MaxDB

Error Diagnosis
Release 7.6

Heike Gursch

Overview

Types of errors / classification of errors

Errors during installation

Log files / traces / dumps

Check table / check backup

Analysis examples

- Connection problems
- DB full / log full situations
- Hanger situations
- Backup/restore problems
- Analyzing system errors -9028 / -9026

Extracting pages (x_diagnose)

Types of errors

Installation problems

Connect problems

Crash situations

Hanger situations

System errors (-10000 < error number <= -9000)

- with or without crash; reproducible ?

SQL errors (e.g. wrong result sets)

- reproducible effect ?

Save / restore errors

- valid backup ?

When an error occurs, the first step is to categorize it, as different types of errors call for different corrective measures. In some of the cases described here, categorization is simple. In some cases, however, it may not be immediately clear which of the active components is responsible for the undesirable system behavior. It may not even be clear whether the problem has to do with the database, and thus whether the diagnosis options described in this chapter will be helpful.

Problems arising from the installation of the software are easy to identify as such.

In an SAP environment, connection problems generally manifest themselves in that the database is, in principle, ready for operation, but the client processes cannot get a connection.

If the database, without any conscious action having been taken, is no longer ready for operation, it may have crashed. However, it is also possible that the database has consciously performed an emergency shutdown due to an existing error situation.

In the SAP environment, a hang situation is indicated by the presence of an hourglass. Determining the area in which a problem exists (lock collision, "blockage" of the system resources, etc.) is not necessarily insignificant.

System errors are serious errors and in the SAP system are often logged as error -602. The unique error number can be found in the ***knldiag*** file.

As a rule, transactions terminate with a short dump if an SQL error occurs.

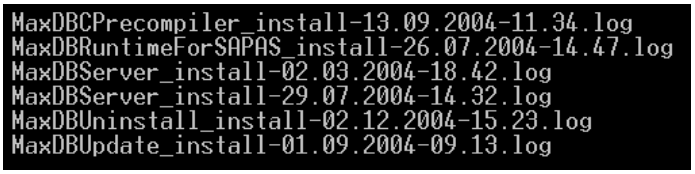
Backup or restore problems are recognizable as such; however, problems in this area frequently are due not to the database itself, but rather have an external cause operating system, external backup tool).

Installation

Call: sdbinst/sdbupd

Option –help lists valuable options.

Installation logs are written in <indepdatapath>/wrk.



Deinstallation of MaxDB software: sdbuninst

No deletion with operating system!

As of version 7.6, the tool SDBSETUP with GUI interface is available for all supported platforms

Installation of the MaxDB software is done with the tools ***sdbinst*** and ***sdbupd***. They are available for use after unpacking the software package provided on the Service Marketplace.

For an initial installation, *sbdinst* is used; overwriting an older version with a current one is done with *sdbupd.*

You get the call options by entering the option *-help*.

More extensive migration work may be necessary if you are making a big release-jump; the procedure is described in the corresponding notes.

If you encounter problems during installation, a corresponding message is issued. The message should inform you as to what error has occurred and how to fix it.

Installation logs are stored in the directory <indepdatapath>/wrk ; these can be useful, even at a later point in time. The name contains the type of installation as well as a time stamp, which makes it possible to determine the corresponding package.

sdbinst/sdbupd enable you to install individual components from the overall package. You use the *–package* option to specify the component(s).

You use ***sdbuninst*** to deinstall the software. Do not delete the MaxDB software with operating system resources under any circumstances!

Errors during the Installation

Examples (look for STDERR):

```
STDOUT: start extraction test run of "SAPDBBAS.TGZ"
STDERR: test run failed: cannot extract bin/dbmrfc: cannot write to /sapdb/pro
grams/bin/dbmrfc: Text file busy - no file(s) of "SAPDBBAS.TGZ" extracted!
STDERR: maybe any sap db software is running... please stop all
STDERR: installation exited abnormally  at Th, Aug 14, 2003 at 14:46:41
```

```
...
STDOUT: start extraction test run of "SAPDBUTL.TGZ"
STDERR: maybe any sap db software is running... please stop all
STDERR: installation exited abnormally  at Th, Aug 14, 2003 at 14:48:30
...
```

```
WRN: try to install release "7.3.00.32" over existing "7.3.0.36"
MSG: update test: installed release newer
MSG:  update from "7.3.0.36" to "7.3.00.32" not allowed
STDERR: cannot downgrade package
STDOUT: skipping package
```


To quickly find errors in the installation logs, look for the key word STDERR.

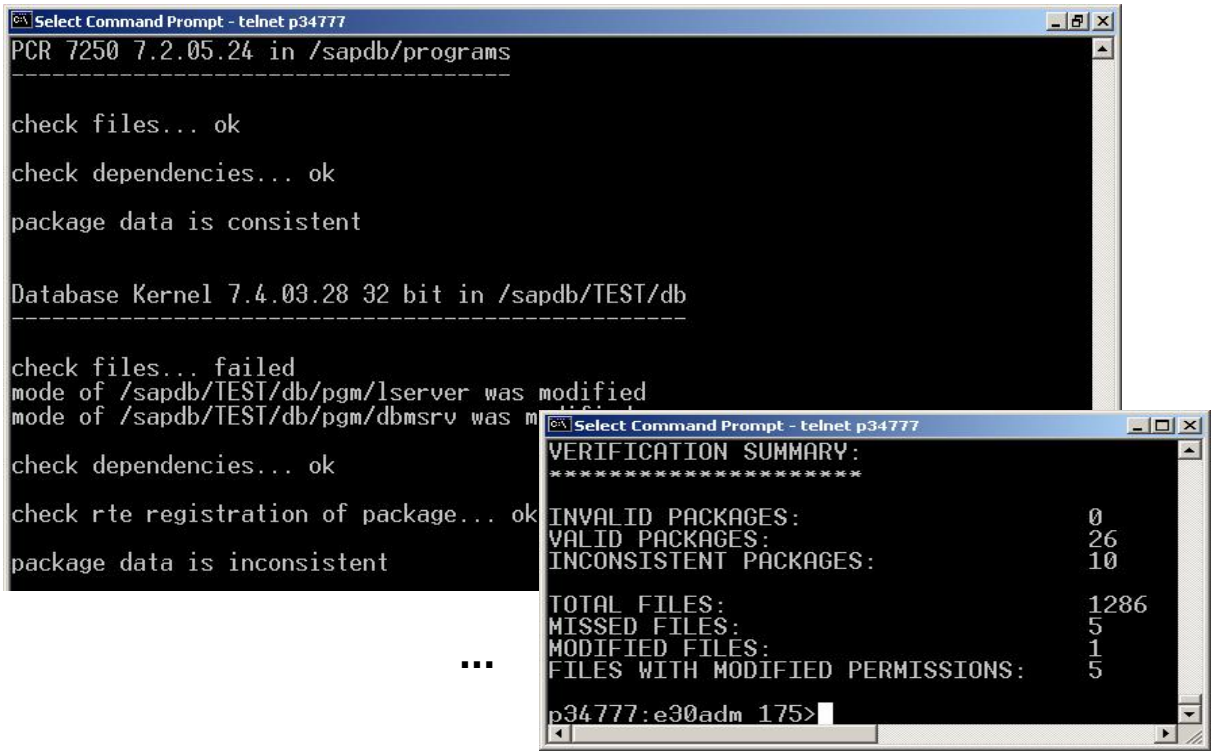
A common cause of errors in Windows systems is that DLLs cannot be overwritten or - more generally - that a component to be overwritten is still running (for example the x_server).

In the first two cases you get the message that the database software has not been completely stopped, so a re-installation would fail. To identify such problems ahead of time and avoid having the installation fail half-way through, the actual installation is preceded by a test run.

As of version 7.6.01 the option –force_extract cares for the substitution of programs and libraries being in use by copying the existing files.

In the third example, the attempt was made to install an older version over a newer one. This is not permitted since downward compatibility cannot be ensured, for instance when loading the system tables.

sdbverify



sdbverify is a tool that checks all installations on a computer for completeness. During the check, any inconsistencies due to impermissible software combinations are detected.

Using the registry entries, *sdbverify* checks whether the status in the file system still corresponds to the contents of the installation package.

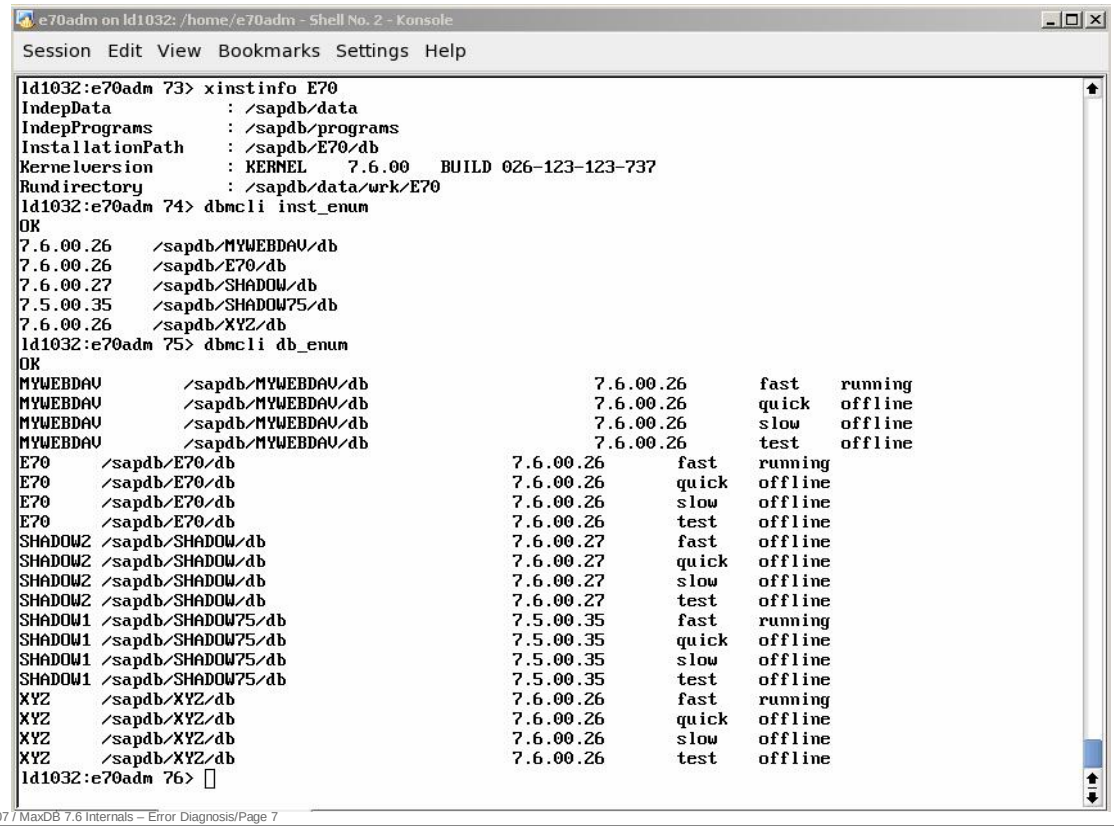
In the example above, it is noted that the access rights (under Unix) were subsequently changed.

