problem with page content: *.cor

These files are created in the rundirectory.

They have to be evaluated using x_diagnose.

The developer will request access to these files if necessary.

DIAG_HISTORY

To save relevant log files in case of a database crash, these logfiles are saved to directory DIAGHISTORY – a sub-directory of the rundirectory.

The location of this directory can be changed using database parameter DIAG_HISTORY_PATH.

The number of stored histories is specified by database parameter DIAG_HISTORY_NUM.

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

Further Information

Learning Objectives

Know how to use SDBSETUP to

- install the MaxDB software and the database instance
- upgrade the MaxDB software and the database instance
- drop the database instance and uninstall the MaxDB software

Know how to use SDBINST to

- install the MaxDB software

Know how to use SDBUPD to

- upgrade the MaxDB software and the database instance

Know how to use SDBUNINST to

- uninstall the MaxDB software

Installation Tools

When you install an SAP system, the MaxDB Software is installed automatically during the SAP installation with SAPINST.

However, when you would like to set up a standby instance or create a system copy, you might want to install just the database software – without an SAP system.

In this case you can use either SDBSETUP or SDBINST.

For updating an existing database instance to a newer Build of the same major database version, use SDBUPD.

For upgrading the database instance to a new major release, please follow the instructions in the upgrade guide.

If you want to remove all MaxDB software from your server, then use SDBUNINST.

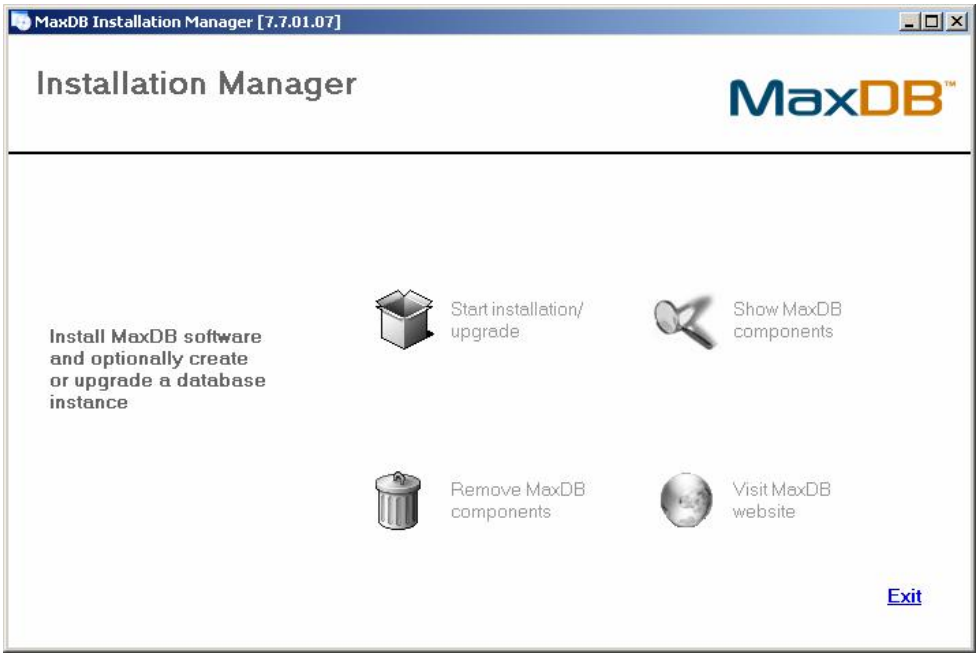
Installation Manager

SDBSETUP (= Installation Manager) can be used to install, upgrade and uninstall the database software.

The Installation Manager allows also to install (and drop) a database instance and to load demo data into this database instance.

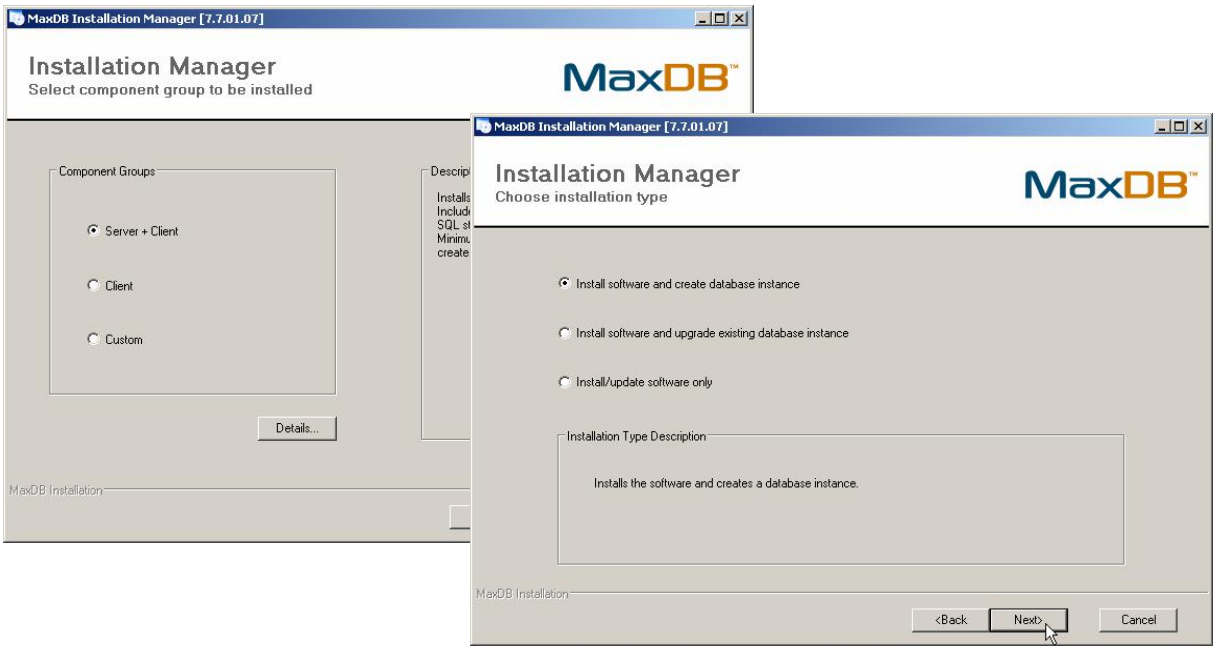
The Installation Manager is a graphical user interface which is available for Windows, Linux and UNIX systems.

Installation Manager: Start Screen



To install MaxDB software choose *Start installation/upgrade*.

Installation Manager: What Should be Installed

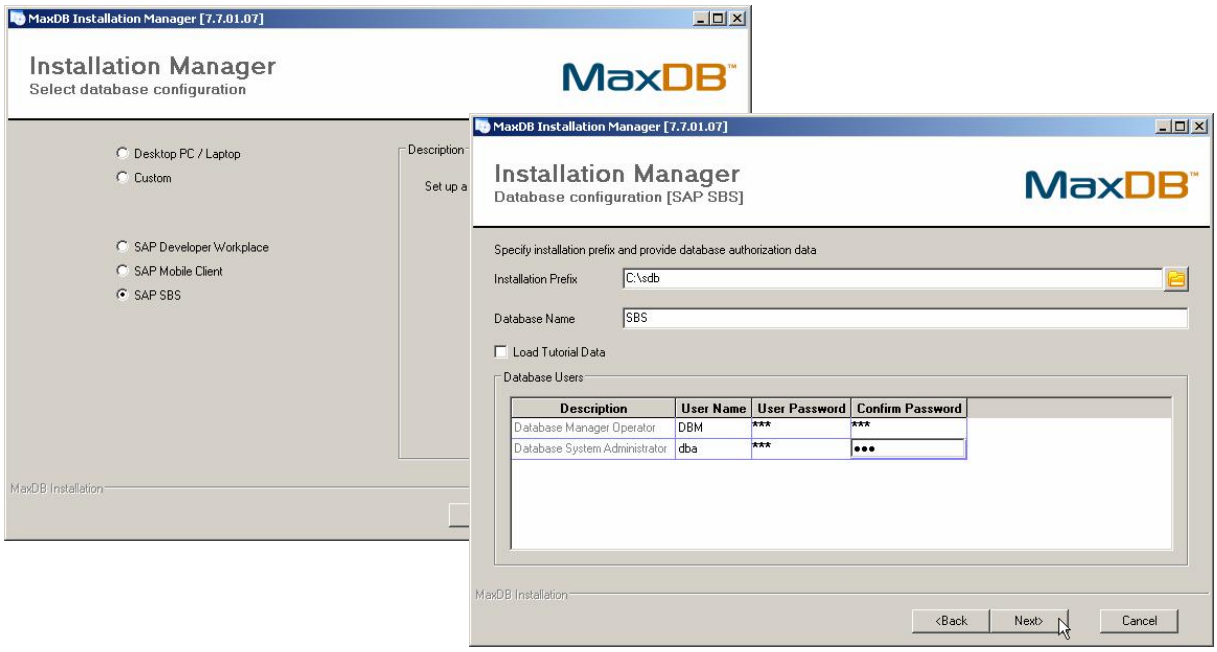


In the first step you have to specify which part of the database software you'd like to install. *Server+Client* is the complete software package which you need on the database sever.

The *Client* package needs to get installed on other application servers. This software is required to connect from the application server to the remote database instance.

In the next step you can decide if you would like to install just the database software or if also a database instance should be created. You can also upgrade an existing database instance to the new software version.

Installation Manager: Template Selection



If you choose to install a database instance, SDBSETUP provides different configuration templates. Using one of these templates, most parameters are predefined.

However, you still have to specify the installation directory, the database name and users and passwords for the database users.

Installation Manager: Database Configuration

Adjust or confirm the values for the data and log volumes.

Adjust or confirm the amount of memory and the number of CPUs the database should use as well as the max. number of parallel database sessions.

Installation Manager: Advanced Database Settings

In the next step you can activate some automatic features for your database instance.

If the summary is correct, start the installation.

Installation with SDBINST 1/3

Prerequisites on Windows

- You have administration rights for your host.
- Download the MaxDB software from the Service Marketplace.
- Use SAPCAR to unpack the software package to a local directory of your choice.
- In this local directory, the system creates a subdirectory *maxdb-server-<OS>-<32/64-Bit>-<PA>-<Build>*
- and stores the MaxDB software there.

Prerequisites on UNIX/Linux:

1. You are in a root shell

2. Download the MaxDB software from the Service Marketplace.

3. Use SAPCAR to unpack the software package to a local directory of your choice.

In this local directory, the system creates a subdirectory
`maxdb-server-<OS>-<32/64-Bit>-<PA>-<Build>`
and stores the MaxDB software there.

Preparations on UNIX/Linux:

The installation program enters the required services sql6 and sapdbni72 in the file /etc/services if they do not already exist there. If these services are managed centrally for your system on the network (NIS), you must enter them there as follows.
sql6 7210/tcp
sapdbni72 7269/tcp

Create the needed operating system user/group:

- recommended user: sdb
- recommended user group: sdba
- The primary group of sdb has to be sdba.
- The user account has to be locked.

If you manage user groups and owners locally on your server, then we recommend that you register the names of the user groups and the owner on the operating system before you start the installation. However, you can also specify them during the installation.

If you manage user groups and owners for your system at a central location in the network, then you must create them here before you start the installation.

If the path <independent_program_path>/bin is not yet entered in your environment variable PATH, enter it now for all users who want to use MaxDB.

Installation with SDBINST 2/3

Installation Procedure on Windows

- Open the Command Prompt or the Windows Explorer and navigate to the local directory that contains the installation files:

  maxdb-server-<OS>-<32/64-Bit>-<PA>-<Build>

- Enter SDBINST or double-click this program to start the installation program.
- Confirm the installation profile all.
- If MaxDB software installations already exist on your server, a list of these installations is displayed.
  - If you want to install new database software in parallel with an existing installation, enter none. Specify an installation path that is different from the existing installations. Define an installation path in parallel with the existing installation paths.

Installation Procedure on UNIX/Linux

Go to the directory
`maxdb-server-<OS>-<32/64-Bit>-<PA>-<Build>`,

into which you unpacked the installation package.

Enter ./SDBINST to start the installation program.

Confirm the installation profile all.

Enter the user/group information:

Special operating system user sdb (owner of the MaxDB database software)

Administrator group sdba

If the group and the owner have not yet been created, the system asks whether you want to create them. Then the group and the owner are created locally on your server.

If MaxDB software installations already exist on your server, a list of these installations is displayed.

If you want to install new database software in parallel with an existing installation, enter none. Specify an installation path that is different from the existing installations. Define an installation path in parallel with the existing installation paths.

Installation with SDBINST 3/3

Installation Procedure on Windows - continued

- If there are no MaxDB installations on your host, enter the following paths:
  - Path for storing the data, configuration, and run directories of MaxDB database instances and MaxDB applications (change the system default C:\Program Files\sdb\data to a path without blanks)
  - Path for storing the programs and libraries shared by the MaxDB database instances and MaxDB applications (change the system default C:\Program Files\sdb\programs to a path without blanks)
  - Path for storing the server software that depends on the database version(change the system default *C:\Program Files\sdb\<version>* to a path without blanks)
- You see a success message about the installation.
- If there was no MaxDB installation on your host until now, shutdown and restart your host to make the independent program path known to the system.

Installation Procedure on UNIX/Linux – continued

If there are no MaxDB installations on your host, enter the following paths:

Path for storing the data, configuration, and run directories of MaxDB database instances and MaxDB applications (the system default value is /var/opt/sdb/data – for SAP installations we recommend /sapdb/data)

Path for storing the programs and libraries shared by the MaxDB database instances and MaxDB applications (the system default value is /opt/sdb/programs – for SAP installations we recommend /sapdb/programs)

Path for storing the server software that depends on the database version.
This path must be unique. Multiple directories with different versions can exist alongside each other (the system default value is /opt/sdb/7500 – for SAP installations we recommend /sapdb/<db_name>/db)

You see a success message about the installation.

Installation of Database Manager GUI and SQL Studio on Windows

The MaxDB tools Database Manager GUI and SQL Studio have to be installed separately using the self-extracting files.

Upgrade Prerequisites

Prerequisites for Using SDBUPD or SDBSETUP

- Program SDBINST or SDBSETUP (or SAPINST) was used to install the existing software.
- Download the MaxDB software from the Service Marketplace.
- Unpack the software package to a local directory of your choice. In this local directory, the system creates a subdirectory

  maxdb-server-<OS>-<32/64-Bit>-<PA>-<Build>

  and stores the MaxDB software there.
- On Windows: Stop all database instances.
- The database parameters of the database instance you want to upgrade must not have been changed since the last restart.
- The database instance that you want to upgrade is the only instance that refers to the installation path of the software you want to update.

Each time you upgrade database instances, we recommend that you watch out for any errors. Make sure that you have made complete backups of all data of the relevant database instances, and that the database software of the source version is at hand. Only then can you return to the source version straight away, if necessary, and avoid long periods of system downtime.

On Windows no remote communication is possible between the database instances and applications during the upgrade, since the X Server software may have to be substituted, depending on the source and target version of the upgrade. Therefore, stop all database instances, so that SDBUPD can stop the X Server program that is still running. Alternatively, you can stop the X Server program yourself before the upgrade.

On UNIX/Linux systems the X Server can be substituted online, i.e. without stopping the X Server and other database instances.

Upgrade Procedure with SDBSETUP

If you use SDBSETUP for a database upgrade, choose

- “Start Installation/upgrade”
- “Server + Client”
- “Install software and upgrade existing database instance”

Select the database instance to be updated and enter the Database System Administrator (dba).

SDBSETUP performs the upgrade and displays a success message if the upgrade is completed successfully. It then flags the installation as complete.

Upgrade Procedure with SDBUPD

To use SDBSETUP for a database upgrade, go to the directory *maxdb-all-<os>-32|64bit-<arch>-<version>_<build>,*

into which you unpacked the installation package.

Start the SDBUPD program with the following command:

UNIX/Linux:

./SDBUPD –d <db_name> -u <dbm_user>,<password>

Microsoft Windows:

sdbupd –d <db_name> -u <dbm_user>,<password>

SDBUPD performs the upgrade and displays a success message if the upgrade is completed successfully. It then flags the installation

as complete.

Uninstall Procedure

If you use SDBSETUP to uninstall the MaxDB software, choose

- “Remove MaxDB components”

Select the components to be uninstalled

- “Base” to uninstall the database instance and all MaxDB software
- “Database Kernel <db_name>” to drop the database instance <db_name> and to uninstall the database kernel software (dependent directory)

Confirm the drop database dialog and enter the Database System Administrator (dba).

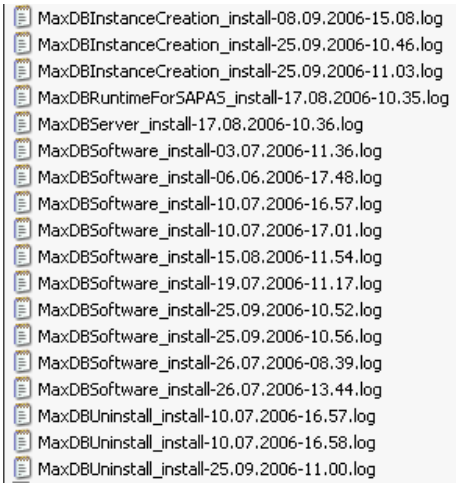
SDBSETUP performs the uninstall and displays a success message if it is completed successfully.

The uninstall procedure with SDBUNINST is described in note #599129.

Installation Log Files

All installation tools writes their log files into directory <indep_data_path>\wrk

The name of the log file depends on the kind of installation and contains the timestamp:



Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

Further Information

Learning Objectives

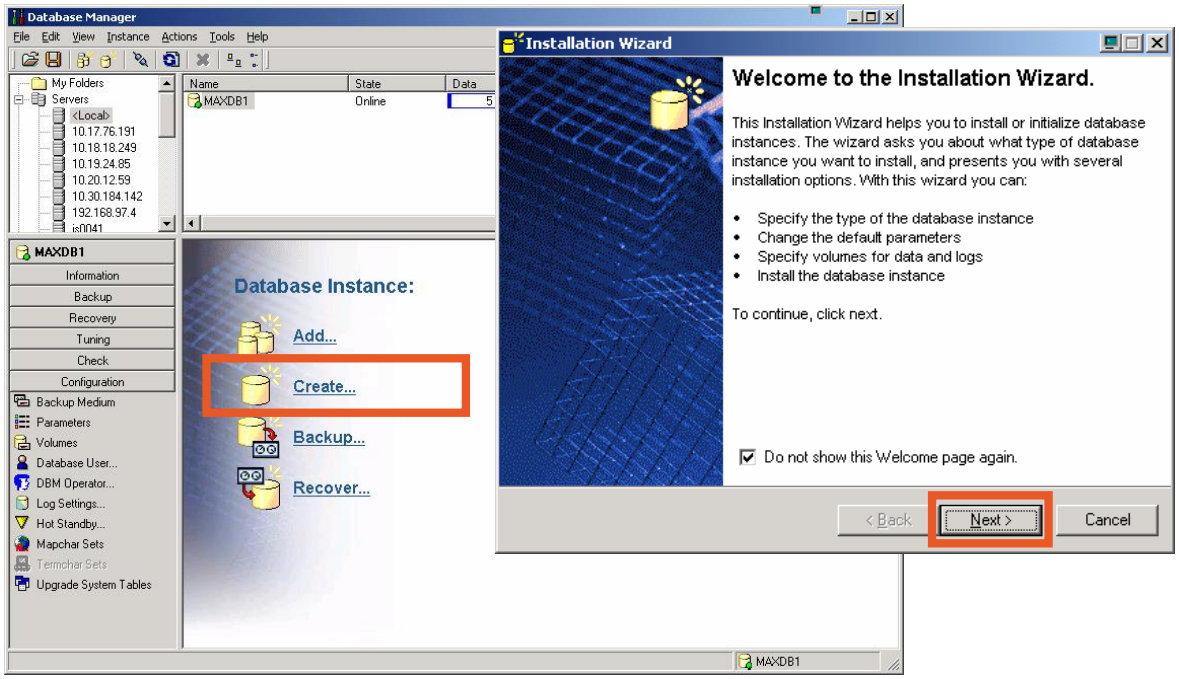
Know how to install a database instance using DBMGUI.

Instance Installation

Instance installation steps in the Database Manager GUI:

- Selecting an installation template
- Specifying a database name and server
- Selecting the software version
- Specifying the Database Manager Operator
- Choosing parameter initialization mode
- Specifying the instance type
- Adjusting parameter settings
- Specifying the data and log volumes
- Specifying backup media
- Specifying the Database System Operator
- Double-checking the information and starting the installation

Instance Installation 1/12
Starting the Installation Wizard



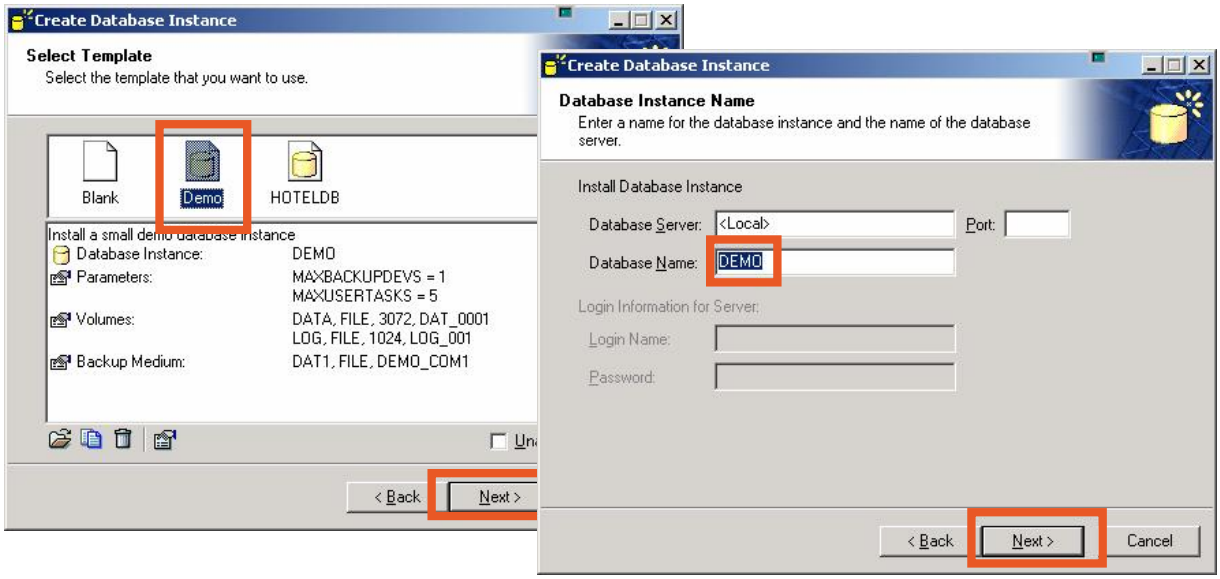
The Database Manager (as of version 7.5) is designed to support the most important processes by special wizards. These are the installation, backup and recovery wizards.

The Add... feature serves to bind database instances to this Database Manager.

To install a database instance enter the installation wizard by choosing Create....

The initial welcome popup can be suppressed for further use.

Instance Installation 2/12
Selecting Installation Template, DB Name and Server

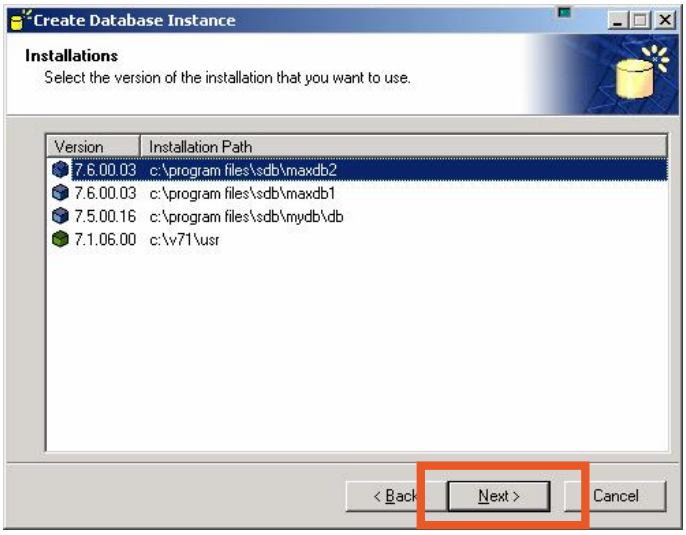


First you have to select an installation template. According to the selcted template the database paramters are initialized with reasonable values for a demo database or a database for the tutorial data. With template Blank the parameters are initialized with the default values. However, the parameter values can be changed later during the installation.

Enter the name of your new database instance.

If you install a local instance, server and related fields have to be left free. Otherwise the server name and the adminstration user together with his/her password have to be specified.

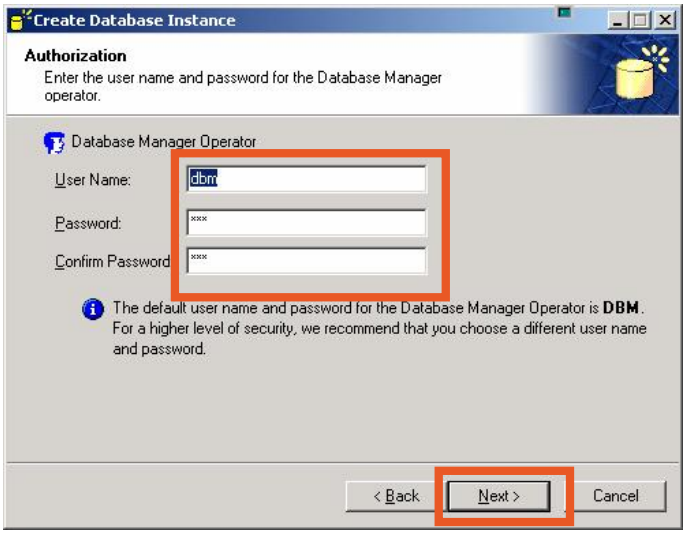
Instance Installation 3/12
Selecting the Software Version



A list of all installed MaxDB and SAP DB software versions is presented.

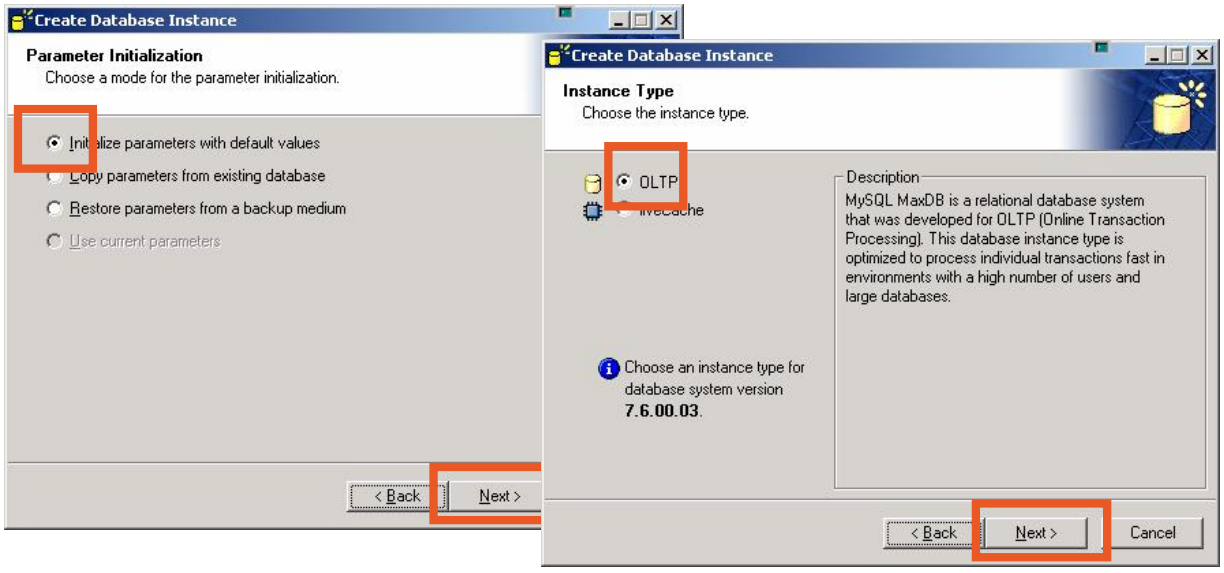
Select the database version for your instance.

Instance Installation 4/12
Specifying the Database Manager Operator



The Database Manager user ‘dbm‘ with password ‘dbm‘ is presented for quick set-up but of course user and password can be changed.

Instance Installation 5/12
Choosing Parameter Initialization and Instance Type



There are 4 possibilities to initialize the parameters:

You can use ‘Initialize parameters with default values‘ for a new installation.

You can copy the parameters from another (existing) database

You can restore the values from a backup medium.

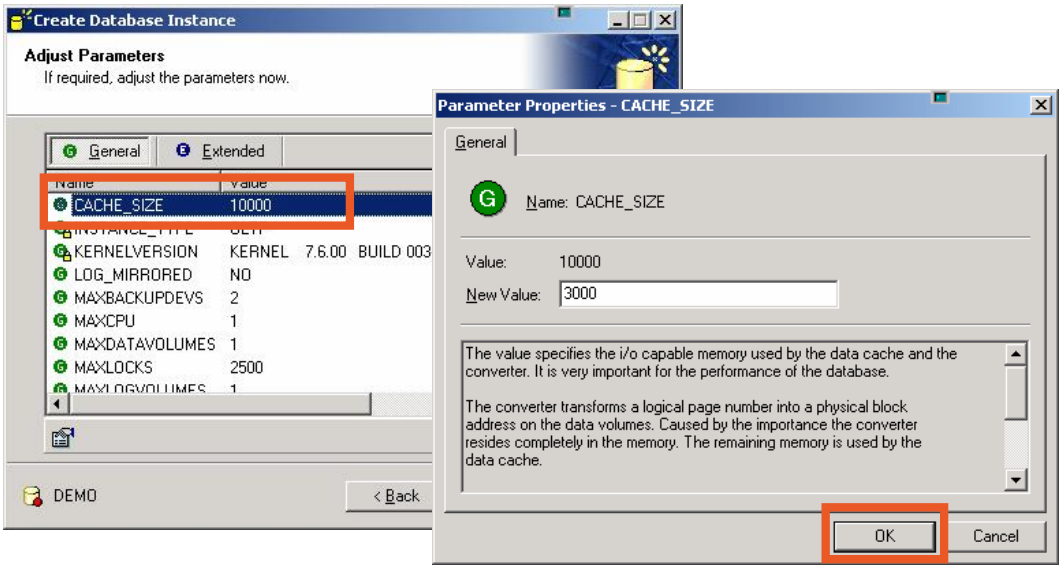
You can use the current values if you reinstall an existing database

It is possible to install different types of database instances:

An OLTP database which is the ‘normal‘ instance type,

A liveCache instance which is used for APO (SAP only)

Instance Installation 6/12
Adjusting Parameter Settings

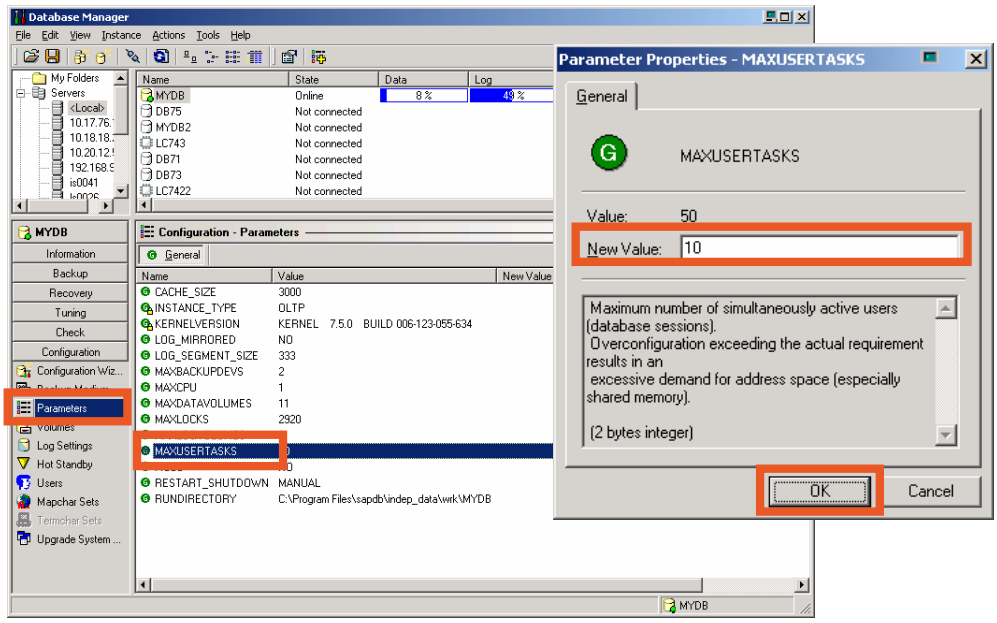


On this screen you can change the initial parameter values. A short explanation for every parameter can be found in the column ‘description‘.

The ‚General‘ view shows only the standard and thus most significant parameters. The ‚Extended‘ view offers many more parameters that normally do not have to be changed except for support cases.

To change a parameter value, double-click on the parameter name. The maintenance window with additional information will appear.

Parenthesis:
Changing Database Parameters



If you have to change a database parameter after you have finished the installation, select the menu ‚Configuration → Parameters‘.

Select the parameter by double-clicking the parameter name.

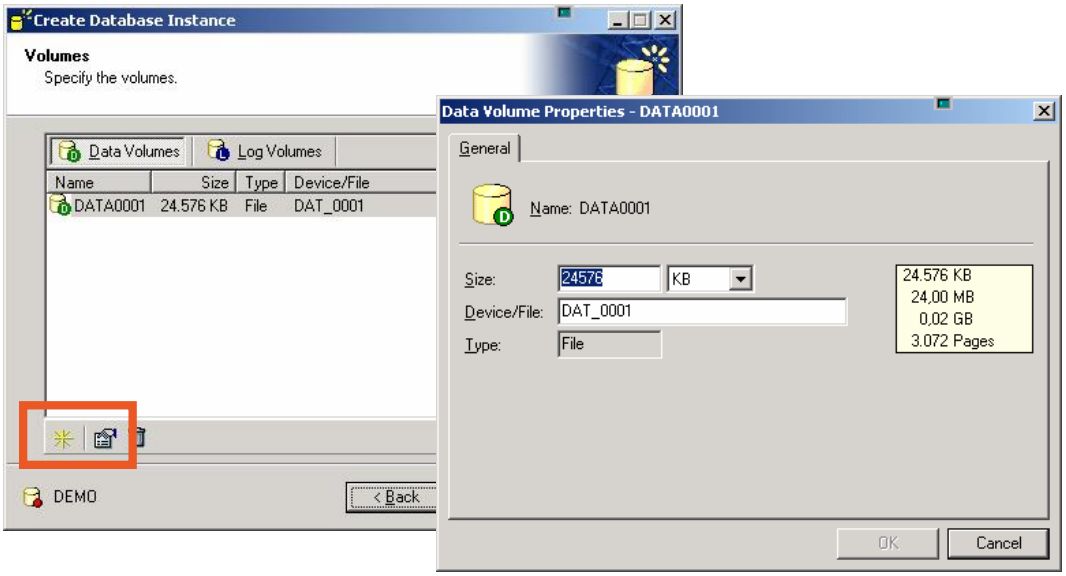
Specify the new value and press ‚OK‘.

The change will become visible in the ‚New Value‘ Column of the DBMGUI display.

The new parameter value will become valid only after a database restart.

After you have changed parameters, you have to stop the database to mode ‚ADMIN‘ or ‚OFFLINE‘ and then restart the database to ‚ONLINE‘ mode.

Instance Installation 7/12
Specifying the Data Volumes



The next installation step is to specify the database volumes for Data and Log.

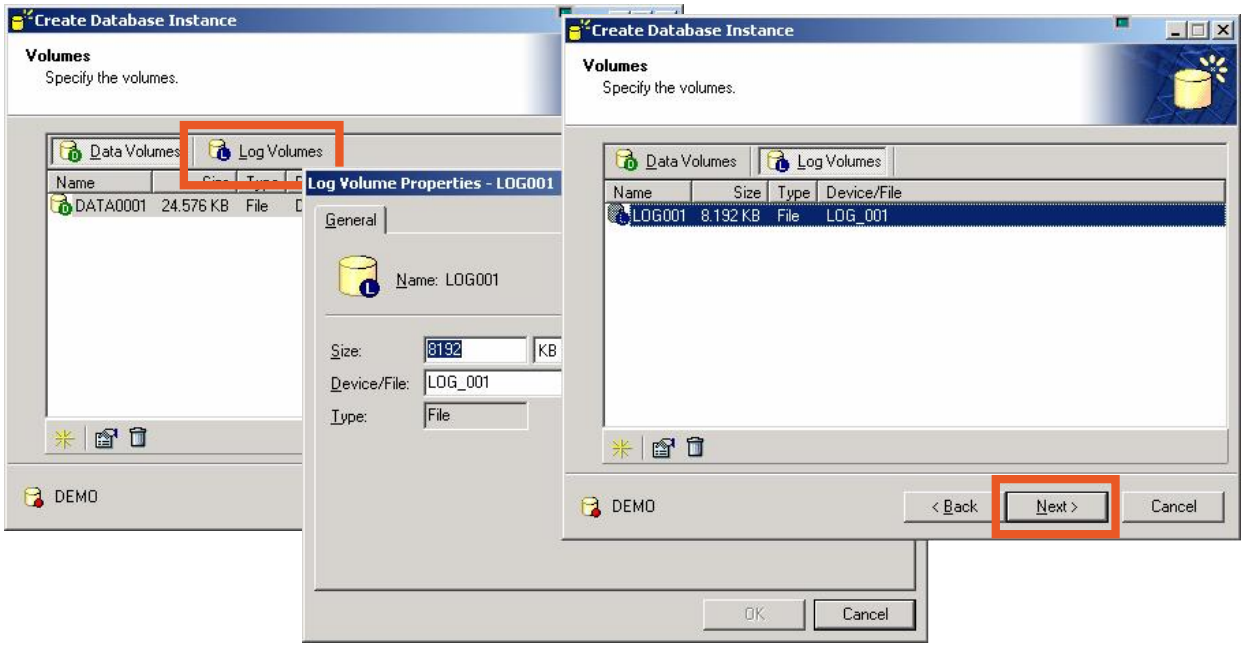
Select the tab strip (Data or Log).

Click on the create asterisk to enter the size, device type and location or on the Properties button to change the proposed values.

Location specifies the path (complete path is recommended) to the volume. If only the file name is specified, the file is created in the rundirectory.

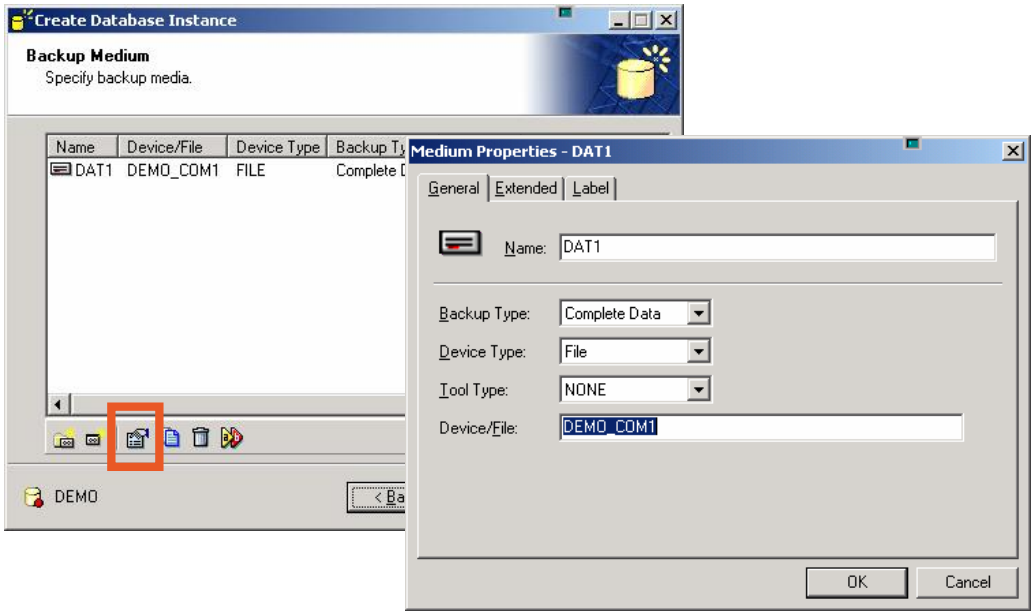
Type specifies if the volume is a file or a raw device.