The result is a summary of the installations that have been checked.

Start ***sdbverify*** with user root under Linux/UNIX.

To get an overview of the registered installations, you can also use the tool ***sdbregview***. If you call it with the option -I, it outputs a short list; without the option it outputs comprehensive information about every installation.

xinstinfo, inst_enum, db_enum



The tool ***xinstinfo*** provides a quick overview of the installation paths used on a system. It displays the "Independent Data Path" and the "Independent Program Path", or in other words, the paths that are valid for all the databases installed on the computer. The programs found there are always operated in the highest installed version (for example the x_server).

If ***xinstinfo*** is called with a database name, you also get database-specific information.

Using ***dbmcli*** with the entry ***inst_enum***, you get a list of the versions installed on the computer (dependent paths). The command ***db_enum*** lists the databases in their different variants (fast, quick, slow, test) as well as their current operational states.

Overview (Diagnostic and Trace Files)

System log (SM21)

ABAP Short Dump (ST22)

dev logs

SQL Trace (ST05)

Precompiler Trace

SQLDBC

appldiag

xserver_<hostname>.prt

dbm.prt, dbm.utl, dbm.knl, dbm.ebp, dbm.ebl

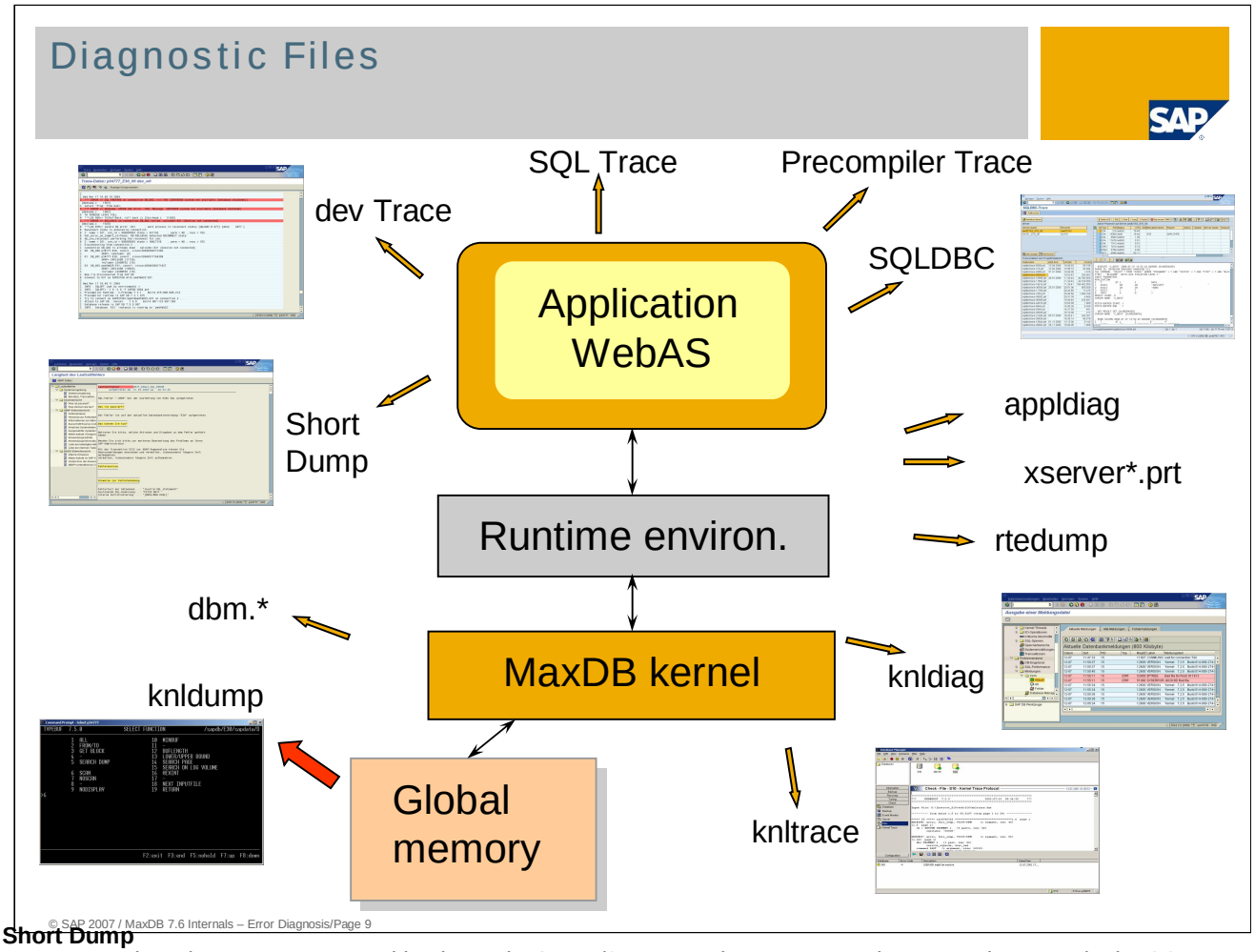
knldiag

Event Viewer

knltrace

knldump

rtedump



Short Dump
ABAP short dumps are generated by the WebAS or R/3 system when unexpected return codes occur in the SQL environment.

dev-Trace
The Developer Traces are logs of the disp+work processes of the SAP system.

SQL Trace
SQL commands and their runtimes are logged.

Precompiler Trace
SQL trace of the order interface.

SQLDBC
Trace for the SQLDBC (SQL Database Connectivity) interface.

appldiag
If errors between the runtime environment and the kernel occur, they are entered in the *appldiag* file. This file is created for each operating system user.

xserver_<hostname>.prt
If errors occur during communications via the x_server, they are entered in the *xserver_<hostname>.prt* file.

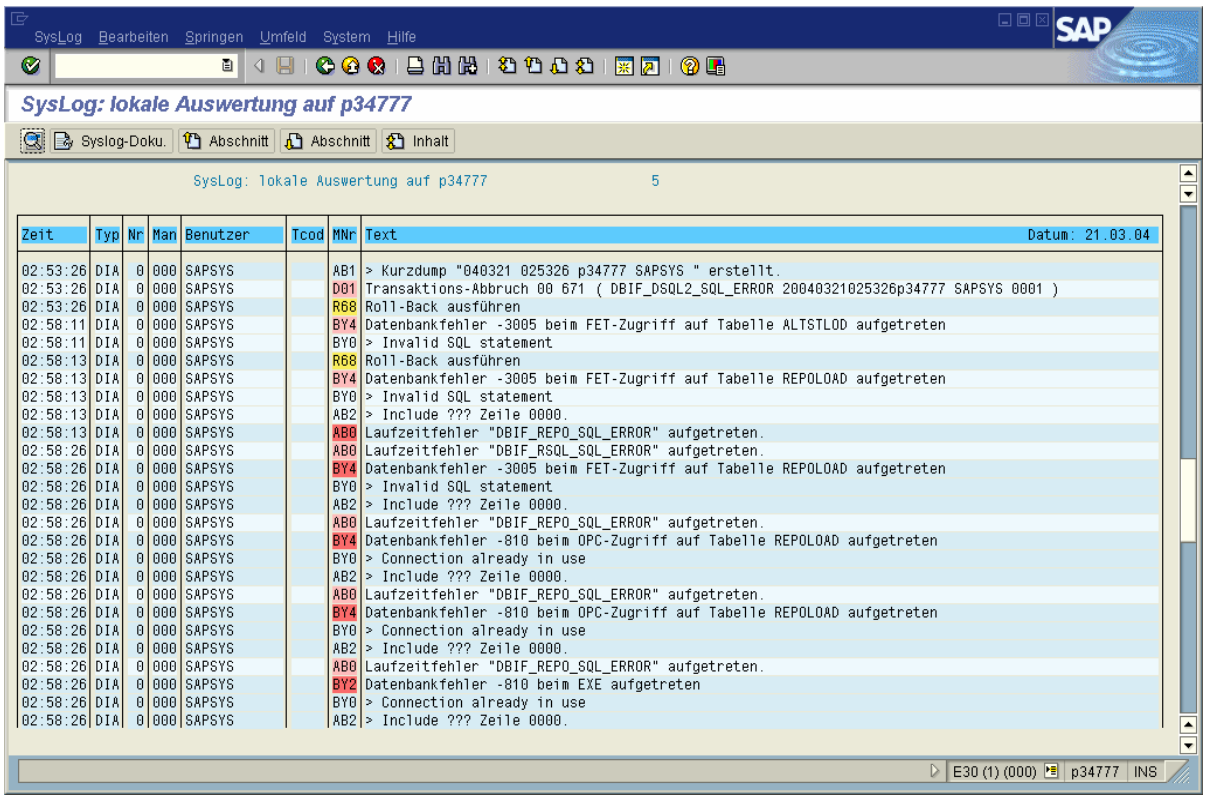
rtedump
If a crash occurs, the runtime environment writes its status in the *rtedump* file. It is an ASCII output of the command
`x_cons <SERVERDB> show all`

dbm.*
Various log files for the backup environment or for logging DBM server commands.**knldiag**
The kernel writes information and messages to the *knldiag* file. It has a fixed size and is overwritten cyclically. After a crash, it also contains the backtrace.

knltrace
This file is written by the kernel when the Vtrace is activated and following a crash. It has a predefined, fixed length.