Instance Installation 8/12
Specifying the Log Volumes



The values for the log volume(s) are specified in the same way as for the data volumes.

Instance Installation 9/12
Specifying Backup Media



Now you are requested to set up your first backup medium. A backup medium is assigned to every backup you carry out. Backup media include files, tapes, and pipes. The media for all types of backup and recovery operations can be reused under this logical name. You can define a single backup medium or a group of several backup media.

You can choose any name for the medium name.

Besides the medium name you have to specify a location. You have to enter the complete path of the medium. If you specify only a file name this file will be created in the rundirectory of the database.

Each backup medium is bound to a type of backup.

Complete Backup

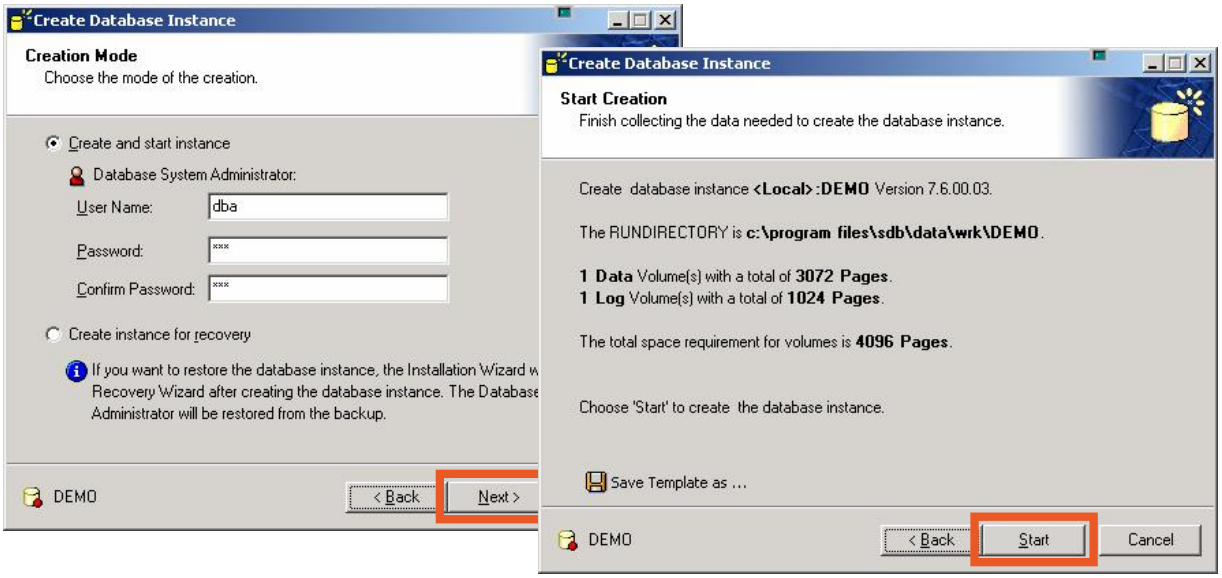
Incremental Backup (Changed pages since last complete backup)

Log Backup

Furthermore an external backup tool can be assigned to this medium.

For template DEMO a single backup medium is already defined.

Instance Installation 10/12
Specifying DB System Admin / Starting the Installation

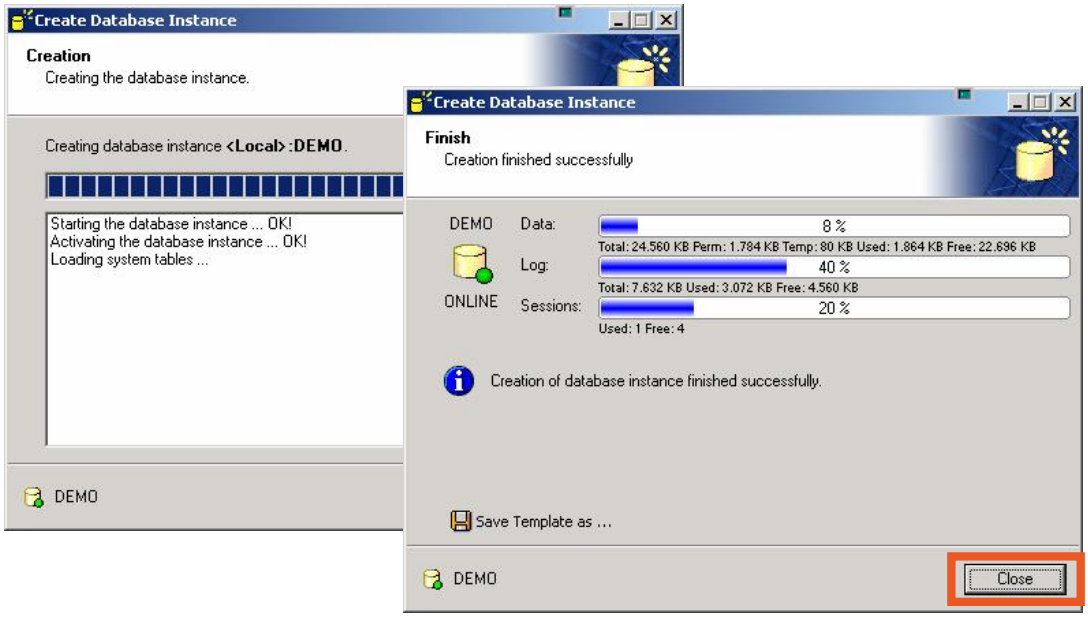


At this point, all data that is required to run the installation has been collected, except for the database user name and password, which has to be entered now.

For quick processing the user ‚dbadmin' with password ‚secret' is preset but of course this can be modified.

Before the installation starts, the installation summary is shown. Select the ‚Start' button to start the installation.

Instance Installation 11/12
Waiting for Completion



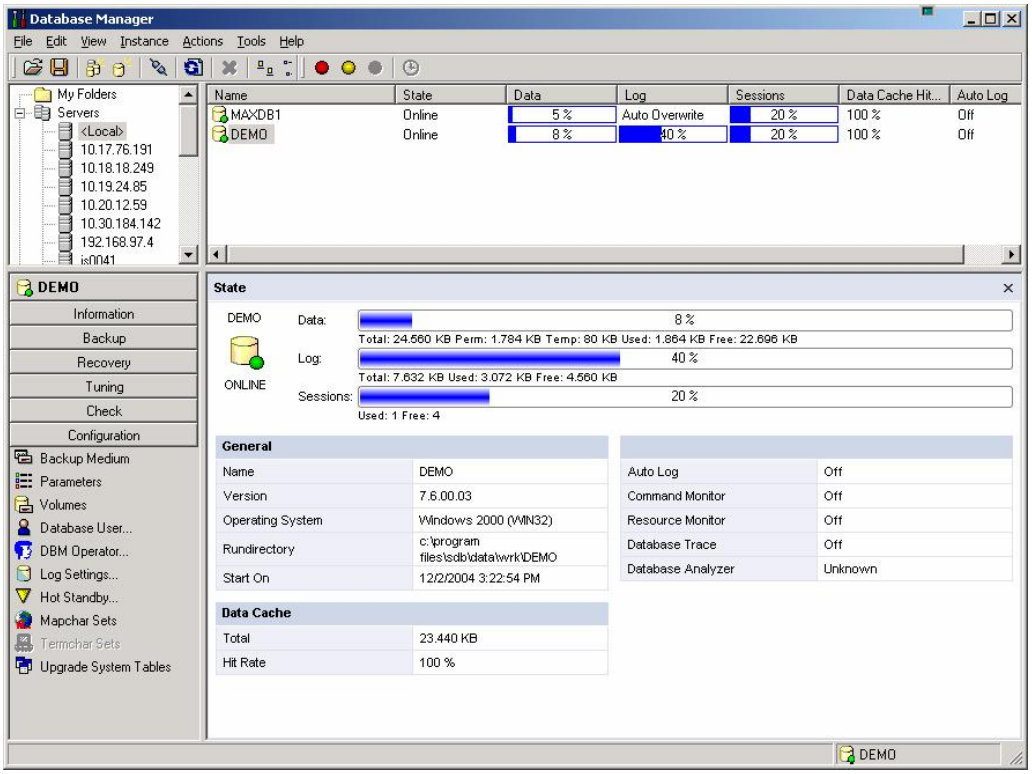
During installation a progress bar and the steps of the installation are displayed.

The loading of the sytem tables is the final step.

When the installation is finished the wizard window shows the database overview.

You can save the details you entered when you install or initialize a database instance in a template. The Installation Wizard proposes a path and name for the template. You can use this template to install and initialize other database instances.

Instance Installation 12/12
Database Manager GUI Overview



Now the instance has been registered and can be administered with the database manager. You can easily assign your whole landscape of MaxDB database instances to the Database Manager by simply registering them with the menu option Instance → Add.

Presented above is the main screen of the Database Manager

On the left side you see all possible actions. They are grouped by command types.

On the right side you see the information selected: in this example the filling level of data and log devspaces and the cache hit rates.

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

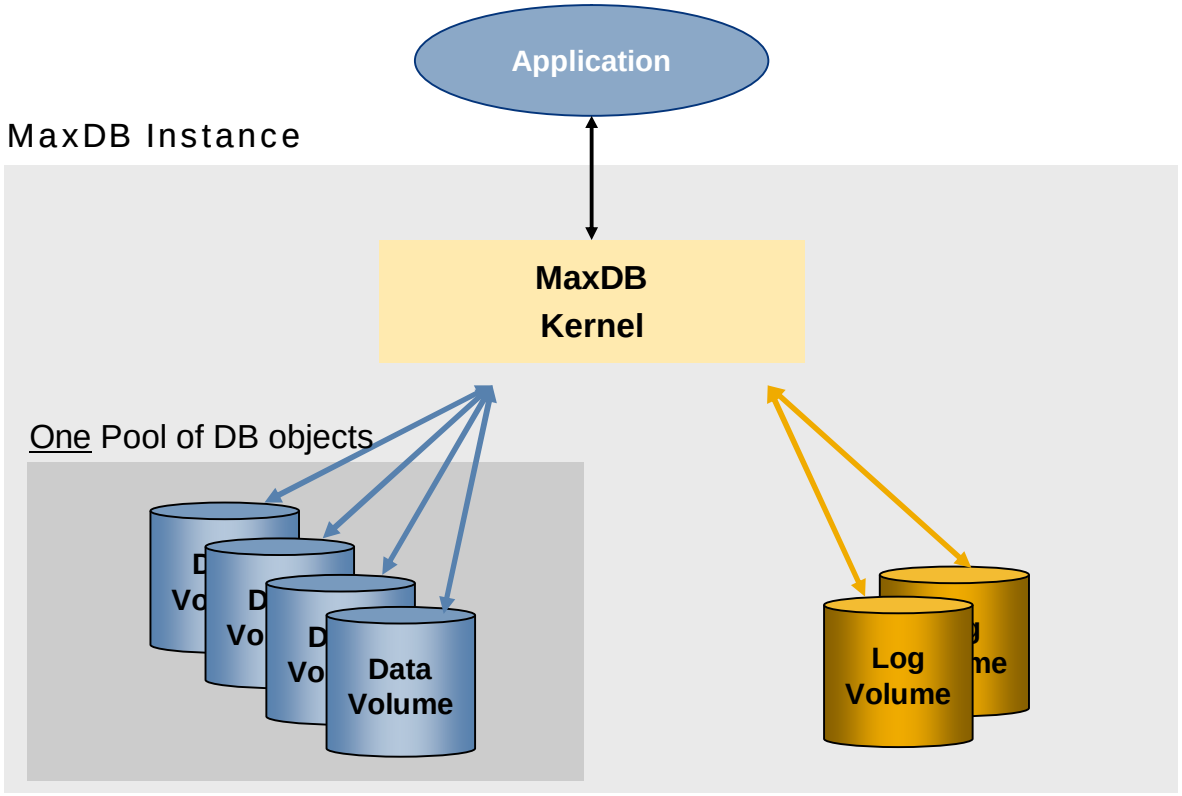
Further Information

Learning Objectives

Knowing the internal task structure of MaxDB

Knowing the most important memory areas used by MaxDB

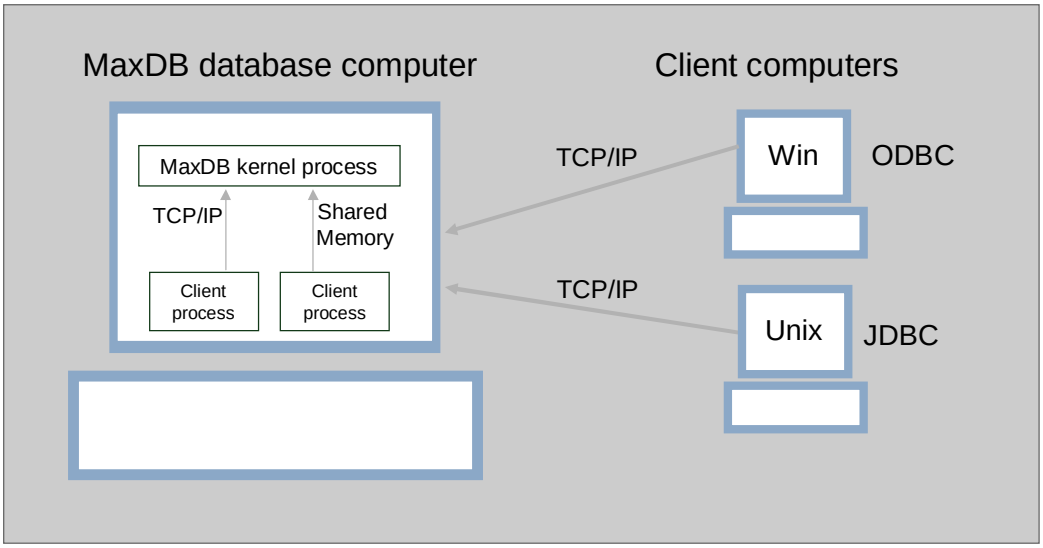
Anatomy of a MaxDB Instance



MaxDB Server & Clients

Clients/applications can run on a local or a remote computer

Separate computers can have different operating systems

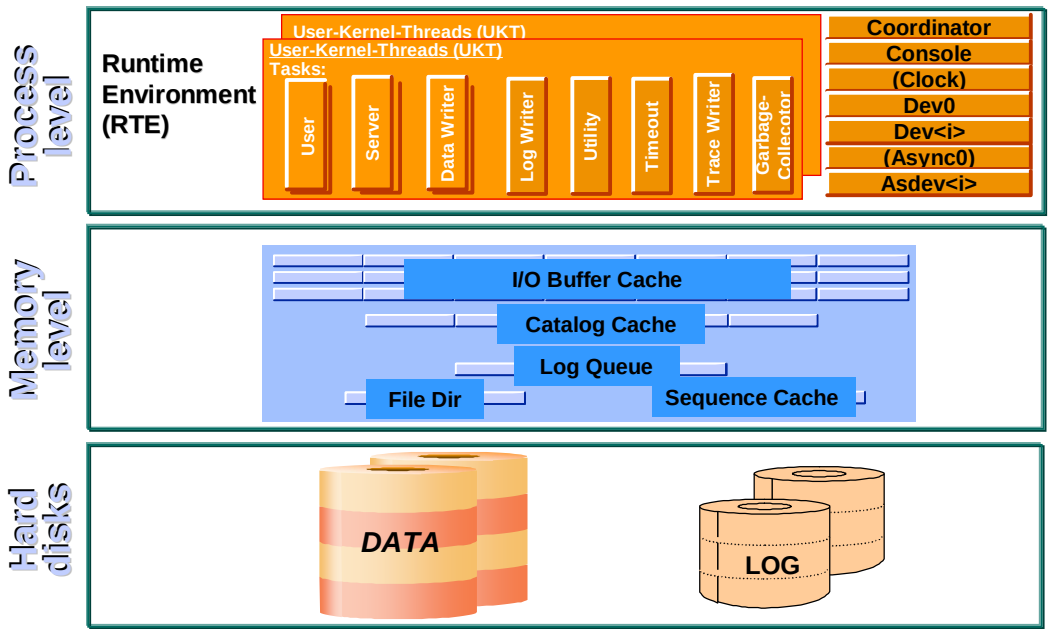


Processes/Threads

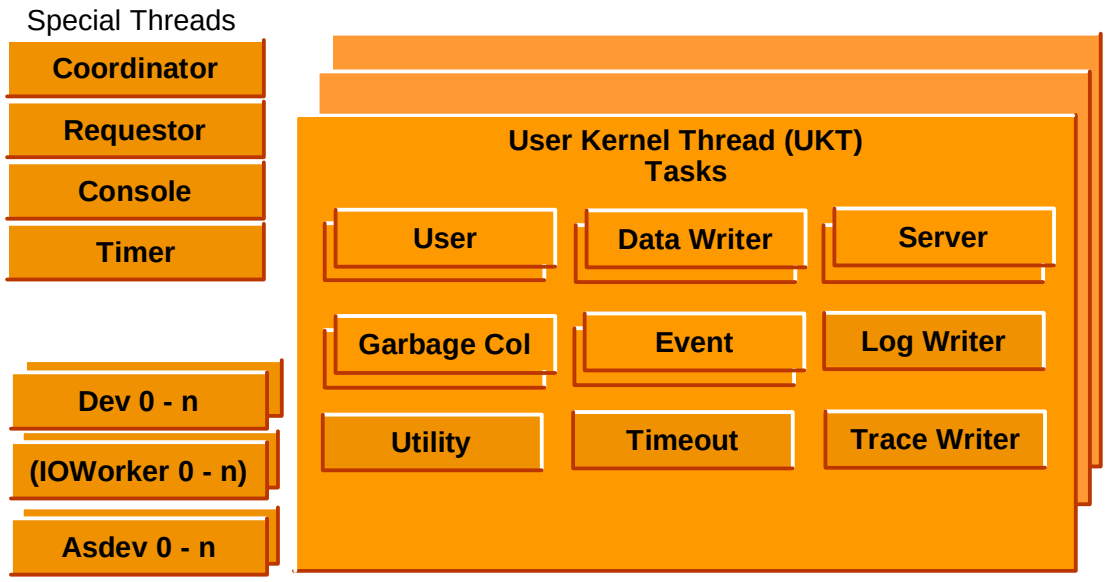
The database kernel runs as one process which is divided in threads.

The following slides show the process/thread structure and the function of the different threads.

Database instance



Process structure



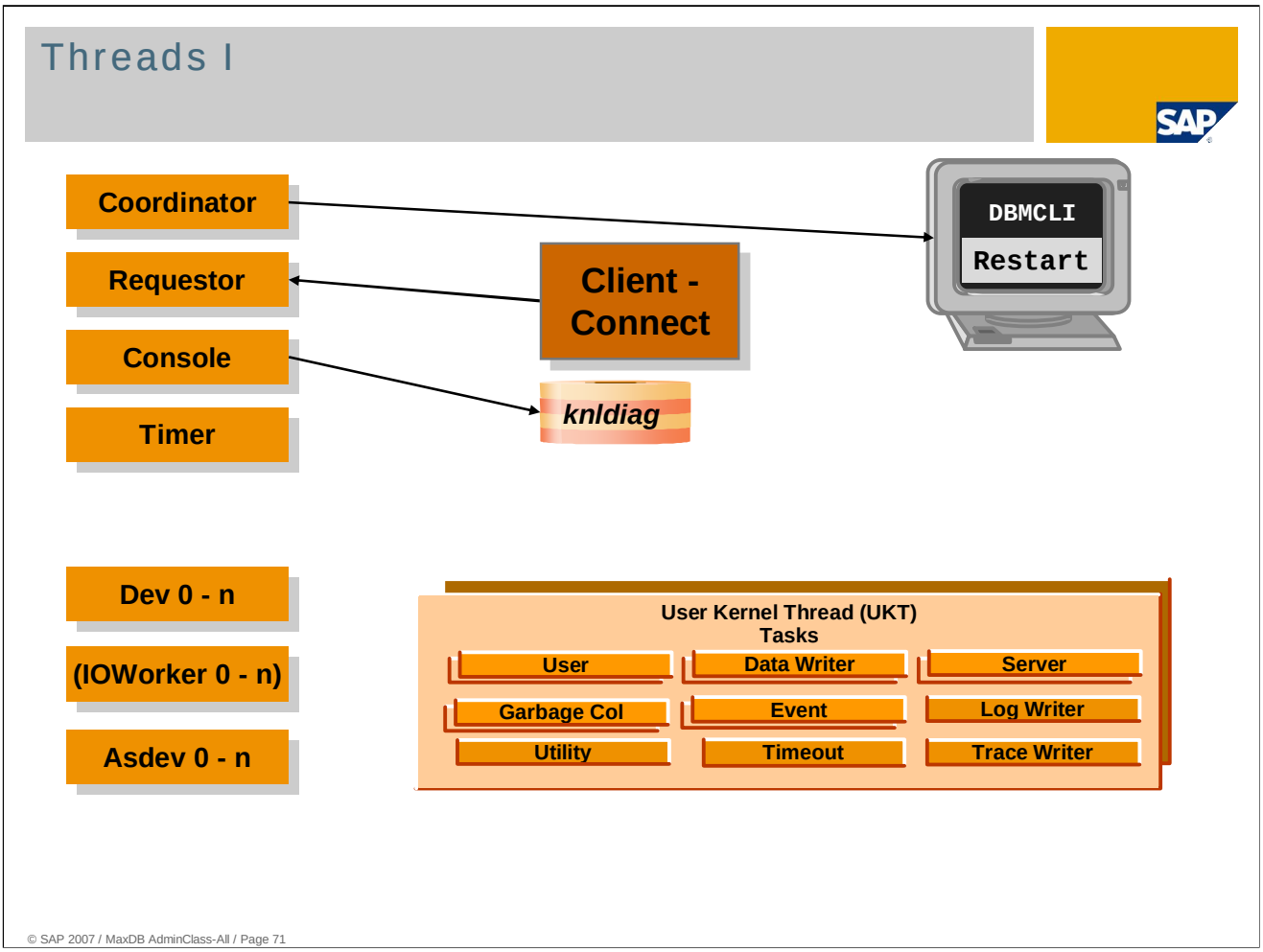
The database kernel runs as one process which is divided into threads.

On UNIX there are two processes – the database kernel forks itself after initialization to be able to clean up after termination (watchdog process).

Inside the operating system several threads can be active in parallel. Different threads perform different functions.

User Kernel Threads (UKT) contain several tasks, which perform different functions. This tasking allows a more effective coordination of the functions as the operating system would do with several threads.

The Runtime Environment (RTE) defines the process and user kernel thread structure.



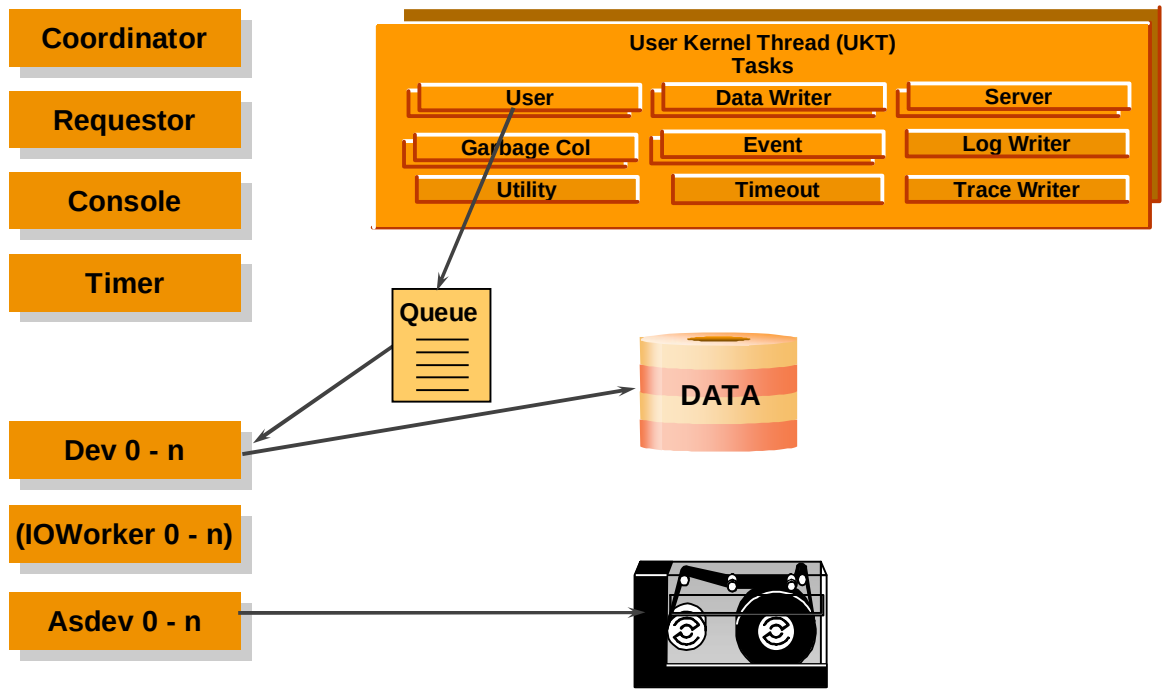
During startup of the RTE, i.e. during start of the database instance into admin state, the Coordinator Thread is created first. It has a special meaning.

- The Coordinator Thread uses database parameters during the start to get to know the memory and process configuration of the database instance. For this reason changed database parameters mostly become active only after the following start of the database instance (restart).
- The Coordinator Thread coordinates the starting of the other threads and controls them during the runtime of the database instance.
- In case of errors, the Coordinator Thread is able to stop other database threads.

The Requestor Thread receives connect requests from client tasks to the database. The connect request is attached to a task in a User Kernel Thread.

The Console Thread allows to monitor database kernel internal states while it is connected to the x_cons program.

Threads II



Several Dev Threads (Dev is short form of devspace, formerly used where now the word volume is used) are responsible for handling the read and write calls (sent by the corresponding tasks) to and from data and log volumes. MaxDB supports asynchronous I/O calls. With Windows NT/2000 the asynchronous I/O of the operating system is used.

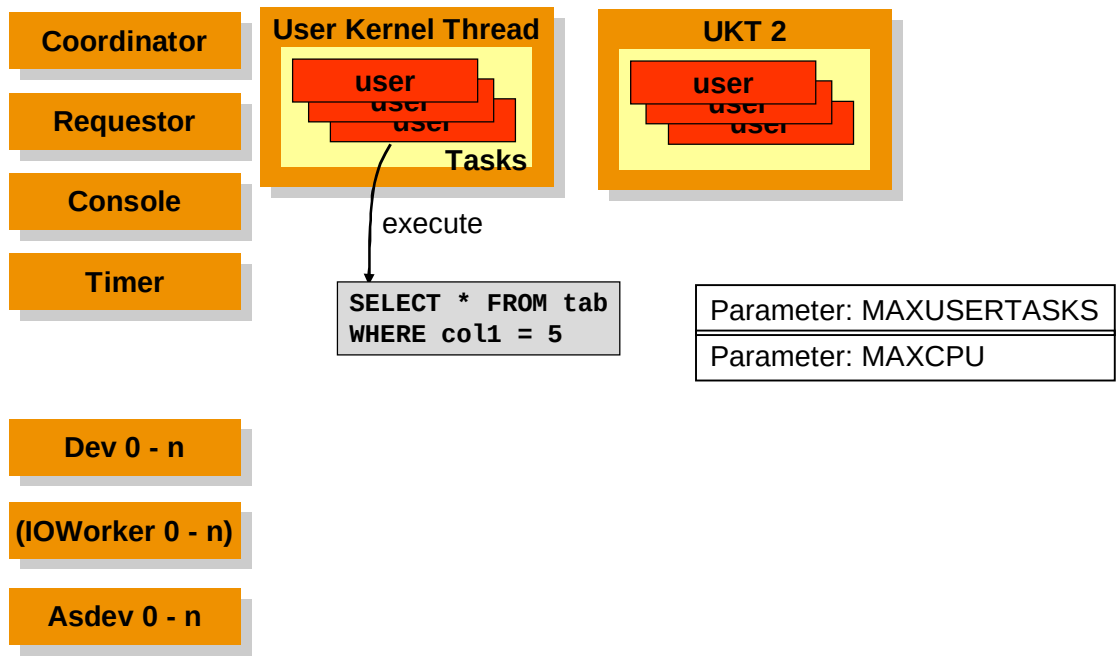
The number of Dev Threads mainly depends on the number of volumes in the database installation. Usually for every log and data volume there are two and for writing the database trace there is one Dev Thread activated. If the asynchronous I/O of the operating system is used, only one Dev Thread per volume is started.

Thread dev0 has a special function:
dev0 is responsible for coordinating and controlling the Dev Threads. If, for example, a mirrored log volume fails during working (bad volume), dev0 will stop the corresponding Dev Threads. If a new data volume is added to the database instance during database activities, dev0 is responsible for creating new Dev Threads.

If the asynchronous I/O calls of the operating system are used, the User Kernel Threads directly send their calls to the OS. The answer of the OS is put into a queue which is read by the I/O Worker Threads.

For creating backups temporary threads are activated for reading and writing the data. These threads are called asdev<i>. Their number depends on the number of data volumes and backup devices.

User Kernel Threads und Tasks

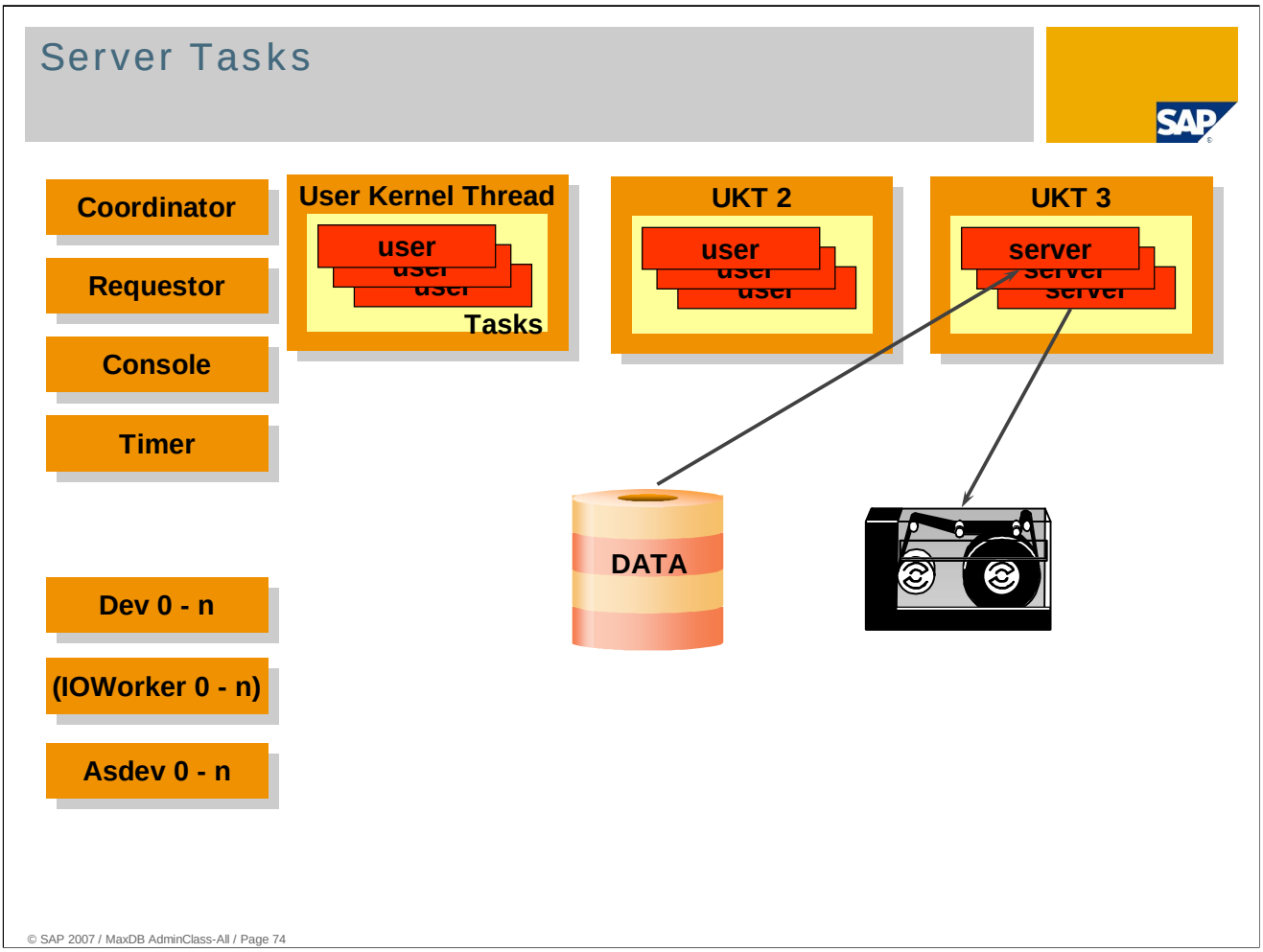


Each user respectively each application process/thread is assigned to exactly one User Task when handling the connect request. The maximum number of User Tasks available depends on the database parameter MAXUSERTASKS. This parameter therefore sets the upper limit for the number of parallel sessions for the database.
The parameter _MAXTASK_STACK defines the stack usage of the User Tasks.

The database parameter MAXCPU defines, how many User Kernel Threads are created for the User Tasks. The other tasks as well as the global threads do not need much CPU time. So the parameter MAXCPU defines how many processors the database may permanently use in parallel.

With version 7.5 each user task can be combined with one thread (database parameter USE_COROUTINES = NO). Several threads are bundled in one UKT. Therefore the possibility to restrict the number of CPUs used in parallel (parameter MAXCPU) remains.

Furthermore there is the chance for user tasks to switch to another UKT if the UKT used so far is overloaded with work.

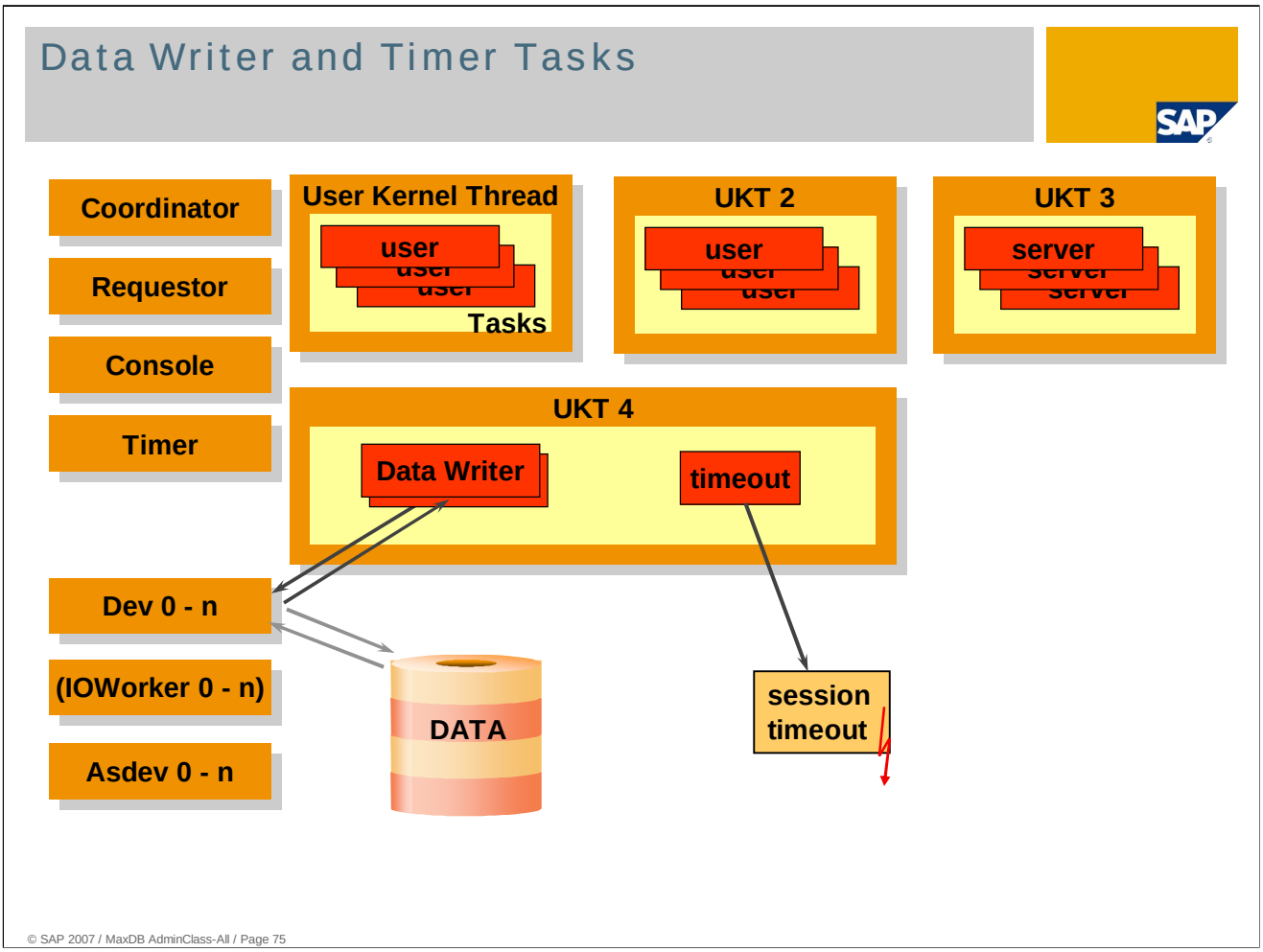


Server Tasks mainly are used for backups. Some Server Tasks read from the data volumes, others write onto the backup devices.

During CREATE INDEX commands multiple Server Tasks read the data of the table from disk concurrently.

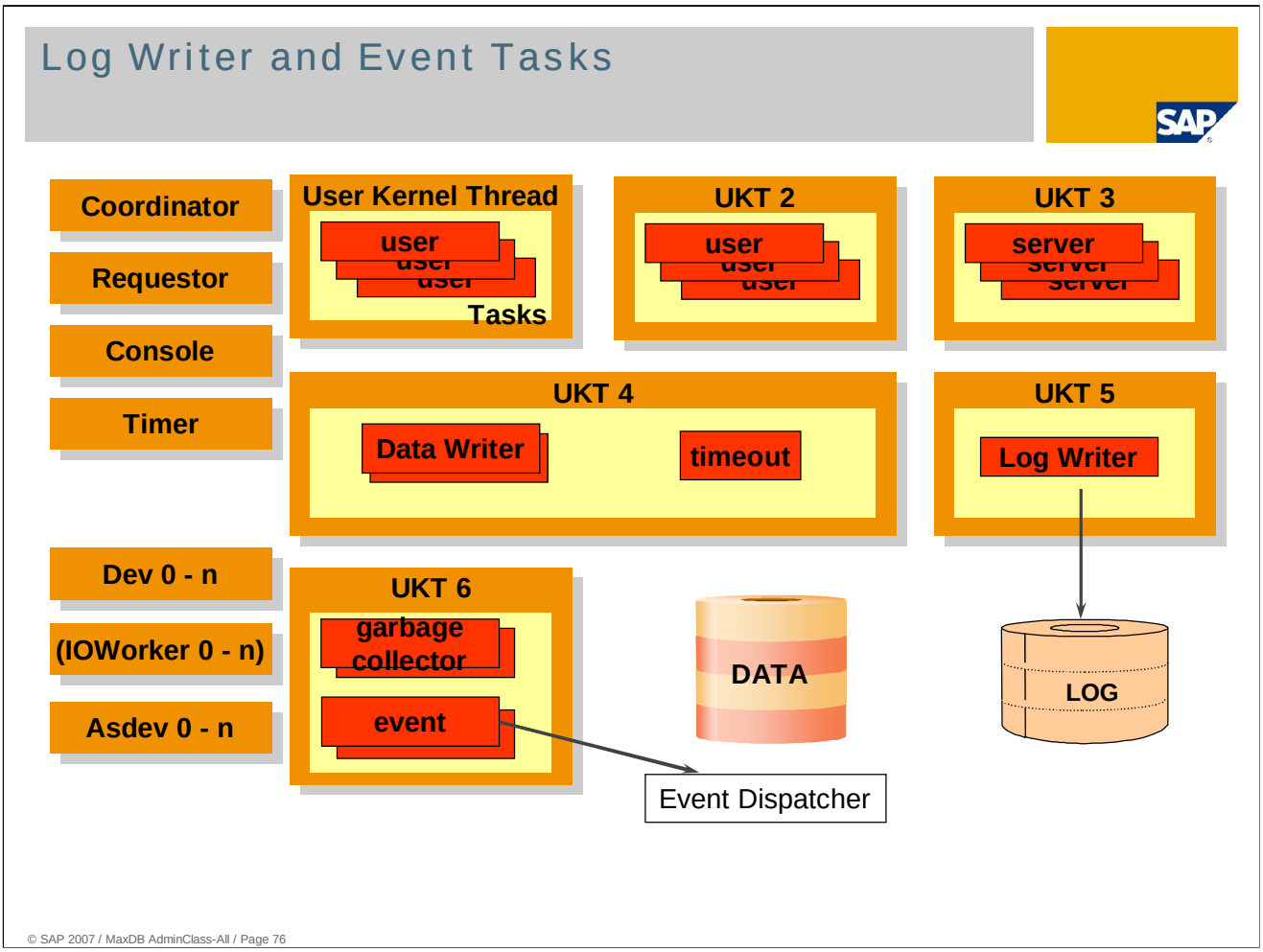
With DROP TABLE commands Server Tasks delete the data of the table asynchronously. The user does not have to wait for all data being deleted.

The database system calculates the number of Server Tasks automatically during configuration of the database instance, taking the number of data volumes and planned backup media into account.



Data Writer Tasks are responsible for writing data from the I/O buffer cache onto data volumes. They become active when a SAVEPOINT has to be done.
The number of Data Writer Tasks is calculated automatically by the database system. It depends mainly on the size of the I/O buffer cache and the number of data volumes.

The Timer Task is used for the handling of timeout situations (for example: Session Timeout, Lock Request Timeout).

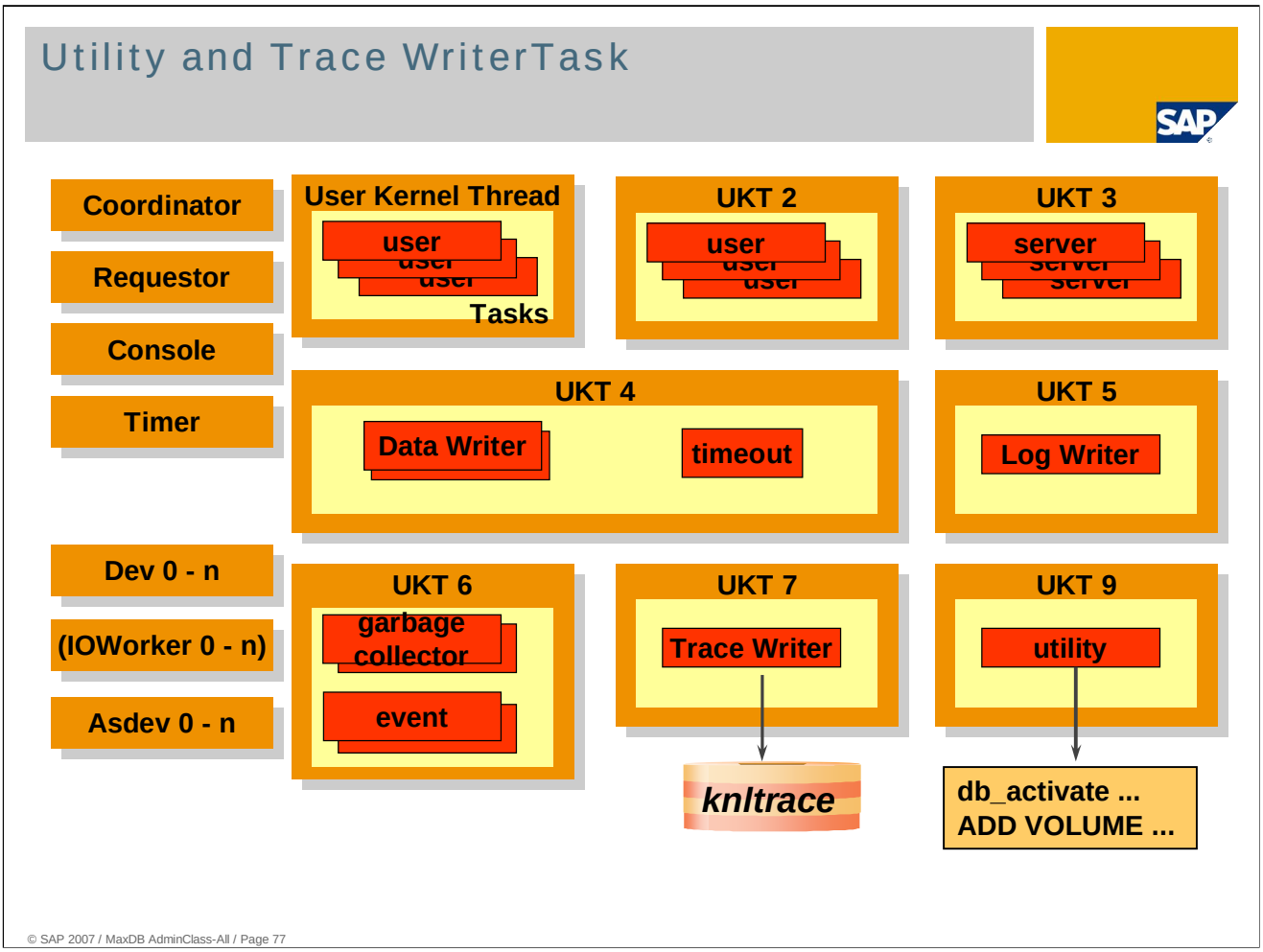


The Log Writer Task is responsible for writing after images onto the log volumes.

Event Tasks allow to wait for events in the database (filling of database above xy percent and so on). For example the Event Dispatcher connects to the Event Task to be informed if an event occurs. The Event Dispatches starts so called Event Handler if an event occurs. This allows f.i. to add a data volume automatically if necessary.

The messages are written to the file knldiag.evt as well.

The Garbage Collectors are only used if MaxDB runs in special SAP applications.

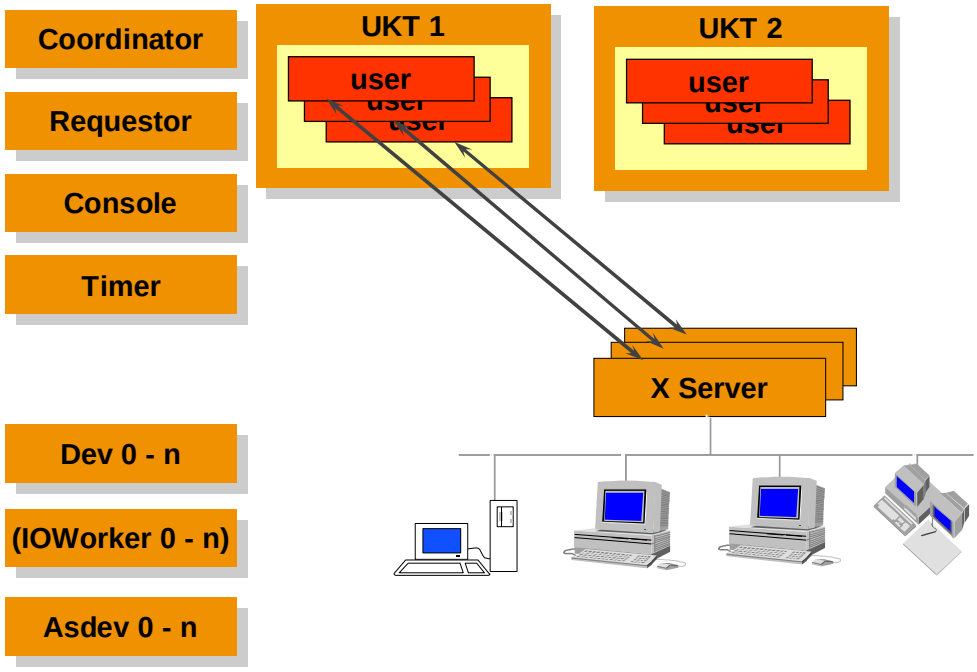


MaxDB allows to write a database trace. If this writing is enabled, the Trace Writer Task becomes active.
Active tasks write trace into a buffer. The Trace Writer Task then writes these buffers into the file knltrace if requested.

The Utility Task is the only one for managing the database instance.
As there is only one Utility Task per database instance, managing duties can not be done in parallel. Conflicts are therefore prevented.
The automatic log backup can be done in parallel to some other managing duties because it does not need the Utility Task any more once started.

As of version 7.5 administrative tasks can also be performed by User Tasks. The database kernel prohibits the concurrent execution of conflicting activities. The Database Manager still uses the Utility Task.

Remote SQL Server



When you start the X Server on UNIX to enable remote database accesses, an additional process is created. It is named vserver. For every application process with a remote connection a new vserver process is created. The ‘old’ process works for the new application, the ‘new’ vserver process waits for the next remote connection.

On Windows platforms an additional thread is started for the application connection.

On Windows platforms the X Server is a service.

For local communication shared memory is used.

Processes with ps

When the database instance is started there are always two processes in UNIX.

The process with process ID 1 as its father is the listener who starts the second process.

The second process is the actual worker process

$ ps -afe | grep kernel

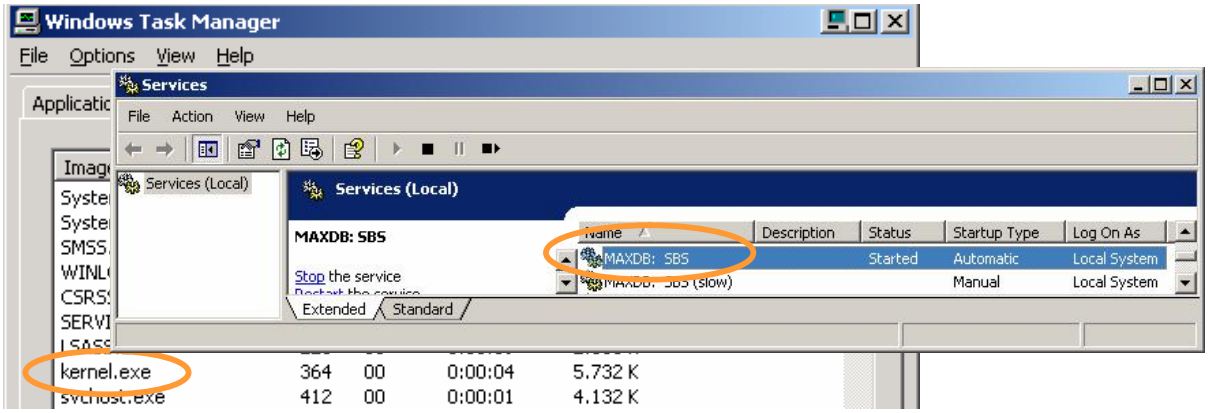
```
sdb  445002      1  0.0   Oct 28 ??     0:01.47 /sapdb/<db_name>/db/pgm/kernel <db_name>
sdb  446847 445002  0.0   Oct 28 ??     0:01.37 /sapdb/<db_name>/db/pgm/kernel <db_name>
```


In Linux there is one clone process per thread.

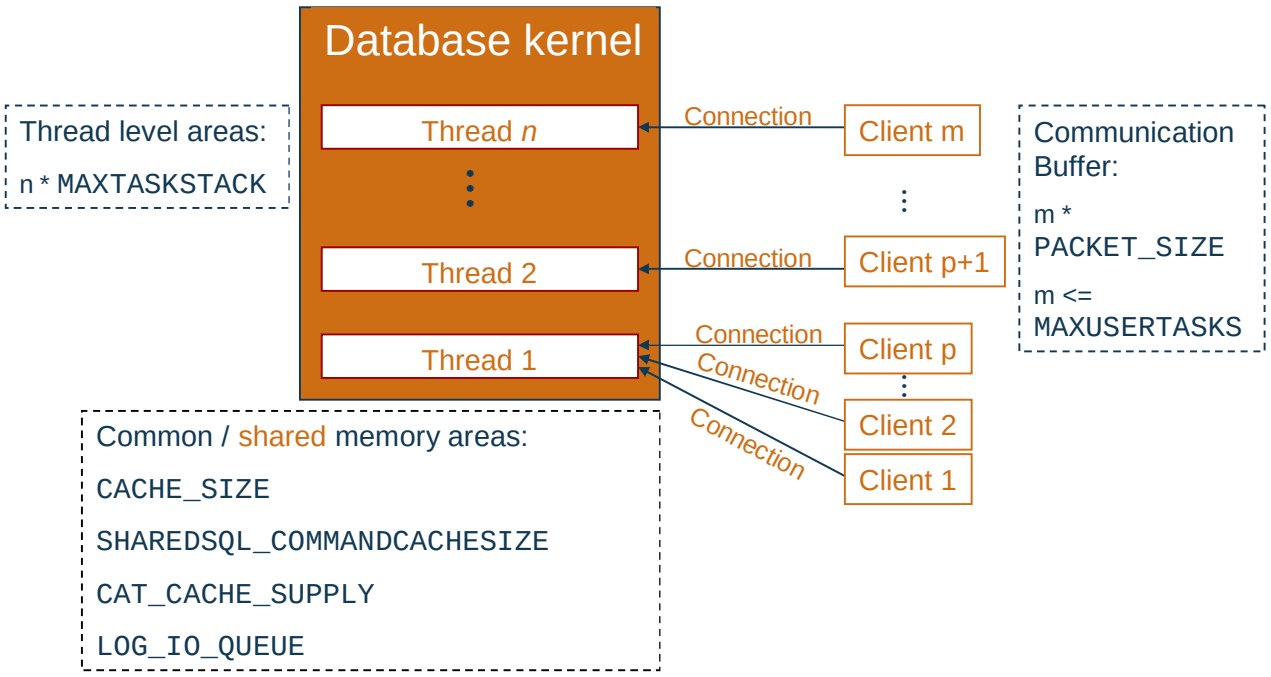
Processes on Windows

When the database instance is started there is always one process on Windows named kernel.exe

The database kernel runs as a service



Memory Footprint



The following parameters specify also some memory requirements:

TRACE_PAGES_TI

Size (in pages) of memory used for kernel trace for each task of type Timer.

TRACE_PAGES_GC

Size (in pages) of memory used for kernel trace for each task of type garbage collector.

TRACE_PAGES_BUP

Size (in pages) of memory used for kernel trace for each task of type backup.

TRACE_PAGES_LW

Size (in pages) of memory used for kernel trace for each task of type log writer.

TRACE_PAGES_PG

Size (in pages) of memory used for kernel trace for each task of type data writer.

TRACE_PAGES_US

Size (in pages) of memory used for kernel trace for each task of type user.

TRACE_PAGES_UT

Size (in pages) of memory used for kernel trace for each task of type utility.

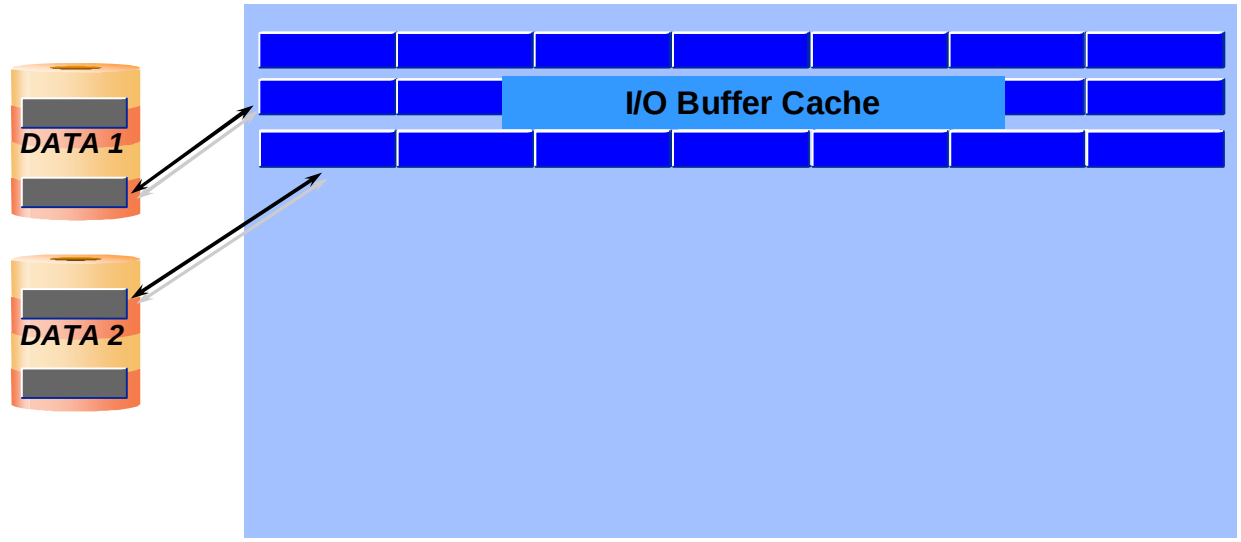
TRACE_PAGES_SV

Size (in pages) of memory used for kernel trace for each task of type server.

TRACE_PAGES_EV

Size (in pages) of memory used for kernel trace for each task of type event.

I/O Buffer Cache



To avoid time-critical disk I/O read and write accesses to volume buffers are used. The corresponding main memory structures are named caches. Their sizes are defined by the database administrator.

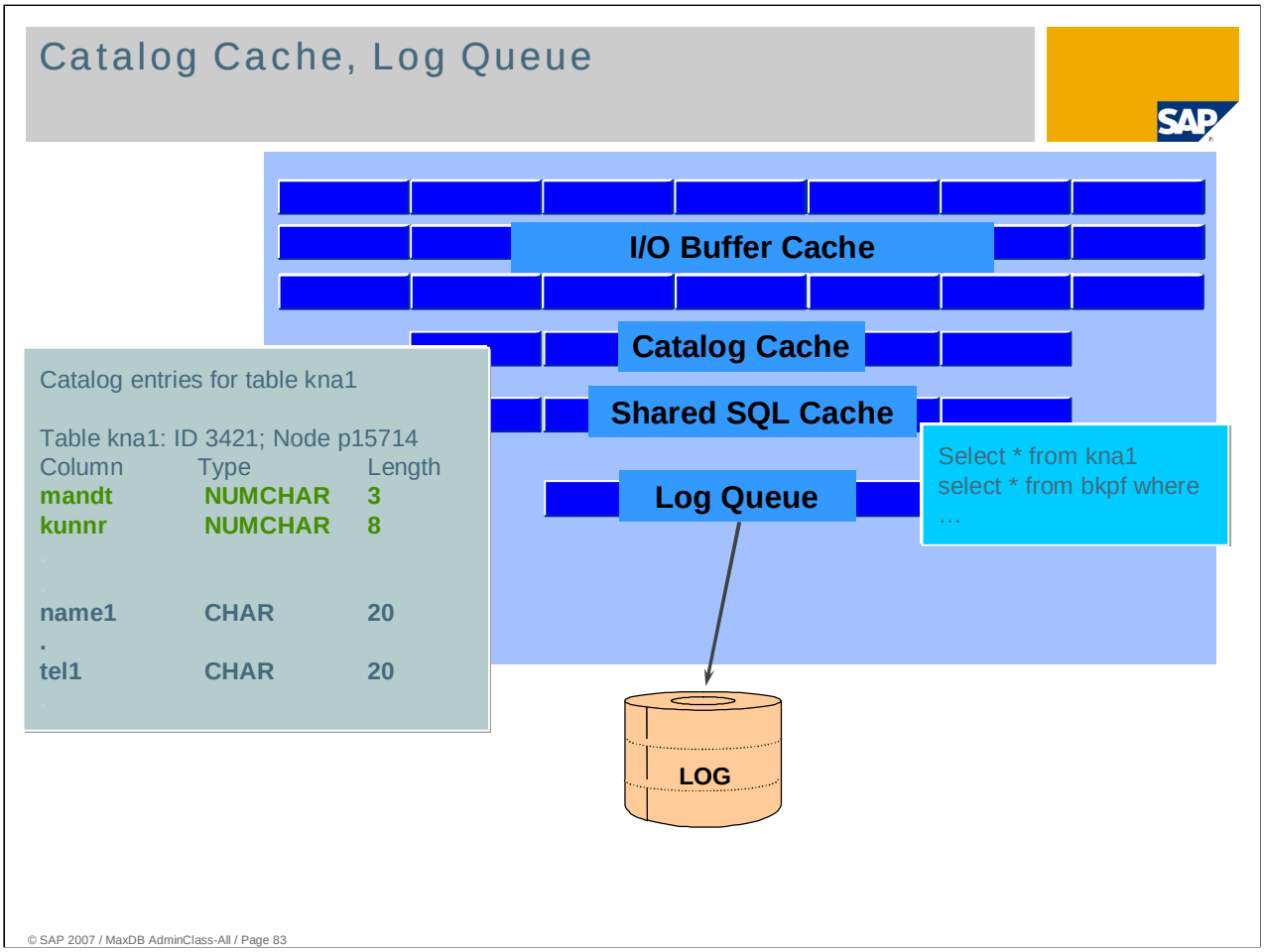
The I/O Buffer Cache (Data Cache) contains those pages of the data volumes, which where read or written recently (that includes data and index pages). All active tasks use it concurrently.
I/O buffer cache is static memory; it is allocated in full size when the MaxDB is started. The size is configured by MaxDB configuration parameter CACHE_SIZE, in page units (1 paqe = 8 KB).

The hit rate, i.e. the number of successful accesses compared to the number of all accesses in the I/O Buffer Cache is decisive for the performance of the database instance. It should exceed 98%. An access is called successful if the wanted data is found in the cache without need for I/O.

As well as data pages the I/O Buffer Cache contains Converter pages. Converter pages are stored on data volumes like data pages. In these pages the link between the logical page numbers and their physical position in the data volumes is stored.

The number of converter pages is calculated automatically. This number may increase when increasing the amount of data stored in the database. During deletes converter pages are freed.

All converter pages remain in cache. The size of the converter depends on the size of the database. Each converter page contains 1861 entries for data pages.



The Catalog Cache contains user specific and global catalog objects. Data displaced from the catalog cache is put into the I/O Buffer Cache. The hit rate of the Catalog Cache should exceed 85%.
The size of the Catalog Cache is specified using the kernel parameter CAT_CACHE_SUPPLY.

All DML statements and their execution plans are stored in the Shared SQL Cache. The data of all users is stored in the same Shared SQL Cache – each executed statement is inserted just once. If Shared SQL is switched off (parameter SHAREDSQL), the statements are stored for each user in the Catalog Cache.

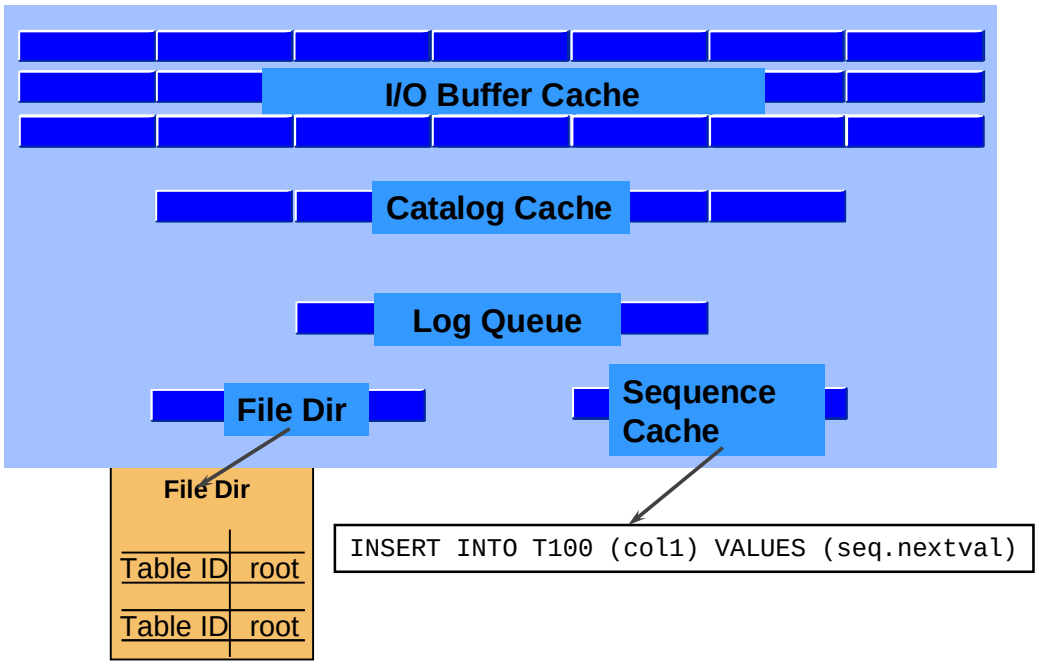
The Shared SQL Cache contains also the text of the executed statements – so you can determine at any time the currently running statements. Furthermore runtime information of these statements (like accessed pages, runtime, qualified rows, …) are stored in the Shared SQL Cache which can be used for performance analysis.

The Log Queue is used as a buffer to store data changed by the application (after images of records) that needs to be transferred to log volume before the application can successfully perform a commit.

The log queue is organized in pages. Several applications can write their changed data (log entries) to the same log page. Whenever a log page is full, it will be written to the log volume by the Log Writer task.

While the Log Writer task writes a log page to disk, other pages in the log queue can be filled. If several pages are used, the Log Writer will write up to eight pages using just one physical I/O.

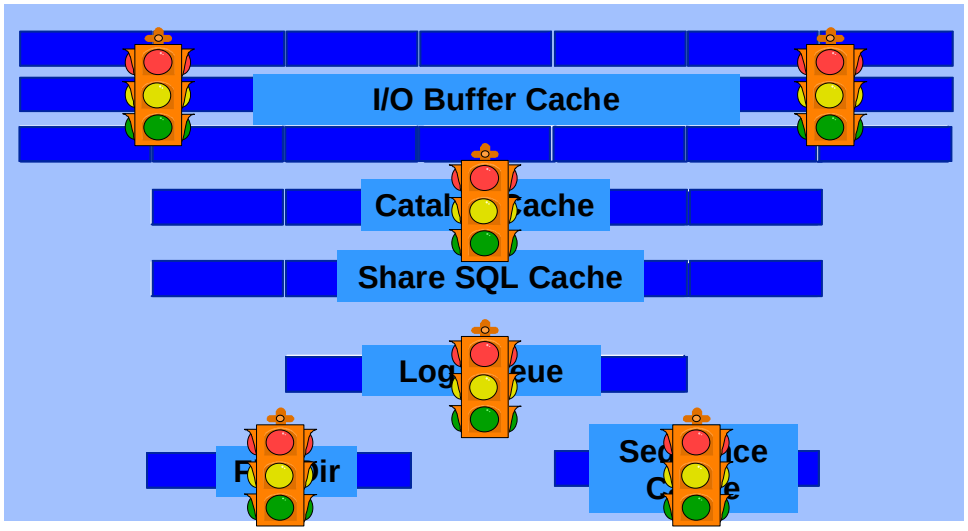
File Directory, Sequence Cache



The File Directory is for internal management of the database instance. In it at least the page numbers of the root pages for all data trees are managed.

The Sequence Cache contains current values for sequences.

Cache Access



Critical sections in memory are protected by synchronization mechanisms (Regions)

Accesses to caches can be synchronized over one or more regions. Depending on its size, the data cache is comprised of 8 to 64 segments of the same size, each of which is protected by exactly one region.

If a task or thread accesses a critical section, the region locks this session for all other tasks or processes.

Other main memory structures are also managed via synchronization mechanisms provided by the database.

Reader-writer locks are used to synchronize the shared SQL cache. Reader-writer locks are used in version 7.5 and up. In contrast to regions, reader-writer locks make it possible to distinguish between shared and exclusive locks.

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

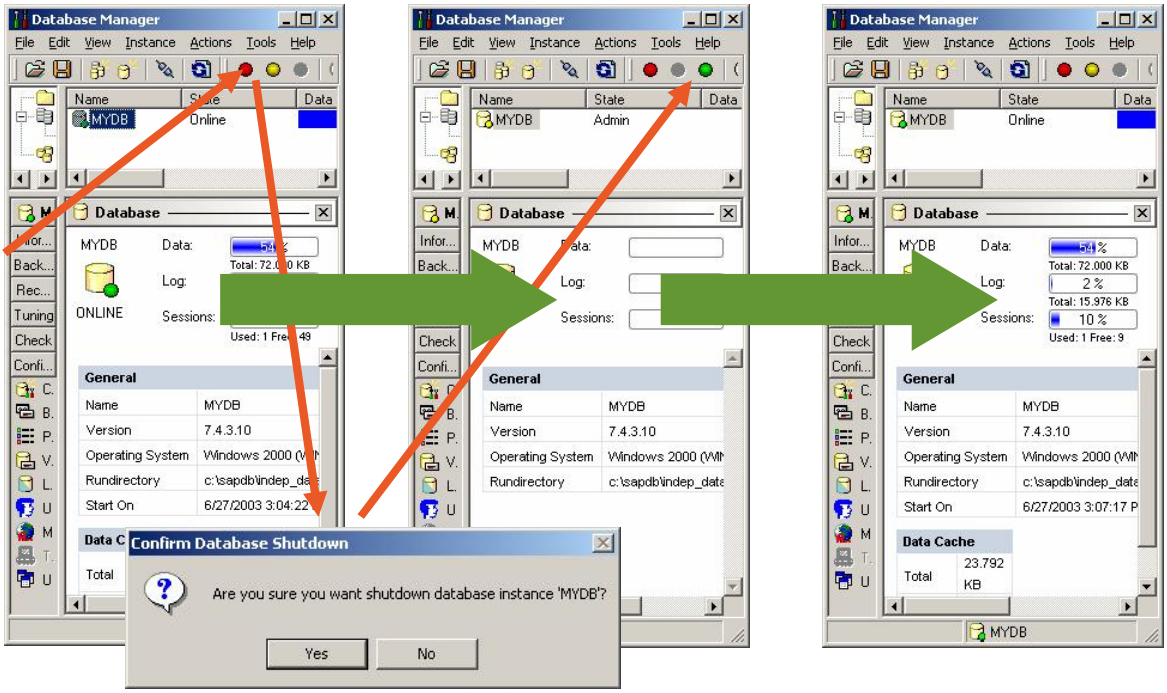
Further Information

Learning Objectives

Know how to

- start and stop the database instance using DBMGUI and DBMCLI
- create backups
- perform a recovery with/without initialization
- check the database structure
- add volumes
- alter the log settings
- create and use snapshots

Startup and Shutdown 1/2



The database always shuts down (offline) when it is switched from online to admin mode, you can either switch to offline and then to admin or switch directly to admin. There is no difference!

Startup and Shutdown 2/2

Shutdown:

dbmcli –d <db_name> -u <dbm_user>,<password> db_offline

Restart:

dbmcli –d <db_name> -u <dbm_user>,<password> db_online

If you want to perform some administrative tasks and no user should be connected to the database instance, you can put the database into state ADMIN:

shell> dbmcli –d <db_name> -u <dbm_user>,<password> db_admin

Backup Basics 1/2

Backups are performed by the database process.

Online backups of the volumes made with operating system tools (e.g. dd, copy) can not be used.

There are three backup types :

- Complete Data Backup: All data pages, configuration and parameter information.
- Incremental Data Backup: Backs up all data pages changed since the last complete data backup.
- Log Backup: All pages of log (in units of log segments) that have not already been backed up.
- Additionally ‚Set AutoLog On‘ automatically backs up the log as soon as a log segment is completed.

Backup Basics 2/2

You should keep several generations of backups. For example tapes containing data or log backups should not be directly overwritten with the next backup.

When the log writing is not disabled:

- The automatic log backup (autosave log mechanism) should always be active.
- If automatic log backup is not active, you must regularly check whether there is enough storage space available in the log area.
- You should regularly archive the version files written during a log backup to a medium of your choice.

Due to the database converter concept, MaxDB does not support backup / restore of single tables/schemas. However, single tables/schemas can be extracted/loaded using the MaxDB loader.

If you retain, say, the last four backup generations, it may be possible to use an older backup if a media failure occurs.

Note that in case of a restore a more up-to-date data backup means that fewer log entries need to be redone. Therefore, perform data backups as often as possible.

Perform a complete data backup on each day of production.

If you cannot or do not want to perform a data backup every day, you should at least perform an incremental data backup on each day of production.

While a complete data backup is active, incremental data and log backups cannot be started.

You can perform data backups in parallel to reduce the time required for the backups.

Complete and incremental data backups are also possible while the automatic log backup is active.

If new log entries cannot be written to the log area because there is not enough storage space, the database stops. If necessary, back up the log area immediately by starting an interactive log backup.

External Backup Tools

MaxDB supports three kinds of external backup tools:

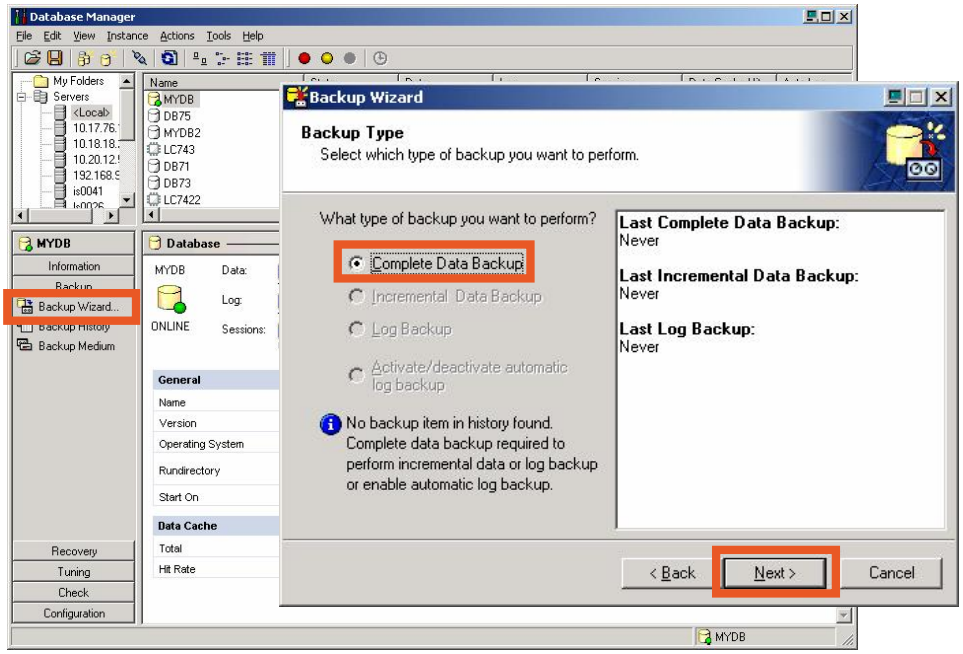
- Tivoli Storage Manager
- Networker
- Tools which support the Interface BackInt for Oracle or Backint for MaxDB

To use one these tools you have to choose the device type Pipe for your backup medium. Furthermore you have to specify, which backup tool should be used: ADSM, NSR or BACK.

For Windows NT media location must be specified as '\\.\<PipeName>' where <Pipename> stands for any name. On a UNIX/Linux platform the location can be any file name of a non existing file.

For details about using external backup tools please see the online documentation.

Complete Backup 1/3



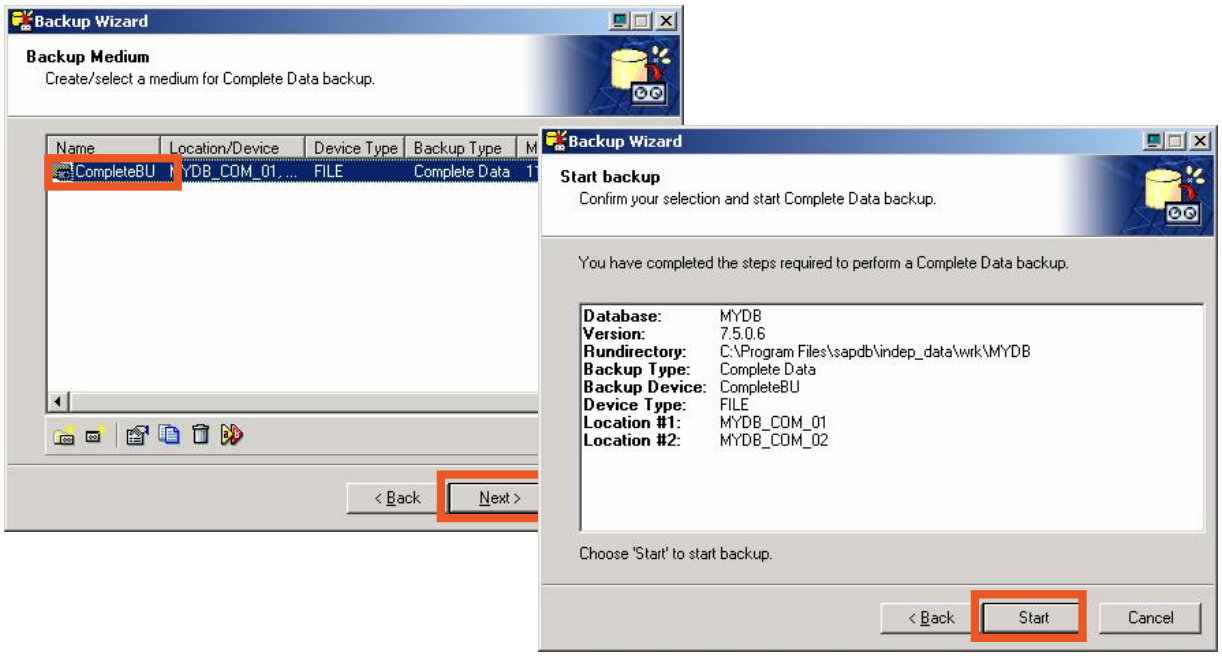
A complete backup saves all occupied pages of the data volume. In addition, the MaxDB parameter file is written to the backup.

The complete backup as well as the incremental backups are always consistent on the level of transactions since the before images of running transactions are stored in the data area; i.e. they are included in the backup.

Each backup gets a label reflecting the sequence of the backups. This label is used by the administrative tools to distinguish the backups. A mapping from the logical backup medium name to the backup label can be found in the file dbm.mdf in the Rundirectory of the database instance.

Each backup is logged in the file dbm.knl in the Rundirectory.

Complete Backup 2/3



For a complete or incremental backup you can choose one of the three device types ‘file’, ‘tape’ or ‘pipe’.

After you have selected a backup medium the backup process has all the information needed for the backup. The wizard responds with a summary and the ‚start‘ request to crosscheck the backup action before starting.

medium_put syntax for dbmcli:

medium_put <media_name> <location> <FILE/TAPE/PIPE> <backup_type> [<size> <blocksize> <owerwrite> <autoloader> <os_command>]

Defining a File as Overwritable Medium for a Complete Data Backup (named completeF)

dbmcli -d <db_name> -u <dbm_user>,<password> medium_put completeF /usr/maxdb/complete FILE DATA 0 8 YES

Defining a Tape Device as Medium for a Complete Data Backup (named completeT)

dbmcli -d <db_name> -u <dbm_user>,<password> medium_put completeT /dev/rft0 TAPE DATA 64000 8 NO

Media Change

If the capacity of the medium is insufficient for the entire backup, you must carry out a media change. In other words, the backup is written to the first medium until this is full, and then a succeeding medium is used. If you foresee this occurring, call up the Database Manager CLI in session mode, because you must not interrupt the session during the backup operation.