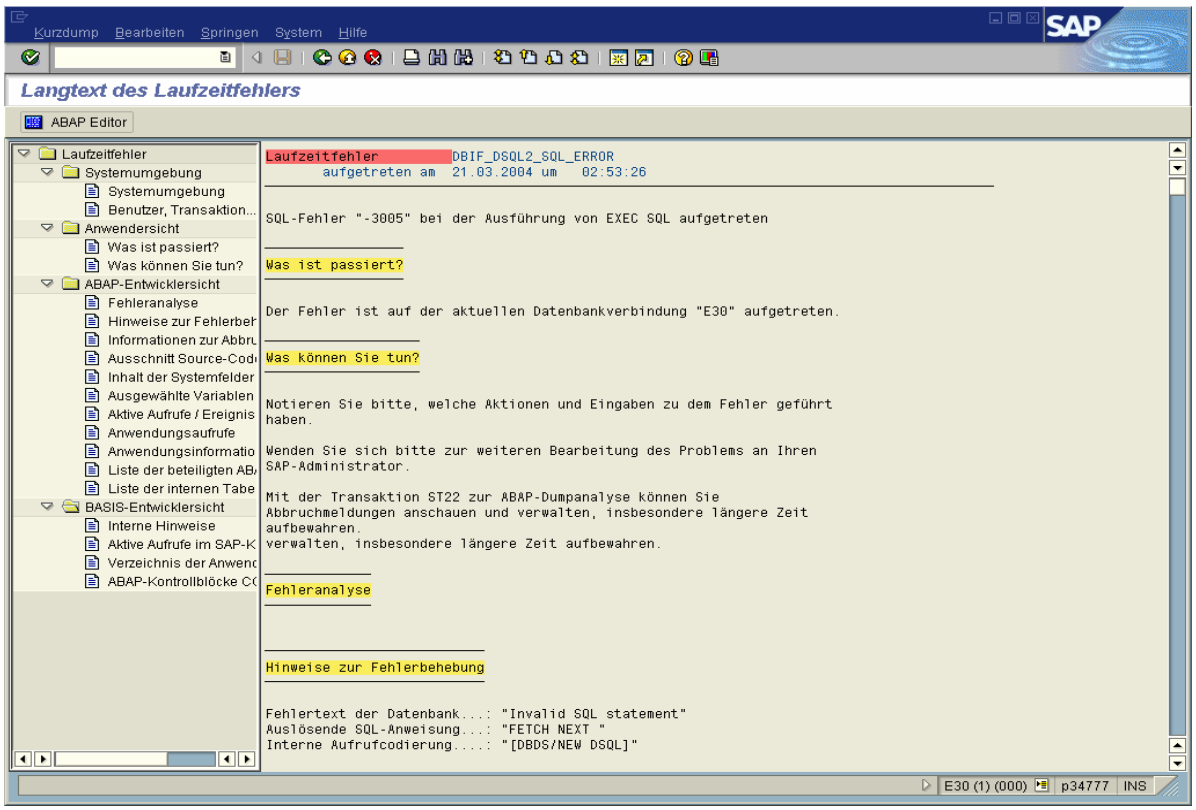
knldump
During an emergency shutdown, the global memory is written to the *knldump* file. The corresponding file system should be sufficiently large.

SM21 – System Log



Transaction sm21 displays the system log of the SAP system. The system log is not written under the control of the database, but it does contain information about database errors.

ABAP – Short Dump



In the SAP system, SQL errors in the database result in APAB short dumps when unexpected return codes occur.

They are not written under the control of the database, but they can be useful for analyzing error situations because they present a full picture of the error. SQL errors are otherwise not logged by the database, unless the Vtrace has been explicitly activated.

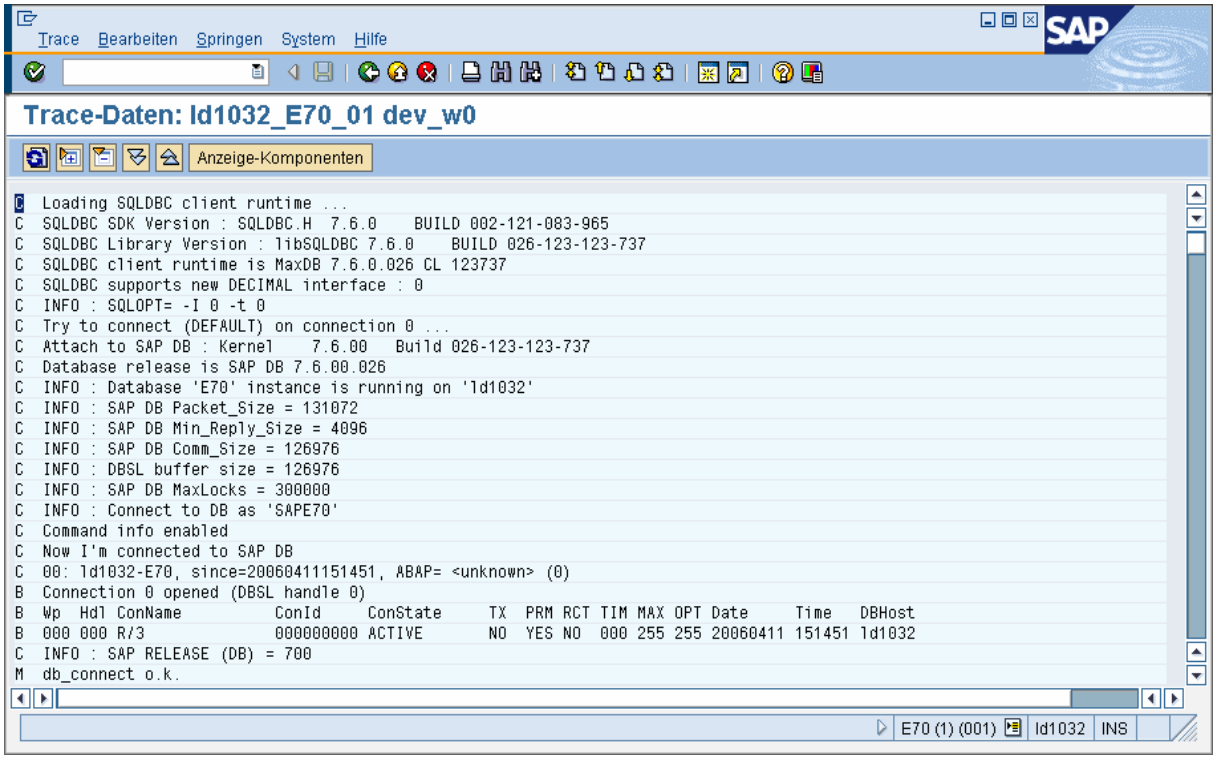
You can get a list of the short dumps that have occurred with transaction st22.

The short dump itself records which program and which ABAP command within it caused the error. You can then search for the error code in the notes.

If an unknown error occurs, it is often desirable to identify the command in "native SQL." To do this, in a reproducible case the SQL trace must then be activated with st05. At the same time, an analysis with the Vtrace may also be useful.

If necessary, you can find helpful information about the versions you are using in the "Notes for error analysis".

Dev Logs (1)



The developer traces are not written directly by the database, but rather log actions of the disp+work processes of the SAP system.

The dev logs are stored in the work directory of the SAP system and have the designation ***dev_w**.* You can access them directly with transaction st11.

They are active by default; only higher trace levels have to be activated explicitly. This is generally done by the developers themselves.

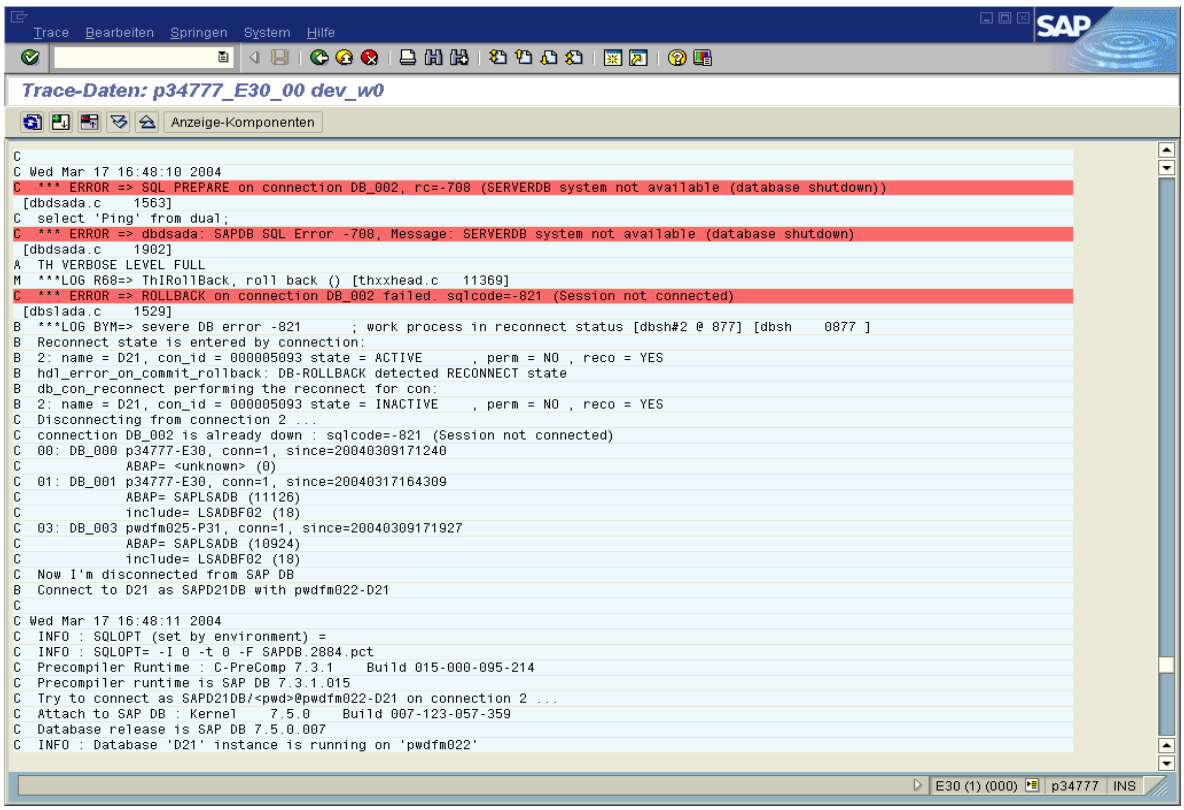
However, they were included in this unit because other information relevant for the database is also stored there.