Creating a Complete Data Backup

dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL –c backup_start completeT

Commands for Exchanging Media During Backups

(Example of a complete data backup with media exchange)

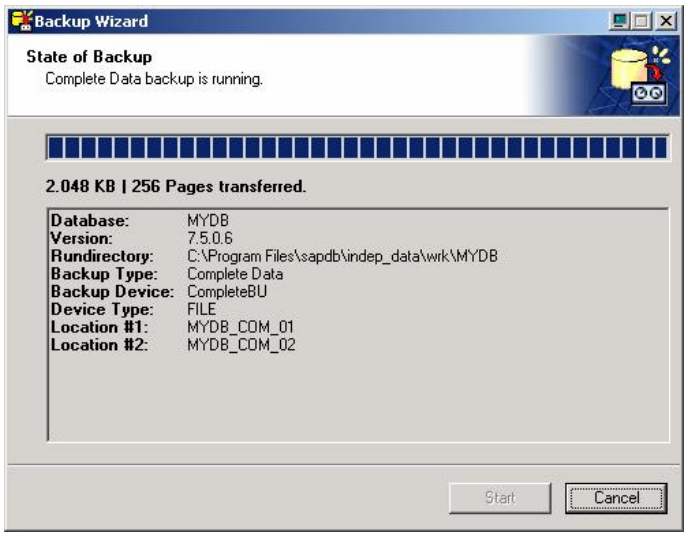
dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL

backup_start completeT

backup_replace completeT

(The backup_replace command has to be executed AFTER the tape has been exchanged, so that the first part of the backup is not overwritten by the next part.)

Complete Backup 3/3



During backup, a progress bar is displayed.

Finally the result and environment information is displayed.

Incremental Backup

An incremental backup is also performed using the backup wizard

Instead of ‚Complete Data Backup' ‚Incremental Data Backup' has to be selected

A backup medium bound to this backup type has to be used – if it does not exist, it has to be created

All data pages changed since the last complete data backup are backed up

Prerequisite: a complete backup has been created successfully after the installation

For a complete or incremental backup you can choose one of the three device types ‘file’, ‘tape’ or ‘pipe’.

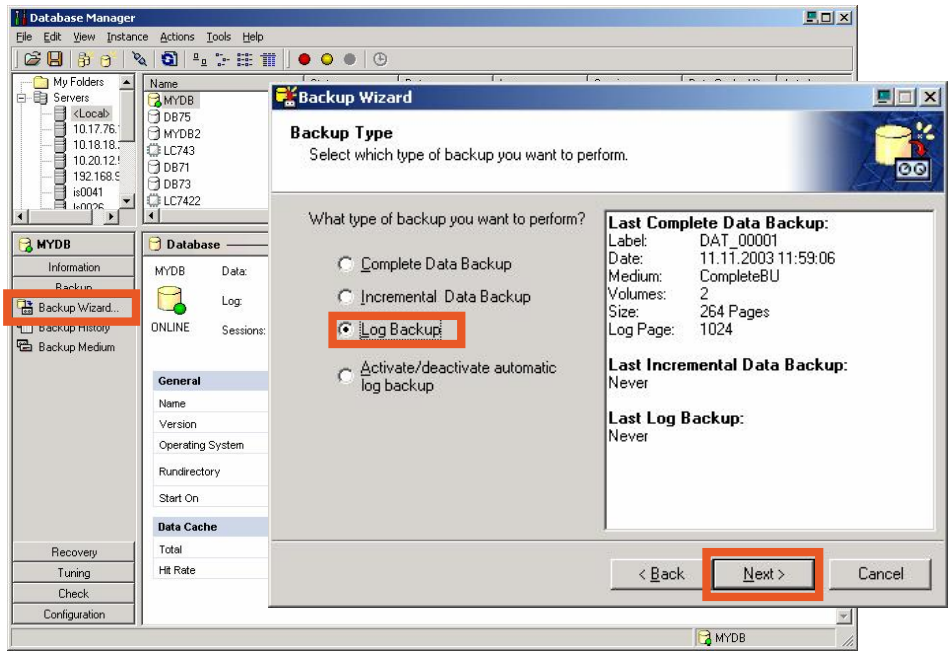
Defining a File as Medium for an Incremental Data Backup (named incrF)

dbmcli -d <db_name> -u <dbm_user>,<password> medium_put incrF /usr/maxdb/incr FILE PAGES

Creating an Incremental Data Backup

dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL -c backup_start incrF

Log Backup 1/2



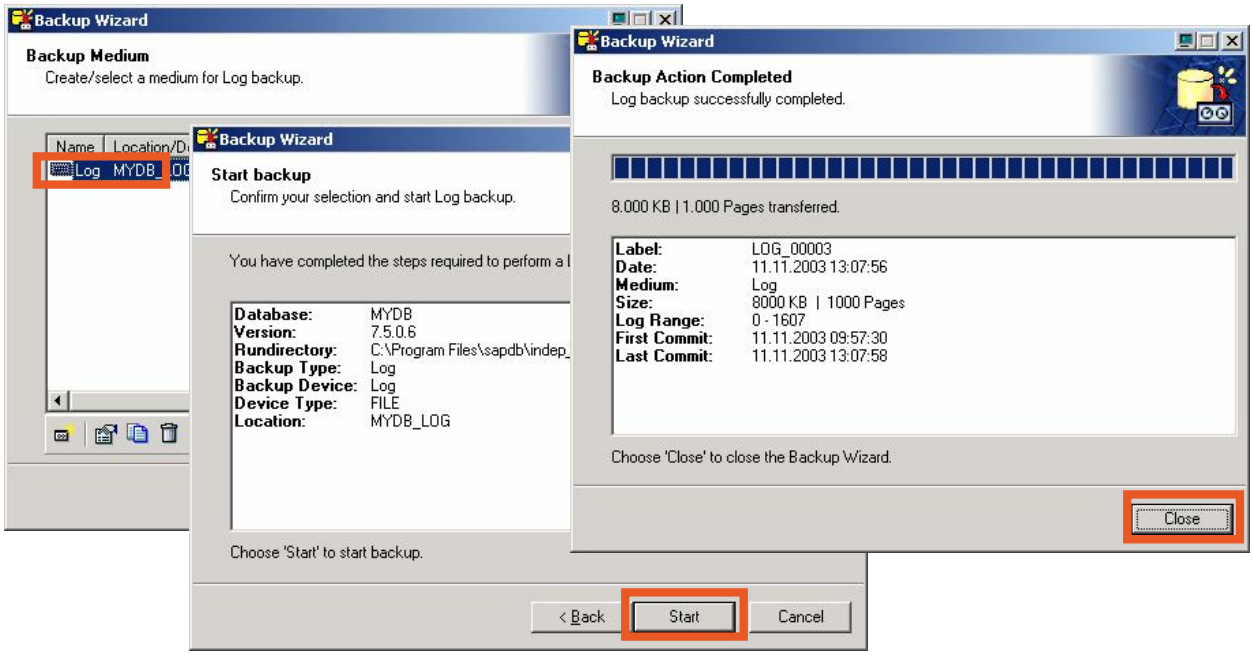
Prerequisite: a complete backup has been created successfully

SAVE LOG saves all occupied log pages from the archive log which have not been saved before.

We recommend to save the log into version files. One version file will be created for each log segment. Version files automatically get a number as extension (e.g. SAVE.LOG.0001, SAVE.LOG.0002, ...)

If the log option ‘auto overwrite’ is enabled, no log backups have to be performed. Trying to create an log backup would result in error -128,Log backup is not possible because the log overwrite mode has been enabled.

Log Backup 2/2



A backup medium bound to the backup type ‘Log Backup’ has to be used – if it does not exist, it has to be created

For a log backup you can choose ‘file’ or ‘pipe’. It is not possible to save log segments directly to tape.

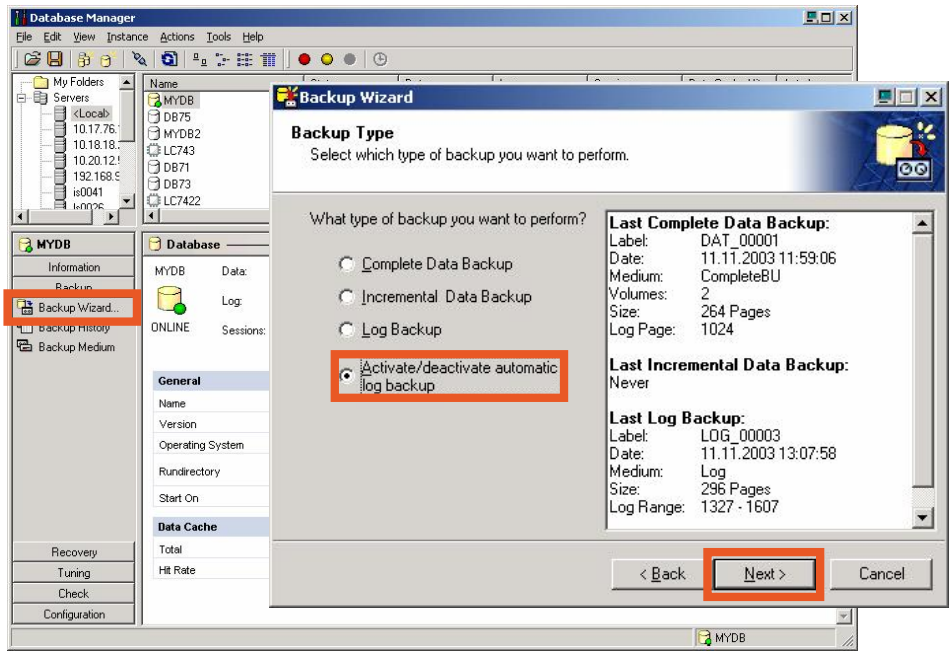
Defining a File as Medium for an Interactive Log Backup (named logsave)

dbmcli -d <db_name> -u <dbm_user>,<password> medium_put logsave /usr/maxdb/log FILE LOG

Creating an Interactive Log Backup

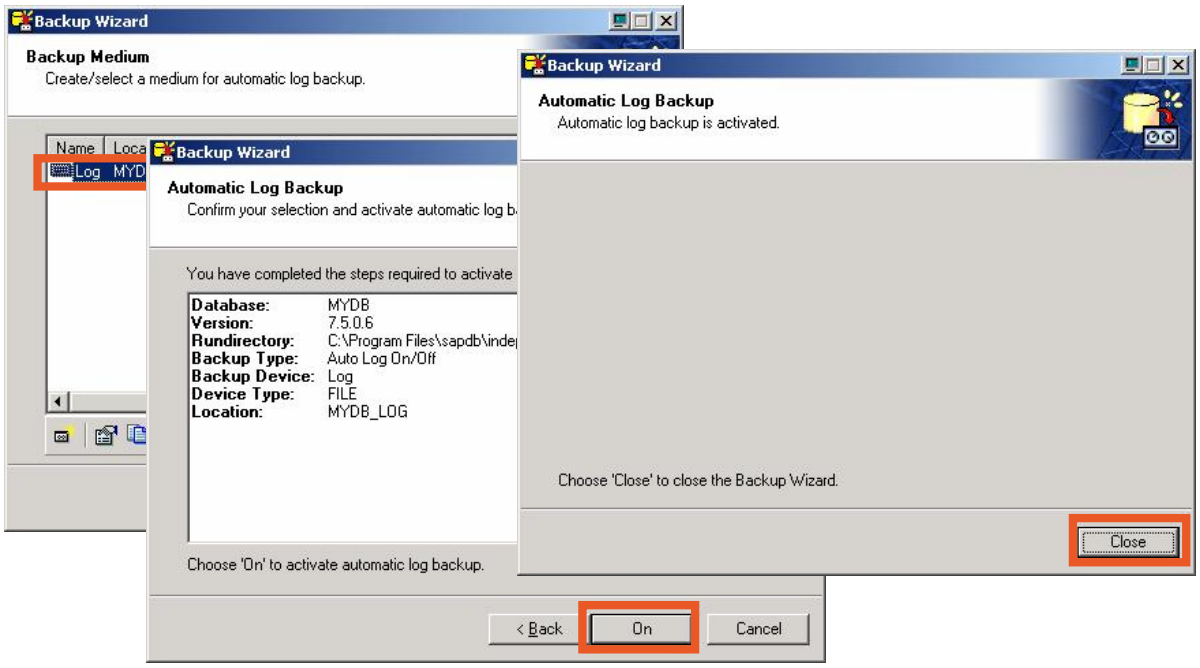
dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL -c backup_start logsave

Automatic Log Backup 1/2



When the Autosave Log mechanism is activated, log segments are automatically backed up as soon as they are completed.

Automatic Log Backup 2/2



A backup medium bound to the backup type ‘Log Backup’ has to be used – if it does not exist, it has to be created

Defining a File as Medium for an Automatic Log Backup (named autosave)

dbmcli -d <db_name> -u <dbm_user>,<password> medium_put autosave /usr/maxdb/auto FILE LOG

Activating the Automatic Log Backup with Medium autosave

dbmcli -d <db_name> -u <dbm_user>,<password> autolog_on autosave

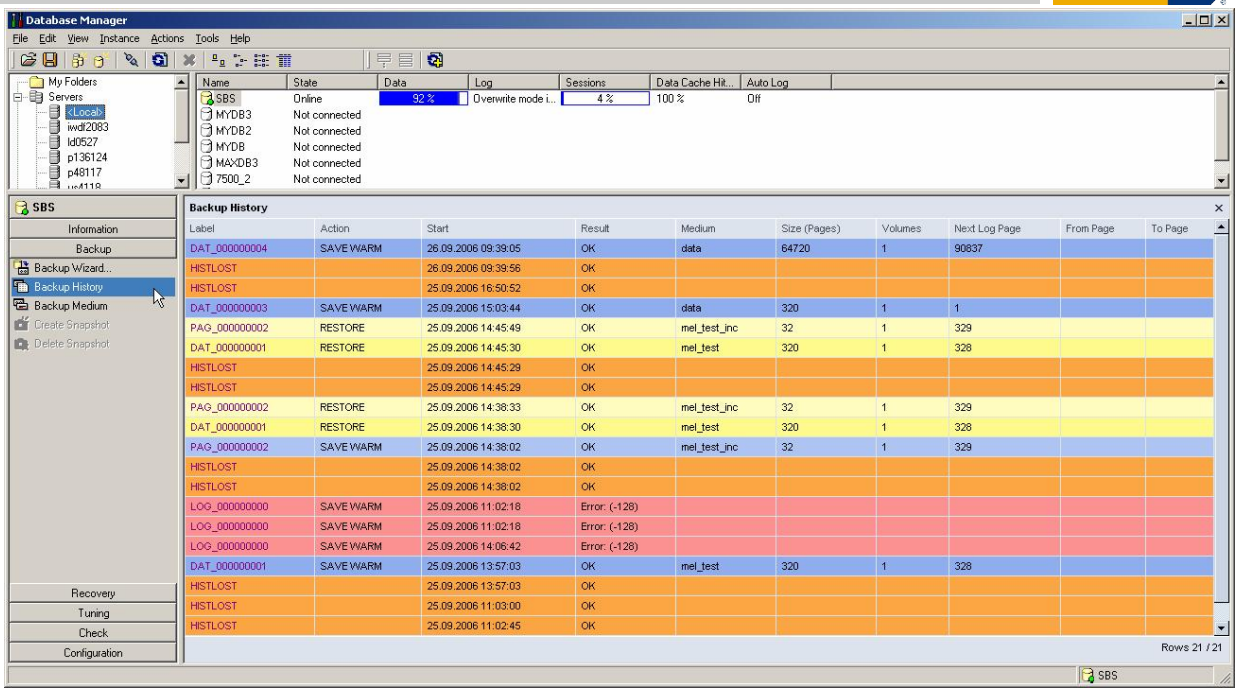
Deactivating the Automatic Log Backup

dbmcli -d <db_name> -u <dbm_user>,<password> autolog_off

Checking the Status of the Automatic Log Backup

dbmcli -d <db_name> -u <dbm_user>,<password> autolog_show

Backup History



To check wether the backup actions were succesful, have a look to the backup history (file dbm.knl).

A red entry shows an erroneous backup action.

To figure out what went wrong with this backup action, check the files dbm.prt, dbm.ebp and knldiag.

Recovery

In order to restore the tables to the state before a crash you need a complete backup and the corresponding log or incremental backups

- From the complete backup file you can restore the tables to the state they were at the time of the backup.
- From log backup all actions are redone, which were performed between the backup and the crash.

Before you recover the database instance after a database error, you have to decide which earlier version of the database instance you want to recover. To recover the current database status after a database error (in other words, the status immediately prior to the error), you must reload all the log backups to the system that were carried out after the start of the data backup involved. An older database status can be recovered by using a data backup and only some of the log backups.

You must import the existing backups in the following sequence:

A complete data backup

An incremental data backup if one exists

Then the existing log backups

Files from log backups that were saved by means of the operating system to other locations - for example, tape devices - must be made available again in file form prior to the start of the recovery operation. If the log information is still available on the log volume, the database will use the information available on the log volume to redo the actions. This is faster then reading the needed information from the backup.

When you recover data with the Database Manager CLI, you must use this tool in session mode, since recovery operations do not permit you to interrupt the session.

You can only recover the database instance in the ADMIN operational mode of the database.

Recovery With Initialization

If you perform a recovery with Initialization, the log volumes are initialized – all data is deleted from the log volumes.

In this case you can only restore the database to the state of the complete or incremental backup.

You have to choose this option

- if you perform a system copy
- if the log information on the log volume does not fit to the complete or incremental backup to be restored
- if the log information on the log volume does fit to the complete or incremental backup but you do not want to restore the database to the latest state but to the state of the complete or incremental backup

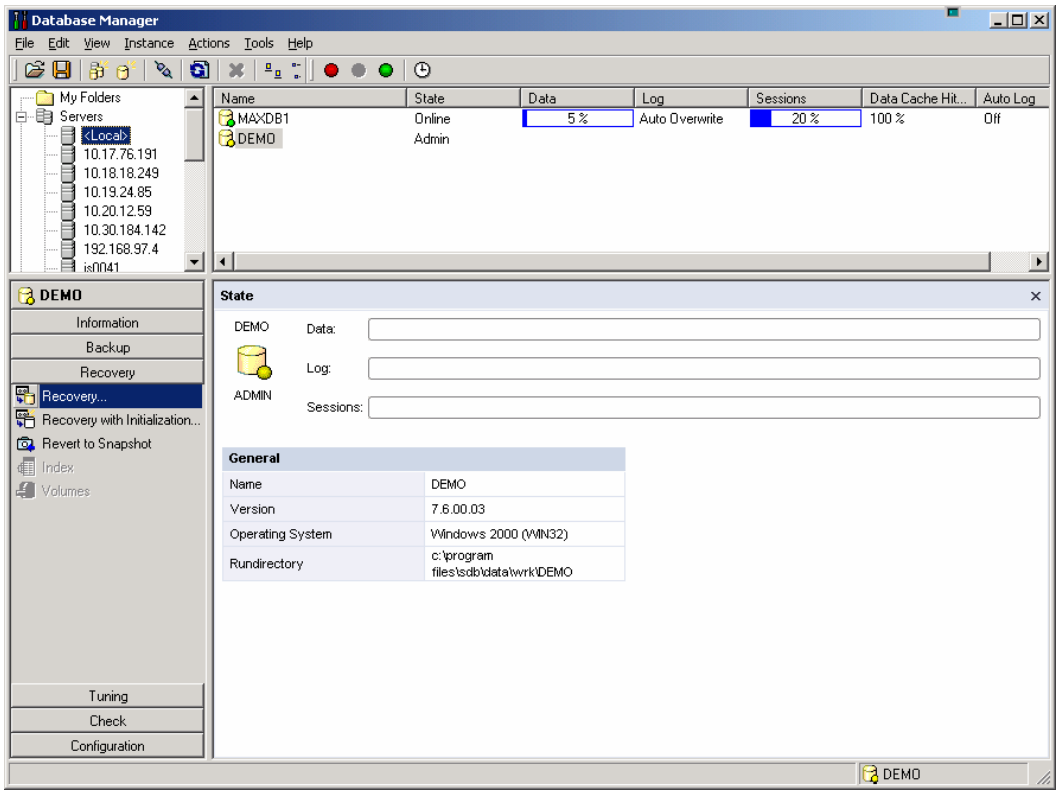
Recovery Considerations

Recovering individual tables is not possible.

It is possible to restore a complete backup to a different database instance. Afterwards you can extract the required table from the new database and load it into the original database.

This can cause data inconsistencies as there might be dependencies between several tables. You have to know the application logic very well if you want to restore individual tables!

Recovery 1/4



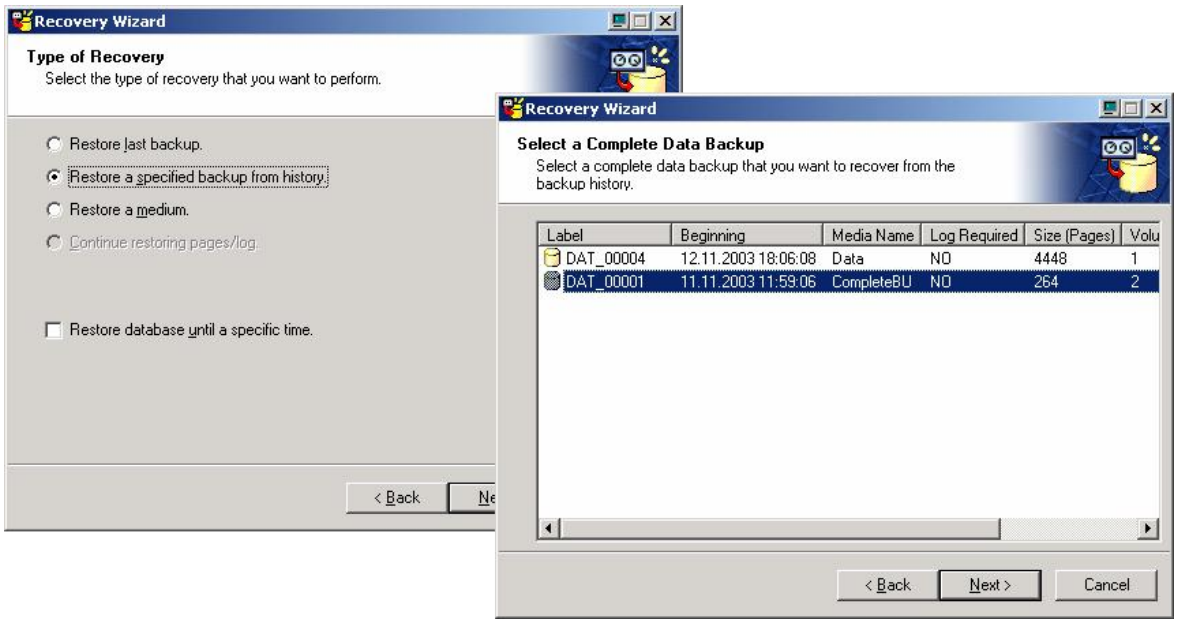
To recover the database it has to be in state ADMIN.

The Recovery Wizard supports the recovery.

If you choose ‚Recovery with Initialization', the database instance is initialized. That means that the volumes are reformatted and all data on the data and the log volumes is lost! Normally this is only necessary if the log volume is corrupted or if you perform a system copy.

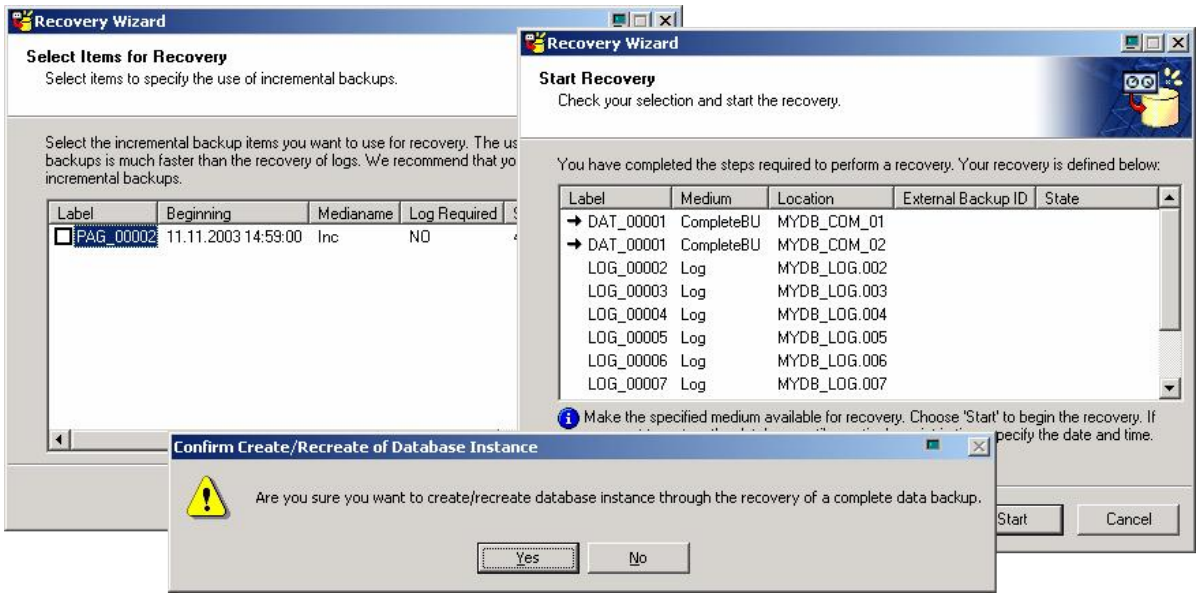
However, the steps to be performed in the Recovery wizard are the same with and without initialization.

Recovery 2/4



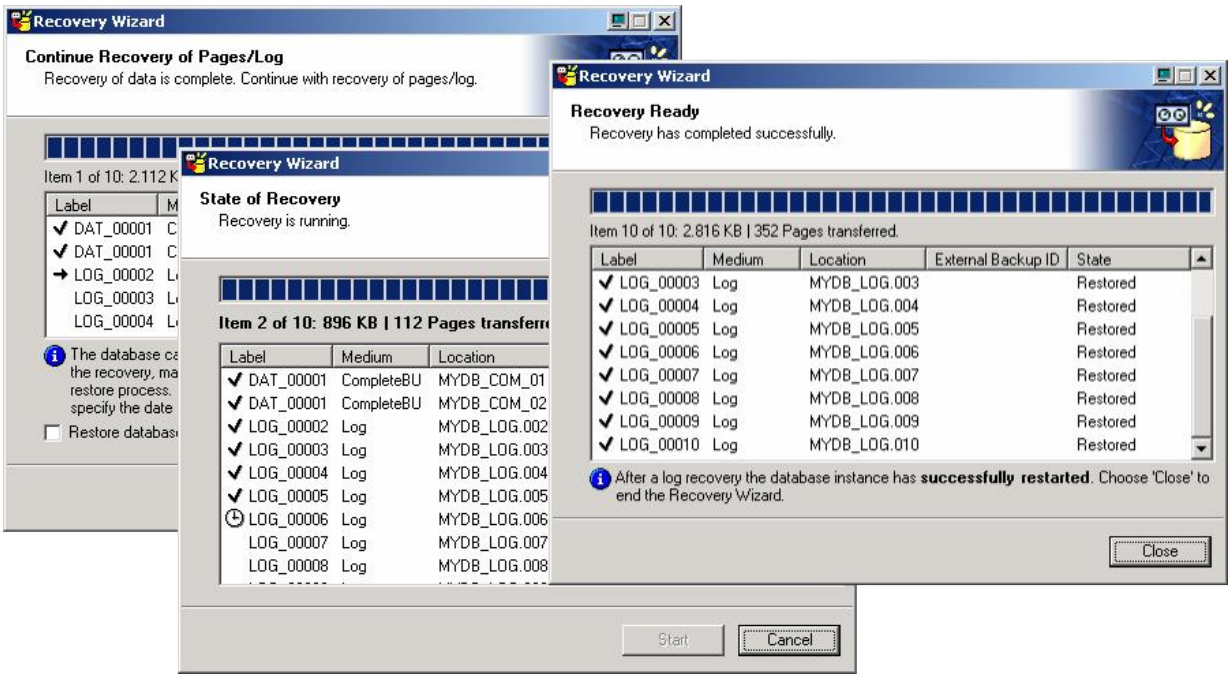
- You can recover using the following options:
  - restoring the last backup,
  - restoring a specified backup, e.g. from an older backup generation,
  - or by directly assigning a medium which contains the complete data backup.
- If you want to recover to a certain point in time instead of the point of failure you can mark the corresponding box. The wizard will ask for the point in time later during the recovery process (this option is only usable if log backups are restored).

Recovery 3/4



A popup window asks to confirm the initialization of the volumes if this menu item has been choosen.

Recovery 4/4



After the complete backup has been restored the Recovery Wizard stops. You have the chance to change the tape if necessary. Confirm to continue the restore when you are ready.

All log backups are restored automatically (if available). Afterwards the database instance is restarted.

Restoring a complete backup called DemoDataCompl and an incremental backup called DemoDataInc using dbmcli:

dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL

recover_start DemoDataCompl

recover_start DemoDataInc

db_online

During startup of the database the log information on the log volume is redone.

Restoring a complete backup (called DemoDataCompl) with initialization and an incremental backup (called DemoDataInc) using dbmcli:

dbmcli -d <db_name> -u <dbm_user>,<password> -uUTL

db_activate RECOVER DemoDataCompl

recover_start DemoDataInc

db_online

After startup the database contains the data of the complete and the incremental backup – no log information is redone as the log volumes are initialized.

Checking the Database Structure

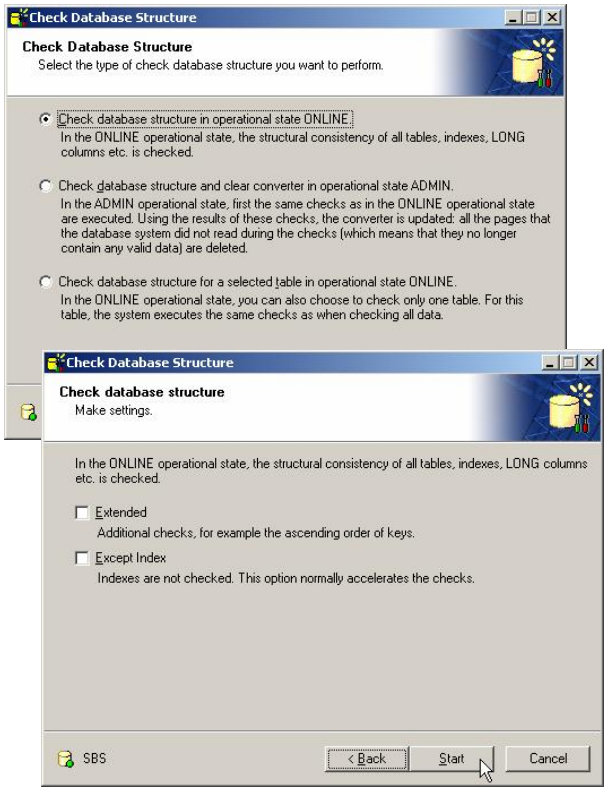
DBMGUI -> Check -> Database Structure

CHECK DATA checks the structure of the complete database: tables, indexes and BLOBs.

The table which is currently checked is locked.

The performance of your system is decreased while the database structure is being checked!

If CHECK DATA reports errors, you have to check the hardware and to restore a backup.




```
dbmcli –d <db_name> -u <dbm_user>,<password> -uUTL -c db_execute

            CHECK DATA [EXTENDED] WITH UPDATE (state ADMIN only)

            CHECK DATA [EXTENDED] [WITHOUT INDEXES] (state ONLINE
            only)
```

If an unqualified delete or a drop table is executed for a table which contains corrupt pages it might be that some pages of this B* tree remain in the database. These pages are deleted during a ‚check data with update' which can only be executed in database state ADMIN.

EXTENDED: checks if the separators are in the correct order.

WITHOUT INDEXES: the B* trees of Indexes are not checked.

When errors are reported, you need to check the hardware. If only indexes are corrupted, these can be recreated using DBMGUI. Otherwise you have to restore a backup.

Checking the Structure of a Single Table

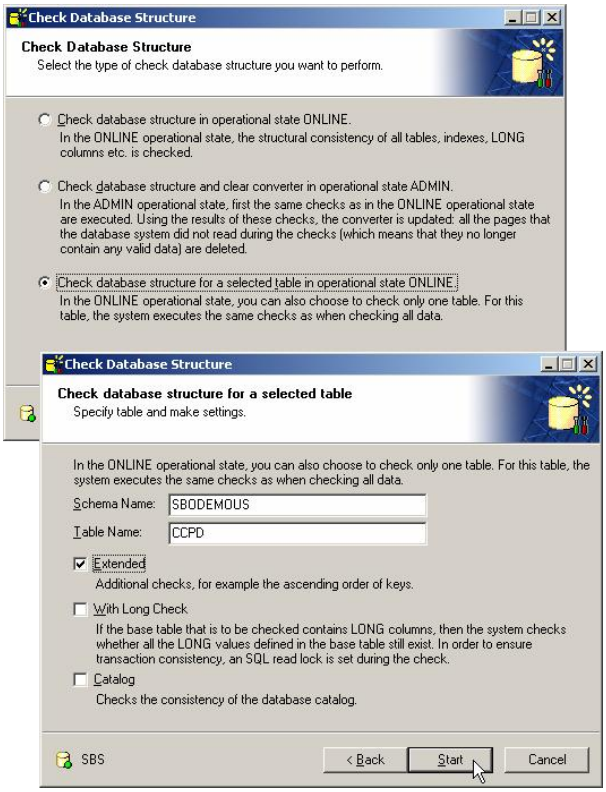
DBMGUI -> Check -> Database Structure

dbmcli –d <db_name> -u <dbm_user>, <password> -uSQL –c sql_execute CHECK TABLE <tablename> [EXTENDED] [WITH SHARE LOCK]

EXTENDED: CHECK TABLE checks if the separators are in the correct order

WITH SHARE LOCK: a check is made to see whether a BLOB actually exists for each BLOB surrogate in the table

BLOBs formerly were called LONG



Check Backup

DBMGUI -> Check -> Backup

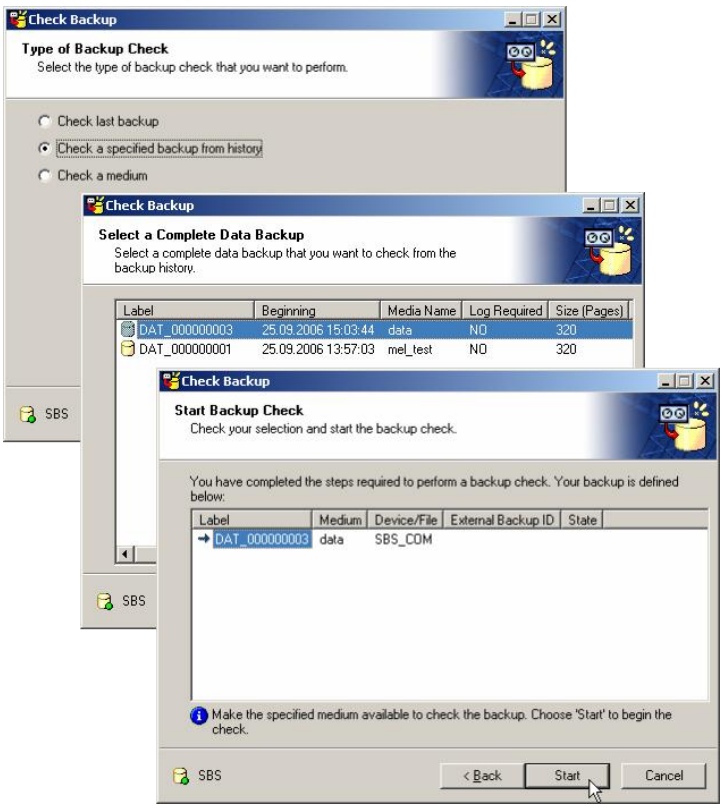
dbmcli –d <db_name>
-u <dbm_user>,<password>

service connect

recover_check <medium> <savetype>

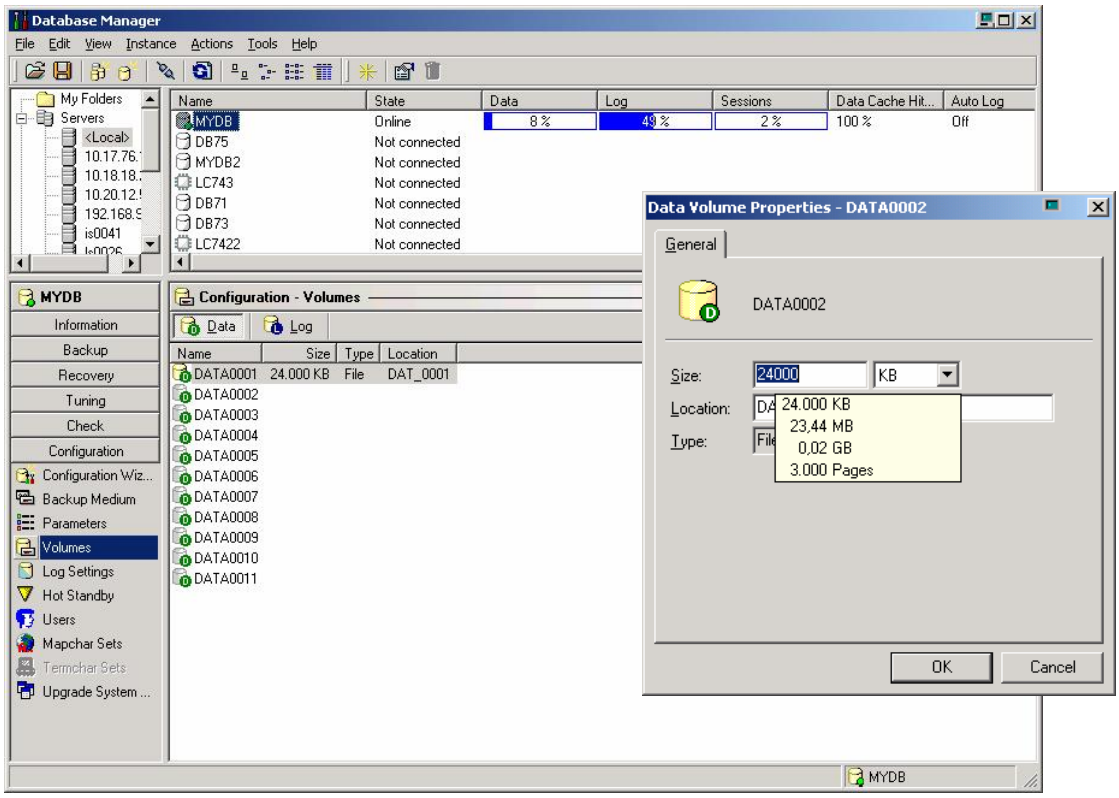
service_release

Uses the service database. Therefore the productive instance is not influenced



Tape devices might not work correctly, tapes might be damaged -> Check the integrity of your backups regularly.

Adding Data Space 1/2



To add a data volume select the menu path ‘Configuration -> Volumes‘, tab ‚Data Volumes‘ and select the next free data volume.

Assign the values for size, location (complete path recommended) and type to the new volume and confirm with ‚OK‘.

Example with DBMCLI:

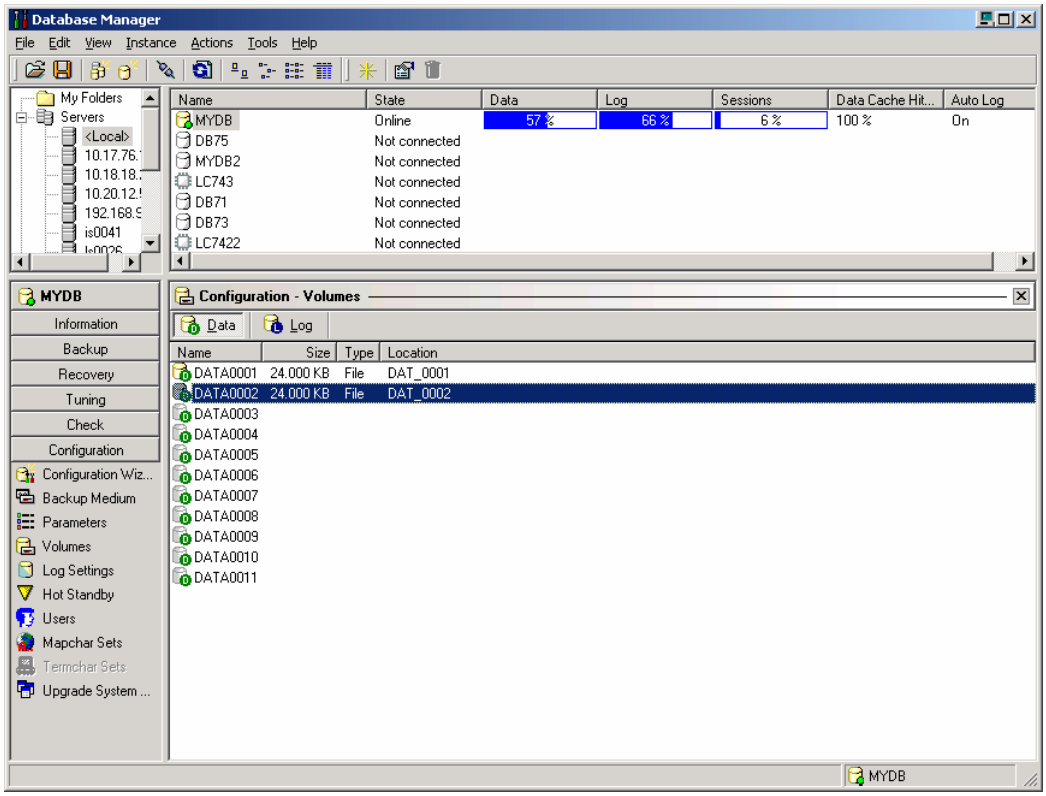
dbmcli –d <db_name> -u <dbm_user>,<password> db_addvolume DATA c:\sapdb\volumes\DAT_0002 F 3000

c:\sapdb\volumes\DAT_0002: location of the new volume (without the path specification the file DAT_0002 will be created in the rundirectory)

F: device type FILE

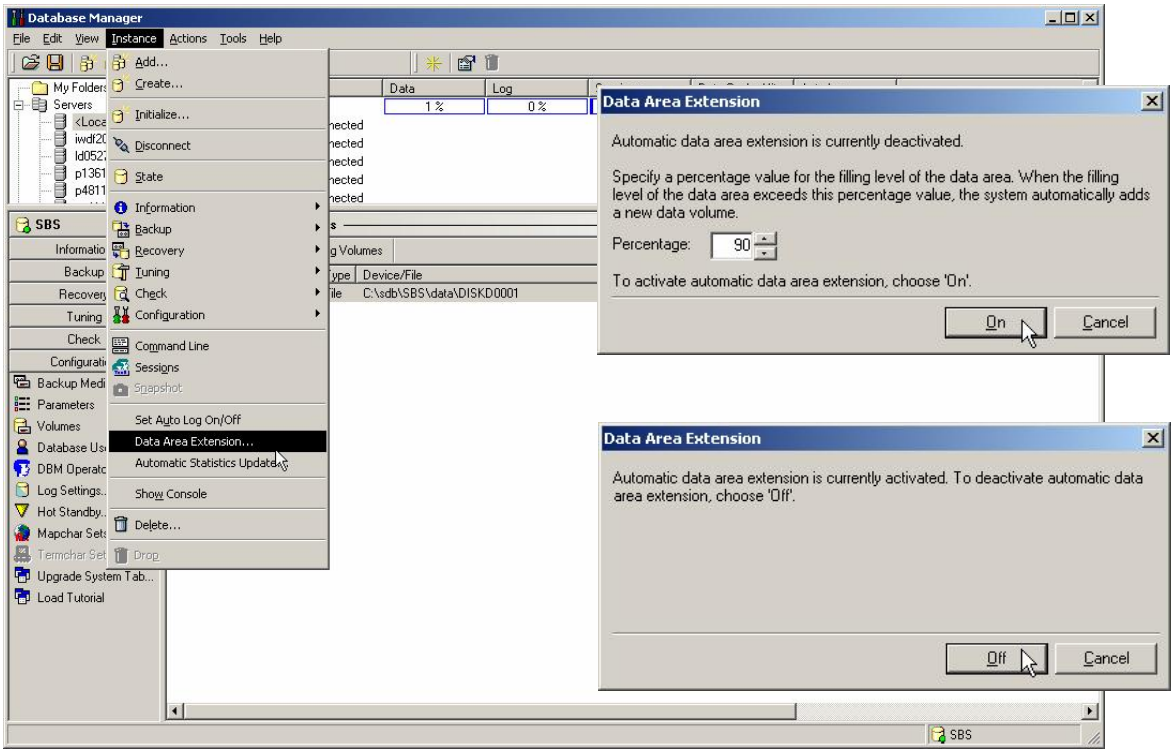
3000: Size in Pages

Adding Data Space 2/2



The new data volume is available immediately.

Automatic Data Area Extension



dbmcli –d <db_name> -u <dbm_user>,<password>
auto_extend ON 90

dbmcli –d <db_name> -u <dbm_user>,<password>
auto_extend OFF

The name of the new data volume is choosen depending on the already existing data volumes.

Using the Database Manager Parameter AutoExtDir you can specify in which directory the new data volumes are created. If no directory is specified, the new volumes are created in the same directory as the lastly added volume.

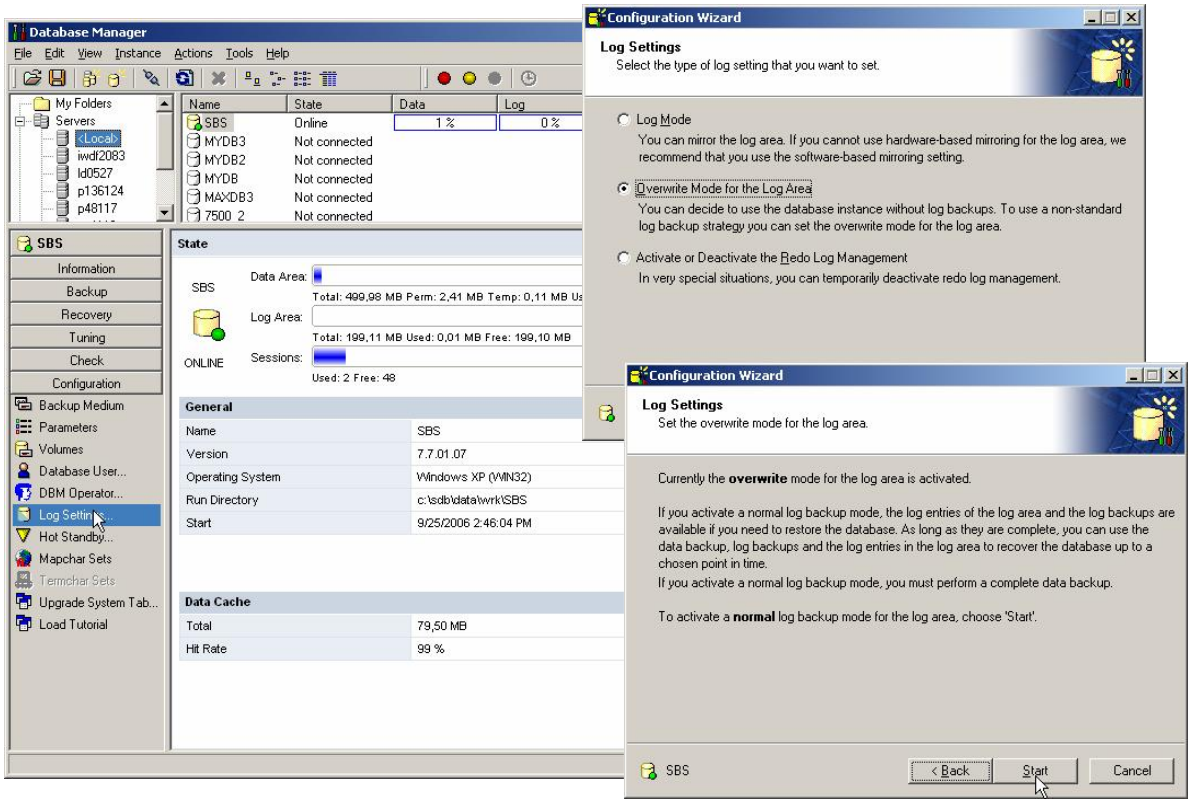
Using the Database Manager Parameter AutoExtSize you can specify the size of the new data volumes – either in Pages or in % of the database size (at the point in time when the new volume is added). If the parameter is not set, the new volume gets the same size as the lastly added volume.

Alter Log Settings 1/2

MaxDB offers different modes to run the log:

- You can mirror the log area.
- You can decide to do without log backups by setting the overwrite mode for the log area accordingly.
- In very special situations, you can deactivate the redo log management temporarily.

Alter Log Settings 2/2



You can make the following log settings:

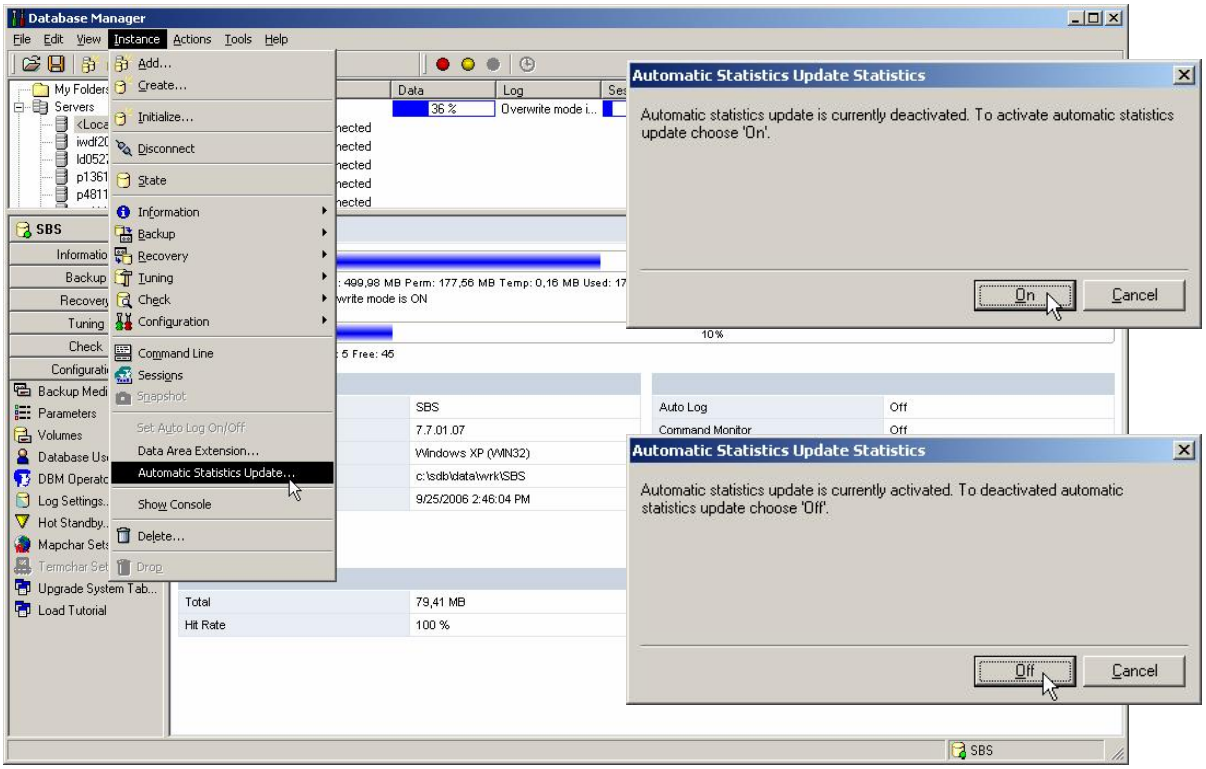
You can mirror the log area. If you cannot use hardware-based mirroring for the log area (which is recommended), you can use the software-based mirroring setting from MaxDB. When you choose software-based mirroring, you specify that the log entries are written to two log areas in parallel (mirrored). However, the log entries are read from one log area only.

You can decide to do without log backups by setting the overwrite mode for the log area accordingly. You select the overwrite mode with the DBMCLI command
db_execute SET LOG AUTO OVERWRITE ON
(or using the corresponding option in DBMGUI as shown on the slide). When you set the overwrite mode, the log area will be overwritten cyclically without the log entries having to be backed up first. The overwrite mode remains activated even after the database instance is stopped. You have to deactivate the overwrite mode explicitly with
db_execute SET LOG AUTO OVERWRITE OFF
(or using the corresponding option in DBMGUI). To enable log backups again, you have to restart the backup history with a complete data backup.

In very special situations, you can deactivate redo log management temporarily using the DBMCLI command
db_execute SET LOG WRITER OFF
in the operational state ADMIN (or using the corresponding option in DBMGUI). When you deactivate redo log management, transactions no longer write their redo log entries to the log queue. Redo log management remains deactivated even after the database instance is stopped. You have to reactivate redo log management explicitly with
db_execute SET LOG WRITER ON
in state ADMIN (or using the corresponding option in DBMGUI) and starting a new backup history with a complete data backup.

The status of the auto overwrite feature is stored in the log volume. After a recovery with initialization this status is reset to the default (which means that the auto overwrite feature is deactivated). After a recovery with initialization has been executed, the auto overwrite feature has to be activated again.

Automatic Statistics Update



dbmcli –d <db_name> -u <dbm_user>,<password> -uUTL
auto_update_statistics ON

dbmcli –d <db_name> -u <dbm_user>,<password> -uUTL
auto_update_statistics OFF

Snapshots

Freeze a consistent state of the database (for a future restore)

Instantaneous backup of the complete database

All subsequent changes are written to new pages

Recovery to previous snapshot

Restore snapshot

Restart

Usage scenarios

Restore of demo or training systems to a previous state

Very fast point-in-time resetting

Creating and Deleting Snapshots

Creating a snapshot

The database has to be in operational state ADMIN to create a snapshot.

DBMCLI command: ***db_execute CREATE SNAPSHOT***

Only one snaphot can exist at the same time. If a snapshot is created, an already existing snaphot is overwritten.

Deleting a snapshot

The database has to be in operational state ADMIN to delete a snapshot.

DBMCLI command: ***db_execute DROP SNAPSHOT***

If there is no snapshot this command returns OK anyway.

Attention: the following four database operations remove an existing snapshot:

- create instance (db_activate)
- restore a data backup
- drop snapshot
- create snapshot

The restore of an incremental backup does not remove a snapshot.

Reverting to a Snapshot

Restoring a snapshot

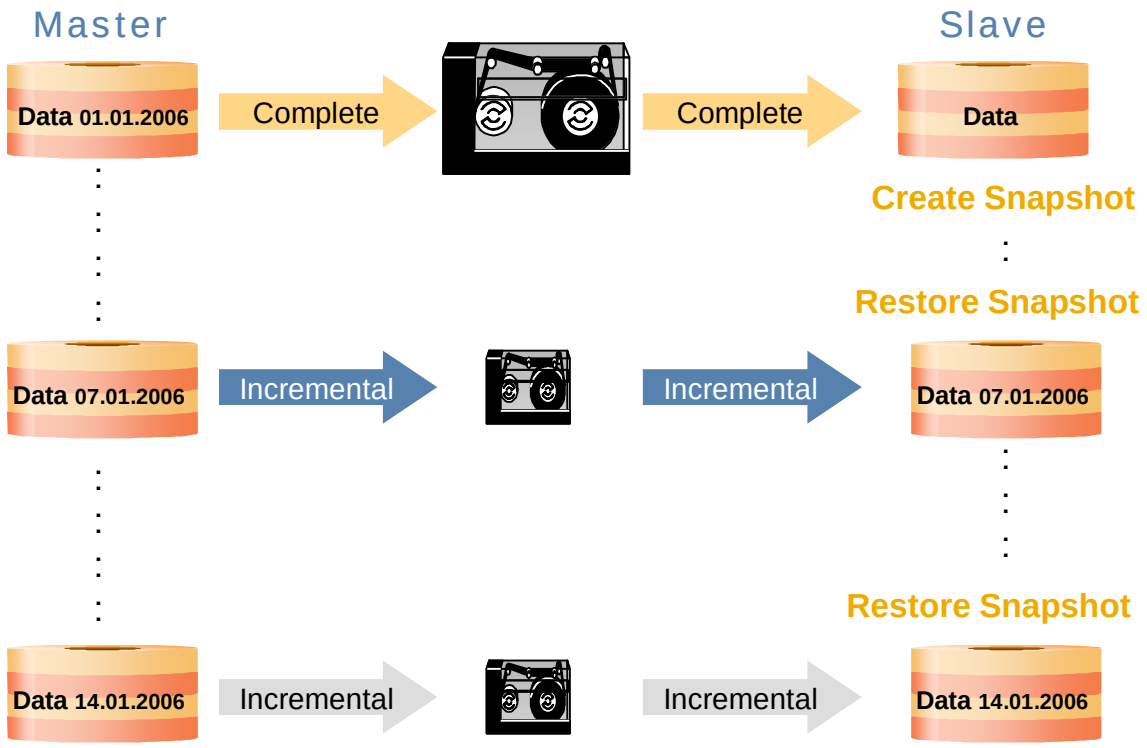
The database has to be in operational state ADMIN to restore a snapshot.

DBMCLI command: ***db_execute RESTORE SNAPSHOT***

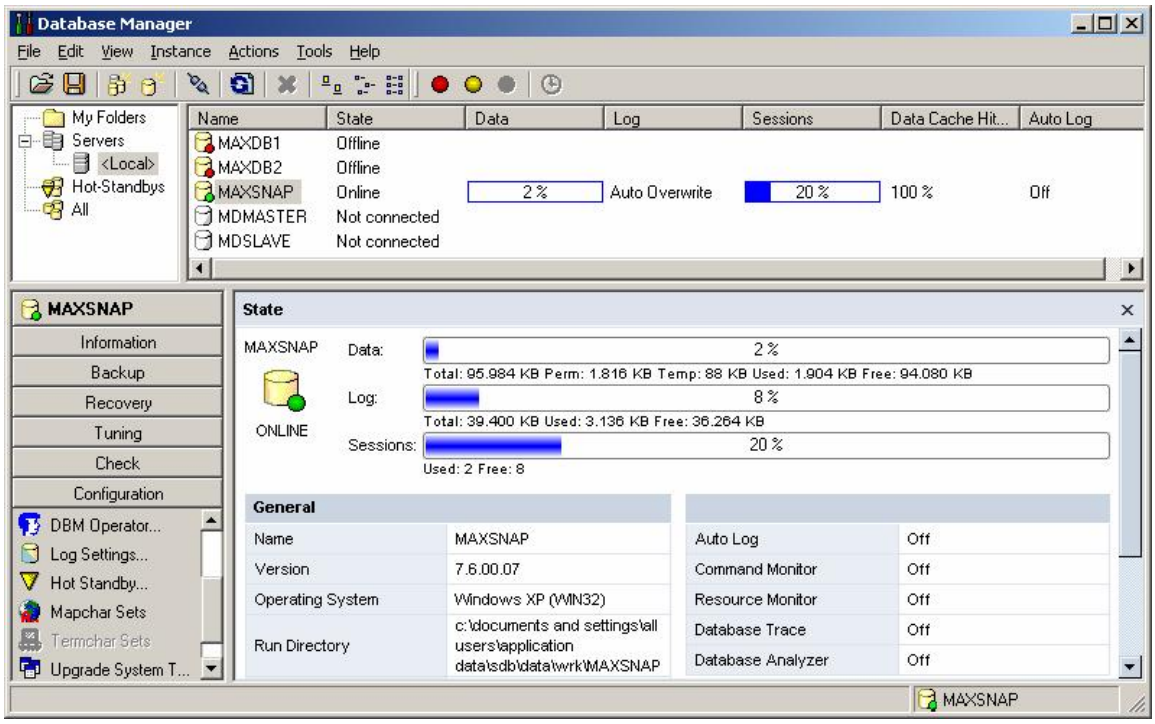
After it is restored the snapshot is still valid – that means it can be restored several times.

If there is no snaphot, the command returns „–8051 No snapshot available“.

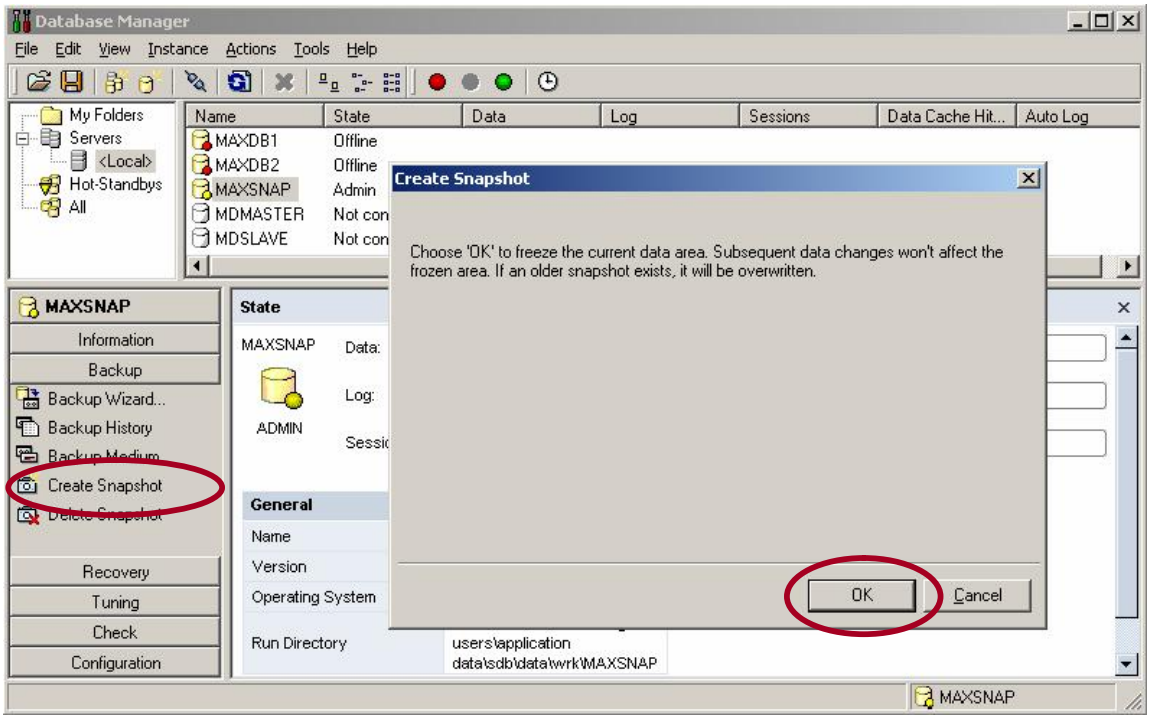
Master – Slave Support w/ Snapshots



Snapshots: Database After Setup

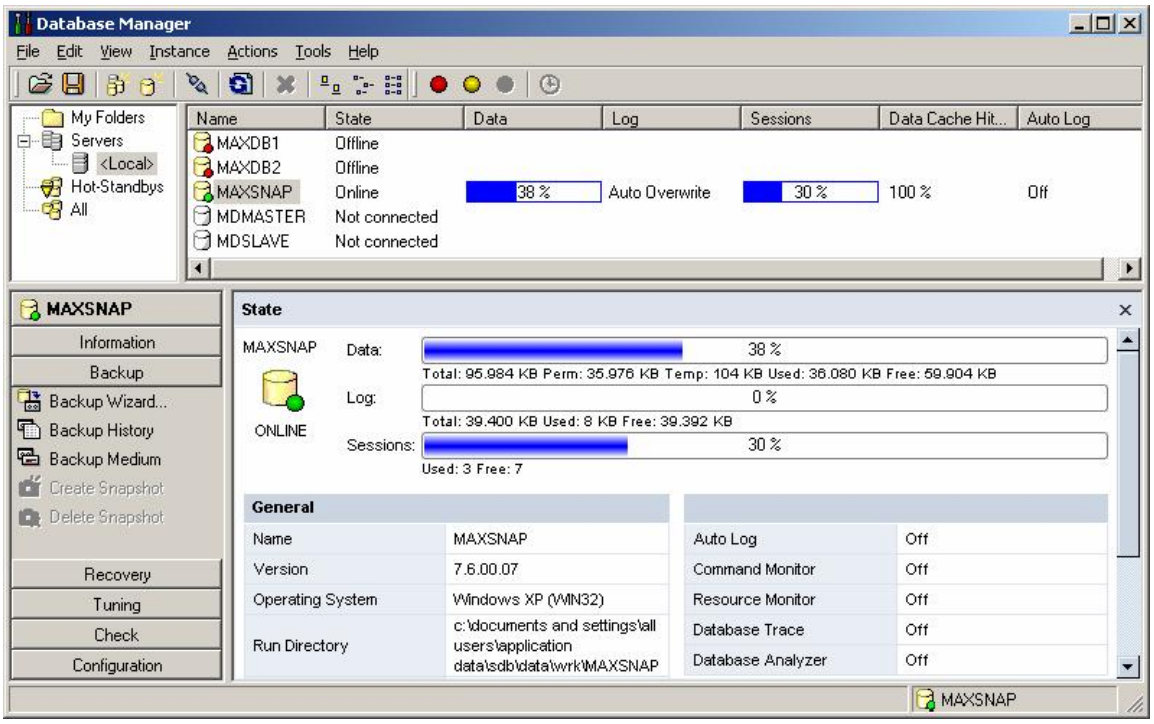


Taking a Snapshot



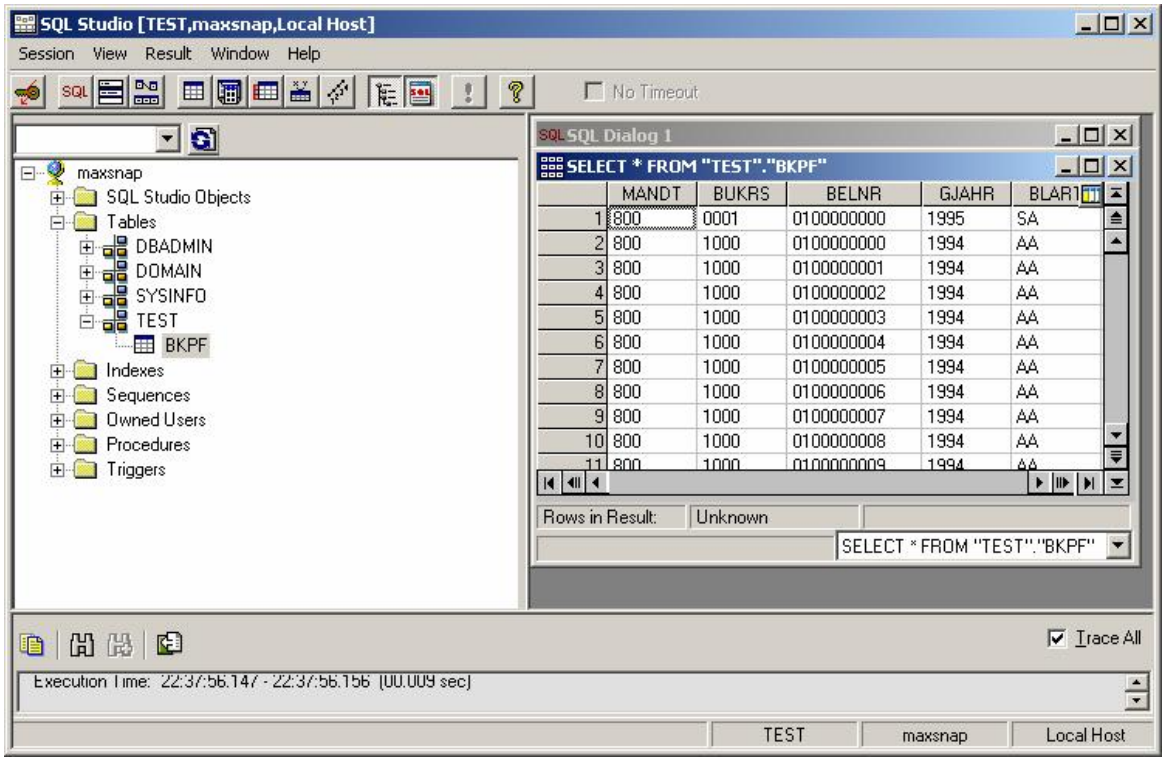
If you would like to create a snapshot, the database has to be in ADMIN mode. *DBMGUI* allows to create the snapshot with menu *Backup -> Create Snapshot*.

Some Data Has Been Loaded

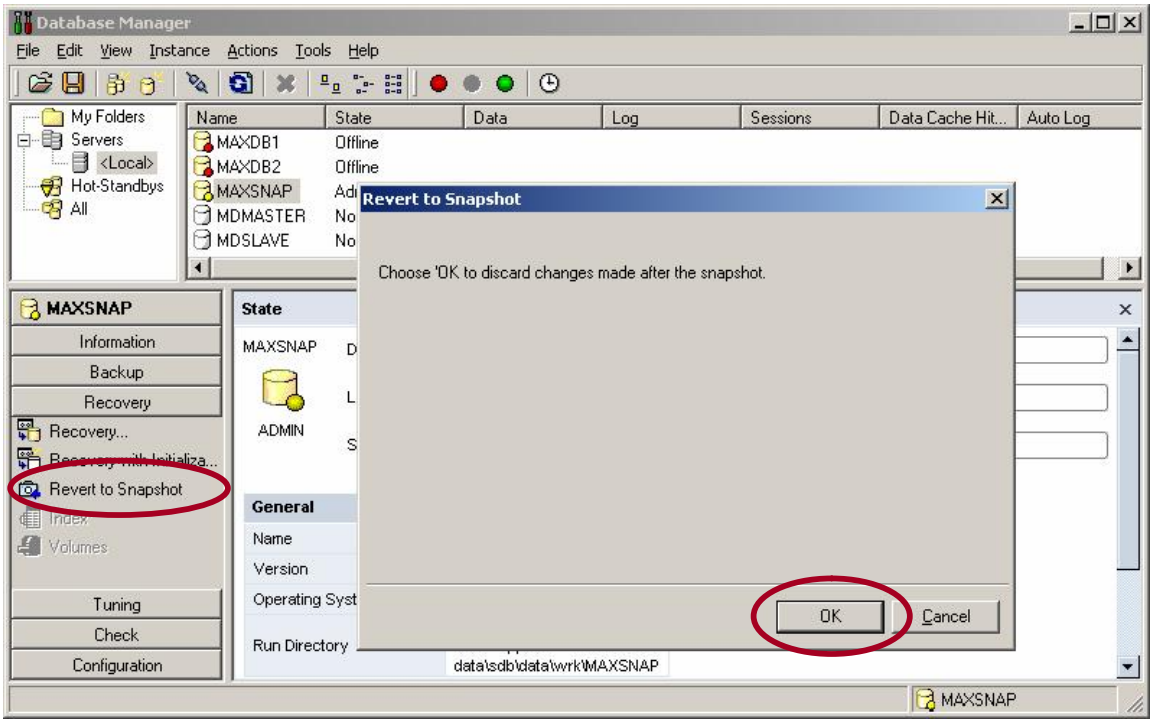


Then you can use your database instance normally – e.g. you can load some data.

Checking Data in SQL Studio

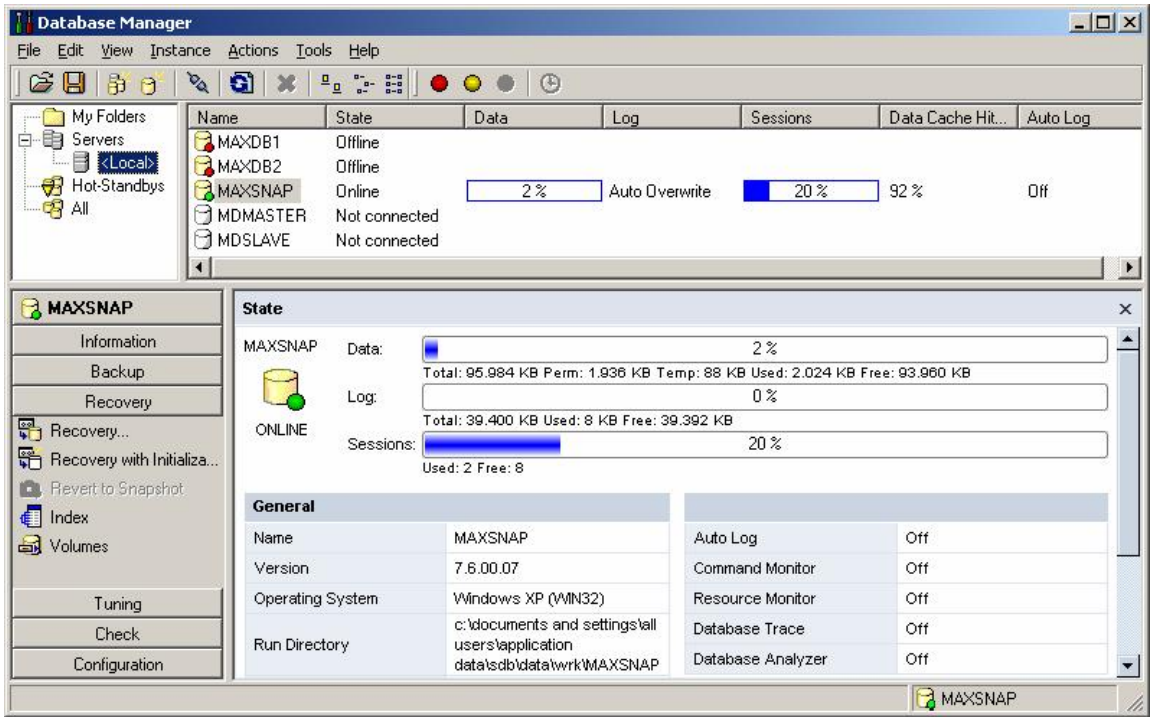


Admin Mode: Revert to Snapshot



If you would like to revert to the snapshot, choose menu Recovery -> Revert to Snapshot. This can be done in ADMIN mode, only.

After Reverting to Initial State



After you restored the snapshot, all data loaded after the creation of the snapshot is gone, data which was deleted, is available again.

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

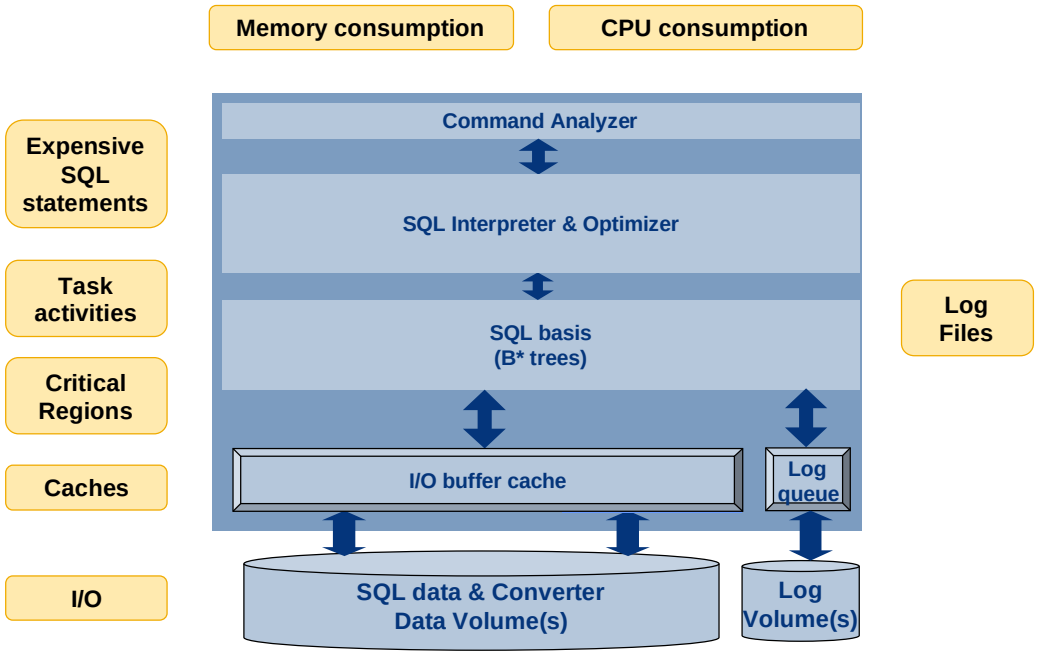
Further Information

Learning Objectives

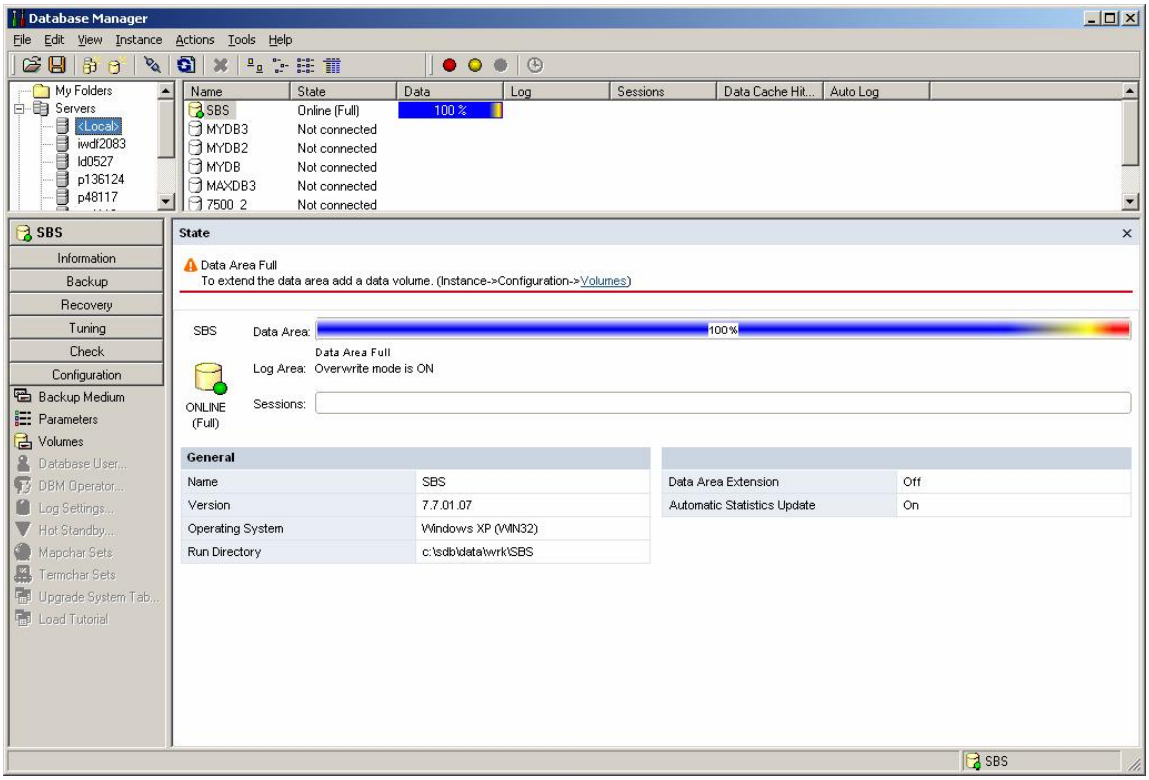
Know how to...

- detect a database full or log full situation
- monitor the data cache hitrate
- detect lock collisions
- detect and rebuild corrupted indexes
- use the database analyzer
- use the command and resource monitor
- switch on the database and SQLDBC traces

MaxDB Monitoring



Monitoring: Database Filling Level



To monitor the database filling level, double-click on the database name in the top window.

The bar view allows a quick overview of the filling level.