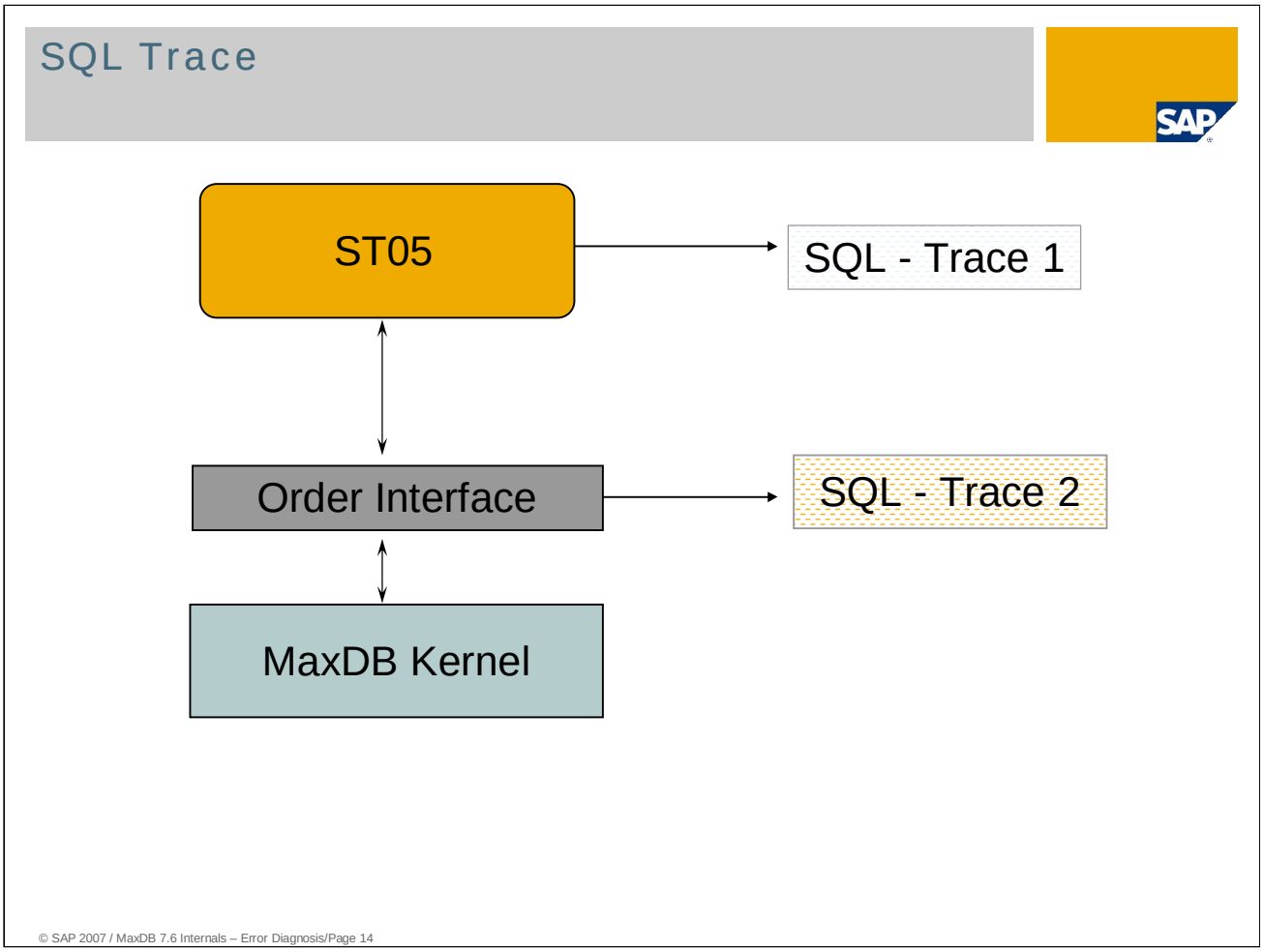
If connection problems between the disp+work processes and the DB occur, developer traces are often helpful.

Errors have a red background and thus are easy to find.

Among other things, it is easy to find the version of the precompiler environment being used.

Dev Logs (2)


Trace Bearbeiten Springen System Hilfe
Trace-Daten: p34777_E30_00 dev_w0
Anzeige-Komponenten
C
C Wed Mar 17 16:48:10 2004
C *** ERROR => SQL PREPARE on connection DB_002, rc=-708 (SERVERDB system not available (database shutdown))
[dbdsada.c 1563]
C select 'Ping' from dual;
C *** ERROR => dbdsada: SAPDB SQL Error -708, Message: SERVERDB system not available (database shutdown)
[dbdsada.c 1902]
A TH VERBOSE LEVEL FULL
M ***LOG R68=> ThIRollBack, roll back () [thxxhead.c 11369]
C *** ERROR => ROLLBACK on connection DB_002 failed. sqlcode=-821 (Session not connected)
[dbslada.c 1529]
B ***LOG BYM=> severe DB error -821 ; work process in reconnect status [dbsh#2 @ 877] [dbsh 0877]
B Reconnect state is entered by connection:
B 2: name = D21, con_id = 000005093 state = ACTIVE , perm = NO , reco = YES
B hdl_error_on_commit_rollback: DB-ROLLBACK detected RECONNECT state
B db_con_reconnect performing the reconnect for con:
B 2: name = D21, con_id = 000005093 state = INACTIVE , perm = NO , reco = YES
C Disconnecting from connection 2 ...
C connection DB_002 is already down : sqlcode=-821 (Session not connected)
C 00: DB_000 p34777-E30, conn=1, since=20040309171240
C ABAP= <unknown> (0)
C 01: DB_001 p34777-E30, conn=1, since=20040317164309
C ABAP= SAPLSADB (11126)
C include= LSADBF02 (18)
C 03: DB_003 pwdfm025-P31, conn=1, since=20040309171927
C ABAP= SAPLSADB (10924)
C include= LSADBF02 (18)
C Now I'm disconnected from SAP DB
B Connect to D21 as SAPD21DB with pwdfm022-D21
C
C Wed Mar 17 16:48:11 2004
C INFO : SQLOPT (set by environment) =
C INFO : SQLOPT= -I 0 -t 0 -F SAPDB.2884.pct
C Precompiler Runtime : C-PreComp 7.3.1 Build 015-000-095-214
C Precompiler runtime is SAP DB 7.3.1.015
C Try to connect as SAPD21DB/<pwd>@pwdfm022-D21 on connection 2 ...
C Attach to SAP DB : Kernel 7.5.0 Build 007-123-057-359
C Database release is SAP DB 7.5.0.007
C INFO : Database 'D21' instance is running on 'pwdfm022'
E30 (1) (000)
p34777
INS

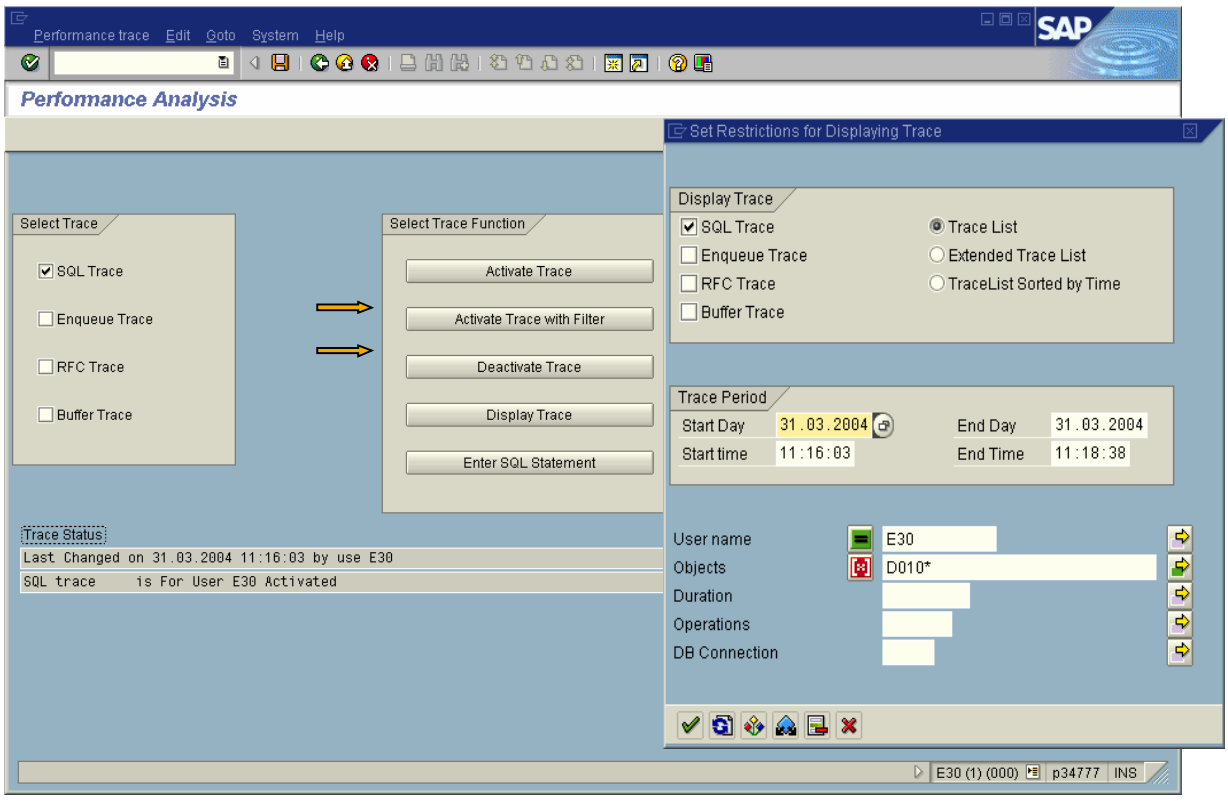


In the SAP system, you activate the SQL trace with transaction st05. The log is written by the database interface. Along with the statements, you'll find the variables, their values and the runtime. The Explain button in transaction st05 displays the database's Optimizer strategy for the command.

This transaction is discussed further in the section on SAP system transactions that are useful for error analysis.

The order interface of the database also writes an SQL trace. Up to version 6.40, the ABAP part of WebAS uses the precompiler interface of the database. As od version 7.0, WebAS uses the new interface SQLDBC. The corresponsing traces show which commands arrive at this interface and which data is transmitted to the clients.

ST05 – SQL Trace



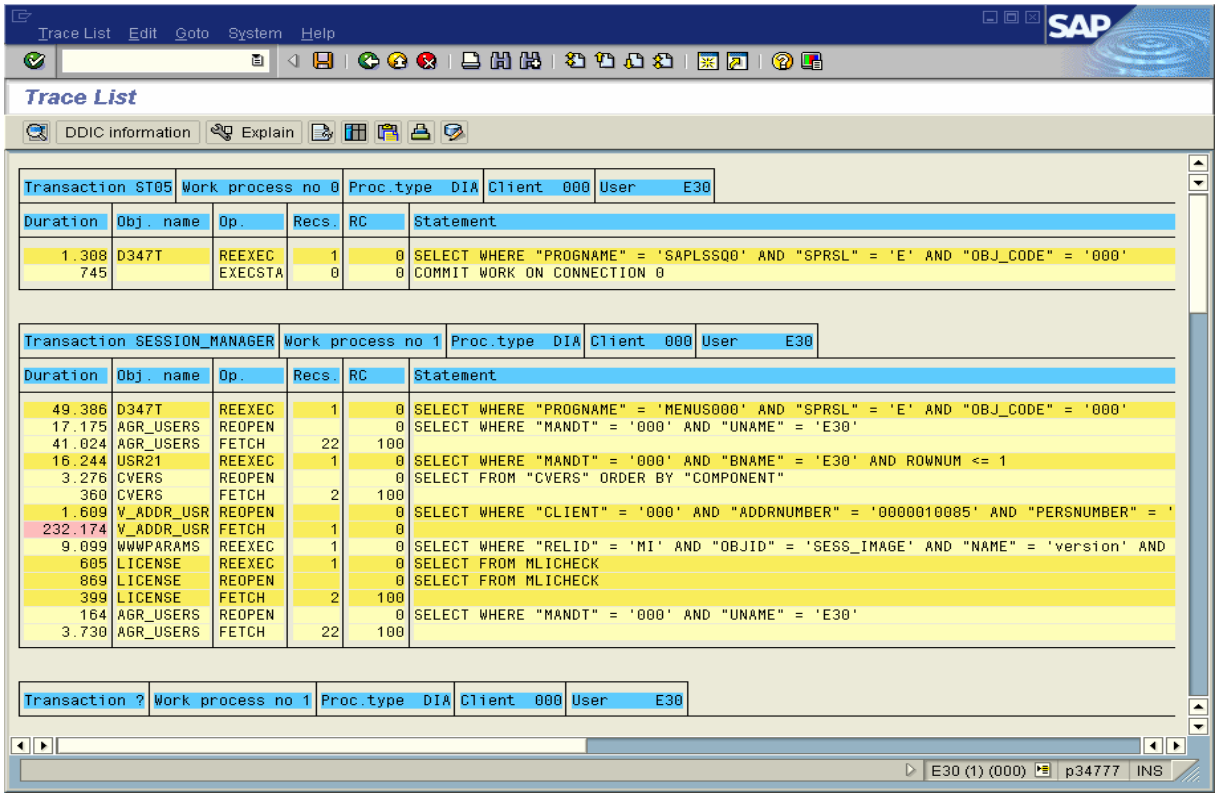
You can use transaction st05 to activate an SQL trace. This is useful for performance analyses or for identifying a command that leads to incorrect result sets.

The SQL trace displays all SQL statements in the form in which they were sent to the database.

Note that when you use the button shown here, the trace must be deactivated before being displayed so that it can be formatted. You can display the trace directly by choosing *Performance Trace->Display Trace Without Prior Deactivation*.

For a more manageable amount of information, you can restrict the display to a specific time period or a particular user or by omitting information about specified tables.

ST05 – SQL Trace (2)



The list of commands identifies which transaction was started, which work process is affected, its type, as well as the client and user.

The command list contains

- information about the duration of the command, where the long-running ones are marked red,

- the affected database object,

- the number of records found,

- the return code of the database and

- the statement.

The statement can be expanded by double-click. It is also possible to insert parameter values in the placeholders so that the statement can be used directly for further testing, for example in the SQL Studio.

The 'Explain' button displays the execution plan of the Optimizer.

Precompiler Trace (1)

SQL – Trace of the order interface:

Profile parameter

- dbs/ada/sql_trace = 0 no trace
  1 short trace
  2 long trace

Environment variable

- SQLOPT = -F file name
  -T short trace
  -X long trace
  -Y statement count

Irtrace

- irtrace –p all –t <trace type>
- irtrace –p <prozess-id> -t <trace type>

The order interface trace is set for Disp+Work processes using a profile parameter. On Windows systems, after changing the profile parameter, only the work process has to be restarted. On Unix systems, the SAP system or the affected application server has to be restarted. The trace files are stored in the work directory of the SAP instance. The name is comprised of the process ID of the work process and the ending **`pct`** .

Other tools that utilize the order interface read the environment variable SQLOPT. Unless otherwise specified with the -F option, the trace file is written to the current directory. The name is comprised of the name of the corresponding C module and the ending **`pct`** .

You can use irtrace to activate the trace without needing to restart the system/application server.

The tool gives you the following options for changing the trace:

- Activate/deactivate/switch trace for a particular process:
  irtrace –p <process id> –t <trace type>
- The following trace types are available:
  long
  short
  off
- Activating/deactivating the trace for all interface processes on the application server:
  irtrace –p all –t <trace type>

Precompiler Trace (2)

```
PRODUCT : liveCache C/C++ Precompiler Runtime
VERSION : 7.1.4
BUILD   : 032-000-055-840

version  :P_1, P_2
SQL STATEMENT  :  FROM MODULE : dbslada            AT LINE : 4186
OUTPUT :  LZU  :  NT/INTEL  7.1.4    Build 032-000-055-840
OUTPUT :  PCR  :  C-PreComp 7.1.4    Build 032-000-055-840
START  :  DATE :  2001-07-13    TIME :  0013:01:01
END    :  DATE :  2001-07-13    TIME :  0013:01:01

SESSION   : 1;
SQLMODE   : SAPR3   AT DATABASE :  DB_000
SERVERDB  : S10
SERVERNODE:
OPTION-CONNECT :
CONNECT "SAPR3             " IDENTIFIED BY :A  SQLMODE SAPR3  ISOLATION LEVEL 0
 TIMEOUT 0
SQL STATEMENT  :  FROM MODULE : dbslada            AT LINE : 6390
START  :  DATE :  2001-07-13    TIME :  0013:01:01
END    :  DATE :  2001-07-13    TIME :  0013:01:01
```

SQLDBC

Runtime library libSQLDBC for the development of database applications and interfaces

- Version-independent runtime library libSQLDBC.dll or libSQLDBC.so
- Version-dependent, e.g. libSQLDBC76.dll
- Runtime library libSQLDBC_C.dll for the support of native C applications
- Stored in directory <indep_program_path>/pgm

Software Development Kit SQLBDC SDK

- Header file SQLDBC.h for C++
- Header file SQLDBC_C.h for C
- Static and dynamic link libraries for C and C++
- Stored in directory <indep_program_path>/sdk/sqldbc

sqldbc_cons

- Creation and control of traces

SQL Database Connectivity (SQLDBC) is a runtime environment for the development of database applications and database interfaces for MaxDB. Through SQLDBC, applications can access MaxDB database instances, execute SQL statements and edit data. SQLDBC is comprised of the three abovementioned components, which are part of the standard and stored in the said directories.

As of version 7.0, WebAS uses SQLDBC.

Traces can be created either directly with sqldbc_cons or using transaction db50.

sqldbc_cons

Configuration and creation of traces for the SQLDBC interface

Properties of the traces:

- sqldbctrace_<pid>.prt
- The trace files are overwritten cyclically.

Traces storage in the following directories:

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

Trace types:

- SQL: SQL statements (sufficient for most analyses)
- Short: method calls
- Long: method calls with call parameters (most comprehensive trace)
- Packet: communication packages

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

sqldbc_cons is a tool for the configuration and control of traces.

The trace files contain a file name of the form sqldbctrace-<pid>.prt, where <pid> is the process ID. It is also possible to choose a name; %p in the name is replaced by the process ID. Traces are stored in the directories <user_home>\Application Data\sdb (Windows) and <user_home>\.sdb (UNIX, Linux). When the configured trace size is reached, the trace is cyclically overwritten.