In this case the database area is completely filled. A warning at the bottom of the window also indicates this critical situation.

At this point, all database tasks are suspended until the db_full situation is sorted out.

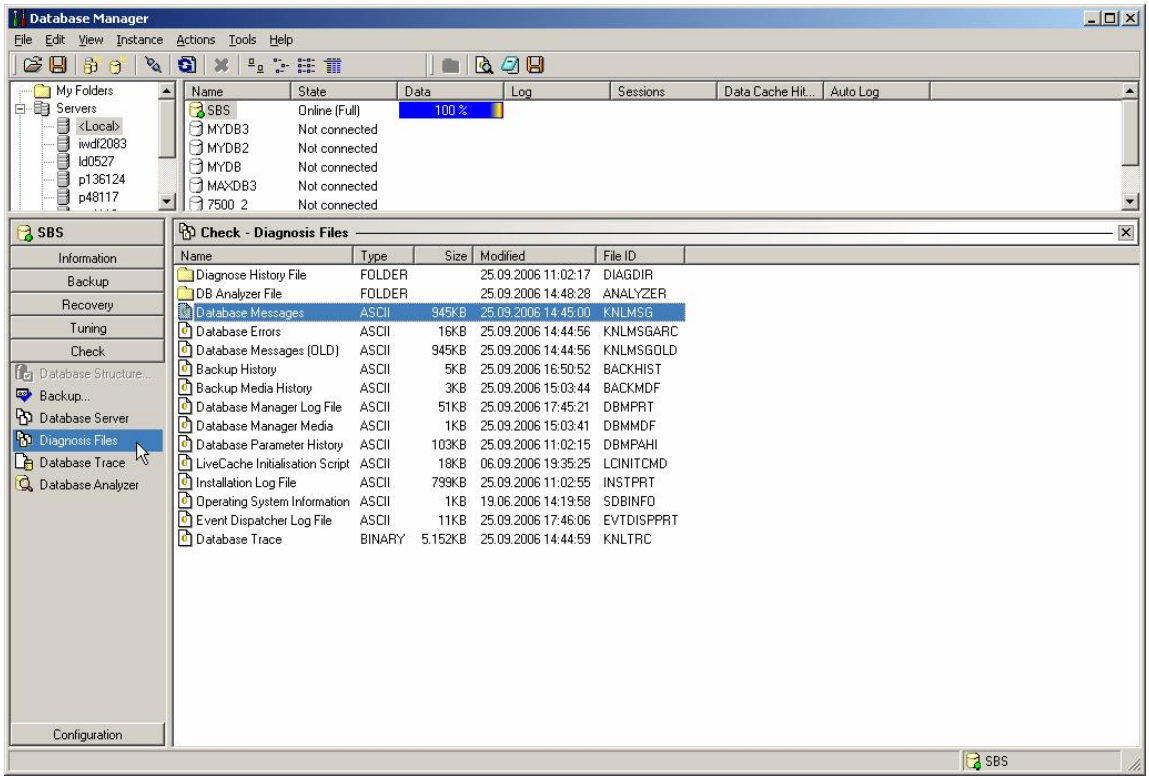
Corresponding DBMCLI command:

dbmcli –d <db_name> -u <dbm_user>,<password> -uSQL <dba_user>,<password> info state

ERR

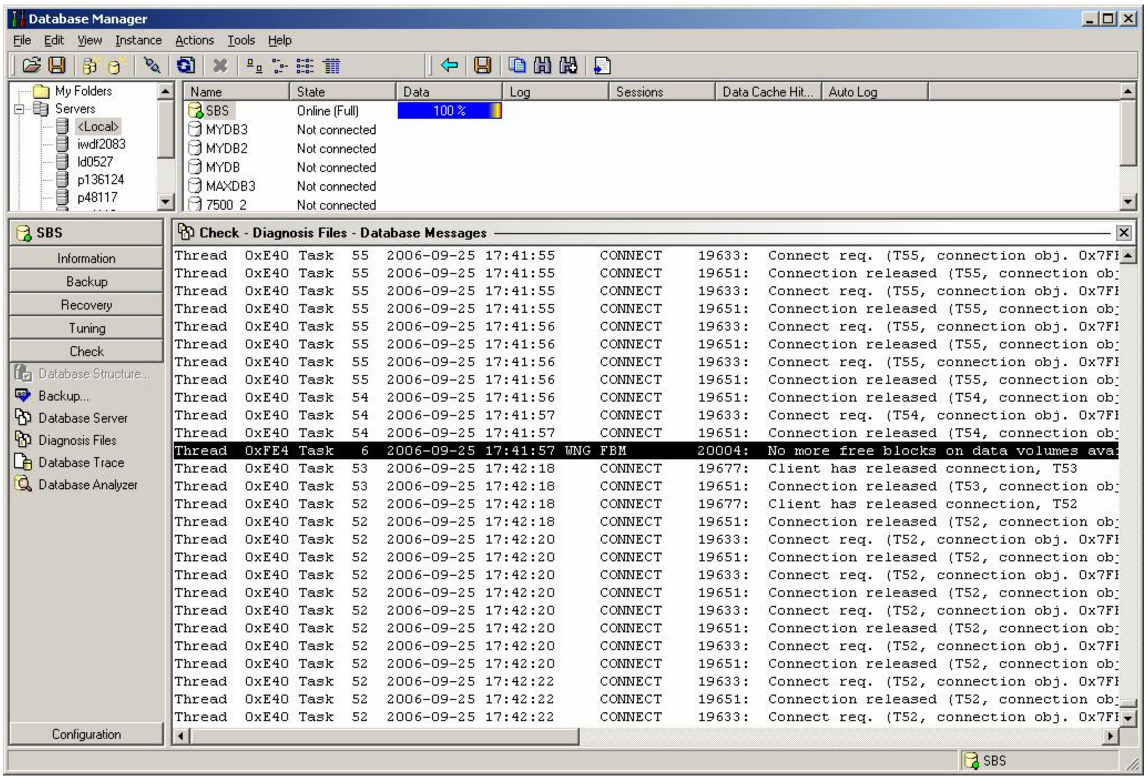
-24798,ERR_DATAFULL: Data area is full

Monitoring: DB Full

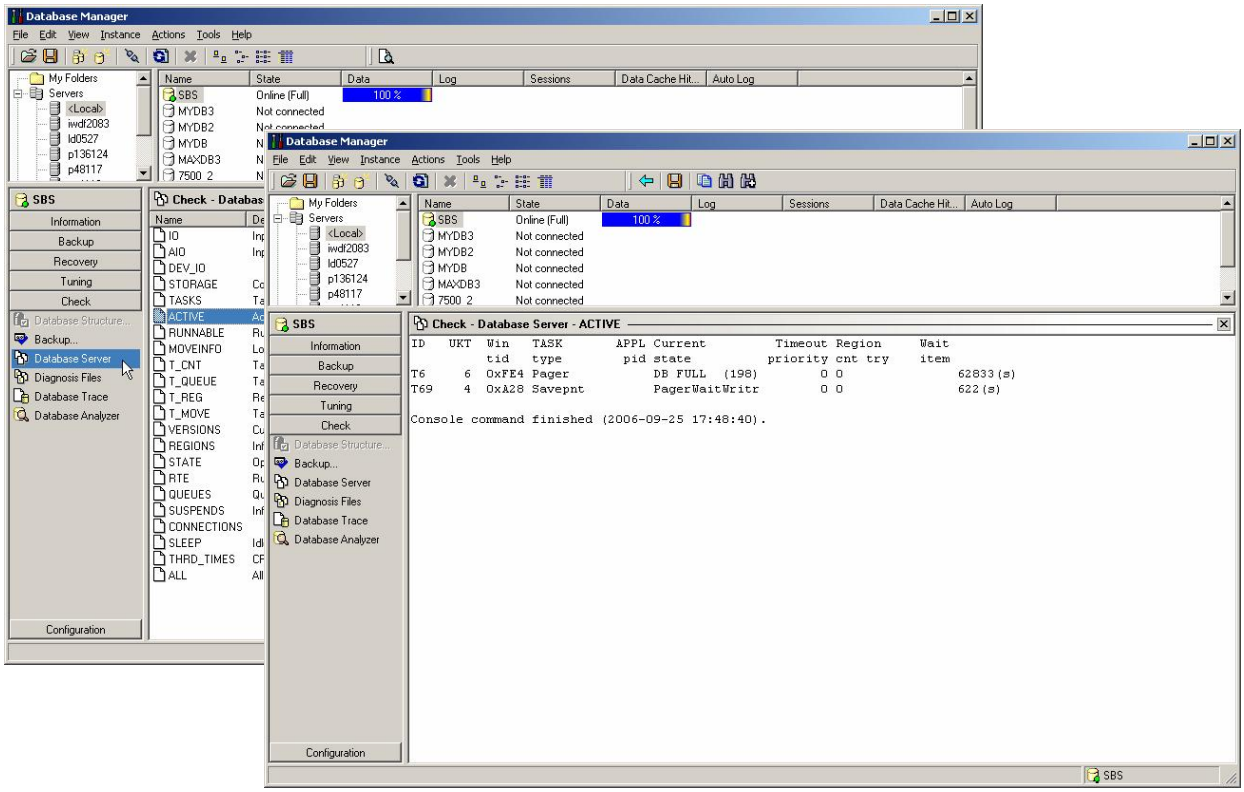


In case of a database standstill, you can also check in file KnlMsg whether it is caused by a db_full situation.

Monitoring: DB Full



Monitoring: DB Full



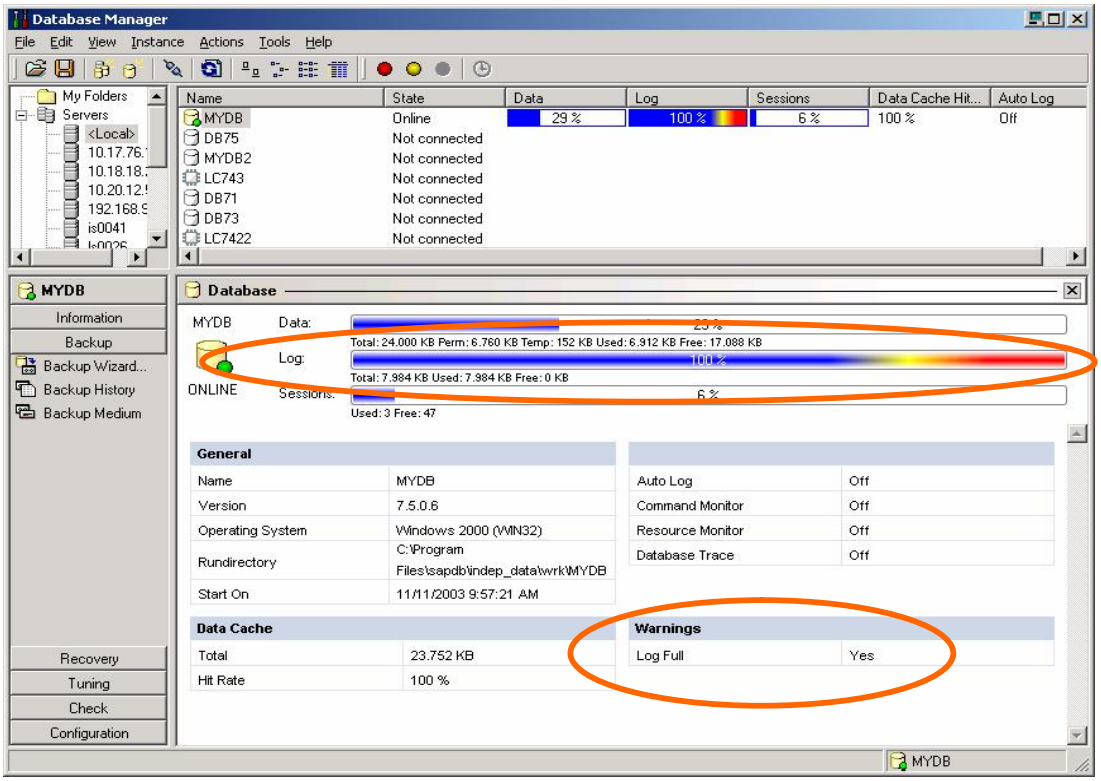
You can also prove this via ‚Check → Database Server → Active' to view the active database tasks

Corresponding DBMCLI command:

dbmcli –d <db_name> -u <dbm_user>,<password> show active

```
OK

SERVERDB: <db_name>

ID   UKT  Win    TASK       APPL Current          Timeout Region     Wait
          tid    type        pid state          priority cnt try    item
T6     6  0xFE4 Pager           DB FULL  (198)        0 0              62855(s)

T69    4  0xA28 Savepnt         PagerWaitWritr       0 0              622(s)

Console command finished (2006-09-25 17:50:35).
```

Monitoring: Log Filling Level



To monitor the log filling level, double-click on the database name in the top window.

The bar view allows a quick overview of the filling level.

In this case the log area is completely filled. A warning at the bottom of the window also indicates this critical situation.

When the log is full it is not possible to execute any SQL statements such as ‘SELECT‘, ‘UPDATE‘, ‘INSERT‘ or ‘DELETE‘. It is also not possible to connect to the database with SQL users.

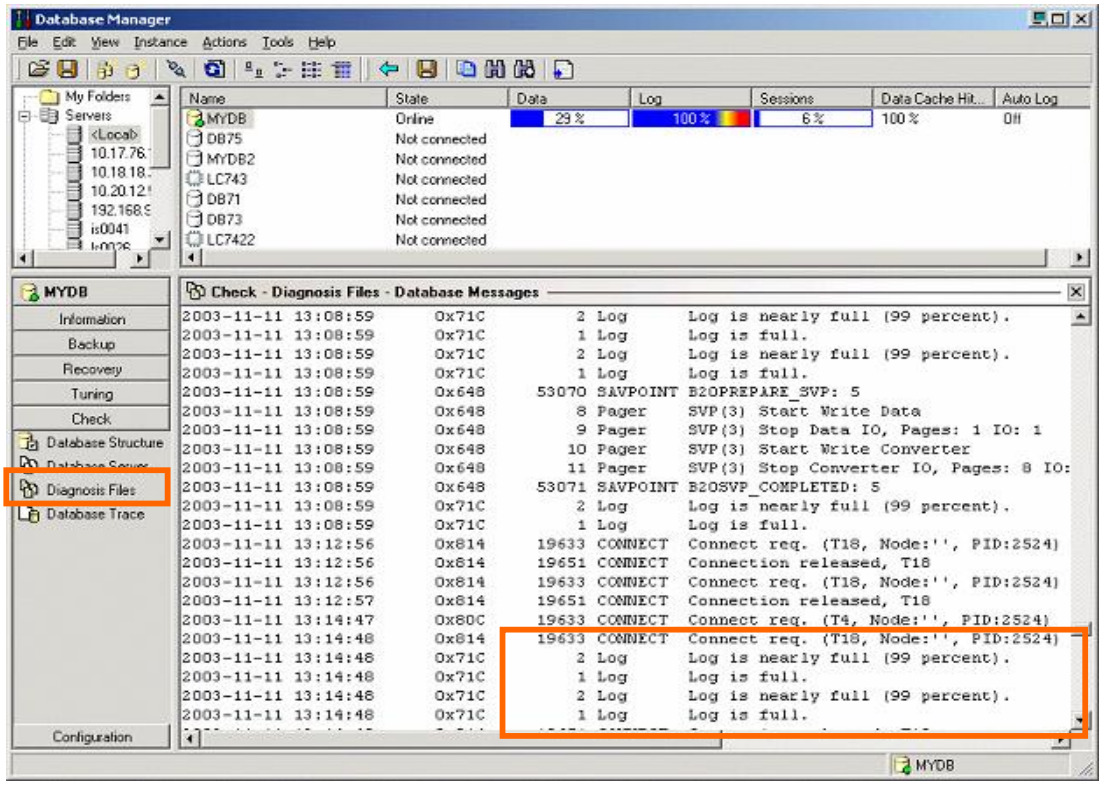
Corresponding DBMCLI command:

dbmcli –d <db_name> -u <dbm_user>,<password> -uSQL <dba_user>,<password> info state

Example output:

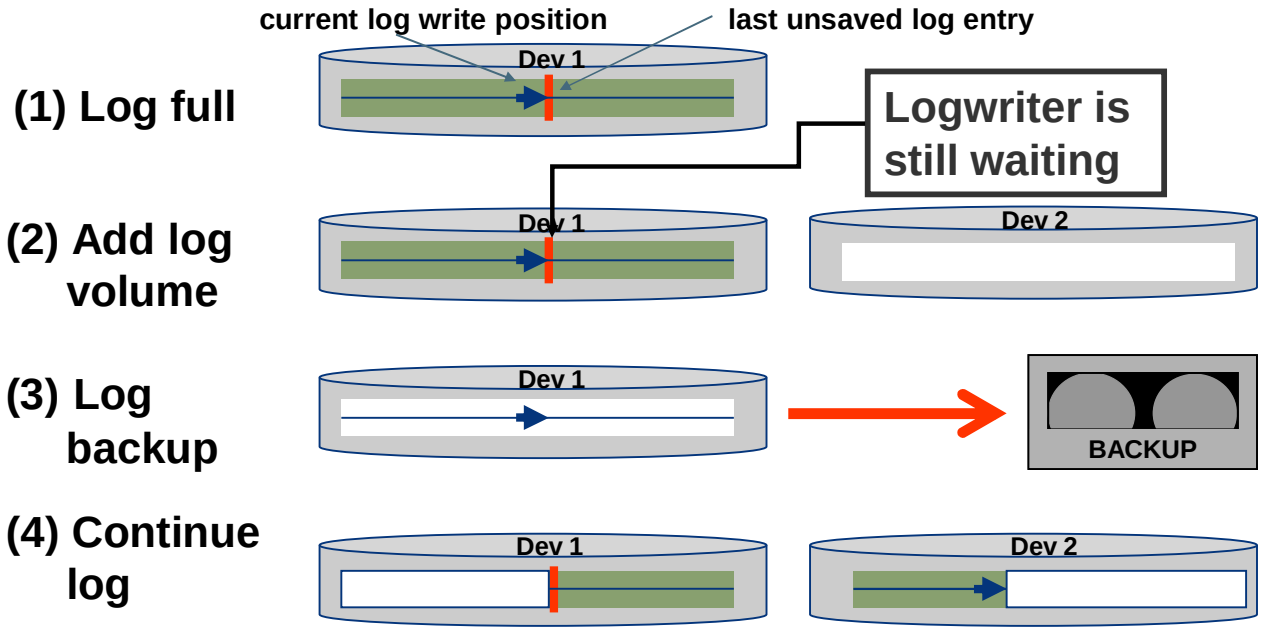
```
Name                    | Value
...
Log             (KB)    = 904
Log           (Pages)   = 113
Log               (%)   = 6
...
Log Full                = No
...
```

Monitoring: Log Full



In file knldiag you can see the message ‘Log is full‘. At this point, all database tasks are suspended until the log full situation is sorted out.

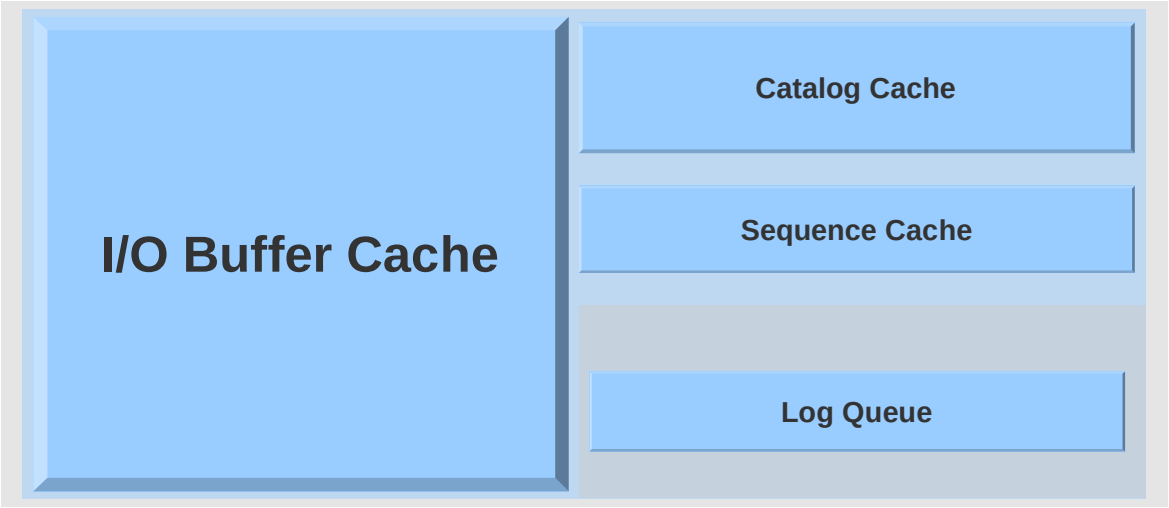
Resolving a Log Full Situation



Note: a log full situation can not be resolved by adding another log volume. MaxDB writes the log cyclically onto the volumes as if they were only one device. This means that even if a new log volume is added, the log writing has to be continued after the last written entry. Therefore, a log volume cannot be used immediately after it was added but the log has to be backed up before (SAVE LOG – interactive log backup).

Note: Prerequisite for a log backup is at least one data backup after installation.

Monitoring: Memory Areas



Relevant memory areas to monitor are *I/O buffer cache*, *Log Queue* and *Catalog cache*

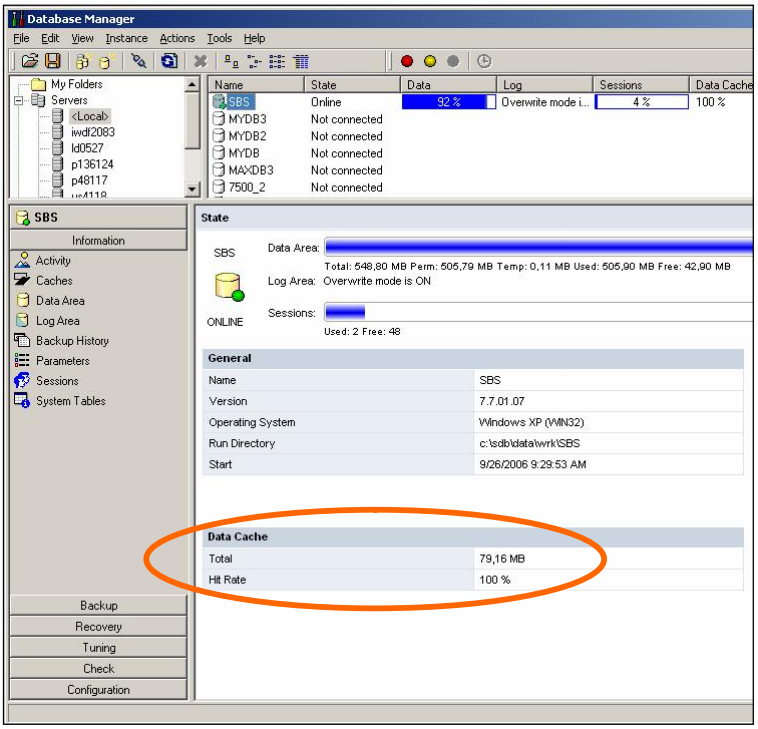
In the physical memory of the MaxDB server, MaxDB allocates the I/O buffer cache and some additional other memory areas. Generally the memory consumption of MaxDB is dominated by the I/O buffer cache.

- Data is transferred from data volumes to I/O buffer cache when data is accessed the first time.
- Any of I/O buffer cache pages may be swapped to data volumes (using an LRU mechanism) if the I/O buffer cache is too small to hold all the information. Exception: Converter pages that contains mapping information (cache <-> disk) are never swapped to data volumes.

Normally the other memory areas are small compared with the I/O buffer cache, but at least catalog cache and log queue should be monitored carefully.

- Catalog cache is dynamic and allocated when needed. It contains mainly information about the database catalog.
- Log queue is the buffer used to temporarily store log entries that need to be written to log volumes by the Log Writer task.

I/O Buffer Cache Hit Rate



I/O buffer cache size
I/O buffer cache hit rate

DBMGUI displays all information about the MaxDB memory areas.

The most important cache to monitor is the I/O buffer cache (Data Cache), where all table and index pages are stored. If data can not be found in I/O buffer cache, it must be read from data volume which is far more expensive than a memory access. Therefore the average I/O buffer cache hit rate should never fall below 98%.

To monitor the I/O buffer cache hit rate (Data Cache hit rate), double-click on the database name in the top window.

Corresponding DBMCLI command:

dbmcli –d <db_name> -u <dbm_user>,<password> -uSQL <dba_user>,<password> info caches

Example output:

```
Type           | Accesses   | Successful   | Unsuccessful | Hit Rate (%)
DATA           | 1849238    | 1847931      | 1307         | 100
SEQUENCE       | 0          | 0            | 0            | (null)
COMMANDPREPARE | 84         | 78           | 6            | 92
COMMANDEXECUTE | 84         | 84           | 0            | 100
CATALOGCACHE   | 563        | 463          | 100          | 82
CATALOG        | 0          | 0            | 0            | (null)
```

This information is the average cache hitrate since the database has been started. To see the current cache hitrate, check the DBAnalyzer log file DBAN_CACHES.csv.

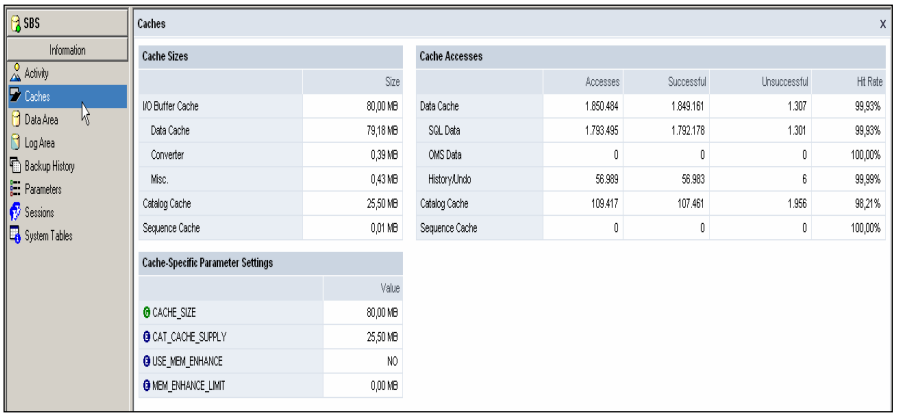
Cache Accesses

I/O buffer cache accesses

I/O buffer cache hit rate should be >= 98%

Catalog Cache accesses

Catalog Cache hit rate should be >= 90%



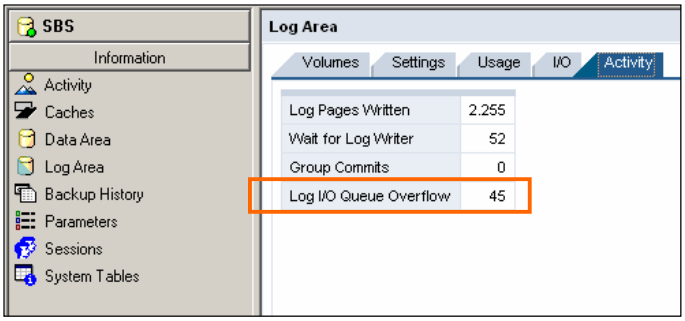
Note: Each “Unsuccessful” access to the I/O buffer cache (Data Cache) results in a physical I/O. Access to disk is at least 1000 times more expensive than access to cache.

A second possibility to monitor the I/O buffer cache hit rate is to use Information -> Caches. Here you can also see the Catalog Cache hit rate.

General recommendation: Create your I/O buffer cache as large as possible. Don’t waste available physical memory in the MaxDB server. Accessing data in a large I/O buffer cache is as expensive as accessing data in a small cache and may avoid the need of reading data from physical disks.

Catalog cache hit rate should be around 90%. The algorithm of internally increasing the catalog cache - up to the limit defined in parameter CAT_CACHE_SUPPLY - uses 90% as lower threshold.

Log Queue



The *log queue* temporarily stores log entries to be written to log disk. If the log queue is full, all update transactions must wait until log pages are written to disk and space in the log queue is released.

A full log queue is counted as "Queue overflow"

The Log queue may fill up due to

- **Many log entries written**
- **Slow log disks**
- **Misconfigured log queue (parameter LOG_IO_QUEUE)**

A log page must be written to disk, even if not filled, if a transaction performs a COMMIT. A COMMIT needs to wait until all changes performed in the transaction have been successfully stored in log volume.

If all space in the log queue is filled, no more update transactions can be performed, as they are not able to store their changed data in the log queue. This is very performance critical and may cause a blocking situation on application processes until the Log Writer task is able to free pages by writing them to the log volume.

Carefully monitor if "Log queue overflows" occur. In this case, check setting of parameter LOG_IO_QUEUE and the performance of the log volume.

Enable Detailed Time Measurement

C:\WINNT>dbmcli -d MYDB -u dbm,dbm

dbmcli on MYDB>db_cons time enable

OK

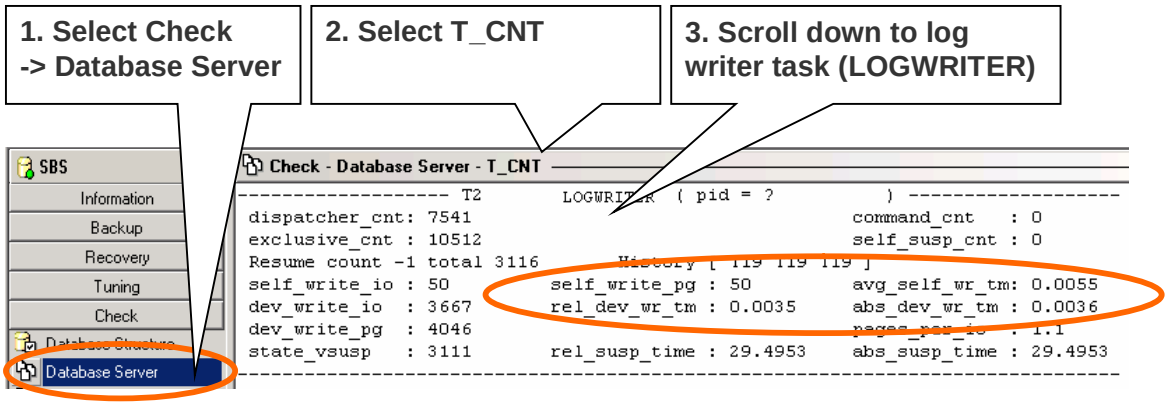
---

dbmcli on MYDB>

Prerequisite that MaxDB collects detailed time values like I/O times is the enabling of detailed time measurement.

To be able to monitor log I/O performance as well as other important performance values, it is necessary to explicitly enable a detailed time measurement in MaxDB. This time measurement should not be activated permanently.

Log I/O Time



To view the average log I/O time display details of the archive log writer task.

Average log I/O time should be well below 20ms

If detailed time measurement in MaxDB is enabled, the log I/O time can be obtained in DBMGUI -> Check – Database Server -> T_CNT.

Log I/O times in fast I/O sub systems are generally well below 5 ms. If log I/O times are around or even above 20 milliseconds, expect performance problems especially in applications with high update load.

(rel_dev_wr_tm * dev_write_io + avg_self_wr_tm * self_write_io) / (dev_write_io + self_write_io) = real average

rel_dev_wr_tm = write time of I/O performed by the Dev Thread (not including wait times caused by internal tasking)

abs_dev_wr_tm = write time of I/O performed by the Dev Thread (including wait times caused by internal tasking)

dev_write_io = number of I/O operations performed by the Dev Thread

avg_self_wr_tm = write time of I/O performed by the Log Writer task

self_write_io = number of I/O operations performed by the Log Writer task

SQL Locks

When a table record is locked exclusively by a transaction, other transactions can not lock the same row.

Exclusive locks are requested when the content of a table record is changed (INSERT, UPDATE, DELETE) or a SELECT … FOR UPDATE is performed.

The lock is released implicitly when the locking transaction performs a COMMIT. If the lock was requested by SELECT…FOR UPDATE, the transaction can release the lock explicitly without performing a COMMIT.

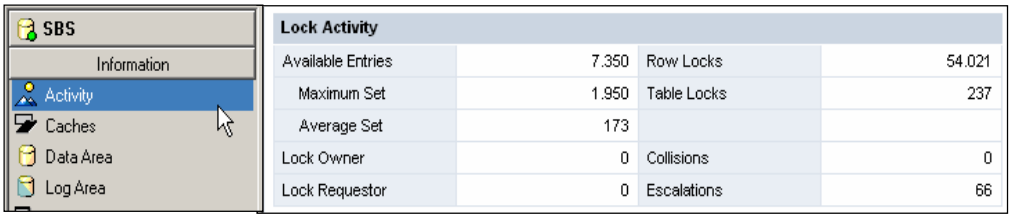
All blocked lock requesting transactions need to wait for the COMMIT or ROLLBACK of the lock holding transaction.

Lock Collisions and Lock Escalations

Collisions on SQL locks can **significantly reduce** the performance of update transactions.

The maximum number of locks on table rows can be configured by parameter ***MAXLOCKS***.

If the application locks a large number of rows of a table, single row locks will be transferred to a full table lock. This is called "**lock escalation**". Generally, lock escalations should be **avoided**, as they prohibit concurrent row locks on this table by other applications.



When data in database tables is changed, the according records will be locked. MaxDB uses a central lock list to administer all lock information.

The number of entries in this lock list is limited and is configured by parameter MAXLOCKS. If – system wide (!) - more than MAXLOCKS locks are required, applications may fail with error –1000 (Too many lock requests). Available Entries represents an internal value – for each lock several entries are created in the internal lock list. Therefore the value of Available Entries is higher than MAXLOCKS.

To avoid this situation, MaxDB tries to transform several single record locks to a full table lock if more of 10% of all configured lock entries are used by one transaction for one table. This is called lock escalation. A severe side effect of a lock escalation is that the table is exclusively locked by one transaction, and other transactions are not able to change any data in this table. As a consequence, those colliding transactions need to wait until the lock holding transaction performs a COMMIT. However, when other tasks are already holding locks on the same table, a lock escalation is not performed.

Lock escalations should be avoided. Increase MAXLOCKS if necessary. The space required in global memory for the lock list is approximately MAXLOCKS*200 bytes.

Corresponding DBMCLI command: dbmcli on MYDB>info locks

Example output:

```
OK
END
Name                 | Value

Max. Entries         = 7350

Average Used Entries = 173

Collisions           = 0

Escalations          = 66

Row Locks            = 54022

Table Locks          = 237

Request Timeout      = 5000
```

More information about lock situations can be found in the database analyzer log files.

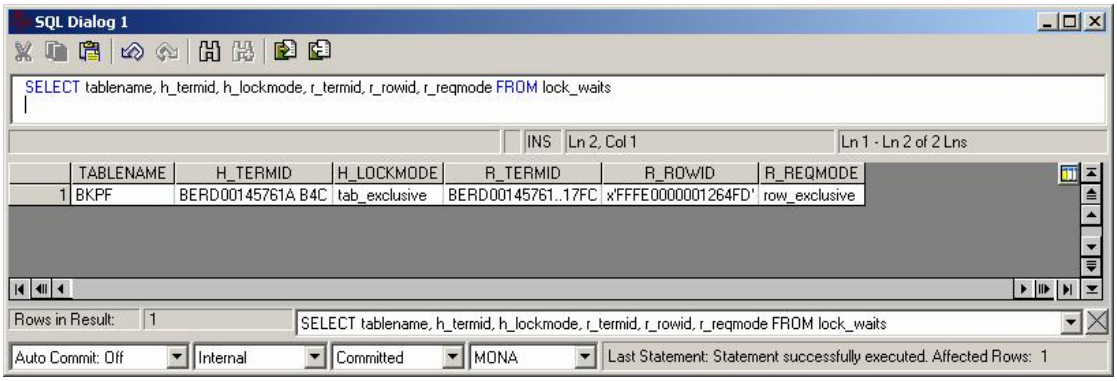
Identify the Locking Process

If a lock situation occurs, the process holding the lock can be identified.

Use the SQL statement '*SELECT * FROM lock_waits*' to display a list of all lock collisions.

For each collision, the lock holder and requestor is displayed.

Use *H_TERMID* to identify the process that holds the lock.



Several lock types exist. The most common are

- row_exclusive: a single row is locked exclusively. A transaction can lock several rows of a single table or different tables.
- tab_exclusive: a whole table is locked. No other transaction may lock this table or a row of this table.

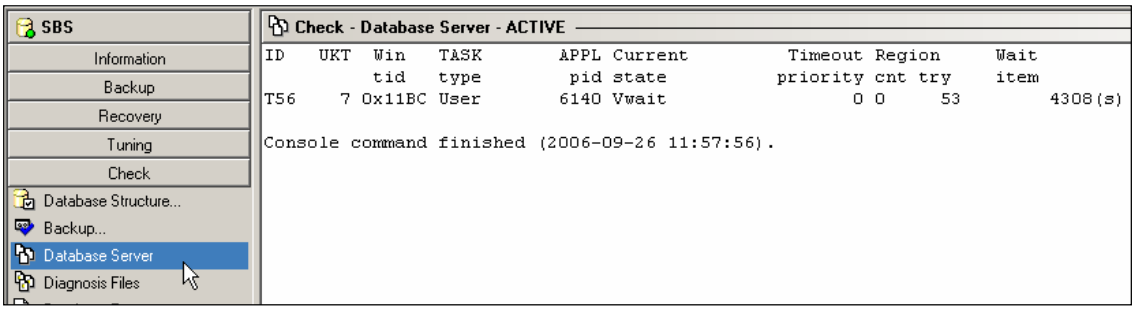
If a transaction has locked a row in share mode, other transactions can also lock this row in share mode, but no transaction can obtain an exclusive lock on this row.

```
sqlcli MYDB=>SELECT tablename, h_termid, h_lockmode, r_termid, r_rowid, r_reqmode FROM
lock_waits

| TABLENAME  | H_TERMID           | H_LOCKMODE    | R_TERMID           | R_ROWID
| R_REQMODE      |

| ---------- | ------------------ | ------------- | ------------------ | -------------
------- | -------------- |

| BKPF       | BERD00145761A B4C  | tab_exclusive | BERD00145761..17FC | x'FFFE0000001
264FD' | row_exclusive  |
```

In the example, the transaction running in the process with PID 0xB4C (=2892 in the Windows Task Manager) holds an exclusive table lock on table BKPF, while another transaction (PID 0x17FC) tries to lock a row of the same table exclusively. This transaction has to wait until process PID 0xB4C performs a COMMIT or ROLLBACK.

Remember that table locks can be triggered automatically by MaxDB, if a transaction requests a high number of row locks on a table. Check for Lock escalations and if those occur, think about increasing parameter MAXLOCKS.

Monitoring Active Task Status 1/2



Check -> Database Server -> ACTIVE **is the internal state of all currently *active* MaxDB tasks. User tasks that are currently waiting for a command from an application process are not displayed.**

DBMGUI -> Check -> Server -> Active displays information about the states of MaxDB tasks that are currently working. In a running system, possible states are:

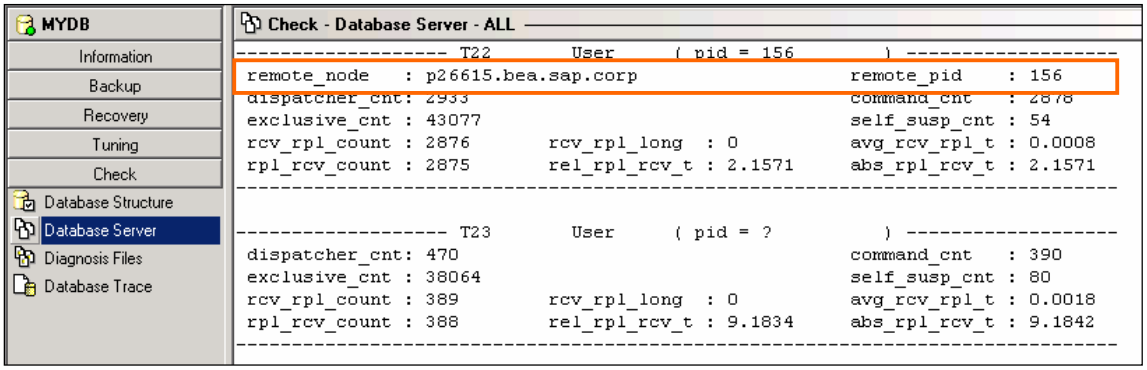
- Running - task is in kernel code of MaxDB and uses CPU
- Runnable, Vsleep - task is in kernel code of MaxDB and waiting for a free slot in its thread (UKT)
- LogIOWait – tasks waits for completion of its log request by archive log writer
- IOWait (R) or IOWait (W) - task waiting for I/O completion (read or write)
- Vbegexcl or Vsuspend - task waiting to acquire an internal lock in MaxDB
- Vwait - task waiting for an SQL lock held by another application process to be released (locks are released after a COMMIT or ROLLBACK)

If the sum of tasks in states Running, Runnable and Vsleep is higher than the number of CPUs on the database server for a long time, there may be a CPU bottleneck. Before you increase the number of CPUs, you may need to analyze the application and its SQL statements in detail.