Possible commands for sqldbc_cons:

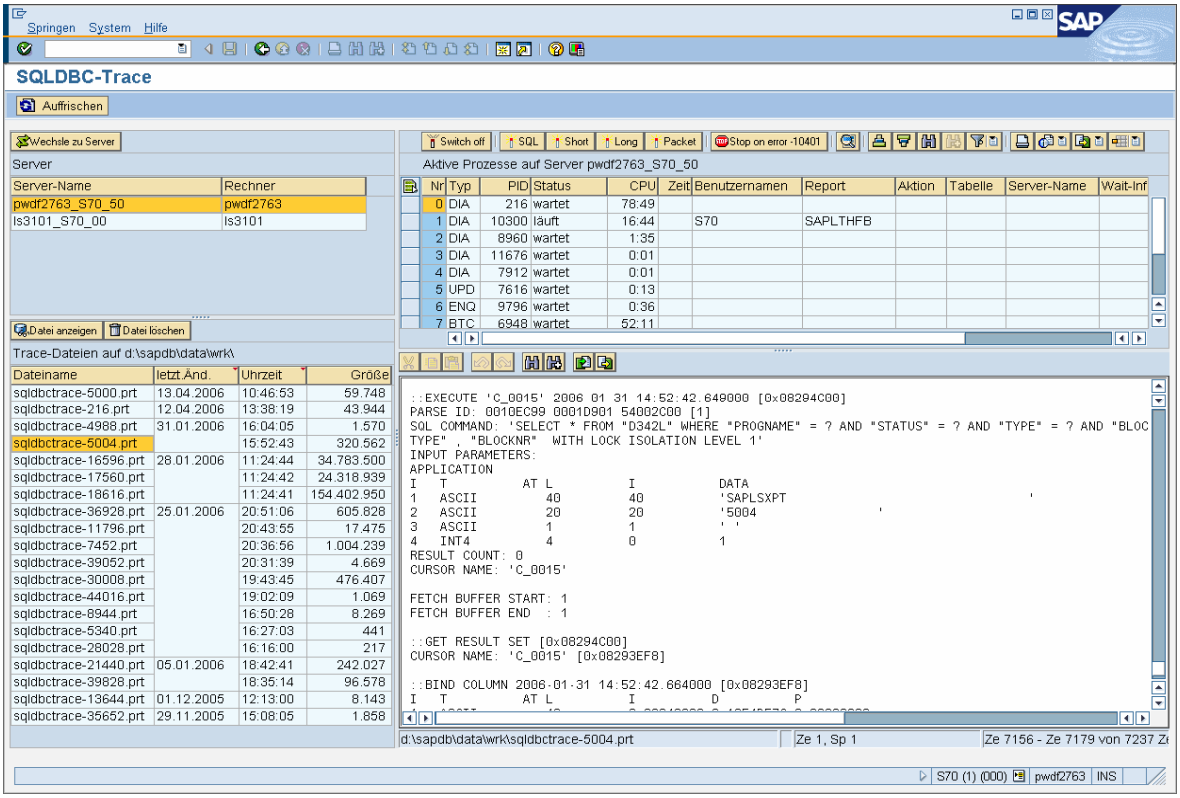
TRACE SQL ON/OFF:	Turns the SQL trace on/off
TRACE PACKET ON/OFF:	Turns the PACKET trace on/off
TRACE SHORT ON/OFF:	Turns the SHORT trace on/off
TRACE LONG ON/OFF:	Switches the detailed LONG trace on/off
TRACE OFF:	Switches all SQLDBC traces off
TRACE FILENAME <file_name>:	determines the name of the trace file
TRACE SIZE <size>:	defines the size (in Bytes) of the trace file
SHOW ALL:	displays the configuration of the traces and current information about the traces
SHOW CONFIG:	displays the configuration of the traces
SHOW TRACESETTINGS	displays current information about the traces

Possible options for sqldbc_cons:

- -f: forces the execution of the command
- -h: Help information
- -p <pid>: executes the command only for the process with the process ID <pid>
- -v: displays detailed information (verbose)

(These options cannot be combined, but only used individually.)

SQLDBC Trace



In transaction db50, choose the path *Tools-> SQLDBC Trace.*

Activating the trace involves three steps:

- Selection of the desired process
- Selection of the trace type (SQL, Short, Long, Packet)
- Specification of trace size (*Go to-> Maximum File Size*)

To switch the trace off, select the process and press the button *Switch off*.

Via menu item*Go to-> Trace Directory* you can choose a trace file name that differs from the default.

To display the trace, select the trace file and press the button *Display File.*

appldiag

Example:

```
06.01 12:53:46  18286 -11205 sqlexec:  system error, not enough space
06.01 12:58:25  18286 -11205 sqlexec:  system error, not enough space
06.01 18:46:23  19025 -11109 database 'S10' is not running
07.01 11:47:37  10959 -11987 sql33_con_msl: task limit
07.01 11:47:59  12031 -11987 sql33_request: connection broken, kernel
                             cleared
08.02 13:11:07  18899 -11987 connection closed by communication partner
08.29 13:13:13  11199 -11987 comseg given away, assuming timeout
```

ID of the process causing the error

The file ***appldiag*** contains error messages that can occur during communication between the applications and the runtime environment.

The file ***appldiag*** or ***<pid>.dia*** (SAP WebAS) is stored in the directory

<indepdatapath>/wrk/<unix user> (UNIX)
<indepdatapath>\wrk (NT)
/usr/sap/<SID>/D*/work (SAP WebAS)

The <indepdatapath> can be determined using the following dbmcli command:

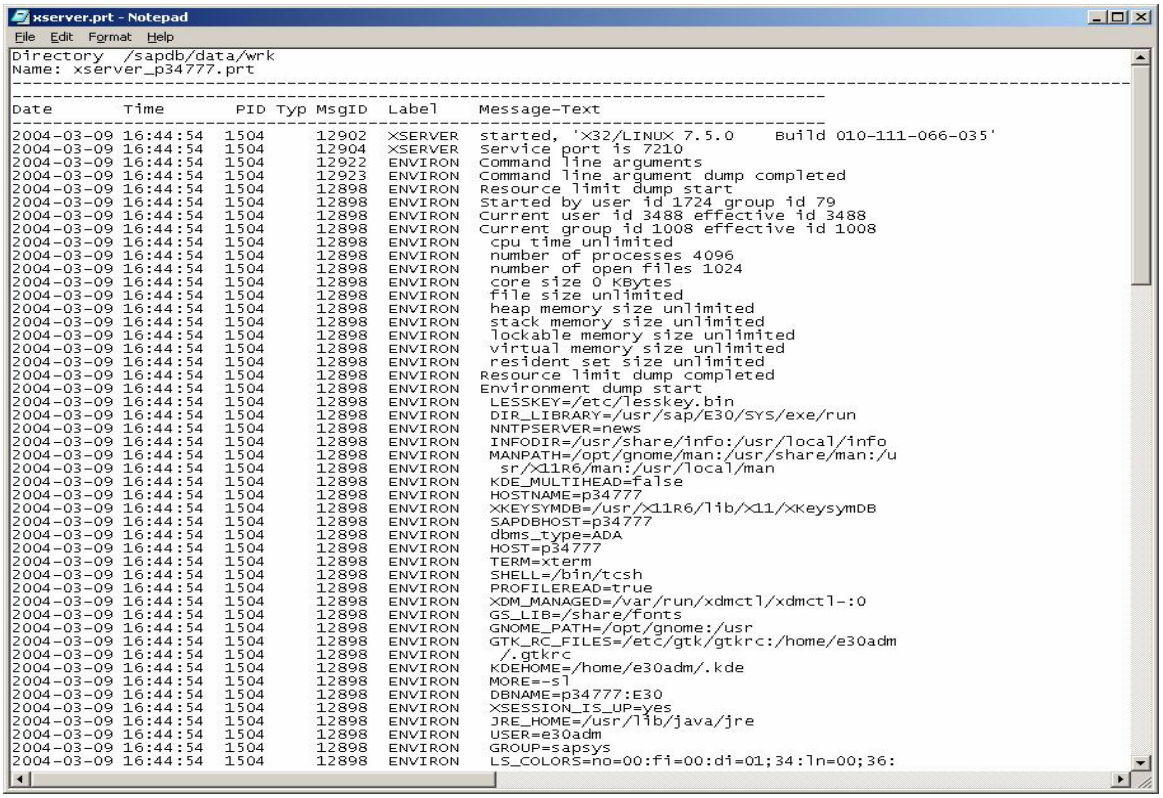
dbmcli –d <SID> -u <dbm-user>,<password> dbm_getpath indepdatapath

Under Windows/NT, ***appldiag*** is only activated if the environment variable is set to DIAGFILE=yes.

The file ***appldiag*** can get very large since it is not cyclically overwritten.

If this file already exists, further messages are added to it; otherwise it is created.

xserver_<hostname>.prt



The file ***xserver_<hostname>.prt*** contains error messages involving the **x_server**. The x_server is used for remote communication and starts vserver processes for each new user who connects to the database remotely.

If multiple database software versions are installed on a computer, the x_server must always be started with the highest version. You can display the highest version with

x_server –V

You can display the possible options for installing, starting and stopping with

x_server –h

The ***xserver_<hostname>.prt*** is stored in the directory <indepdatapath>/wrk .

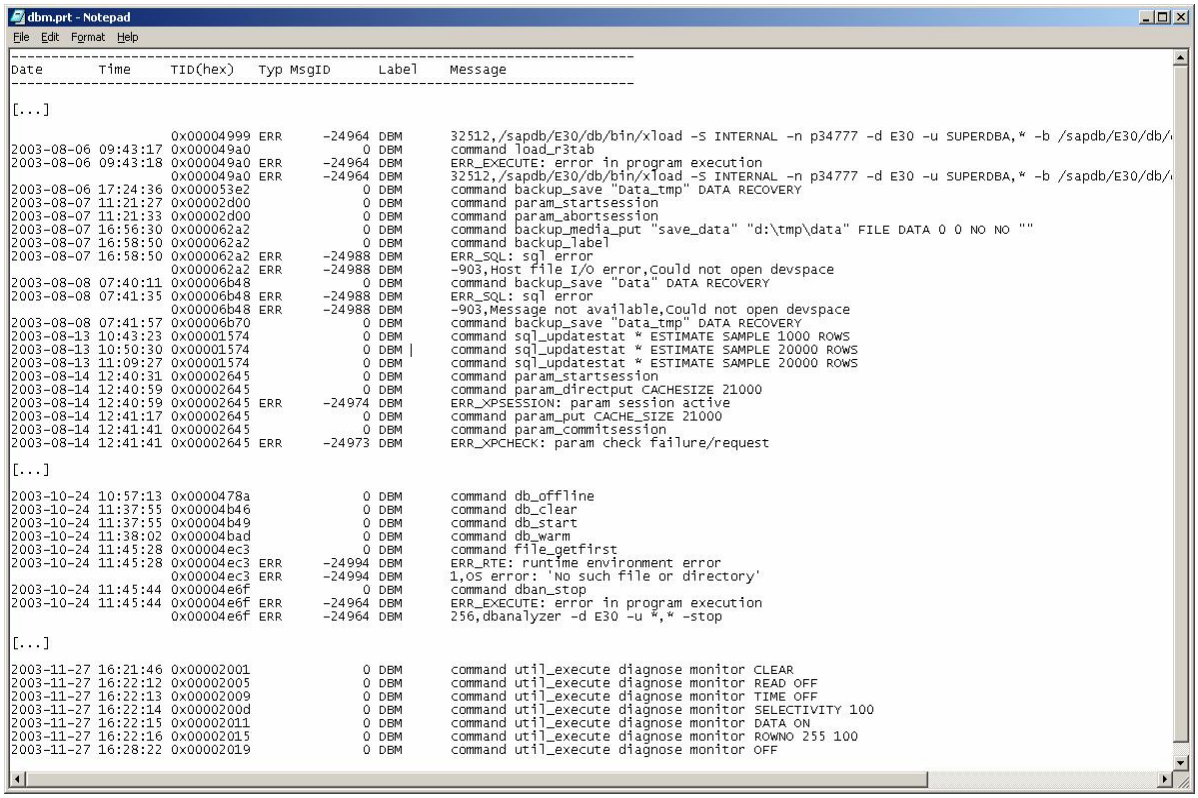
This includes a time stamp, a process ID, in the label the affected software component and an explanatory message text.

If a return code is reported by the operating system, its meaning can be determined with

xsysrc <rc>

The start information (see slide) also contains additional information about operating system settings that are significant for database operation.

dbm.prt



The Database Manager log ***dbm.prt*** comprises the command history of the Database Manager. All change actions and all actions that return error messages are logged.

Because messages show the exact date and time, they can easily be referenced against the outputs of other log files.

If errors occur in the action being executed, they are marked ERR.

The file is stored in the run directory (default: <indepdatapath>/wrk/<SID>).

Access via DBMGUI: ***Check -> Diagnosis Files -> Database Manager Log File (DBMPRT)***

Access via DB50: ***Problem Analysis -> Messages -> Database Manager***

dbm.utl

ld1032_e70_dbm.utl.txt - Notepad

File Edit Format View Help

```
------------------------------------------------------------------------------
Date       Time     Message-Text
------------------------------------------------------------------------------
2006-03-20 18:07:58 441EE16E0001 0000   IC1 CREATE INSTANCE SYSDBA SUPERDBA
2006-03-20 18:15:05 441EE16E0001 0001   RET RETURNCODE     0
2006-03-20 18:17:11 441EE3970004 0000   SHT SHUTDOWN
2006-03-20 18:17:19 441EE39F0001 0000   REQ SET LOG WRITER OFF
2006-03-20 18:17:19 441EE39F0001 0001   RET RETURNCODE     0
2006-03-20 18:17:19 441EE39F0003 0000   RST RESTART
2006-03-20 18:17:20 441EE39F0003 0001   RET RETURNCODE     0
2006-03-20 18:41:58 441EE9660004 0000   SHT SHUTDOWN
2006-03-20 18:42:07 441EE96F0001 0000   REQ SET LOG WRITER ON
2006-03-20 18:42:07 441EE96F0001 0001   RET RETURNCODE     0

[...]

2006-03-24 17:25:21 44241D710003 0000   SDB SAVE DATA QUICK TO '/sapdb/E70/saparch/E70_COM' FILE BLOCKSIZE 8 NO CHECKPOINT MEDIANAME
2006-03-24 17:25:21 44241D710003 0001   RET RETURNCODE  -903
2006-03-24 17:25:40              0000   SDB SAVE DATA CANCEL
2006-03-24 17:25:40              0001   RET RETURNCODE  -104
2006-03-24 17:25:40 44241D840004 0000   SDB SAVE DATA QUICK TO '/sapdb/E70/saparch/E70_COM' FILE BLOCKSIZE 8 NO CHECKPOINT MEDIANAME
2006-03-24 17:33:46 44241D840004 0001   RET RETURNCODE     0
2006-03-24 17:33:46 44241D840004 0002   TAP DATE.............. 2006-03-24
2006-03-24 17:33:46 44241D840004 0003   TAP TIME.............. 17:25:40
2006-03-24 17:33:46 44241D840004 0004   TAP SERVERDB.......... E70
2006-03-24 17:33:46 44241D840004 0005   TAP SERVERNODE........ ld1032
2006-03-24 17:33:46 44241D840004 0006   TAP KERNEL VERSION.... Kernel    7.6.00 Build 018-123-119-055
2006-03-24 17:33:46 44241D840004 0007   TAP PAGES TRANSFERRED.      856784
2006-03-24 17:33:46 44241D840004 0008   TAP PAGES LEFT.........           0
2006-03-24 17:33:46 44241D840004 0009   TAP NO OF VOLUMES.....           1
2006-03-24 17:33:46 44241D840004 000A   TAP MEDIA NAME........ Full
2006-03-24 17:33:46 44241D840004 000B   TAP TAPE NAME......... /sapdb/E70/saparch/E70_COM
2006-03-24 17:33:46 44241D840004 000C   TAP TAPE ERRORTEXT.... UNDEF
2006-03-24 17:33:46 44241D840004 000D   TAP TAPE LABEL........ DAT_000000002
2006-03-24 17:33:46 44241D840004 000E   TAP IS CONSISTENT..... TRUE
2006-03-24 17:33:46 44241D840004 000F   TAP FIRST IO SEQUENCE.      128017
2006-03-24 17:33:46 44241D840004 0010   TAP LAST IO SEQUENCE.. UNDEF
2006-03-24 17:33:46 44241D840004 0011   TAP DBSTAMP1 DATE..... 2006-03-24
2006-03-24 17:33:46 44241D840004 0012   TAP DBSTAMP1 TIME..... 17:25:40
2006-03-24 17:33:46 44241D840004 0013   TAP DBSTAMP2 DATE..... UNDEF
2006-03-24 17:33:46 44241D840004 0014   TAP DBSTAMP2 TIME..... UNDEF
2006-03-24 17:33:46 44241D840004 0015   TAP BD PAGE COUNT.....      856760
2006-03-24 17:33:46 44241D840004 0016   TAP TAPEDEVICES USED..           1
2006-03-24 17:33:46 44241D840004 0017   TAP DB_IDENT.......... ld1032:E70_20060324_172540
2006-03-24 17:33:46 44241D840004 0018   TAP MAX USED DATA PNO.           0
2006-03-24 17:33:46 44241D840004 0019   TAP CONV PAGE COUNT...         464
2006-03-24 17:34:37 44241F9D0005 0000   SHT SHUTDOWN
2006-03-24 17:35:41 44241FDD0001 0000   RST RESTART
2006-03-24 17:35:45 44241FDD0001 0001   RET RETURNCODE     0
```


All commands sent to the database kernel by the utility task are logged in the file ***dbm.utl***.
The file is written by the database kernel.

This file contains detailed information about backup and restore processes, configuration changes such as the addition of volumes, information about update-statistics processes and so on.

In ***dbm.utl*** you can see whether operations have been successful from the point of view of the database kernel. When using external backup tools, it is important to take account of the corresponding log files as well, since errors can also occur on other levels during the transfer of backup information from the kernel to the tools.

The file is stored in the run directory of the database (default: <indepdatapath>/wrk/<SID>).

Access via DBMGUI: ***Check -> Diagnosis Files -> Utility Statements (UTLPRT)***

Access via DB50: ***Problem Analysis -> Logs -> Kernel Administration***

dbm.knl

bsp_dbmknl.txt - Notepad

File Edit Format Help

```
3ED21A1D0003|          |HISTLOST |                     |                     |2003-05-26 15:43:57|                     |          |          |   |
3EE5FA7B0004|DAT_00001 |SAVE WARM|2003-06-10 17:34:19|2003-06-10 17:34:19|2003-06-10 17:34:21|2003-06-10 18:59:15|     20981|          |NO |Data
3EE891540002|DAT_00001 |RESTORE  |2003-06-10 17:34:19|2003-06-10 17:34:19|2003-06-12 16:42:28|2003-06-12 16:53:37|     20981|          |NO |Data_tmp
3EE8947A0004|          |HISTLOST |                     |                     |2003-06-12 16:55:54|                     |          |          |   |
3EE894B70001|DAT_00002 |SAVE COLD|2003-06-12 16:56:22|2003-06-12 16:56:22|2003-06-12 16:56:57|2003-06-12 17:01:46|     21057|          |NO |Data
3EE896F20001|DAT_00003 |SAVE COLD|2003-06-12 17:06:03|2003-06-12 17:06:03|2003-06-12 17:06:28|2003-06-12 17:10:58|     21596|          |NO |Data
3EE89AB60001|          |HISTLOST |                     |                     |2003-06-12 17:22:30|                     |          |          |   |
            |LOG_00000 |SAVE WARM|?          ?       |?          ?       |2003-06-12 17:22:40|2003-06-12 17:25:15|        -1|        -1|   |
3EE89B860002|          |HISTLOST |                     |                     |2003-06-12 17:25:58|                     |          |          |   |
            |LOG_00000 |SAVE WARM|?          ?       |?          ?       |2003-06-12 17:38:34|2003-06-12 17:38:38|        -1|        -1|   |
            |LOG_00000 |SAVE WARM|?          ?       |?          ?       |2003-06-12 17:41:11|2003-06-12 17:41:15|        -1|        -1|   |
3EE8A8EE0003|DAT_00004 |SAVE WARM|2003-06-12 18:23:10|2003-06-12 18:23:10|2003-06-12 18:23:12|2003-06-12 18:34:05|      1394|          |NO |Data_tmp
3F268EE70004|DAT_00005 |SAVE WARM|2003-07-29 17:12:40|2003-07-29 17:12:40|2003-07-29 17:12:43|2003-07-29 17:23:17|    287309|          |NO |Data_tmp
3F2A01690006|LOG_00001 |SAVE WARM|2003-06-12 17:22:48|2003-06-29 02:12:56|2003-08-01 07:58:01|2003-08-01 07:58:46|         0|    103014|   |Log
3F2A01970007|LOG_00002 |SAVE WARM|2003-06-29 02:12:56|2003-07-16 07:33:39|2003-08-01 07:58:47|2003-08-01 07:59:35|    103015|    206147|   |Log
3F2A04430008|LOG_00003 |SAVE WARM|2003-07-16 07:38:39|2003-08-01 08:10:00|2003-08-01 08:10:11|2003-08-01 08:11:05|    206148|    312572|   |Log
3F2FA4DA0003|LOG_00004 |SAVE WARM|2003-08-01 08:10:01|2003-08-05 14:36:16|2003-08-05 14:36:42|2003-08-05 14:37:32|    312573|    388723|   |Log
3F30C9B40004|LOG_00005 |SAVE WARM|2003-08-05 14:36:16|2003-08-06 11:25:41|2003-08-06 11:26:12|2003-08-06 11:27:15|    388724|    466692|   |Log
3F30FD800005|LOG_00006 |SAVE WARM|2003-08-06 11:25:43|2003-08-06 15:06:45|2003-08-06 15:07:12|2003-08-06 15:08:07|    466693|    529822|   |Log
3F311DB40006|DAT_00006 |SAVE WARM|2003-08-06 17:24:52|2003-08-06 17:24:52|2003-08-06 17:24:55|2003-08-06 17:39:47|    542988|          |NO |Data_tmp
3F3337BB0007|DAT_00007 |SAVE WARM|2003-08-08 07:40:12|2003-08-08 07:40:12|2003-08-08 07:40:14|2003-08-08 07:41:35|    549971|          |NO |Data
3F3338250008|DAT_00008 |SAVE WARM|2003-08-08 07:41:57|2003-08-08 07:41:57|2003-08-08 07:41:59|2003-08-08 07:56:27|    549976|          |NO |Data_tmp
3F67A4480003|LOG_00007 |SAVE WARM|2003-08-06 15:06:45|2003-09-17 02:00:55|2003-09-17 02:01:12|2003-09-17 02:02:07|    529823|    749096|   |Log
3FC37C160003|LOG_00008 |SAVE WARM|2003-09-17 02:00:55|2003-11-25 16:57:49|2003-11-25 16:58:14|2003-11-25 16:59:12|    749097|   1007218|   |Log
401687F70004|LOG_00009 |SAVE WARM|2003-11-25 16:57:49|2004-01-27 16:46:42|2004-01-27 16:47:03|2004-01-27 16:48:04|   1007219|   1247065|   |Log
404074D40003|LOG_00010 |SAVE WARM|2004-01-27 16:46:42|2004-02-28 12:00:29|2004-02-28 12:00:36|2004-02-28 12:01:30|   1247066|   1399878|   |Log
404F0F090003|DAT_00009 |SAVE WARM|2004-03-10 13:50:37|2004-03-10 13:50:37|2004-03-10 13:50:41|2004-03-10 14:05:44|   1435397|          |NO |Data_tmp
```