Ideally, user tasks should have state Running.

- If user tasks often have state Vbegexcl, performance may suffer from serialized access to internal MaxDB locks.
  - In MaxDB, internal locks are assigned by regions (corresponding to latches in Oracle). Regions are used to synchronize parallel accesses to shared resources. For instance, searching for a page in the I/O buffer cache is done by region. In each region, only one task at a time can search for a page.
- If user tasks often have state Vwait, the application seems to serialize on SQL locks. Mostly the reason for this behavior can only be found through analyzing the application.
- In case of remote access to the database the APPL pid is marked with an asterisk.

Monitoring Active Task Status 2/2



The PID of the application process associated to the user task and the application server is displayed.

In case of remote access to the database, remote_node is the server name of the connected application and remote_pid the process ID of the connected application process.

In case of local communication these lines are missing.

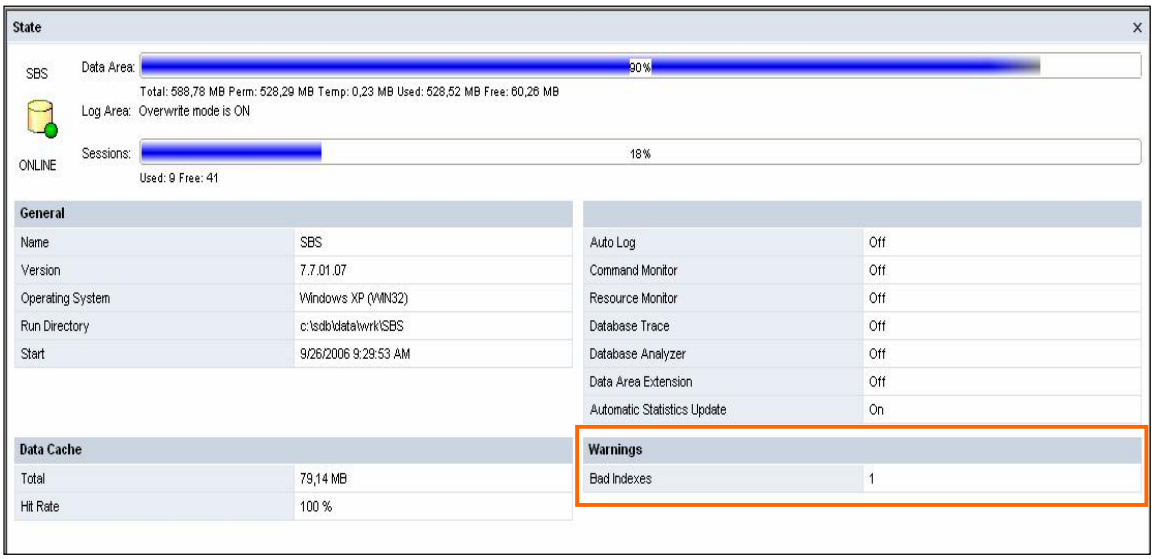
Indexes

Indexes are used to optimize the access path to the data

Performance problems can occur if

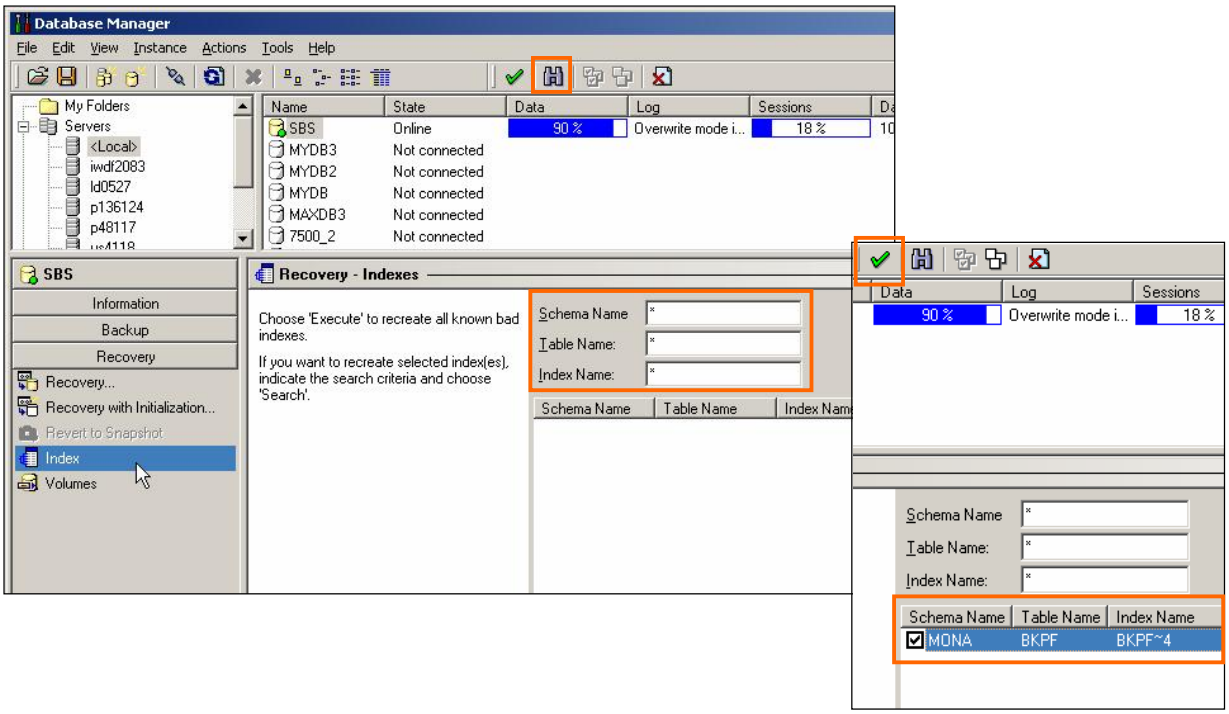
- Indexes are corrupted
- Indexes are disabled
- Appropriate indexes don’t exist

Corrupted Indexes 1/2



If corrupted indexes exist in the database, a warning is shown on the main information screen of DBMGUI.

Corrupted Indexes 2/2



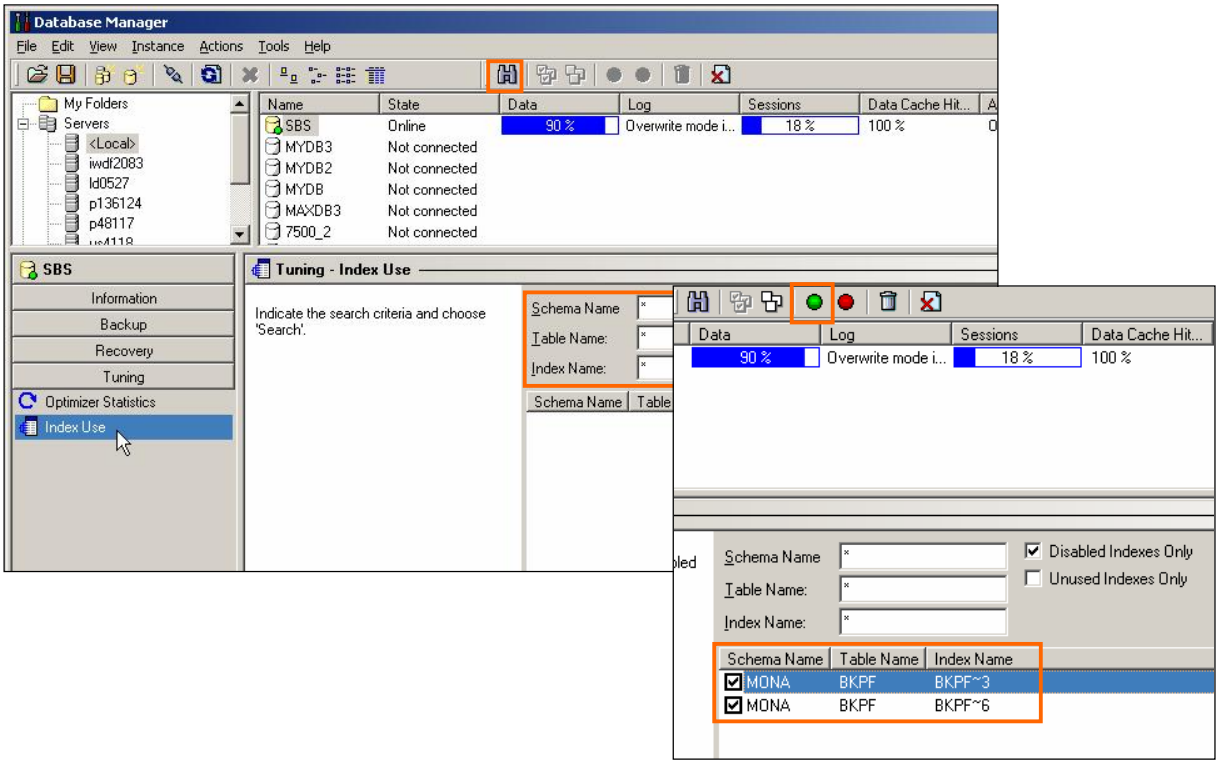
To correct this problem, you have to recreate the corresponding indexes.

Select Recovery -> Index and search for corrupted indexes.

Then mark the corrupted indexes and choose Recreate.

Attention: During the index recreation the corresponding table is locked for write transactions.

Disabled Indexes



Indexes can be disabled to check which strategy the optimizer would use, if the index did not exist.

However, it might be that a disabled index decreases the database performance. To check if an Index is disabled, choose Tuning -> Index Use and search for disabled indexes.

To enable the index usage again, mark the index and press the button with the green traffic light.

Database Analyzer

Rule-based expert system to monitor MaxDB instances

Collects statistical and monitoring data

Collects system messages

Supports remote access

Detects and reports

- Low cache hit rates
- High I/O load
- Low hit rates of DML commands (SELECT, UPDATE, DELETE)
- Log queue overflows
- Lock collisions
- Long running commands
- Long duration and high frequency of system locks

The Database Analyzer is a tool for MaxDB performance analysis. Database Analyzer is a separate executable which allows the analysis of historical MaxDB data. For the most part the collected monitoring data can only be interpreted by the MaxDB development support.

The components of the Database Analyzer are the dbanalyzer executable and the configuration file dbanalyzer.cfg. The executable dbanalyzer collects and logs the monitoring data and interprets the rules provided in dbanalyzer.cfg. The executable is release independent.

The second component of Database Analyzer, the configuration file dbanalyzer.cfg, contains information about the monitored data. Only MaxDB system tables can be used as data source.

For the adoption to new MaxDB releases it is only necessary to adapt the configuration file because the accesses to system tables are defined there. Therefore the configuration file is release dependent and is delivered automatically with MaxDB patches, if the classification rules or monitoring parameters changed between these MaxDB patches.

Starting the Database Analyzer

Starting the Database Analyzer with DBMCLI:

dbmcli –d <db_name> -u <dbm_user>,<password> dban_start <options>

Stopping the Database Analyzer with DBMCLI:

dbmcli –d <db_name> -u <dbm_user>,<password> dban_stop

Getting the state of Database Analyzer with DBMCLI:

dbmcli –d <db_name> -u <dbm_user>,<password> dban_state

Call in UNIX/Linux or DOS shell:
dbanalyzer [-n <server>] -d <database> -u <user,pwd> [-f <configfile>]
-t <interval>[,<number>] [-o <outputdir>] [-c <level>] [-i] [-nologo] [-noenv] [-stop]

Options:

-n <server>
Name of the server on which the database instance is running. If you specify the server name, you must also specify the directory for the protocol files with -o.

-d <database>
Name of the database instance

-u <user,pwd>
User name and password

-f <configfile>
Name of the configuration file. If you do not specify a configuration file, the Database Analyzer uses the file dbanalyzer.cfg in the $INSTROOT/env directory.

-t <interval>,<number>
Time interval (in seconds) between two evaluations and number of desired evaluations. If you specify a value for <number>, the system terminates the Database Analyzer after the corresponding number of evaluations.

-o <outputdir>
Directory for the protocol files. If you do not specify a directory, the Database Analyzer uses the sub directory analyzer in the working directory of the database instance.

-c <outputlevel>
Specifies on a scale of 1 (low) to 4 (high) how much additional information should appear on the console. If you do not specify -c, no output appears on the console.

-i
Deletes protocol files that already exist

-stop
Stops a running Database Analyzer

Log Files

The Database Analyzer log files allow you to monitor the system behaviour:

All data of one day is stored in one directory

Data is stored in separate files, sorted by topic

Display using Microsoft Excel

The Database Analyzer rates the information and bottlenecks:

I: Information = General information, such as the number of executed commands

W1 to W3: Warning levels 1 to 3 = Bottleneck warnings with low, medium, and high priority

DBAN.prt : Logs messages, including analysis rules and current values that caused the message

DBAN_BACKUP.csv: Number of Pages written during backup and I/O times

DBAN_CACHES.csv: Successful and unsuccessful accesses to the caches and hit rates

DBAN_CPU_UTILIZATION.csv: CPU utilization on the database server.

DBAN_FILLING.csv: Fill level of the database instance (such as the size of the data volumes, number of permanently and temporarily used pages)

DBAN_GC.csv: Information about the used history pages

DBAN_IO.csv: Reads and writes to cache pages and data pages

DBAN_IOTHREADS.csv: Reads and writes by io threads

DBAN_LOAD.csv: Accesses and selectivity of SELECT, FETCH, INSERT, UPDATE, and DELETE statements

DBAN_LOGGING.csv: Number of log pages written, fill level of the log queue

DBAN_OVERVIEW.csv: Information about suspends, physical reads, region accesses

...

To be continued on next page

Note continuation page

See notes below

Logfiles Continued:

DBAN_RW_LOCKS.csv: Number of rw locks, number of collisions on rw locks

DBAN_SHARED_SQL.csv: Statsistics about Shared SQL

DBAN_SPINLOCKS.csv: Statistics about spinlock usage

DBAN_STRATEGY_INDEX.csv: Accesses and selectivity for index searches

DBAN_STRATEGY_PRIMKEY.csv: Accesses and selectivity for PRIMARY KEY searches

DBAN_STRATEGY_SCANS.csv: Accesses and selectivity for INDEX SCAN and TABLE SCAN searches

DBAN_TASK_ACTIVITIES.csv: Information on task activities (such as number of SQL statements, number or running user tasks, number of task changes)

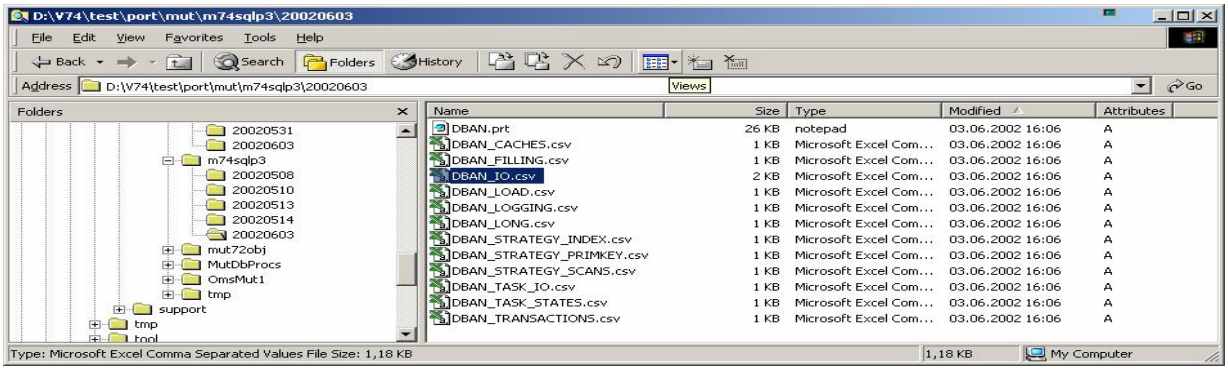
DBAN_TASK_IO.csv: Number and duration of physical writes and reads from perspective of the log writer, the user task, and the pager

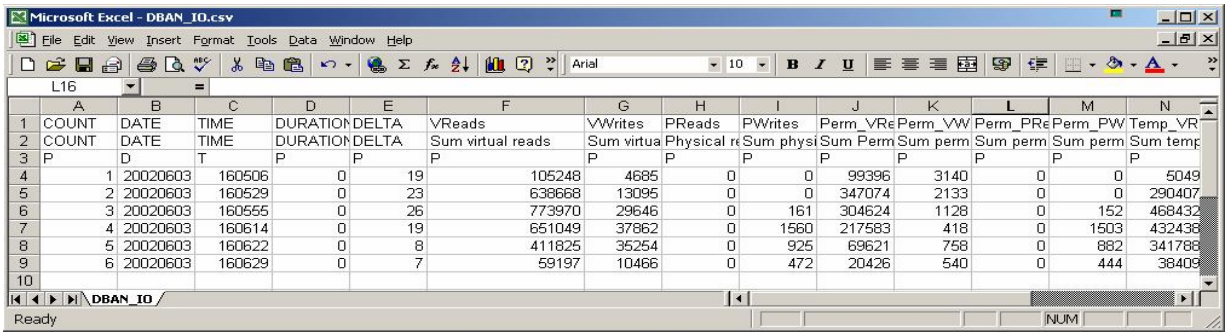
DBAN_TASK_STATES.csv: Number and duration of processed statements. Number and wait duration of the task states Vsuspend, Vwait, Vsleep

DBAN_TRANSACTIONS.csv: Number of SQL statements and procedure calls, PREPARE, EXECUTE, COMMIT, and ROLLBACK statements, subtransactions, TIMEOUTS caused by locks, and conversions of row locks to table locks (Lock List Escalations)

Displaying Log Files

Displaying a log file in Microsoft Excel



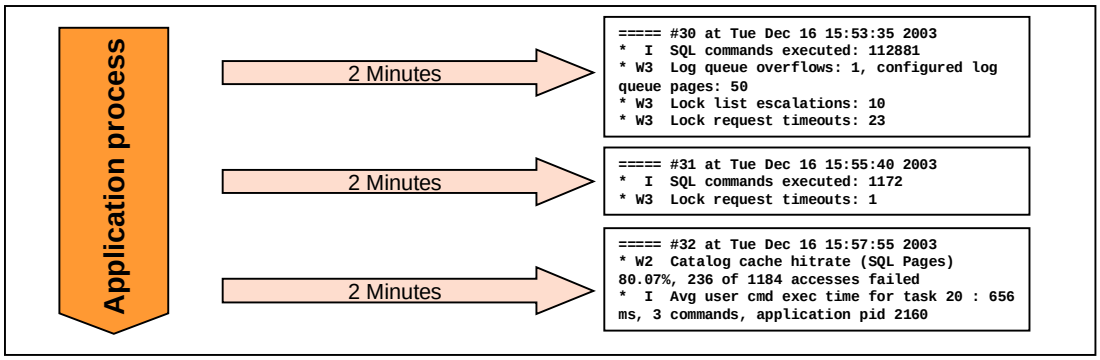


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	COUNT	DATE	TIME	DURATION	DELTA	VReads	VWrites	PReads	PWrites	Perm_VRe	Perm_VW	Perm_PRe	Perm_PW	Temp_VR
2	COUNT	DATE	TIME	DURATION	DELTA	Sum virtual reads	Sum virtua	Physical re	Sum physi	Sum Perm	Sum perm	Sum perm	Sum perm	Sum temp
3	P	D	T	P	P	P	P	P	P	P	P	P	P	P
4	1	20020603	160506	0	19	105248	4685	0	0	99396	3140	0	0	5049
5	2	20020603	160529	0	23	638668	13095	0	0	347074	2133	0	0	290407
6	3	20020603	160555	0	26	773970	29646	0	161	304624	1128	0	152	468432
7	4	20020603	160614	0	19	651049	37862	0	1560	217583	418	0	1503	432438
8	5	20020603	160622	0	8	411825	35254	0	925	69621	758	0	882	341788
9	6	20020603	160629	0	7	59197	10466	0	472	20426	540	0	444	38409
10														

Short Time Analysis

To use Database Analyzer for a more detailed analysis of a short time period (e.g. during the execution of an expensive application), the time interval of data collection should be reduced to 60 or 120 seconds:

dbmcli -u <SYSDBA>,<password> -d <db_name> dban_start -t 120



Expert Analysis: Important Values

CACHES

- DC_Hit: I/O buffer cache hit rate

TASK_STATES

- CmdExecTimeAvg: Average execution time for an SQL statement
- VwaitTimeAvg: Average waiting time for SQL locks

TASK_IO

- AvgRTime_UserPTask, AvgRTime_UserPThread: Average read I/O times on data volumes
- AvgWTime_LogPTask, AvgWTime_LogPThread: Average write I/O times on log volumes

FILLING

- DB_Size: Database size in 8KB pages
- Perm_Used: Data volume filling in 8KB pages (permanent pages)
- DB_Filling: Data volume filling in percent

The *.csv files contain all data collected by Database Analyzer. While several values are useful for MaxDB administrators, other values are intended to be used mainly by MaxDB developers and development supporters.

Detailed historic data provides the opportunity to reconstruct what happened in MaxDB during times of interest. For example the exact I/O buffer cache hit rates, filling of I/O buffer cache, collision rates etc. can be displayed. Remember that all cache hit rates are individually re-calculated for the according time interval – these are not the cache hit rates displayed in DBMGUI that show the average hit rates since MaxDB restart.

Bottlenecks

With the Database Analyzer we head for identifying the bottlenecks that might be responsible for a poor MaxDB performance.

Some of these bottlenecks may be solved easily by changing MaxDB parameter settings or changing the MaxDB configuration.

Some bottlenecks can only be solved by a deeper analysis of the executed application programs.

Some bottlenecks may only be a temporary effect during complex executions or concurrent processes, especially in large multiprocessor environments.

The interpretation of the Database Analyzer results need routine and experience. Not all displayed bottlenecks really cause a measurable decrease of MaxDB performance. They just show that MaxDB doesn't run as smoothly as expected.

Database Analyzer becomes important if users complain about poor response times – especially if the response times were fine before. Then the comparison of current Database Analyzer results with former ones may give important hints for the analysis.

The following pages show some of the most important warnings that you may see in the Database Analyzer logs.

I/O buffer cache Hitrate

Wn I/O buffer cache hitrate <percent>% by <count> accesses, <count> failed

Access to MaxDB data causes physical I/O (failed) as the data is not present in the according caches.

I/O buffer cache hit rate should be above 98% – at least for longer time intervals.

Example

* W3	Data cache hitrate (SQL Pages) 79.93% by 97447 accesses, 19561 failed

The average I/O buffer cache hit rate data should be above 98%, meaning less than 2 physical reads per 100 data accesses. If the I/O buffer cache hit rate becomes lower, expect a significant decrease of MaxDB performance. Short periods of lower hit rate are uncritical, as they may happen e.g. when data of a table that was not accessed for a long time needs to be (re-)loaded into I/O buffer cache.

If the Database Analyzer shows warnings about bad data cache hit rate during several intervals, check I/O buffer cache usage and try to identify the underlying reason:

- Too small sized I/O buffer cache.
- Unfavorable execution of SQL statements so that a large amount of data needs to be read (e.g. full table scans)

Expensive Select Statements

Wn <count> selects and fetches, selectivity <percent>% -> rows read/qual <count> / <count>

Wn <count> <access strategy>, selectivity <percent>% -> rows read/qual <count> / <count>

Access strategies to data in relational SQL tables is bad because a high number of table rows has to be read internally to find a small number of rows that meets the qualification in the WHERE clause.

Try to find the SQL statements that cause the bad access strategy
(-> Command Monitor, -> Resource Monitor)

Example

*W3	15844 selects and fetches, selectivity 0.29% -> rows read/qual 1584352 / 4522
*W3	21 primary key accesses, selectivity 0.00% -> rows read/qual 1422622 / 12

Accesses to relational tables should use an optimized access strategy, e.g. with help of secondary indexes. If e.g. an index is missing or if the table statistics, used by the cost based query optimizer, are outdated, the access strategy may not be optimal. Then the ratio between internally read database records and the result records may be bad, causing an unnecessary high workload on the MaxDB server.

Depending on the number of effected rows (rows read), expect more or less severe side effects on general performance.

Check if the optimizer statistics are up to date and update statistics if necessary.

Cooperate with the application developer to solve such problems.

Expensive Delete Statements

Wn <count> deletes, selectivity <percent>% -> rows read/qual <count> / <count>

Access strategies to data in a delete statement is bad because a high number of table rows has to be read internally to find the rows to be deleted that meet the qualification in the WHERE clause.

To identify the expensive delete statements, use additional tools like Command Monitor or Resource Monitor.

Example

* W3	9 deletes, selectivity 0.00% -> rows read/qual 306880 / 6

The access strategy for DELETE statements is optimized the same way as for SELECT statements. The number of internal database accesses to identify the rows to be deleted should be minimized.

If e.g. an index is missing or if the table statistics, used by the cost based query optimizer, are outdated, the access strategy may not be optimal.

Depending on the number of effected rows (rows read), expect more or less severe side effects on general performance.

Cooperate with the application developer to solve such problems.

Critical Regions

Wn <region-name>: count> collisions (<percent>%) by <count> accesses on region <region-ID>

The collision rate on internal MaxDB locks (critical regions) is high and may cause serialization on accesses to these locks.

Potential reasons:

- MaxDB server is CPU bound
- MaxDB server is paging on OS level

Example

* W1	FBM: collision rate 5.89%, 689 collisions, 3 waits (0.03%), 11706 accesses on region 10
* W2	LOG: collision rate 10.49%, 1536 collisions, 13 waits (0.09%), 14638 accesses on region 18

MaxDB uses critical regions to protect access to internal data structures (I/O buffer cache administration, catalog access etc.) against concurrently active user tasks. Generally critical regions are held only for a very short time (<< 1 microsecond) to reduce the risk of collisions. If the MaxDB server becomes CPU bound, the operating system may dispatch a MaxDB thread that currently holds a critical region. In this case the chance will increase that other threads will collide on the held region. Therefore high collision rates are typical for heavy workload on MaxDB server (CPU, paging).

Check the OS workload.

Command Execution Time

Wn User cmd exec time for T<task-ID> : <duration> ms by <count> commands, appl. pid <PID of MaxDB work process>

The average execution time of commands, executed by a specific MaxDB user task, is very high. Each of the commands has taken the displayed time in average.

Example

* W3	User cmd exec time for T394 : 3410.97 ms by 3 commands, appl. pid 70412
* W3	User cmd exec time for T396 : 3201.52 ms by 3 commands, appl. pid 360078

In the example above, task T394 executed three commands with an average execution time of 3.4 seconds.

This is time critical. In typical dialog oriented environments, the average execution time should not exceed 15 milliseconds. For complex statements, individual execution times can be much higher.

Check which kind of application was active when the high execution time occurred.

Logging

Wn Log queue overflows: <count>

Wn max Log queue filling: <percent> %

The log queue is full. As a result, update transactions must wait until the log writer has successfully transferred log pages to disk and released space in log buffer.

Example

* W3	Log queue overflows: 362
* W3	max Log queue filling: 100 %

If the size of the log queue buffer, that holds the log entries before they are written to the log volume by the Archive Log Writer, is not large enough for the current log volume, concurrent update transactions may temporarily get stuck.

Check the size of the configured log queue (parameter LOG_IO_QUEUE) and increase it if possible. Maximum size is 2000 pages.

Additionally check the performance of physical disks used for log volumes. For log volumes, the fastest available disks should be used, as log has to be written synchronously when a transaction commits - the transaction always has to wait until its log has successfully be written to the log volume. If the committing transaction keeps locks on MaxDB resources, other transaction may collide on these locks and must wait.

The first time when the maximum log queue filling level reaches 100%, Database Analyzer will show a warning.

Further Optimization

If SQL statements are not executed using the best optimizer strategy this may lead to general performance problems.

To analyze the execution plans of SQL statements you have to cooperate with the application developer.

The tools Command Monitor and Resource Monitor should be used to analyze such problems.

Impact of Bad SQL Statements

One major reason for an unsatisfying database performance are SQL statements that are executed in an insufficient way.

Reasons can be:

- Missing indexes for table columns.
- Bad programming: Omit known conditions in SQL statements.
- Insufficient input into input fields by users.

Unfavorable executions of SQL statement result in high execution times of particular application transactions.

As a side effect, unfavorable executed SQL statements may have a severe impact on overall database performance, as they cause a high additional workload (CPU usage, I/O, displacement of data page from I/O buffer cache).

The potential reasons for a bad database performance are various, but often SQL statements that are not executed in an optimal way have a larger impact on transaction and general performance than a small I/O buffer cache, slow disks etc.

If an adequate index is missing for table access, the cost of statement execution easily can be factor 1000 or higher above the optimal costs. Such expensive executions do not only influence the specific application that executes the SQL statement, but also other concurrent transactions, if thousands or even millions of data pages must be read from disk into I/O buffer cache and cause a replacement of other data from I/O buffer cache.

A potential reason is that the end user doesn't use the application as designed, e.g. does not specify values for all essential input fields.

In any case, these long running statements must be identified and the reason for their bad performance must be eliminated, e.g. by

- changing application program
- creating secondary index(es)
- teaching end user
- …

Instruments to Monitor SQL Statements

SQL statements may cause a high workload, because they are

- cheap in individual execution, but executed frequently
- expensive in individual execution

MaxDB provides two different instruments to monitor SQL statements:

- Resource Monitor
  - Collects data of all SQL statements
  - Accumulates runtime values (time, buffer accesses, rows read internally etc.)
- Command Monitor
  - Uses configurable thresholds to catch particularly long running statements
  - Collects input data from the application
  - Shows execution plan

Statements can be “expensive” in two different ways:

- The individual execution is expensive, as e.g. a secondary index is missing. In many cases those statements can be optimized with database methods.
- The individual execution is cheap (e.g. trough primary key access), but the statement is executed very often and therefore causes a high aggregated runtime and workload. Mostly those statements cannot be optimized with database methods because they are already executed in the most efficient way.

Both kinds of expensive statements must be identified, but the method of solving the identified bottlenecks may be totally different.

MaxDB provides two different instruments to deal with different kinds of expensive SQL statements

- The Resource Monitor aggregates the resource consumption over all executions of a statement. It helps to identify the SQL statements that cause the highest workload on the database and whose optimization promises the highest overall effect. The data for the execution plan (EXPLAIN) is not collected.
- The Command Monitor collects specific data about SQL statements whose resource consumption violates configurable thresholds like runtime, page accesses or selectivity. The command monitor is mainly used to catch statements with high individual runtime. The command monitor also collects the exact user input data used during statement execution. This is essential to create the correct execution plan (EXPLAIN) used for statement execution.

Resource Monitor 1/2

Use this command to start/stop the logging of the executed commands:

sqlcli -d <db_name> -u <sql_user>,<password> diagnose analyze on | off

When a command is parsed, it is entered in the system table SYSCMD_ANALYZE and a command ID is generated.

The logging of the resource usage has to be switched on/off separately:

sqlcli -d <db_name> -u <sql_user>,<password> diagnose analyze count on|off

The resource usage values are stored in table SYSDATA_ANALYZE – for each session aggregated on the basis of the command ID.

- The stored data can be deleted:
  sqlcli -d <db_name> -u <sap_user>,<password> diagnose analyze CLEAR COMMAND/DATA/ALL

Resource Monitor 2/2

To analyze the resource monitor data you have to select the data from table SYSDATA_ANALYZE.

The data is aggregated per session. If you want to know the totals over all sessions you have to accumulate the values manually.

Then you have to join these totals with the data from table SYSCMD_ANALYZE.

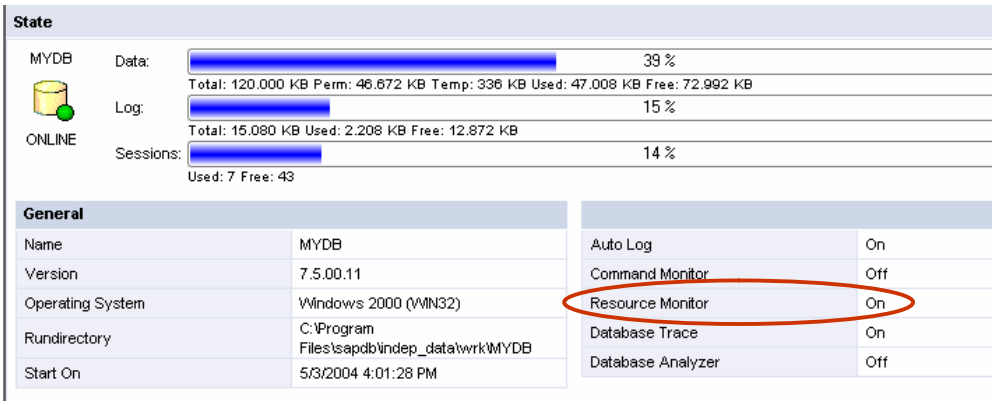

```
create table temp.my$sysdata_analyze as

  select
    t1.cmdid, sum(t1.call_count) call_count,
    sum(t1.rows_read) rows_read, sum(t1.rows_qual) rows_qual,
    sum(t1.virtual_reads) virtual_reads, sum(t1.physical_io) physical_io,
    sum(t1.rows_fetched) rows_fetched, sum(t1.runtime) runtime,
    min(t1.min_runtime) min_runtime, max(t1.max_runtime) max_runtime,
    sum(t1.vwaits) vwaits, sum(t1.vsuspends) vsuspends
  from sysdata_analyze t1
  group by t1.cmdid

select
  rawtohex(t1.cmdid) cmdid, t1.call_count,
  t1.rows_read, t1.rows_qual, t1.virtual_reads, t1.physical_io,
  t1.rows_fetched, t1.runtime, t1.min_runtime, t1.max_runtime,
  t1.vwaits, t1.vsuspends, t2.job, t2.line,
  substr(t2.sql_statement,1,2000), substr(t2.sql_statement,2001,1700)
from temp.my$sysdata_analyze t1, syscmd_analyze t2
where
  t1.cmdid = t2.cmdid
  order by t1.runtime desc
```

If you want to check the data later again (that is the temporary table does already exist), you have to delete the content of the temporary table my$sysdata_analyze and to insert the current data:

```
  delete from temp.my$sysdata_analyze

  insert into temp.my$sysdata_analyze select ...
```

Checking Resource Monitor State

To check if the Resource Monitor is running, have a look at the Database Manager GUI:



- We are currently working on the implementation of the new resource monitor using the information provided by Shared SQL. When this is finished, the resource monitor will always be collecting the required information – you don‘t have to activate it manually anymore. Then the informtaion is stored in table commandcachestatistics.

Command Monitor

Use this command to start the command monitor:

sqlcli -d <db_name> -u <sql_user>,<password> diagnose monitor selectivity <number>

| read <number>

| time <number>

| rowno <number>

| data on

Deactivate the command monitor using this command:

sqlcli –d <db_name> -u <sql_user>,<password> diagnose monitor [selectivity | read | time | rowno | data] off

read: A SQL statement is logged, if the specified number of page accesses is exceeded.

time: A SQL statement is logged, if the specified runtime (in seconds) is exceeded.

selectivity: A SQL statement is logged, if the ratio between qualified and read rows is lower than the specified value per thousand.

data on: Always specify this option, that the command parameters are stored. This is necessary to be able to perform an EXPLAIN command for the SQL statement.

rowno: If the specified number of rows in table SYSMONITOR is reached the entries are overwritten.

If the command monitor is activated, SQL statements which exceed the specified values are stored in the system tables SYSMONITOR and SYSPARSEID. Statements which were already running when the command monitor was activated are not logged. If option DATA ON was specified, the statement parameters are stored in system table SYSMONDATA.

To delete the monitor data use this command:

sqlcli –d <db_name> -u <sql_user>,<password> diagnose monitor clear

Checking Command Monitor State

For a performance analysis first check if the monitoring tools are active.

SELECT parametername, value FROM activeconfiguration WHERE parametername LIKE 'MONITOR%'

PARAMETERNAME	VALUE
MONITOR_READ	1000
MONITOR_TIME	2147483647
MONITOR_SELECTIVITY	10
MONITOR_ROWNO	0

monitor_read / monitor_selectivity / monitor_time: shows the values with which the command monitor was started.

monitor_rowno: max. number of statements in table SYSMONITOR

If the command monitor is not active, these entries are set to default values: monitor_read : 2147483647, monitor_time: 2147483647, monitor_selectivity: 0, monitor_rowno: 0.

Analyzing Command Monitor Data 1/2

To analyze the command monitor data you have to select the data of the system tables: SYSPARSEID and SYSMONITOR:

SELECT /+ordered*/ sysk, rows_read, rows_qual, strategy, runtime, physical_io, sql_statement*

FROM sysmonitor, sysparseid

WHERE sysmonitor.parseid = sysparseid.parseid

ORDER BY runtime DESC

Using the command monitor you can identify long running statements. As the number of logged statements is limited you cannot use this monitor for a long time analysis. You should enter reasonable threshold values that only really expensive SQL statements are logged.

To identify the long running statements select the data from tables SYSMONITOR and SYSPARSEID:

```
SELECT /*+ordered*/ sysk, rows_read, rows_qual, strategy,
runtime, physical_io, substr(sql_statement,1,550) FROM
sysmonitor t1 INNER JOIN sysparseid t2 ON t1.parseid =
t2.parseid

|SYSK               |ROWS_READ|ROWS_QUAL|STRATEGY
|RUNTIME|PHYSICAL_IO|EXPRESSION1                                |

|-------------------|---------|---------|--------------
|-------|-----------|-----------------------------------------|

|0x0000000000000001|       40|       40|SCAN SCAN SCAN
|       0|          0| SELECT * FROM bkpf WHERE mandt = ? AND
bukrs = ? AND belnr > ?      |
```

Analyzing Command Monitor Data 2/2

If you want to execute the EXPLAIN statement for a logged statement and the statement contains parameters, you have to select the parameter data from table SYSMONDATA:

*SELECT **

FROM sysmondata

WHERE sysk = <sysk>


```
SELECT * FROM  sysmondata WHERE sysk = x'0000000000000001'
| SYSK               | PARAMNO | DATA_TYPE | DATA       |
| ------------------ | ------- | --------- | ---------- |
| 0x0000000000000013 | 1       | CHAR      | 811        |
| 0x0000000000000013 | 2       | CHAR      | 2100       |
| 0x0000000000000013 | 3       | CHAR      | 0000000016 |
```

Getting the Execution Plan

After selecting the parameter data from table SYSMONDATA you can execute the EXPLAIN statement:

*EXPLAIN SELECT **

FROM bkpf

WHERE mandt = <value1> AND
bukrs = <value2> AND
belnr > <value3>


```
EXPLAIN SELECT * FROM bkpf WHERE mandt = '811' AND bukrs = '2100' AND belnr >
'0000000016‘

|SCHEMA | TABLENAME | COLUMN_OR_INDEX | STRATEGY                    | PAGECOUNT
|

|-------|-----------|-----------------|-----------------------------|----------
--|

|MONA   | BKPF      | BKPF~2          | RANGE CONDITION FOR INDEX | 2607
|

|       |           | MANDT           | (USED INDEX COLUMN)       |
|

|       |           | BUKRS           | (USED INDEX COLUMN)       |
|

|       |           |   RESULT IS NOT COPIED , COSTVALUE IS       | 89
|
```

MaxDB Traces

MaxDB provides two different traces:

- the database trace called knltrace
- the SQLDBC trace called sqldbctrace-<pid>.prt

The database trace can be activated using DBMGUI or dbmcli, the SQLDBC trace can be activated using sqldbc_cons.

Database Trace

The kernel trace is used to analyze executed SQL statements and database errors.

When the kernel trace is switched on, you have to specify which kernel component should write trace information. In general the default option is sufficient.

The trace information is written into a memory area. This memory area has to be flushed to file knltrace before the trace can be evaluated.

When the trace is evaluated (using dbm command trace_prot) you have to specify which parts should be evaluated – most times abkmx is sufficient (if not requested otherwise).

As a default the kernel trace is switched off. The influence of the default kernel trace on the system performance is very small. Other trace options might decrease the system performance and should only be used when needed for an analysis (requested by the developer).

It is possible to switch on the trace for a specific user session. This is helpful if too many concurrent users are active on the system – otherwise the important trace information could be overwritten.

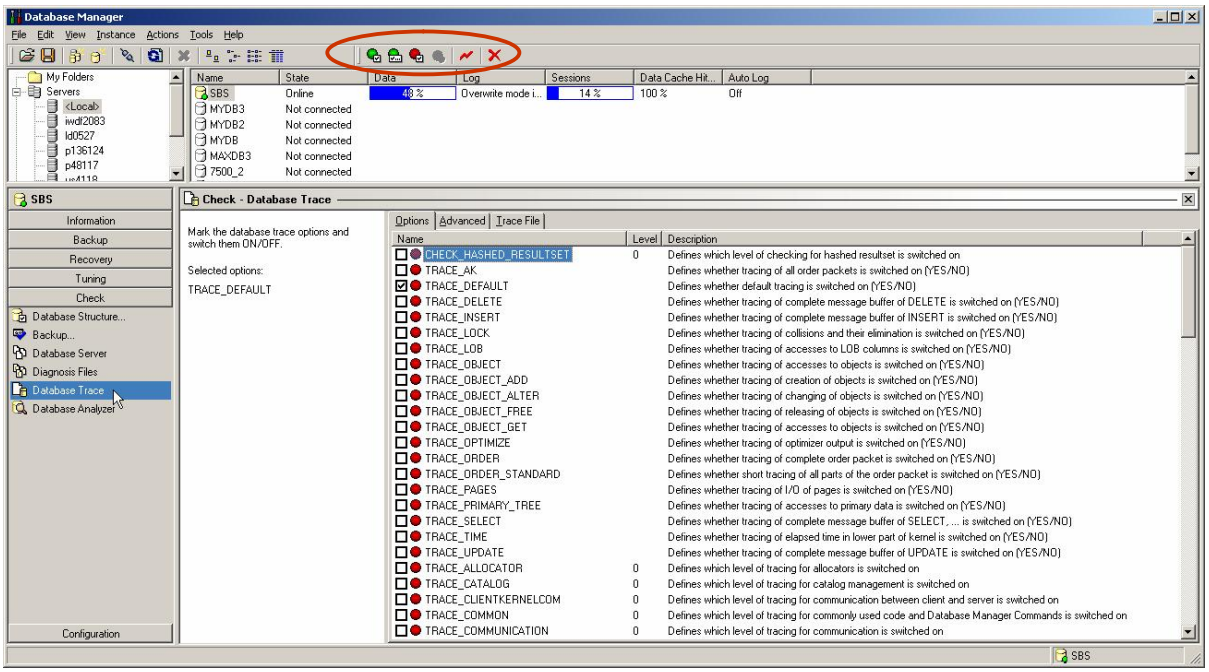
Option STOP ON ERROR allows to deactivate the trace automatically in case a specific error occured. This also prevents that the important trace information is overwritten.

Database Trace

When you are requested to create a kernel trace, proceed as follows:

1. Activate the kernel trace
2. Reproduce the problem (if possible when no other users are active, otherwise try to create a session specific trace)
3. Flush the trace information to disk
4. Deactivate the kernel trace
5. Convert the trace file into ASCII

Activating the Kernel Trace



You can use DBMGUI or DBMCLI to activate, flush, deactivate and evaluate the kernel trace. Even SQL statements in SQL Studio can be used.

The DEFAULT option is sufficient, if the developer did not request any special trace option.

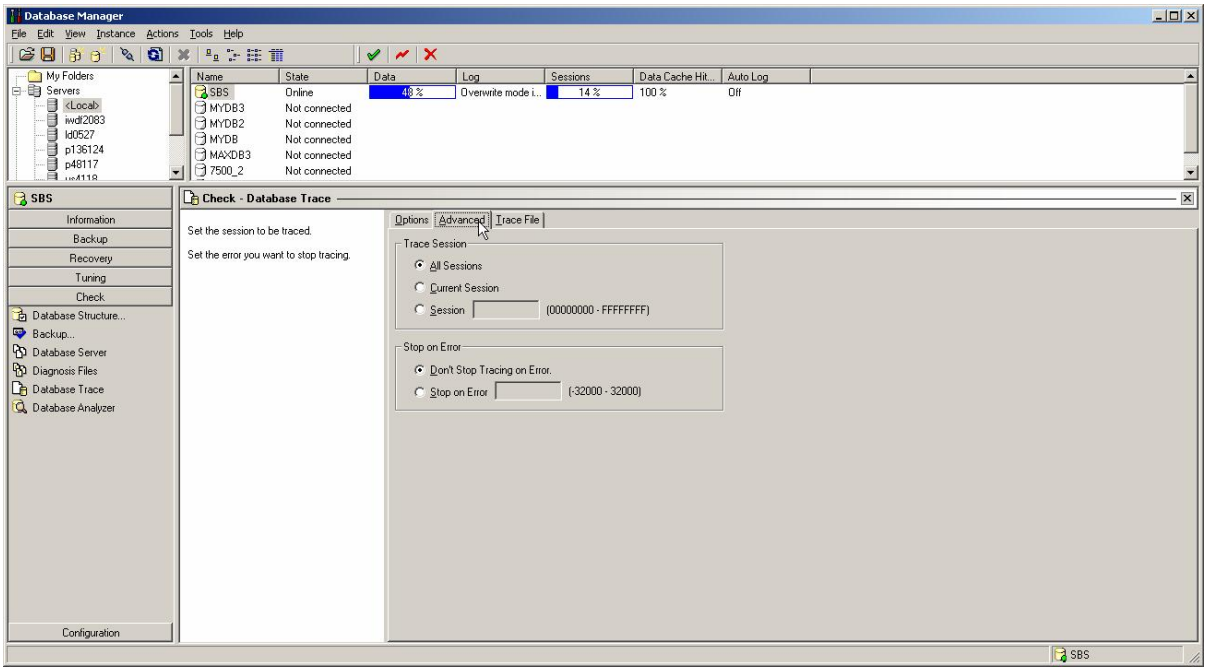
Use the buttons on top to activate, deactivate, flush or initialize the trace.

If you initialze the trace, all trace information is deleted.

dbmcli commands:

- Activate:
- dbmcli –d <SID> -u <dbm-user>,<password> trace_on default
- Flush:
- dbmcli –d <SID> -u <dbm-user>,<password> trace_flush
- Deactivate:
- dbmcli –d <SID> -u <dbm-user>,<password> trace_off
- Evaluate:
- dbmcli –d <SID> -u <dbm-user>,<password> trace_prot <Optionen>

Advanced options

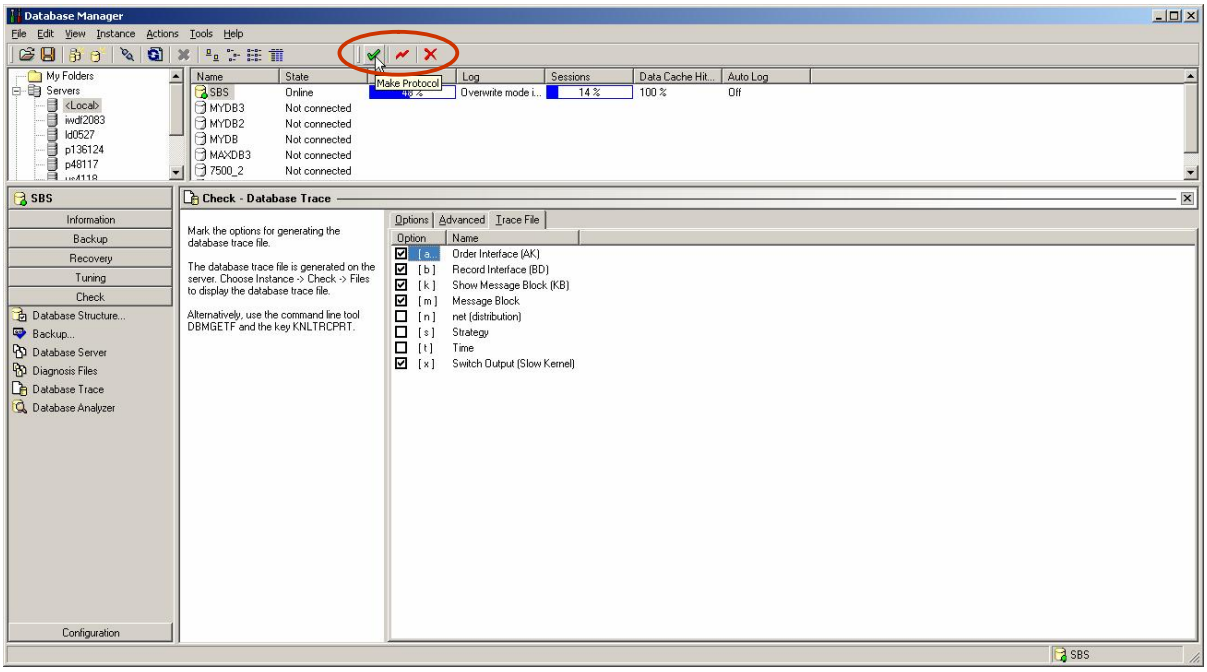


TRACE SESSION
To activate the trace for a specific database session, enter the session ID.

You can use the following commands to determine the session ID:
`x_cons <SID> show active` and
`SELECT * FROM TRANSACTIONS`

STOP ON ERROR
If the database error code is known you can specify this to make sure that the trace is stopped when the specific problem occurs.

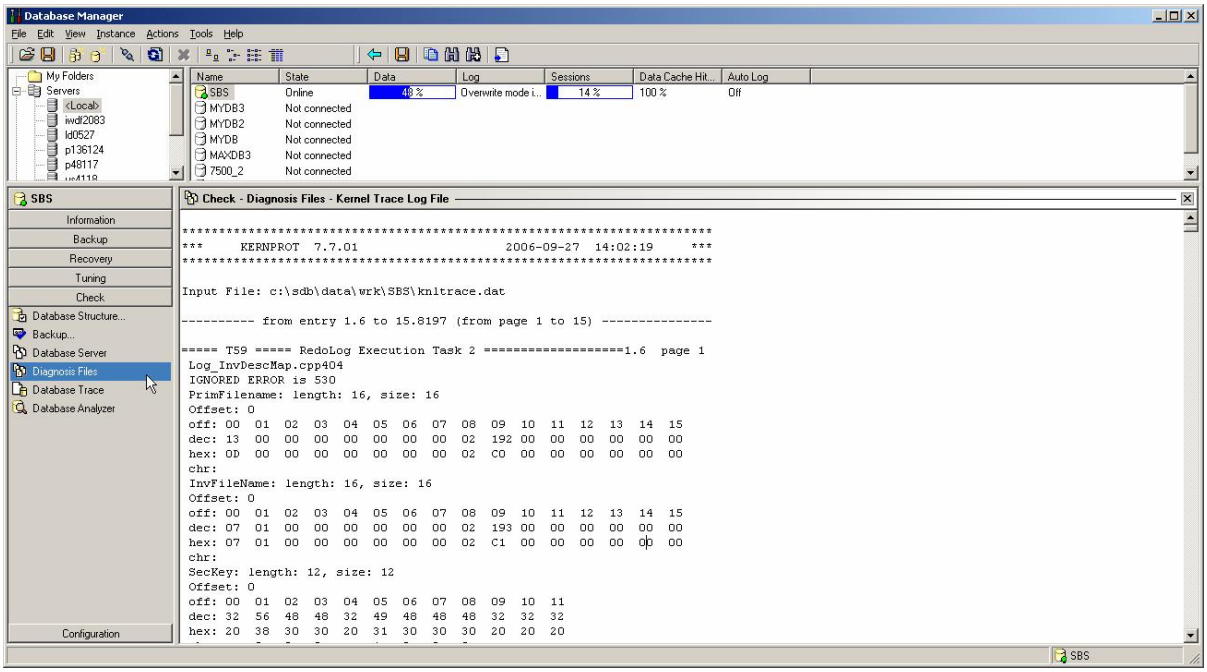
Evaluating the Trace



On tab ‚Trace File' the trace information can be flushed to file knltrace and the trace information can be evaluated.

You have to specify for which modules the trace information should be evaluated, DEFAULT: abkmx.

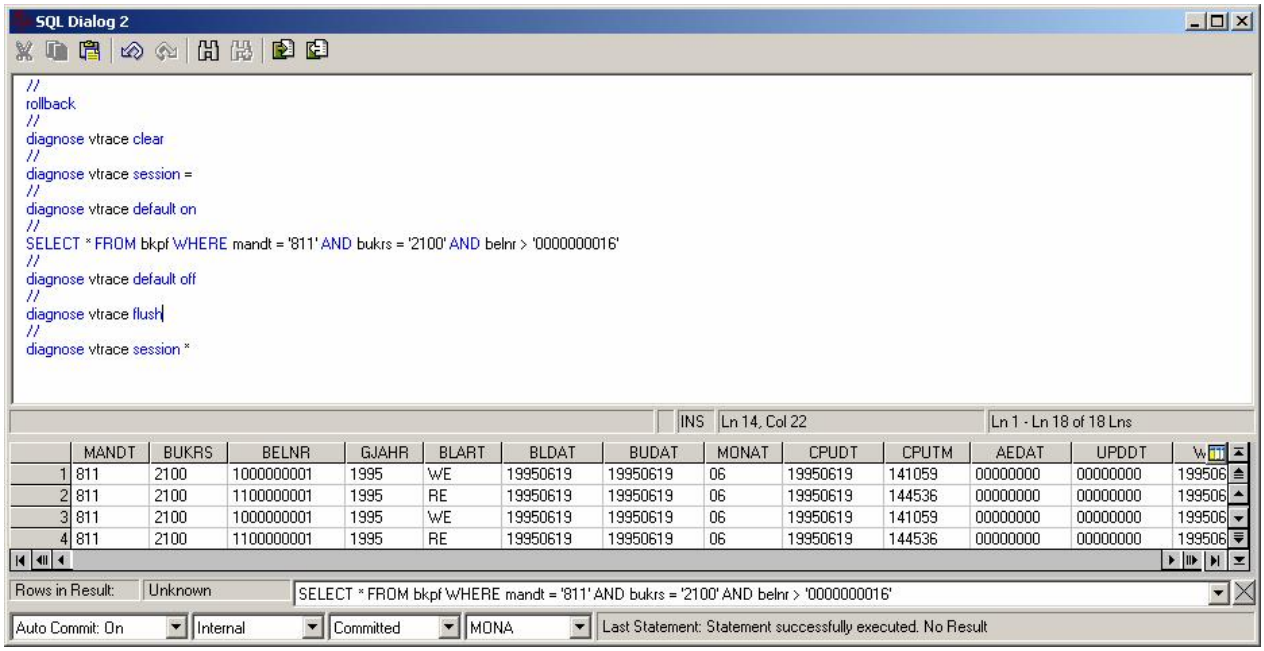
Kernel Trace Protocol



Using menu ‚*Check -> Files -> Kernel Trace Protocol*' you can view the trace protocol.

Normally the trace file has to be sent to the developer to analyze the problem.

Session Specific Kernel Trace



If you have to analyze just one or a few SQL statements, you can create a session specific kernel trace in SQL Studio (execute all statements one after the other by pressing F8):

```
Rollback
//
diagnose vtrace clear
//
diagnose vtrace session =
//
diagnose vtrace default on
//
<SQL statement to be analyzed>
//
diagnose vtrace default off
//
diagnose vtrace flush
//
diagnose vtrace session *
```

Afterwards the trace has to be evaluated using DBMGUI or DBMCLI.

SQLDBC Trace

The SQLDBC trace is used to analyze problems of the SQLDBC interface.

The SQLDBC trace is activated using tool sqldbc_cons.

Syntax: sqldbc_cons [<command>] [<option>]

The trace files are called sqldbctrace_<pid>.prt and the are created in directory

- <user_home>\Application Data\sdb (Windows)
- <user_home>\.sdb (UNIX, Linux)

The location and trace file name can be changed using option
CONFIG TRACE FILENAME <filename>

The trace files are overwritten cyclically. The size is specified using option
TRACE SIZE <size>

The default name of the trace files is sqldbctrace_<pid>.prt. <pid> is the process ID of the application process.

Possible traces:

- SQL: SQL-Statements (normally sufficient)
- Short: Method calls
- Long: Method calls with call parameters (largest trace)
- Packet: Communication packets

Possible commands for sqldbc_cons:

TRACE SQL ON/OFF:	(de)activates the SQL trace
TRACE PACKET ON/OFF:	(de)activates the PACKET trace
TRACE SHORT ON/OFF:	(de)activates the SHORT trace
TRACE LONG ON/OFF:	(de)activates the LONG trace
TRACE OFF:	deactivates all SQLDBC traces
TRACE FILENAME <file_name>:	specifies the name (and directory) of the trace file
TRACE SIZE <size>:	specifies the size (in bytes) of the trace file
SHOW ALL:	shows the trace configuration and current information to the traces
SHOW CONFIG:	shows the trace configuration
SHOW TRACESETTINGS	shows current information to the traces

Possible options for sqldbc_cons:

-h:	help information
-p <pid>:	executes the command for the specified process ID
-v:	shows extended information (verbose)

(These options can only be used separately.)

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

Further Information

Learning Objectives

Know how to set up a standby instance.

Know what a Hot-Standby system is and how it works.

High Availability

There are several possibilities to guarantee high availability of your database:

Standby database (manually updated)

Hot-Standby system in a cluster configuration (automatically updated)

Standby database

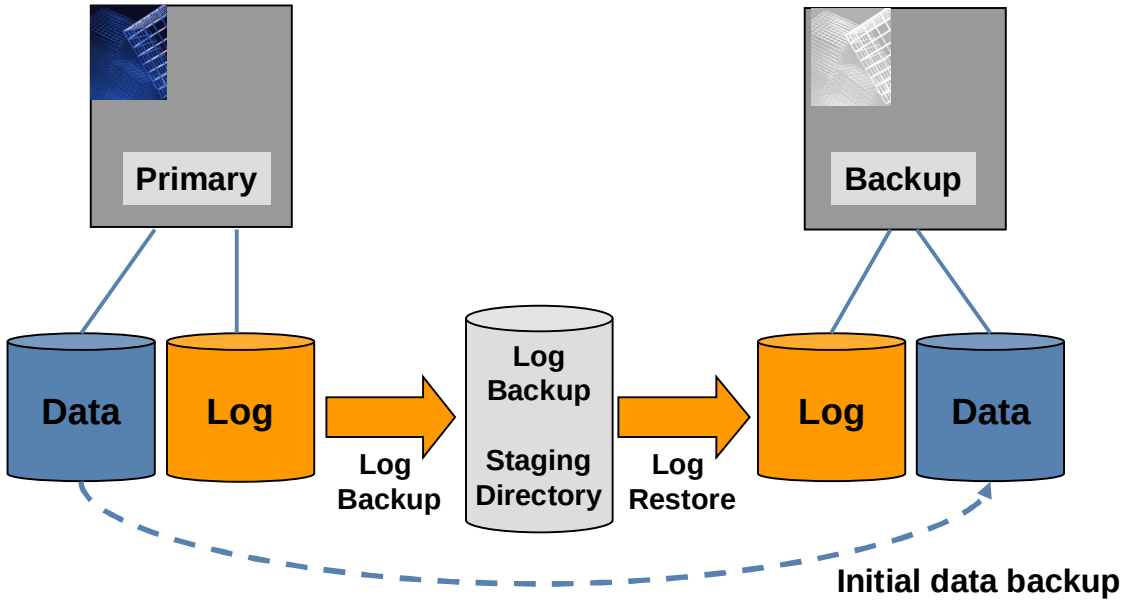
Standby instance: copy of the active database instance (original instance)

- To keep the contents of your standby database up-to-date, the log backups of the original instance are imported at regular intervals. The standby instance always has the operational state ADMIN.

If you experience problems with the original instance, you can start operating the standby instance immediately, and carry on working without a lengthy period of downtime.

Depending on the configuration, you can also restore the standby instance to a specified state in the past.

Standby Database (Log Shipping)



Available e.g. as DBShadow by Libelle (www.libelle.de)

Setting up a standby instance

Use the Database Manager for the following steps:

Create the standby instance by making a copy of your original instance. Do not start the copy in the operational state ONLINE; instead, keep it in the operational state ADMIN.

- Set up the original instance and standby instance on different hosts.

In the standby instance, create a backup medium for importing log backups.

Make log backups in your active original instance at regular intervals.

At regular intervals, import these log backups into your standby instance with a time delay of your choice.

Keep the following points in mind:

The data volumes of the standby instance must be configured with enough space to be able to import the log backups of the original instance.

You can use pipes to transport the data backup. You can transport log backups with ftp, or copy them with an exported file system (such as NFS or a shared file system).

You can start the standby instance in the operational state ONLINE, if you want it to replace the original instance as the active instance. If the standby instance has been in the state ONLINE, log backups from the original system can no longer be imported. So if you still want to use it as a standby instance, you have to initialize the instance, and import a complete data backup of the original instance.

Starting the Standby Instance as an Active Instance

Experiencing problems with the original instance? Stop it and start the standby instance to ONLINE. The standby instance can then take over the role of the active instance immediately.

Use the Database Manager for the following steps:

- Make sure that the original instance is in the operational state OFFLINE.
- Start the standby instance in the operational state ONLINE.
- In this standby instance, restore any indexes that are no longer up-to-date (BAD INDEXES).
- Make a complete data backup of the standby instance.

You can now operate the former standby instance as an active database instance. The original instance can become the standby instance. To do this, initialize the original instance, and proceed as described in Setting Up a Standby Instance.

Loss of Data

In this procedure, after you start the standby instance, it does not contain any data changes that have not yet been imported in log backups from the original instance.

You don’t want to loose data? Choose one of these options (depending on the configuration and state of your original instance):

- Importing Log Backups up to a Specific Time
- Importing Another Manual Log Backup
- Copying the Log Volumes of the Original Instance

The following example of an input script shows you how to use the Database Manager CLI to set up a standby instance and start operating it as an active instance:

```
db_offline
db_admin
medium_put <medium_name_data> <path_data> <medium_type> DATA

util_connect
db_activate recover <medium_name_data>
util_release

medium_put <medium_name_log> <path_log>\save.log FILE LOG

util_connect
recover_start <medium_name_log> LOG 001
recover_replace <medium_name_log> <path_log>\save.log 002
recover_replace <medium_name_log> <path_log>\save.log 003
…

recover_ignore
```

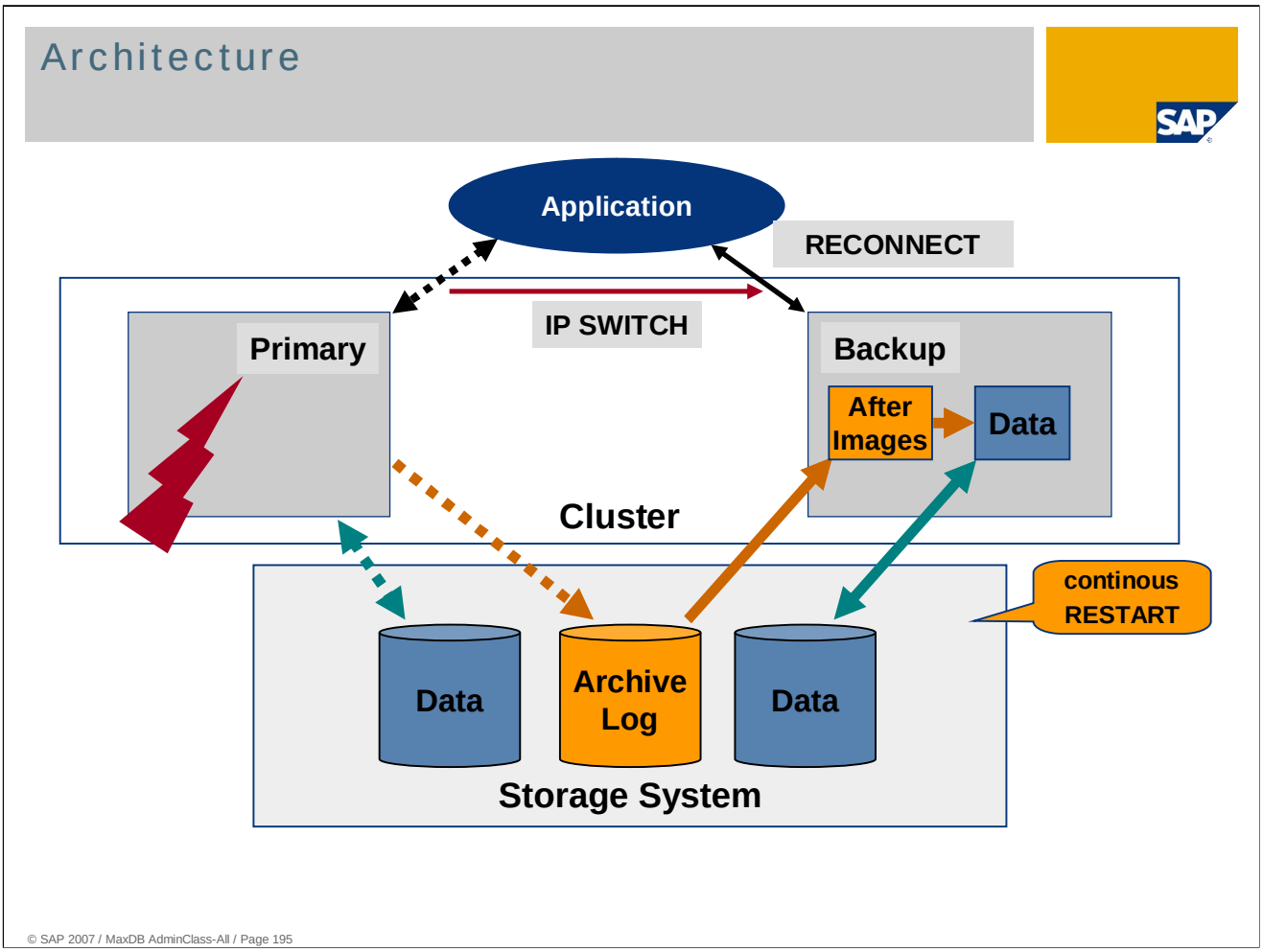
Hot-Standby System

Consists of an active master instance and one or several standby instances, which are installed in a cluster configuration.

In case of an error in the master instance, a standby instance can take over the master role immediately without loss of data.

The Hot-Standby system acts as a single database instance and is reached via a shared official node name.

Internally the components use local server addresses.



The master instance and the standby instances access their own, independent data volumes. They share the log volumes. The standby instances have only read access to the log volumes.

The standby instances are in mode STANDBY, which is somewhere between ADMIN and ONLINE.

Log entries generated on the master instance are redone on the standby instances automatically. Therefore the data in the standby instances is always up-to-date – with a short delay.

Installing a Hot-Standby System

Install a database instance which will be defined as the master instance

Add one or several standby instances

- During initialization of the standby instance, the storage system copies the complete content of the master data area in a consistent state to the standby data area (‚split')

Procedure:

Install a database instance <db_name>. Use the storage system for the data volumes.

Stop this database instance to operational state OFFLINE.

Use the following DBMCLI command to specify this instance as the master instance:

hss_enable lib=<hs_storage_dll> node=<official_node>

Start the master instance to operational state ONLINE

Use the following DBMCLI command to define a standby instance:

hss_addstandby <hs_node_nnn> login=<user>,<password>

To start the standby instance to mode STANDBY and therefore to add it to the Hot-Standby system, use the following DBMCLI command:

db_standby <hs_node_nnn>

The standby instance is initialized and in operational state STANDBY. All necessary configuration parameters are copied from the master instance.

Using a Hot-Standby System

The Hot-Standby system is addressed via the database name which was specified during the master installation and the official node name

All administrative tasks (backups, add volumes, parameter changes) are performed as in a single database instance

You have to perform data and log backups regularly to secure your system against handling errors

Fail-Over in a Hot-Standby System

In case of a problem on the master instance, the Fail-Over mechanism of the cluster configuration is responsible for the switch to a standby instance

The standby instance then gets write access for the log volumes, redoes the last log entries and is started into mode ONLINE

This guarantees a very short downtime and no data is lost

Chapter

MaxDB Overview

Installation of MaxDB software

Installation of a MaxDB instance

Internal Database Structure

Administration

Monitoring and Error Analysis

Fail-Over Solutions

Further Information

Where you can find further information

OSS notes:

- FAQ notes (search for FAQ on component BC-DB-SDB*)
- Notes with parameter recommendations (e.g. # 814704)

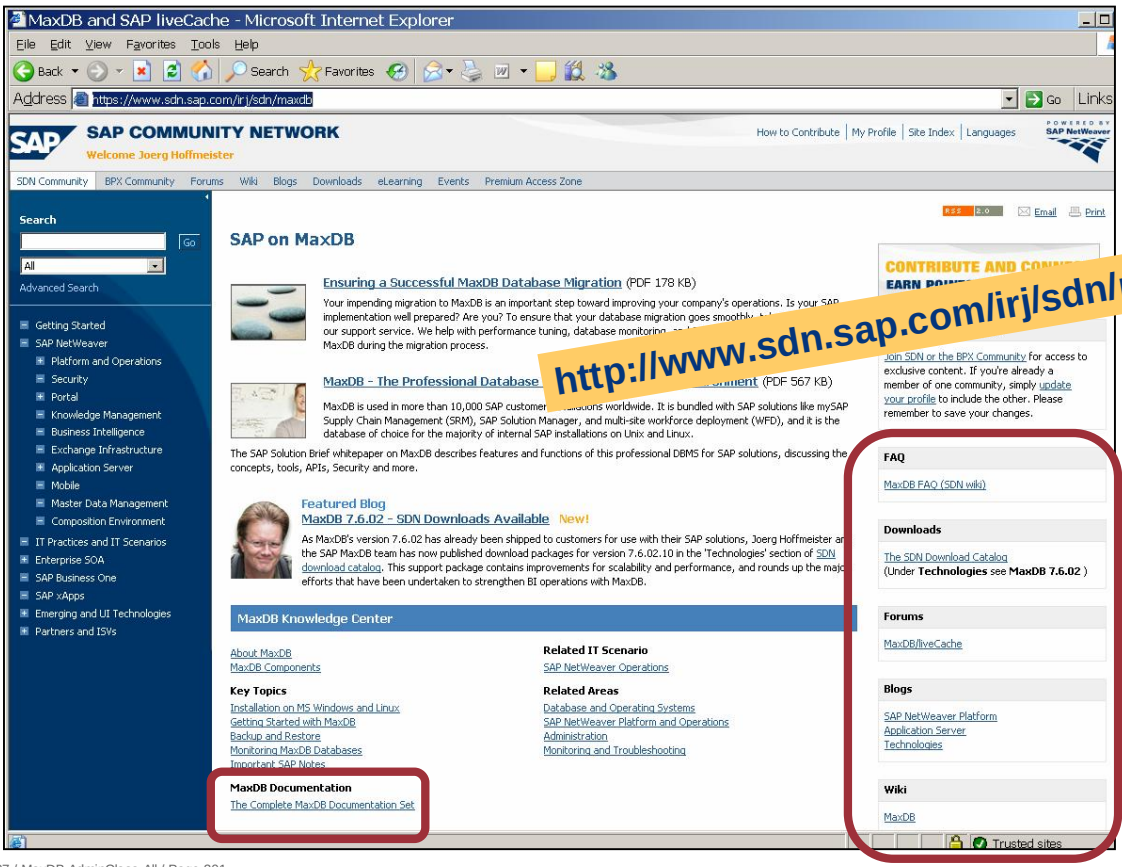
SDN:

- wiki (see next slides)
- forum (see next slides)

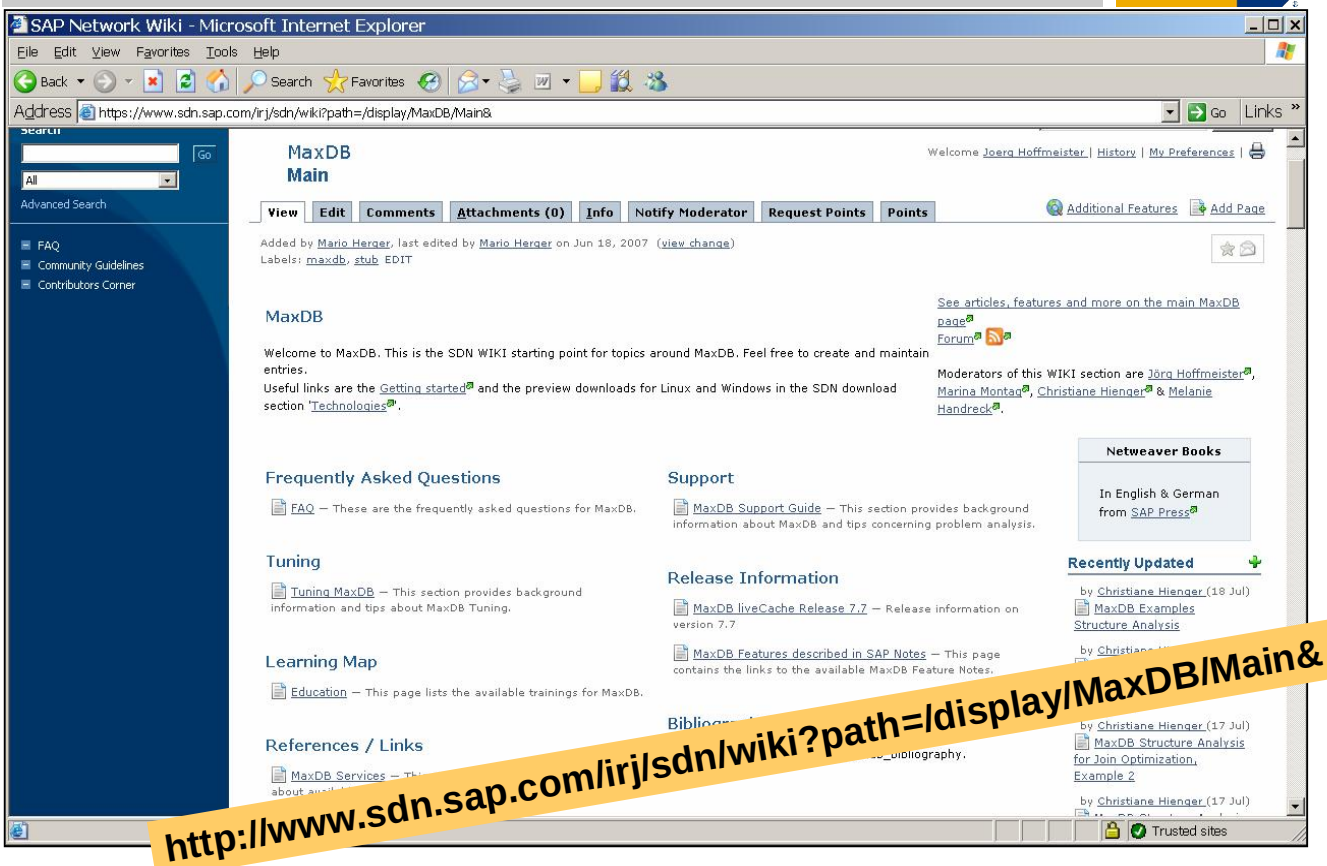
Documentation:

- http://maxdb.sap.com

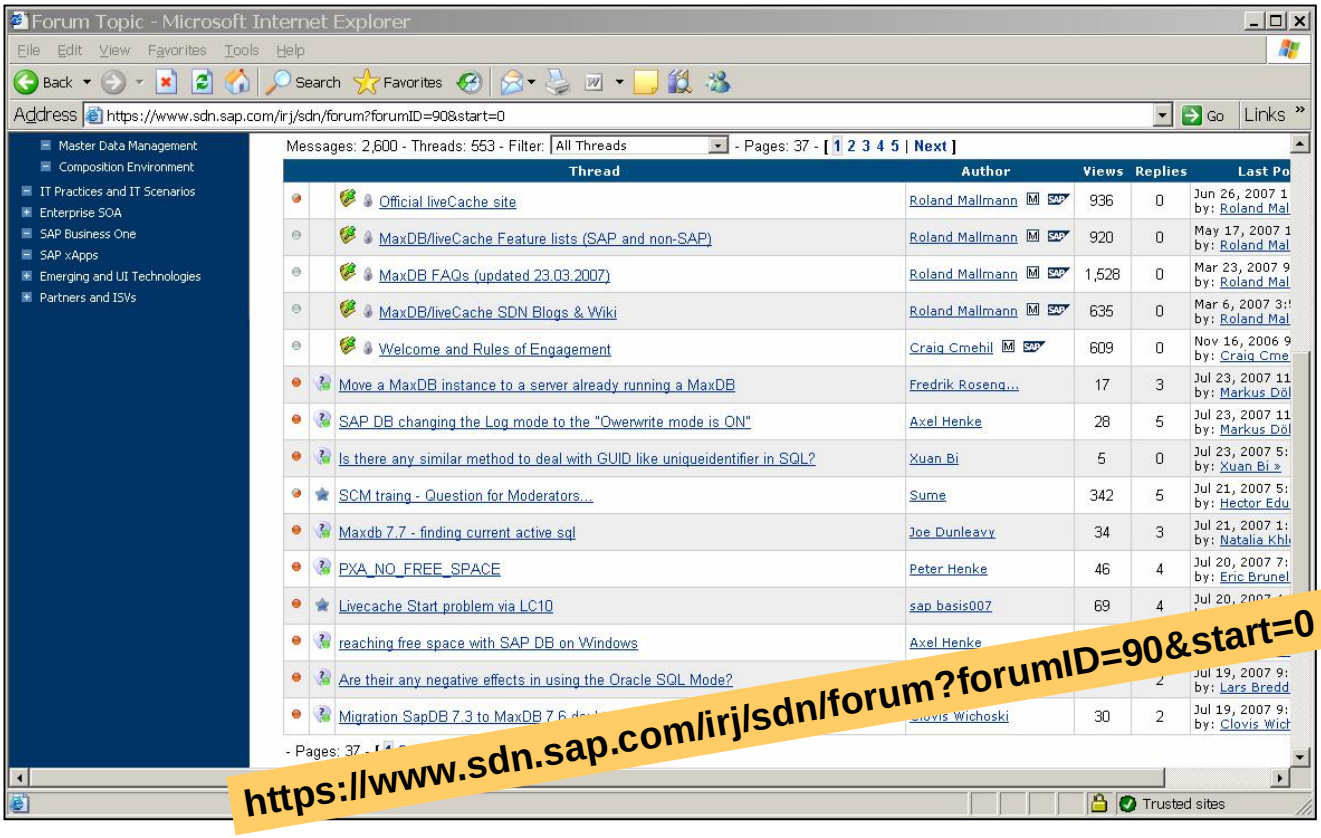
SAP Network: Information on MaxDB



SAP Network: MaxDB Wiki



SAP Network: MaxDB Forum



Disclaimer

This presentation reflects current planning. Contents may be changed without prior notice, and are in no way binding upon SAP.

Thank you!

Copyright 2007 SAP AG
All rights reserved