Two examples of complete lines:

```
|LOG_00000|SAVE WARM|?       ?     |?       ?     |2003-06-12 17:22:40|2003-06-12 17:25:15|      -1|      -1|  |
                              |      0|      0|    -123|

3F3337BB0007|DAT_00007|SAVE WARM|2003-08-08 07:40:12|2003-08-08 07:40:12|2003-08-08 07:40:14|2003-08-08 07:41:35|   549971|        |NO |Data
                              |      0|      0|    -903|Could not open dev
```


The file ***dbm.knl*** contains a list of the backup and restore actions that have been executed.

You can identify what type of backup (DATA, LOG) was executed, in which time period the execution took place, up to which log page number the data was backed up, which medium was used and whether any errors occurred.

Because of the length of the output line, the file is somewhat difficult to work with; it is therefore a good idea to get a formatted display of the backup history with the DBMGUI. Errors are noted at the end of the output line.

When using external backup tools, it is important to observe their logs as well, which are described in the following pages.

The file is written by the database kernel and stored in the run directory (default: <indepdatapath>/wrk/<SID>) abgelegt.

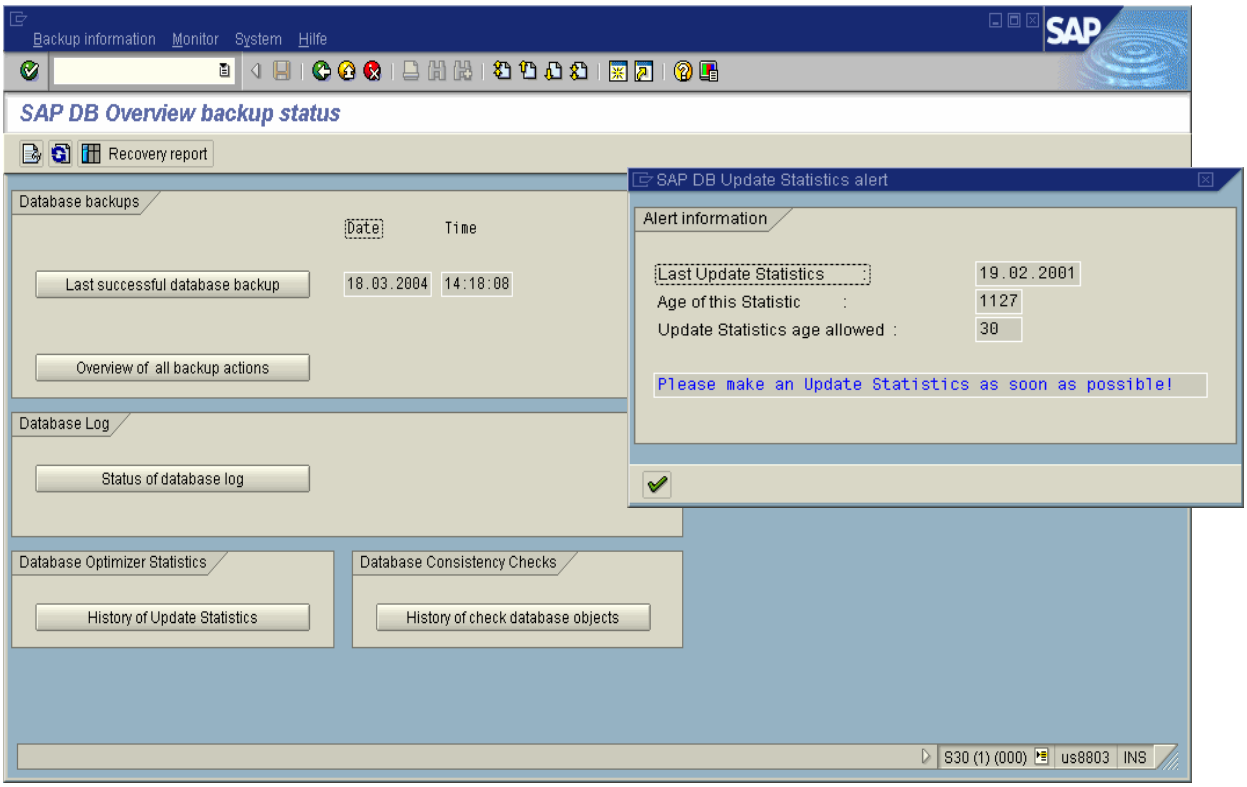
Access via DBMGUI: ***Information -> Backup History*** or ***Check -> Diagnosis Files -> Backup History (BACKHIST)***

Access via DB50: ***Problem Analysis -> Logs -> DBA History -> Backup/Restore (Kernel)***

Because of the length of the output line, the file is somewhat difficult to work with; it is therefore a good idea to get a formatted display of the backup history with the DBMGUI or transaction DB50. Errors are noted at the end of the output line.

The transactions presented on the next few pages, db12 and db13(c) use this information about backups to, for example, propose a recovery procedure.

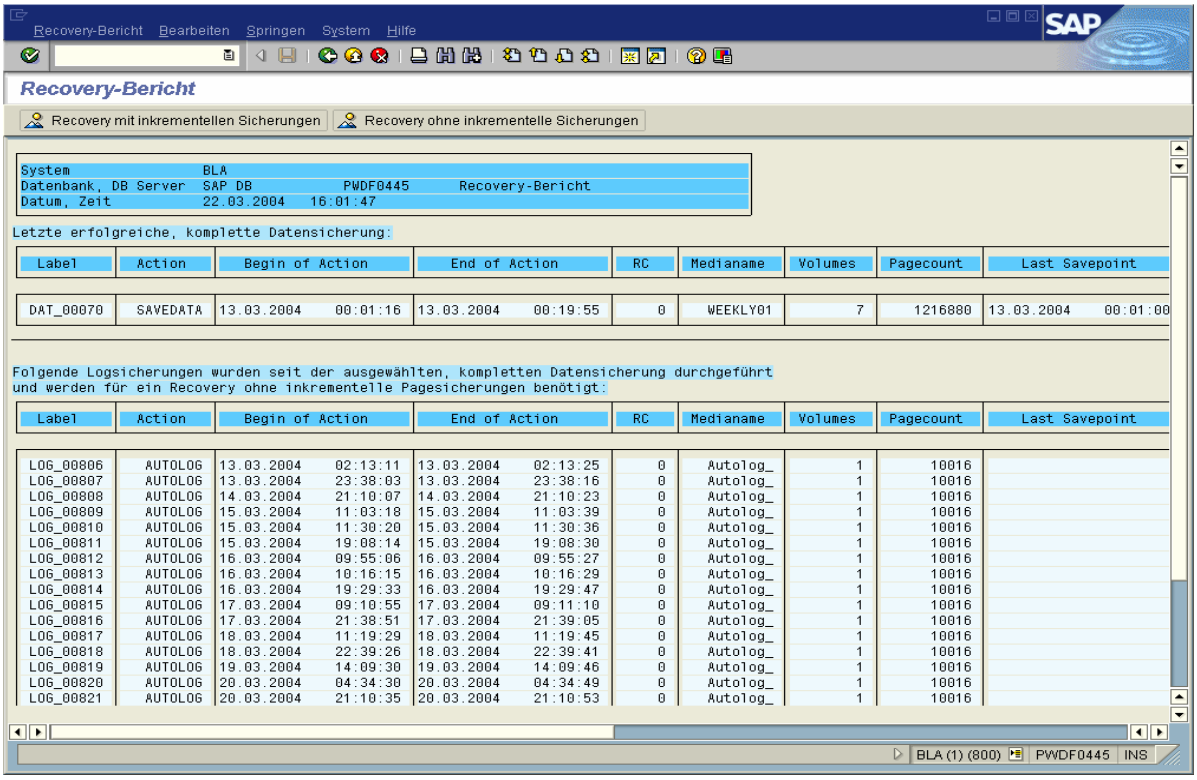
DB12 - Overview



SAP transaction db12 can be used to get an overview of backup and restore actions that have been executed.

Here you can also get information about the scope and frequency of Update Statistics operations as well as a history of consistency checks.

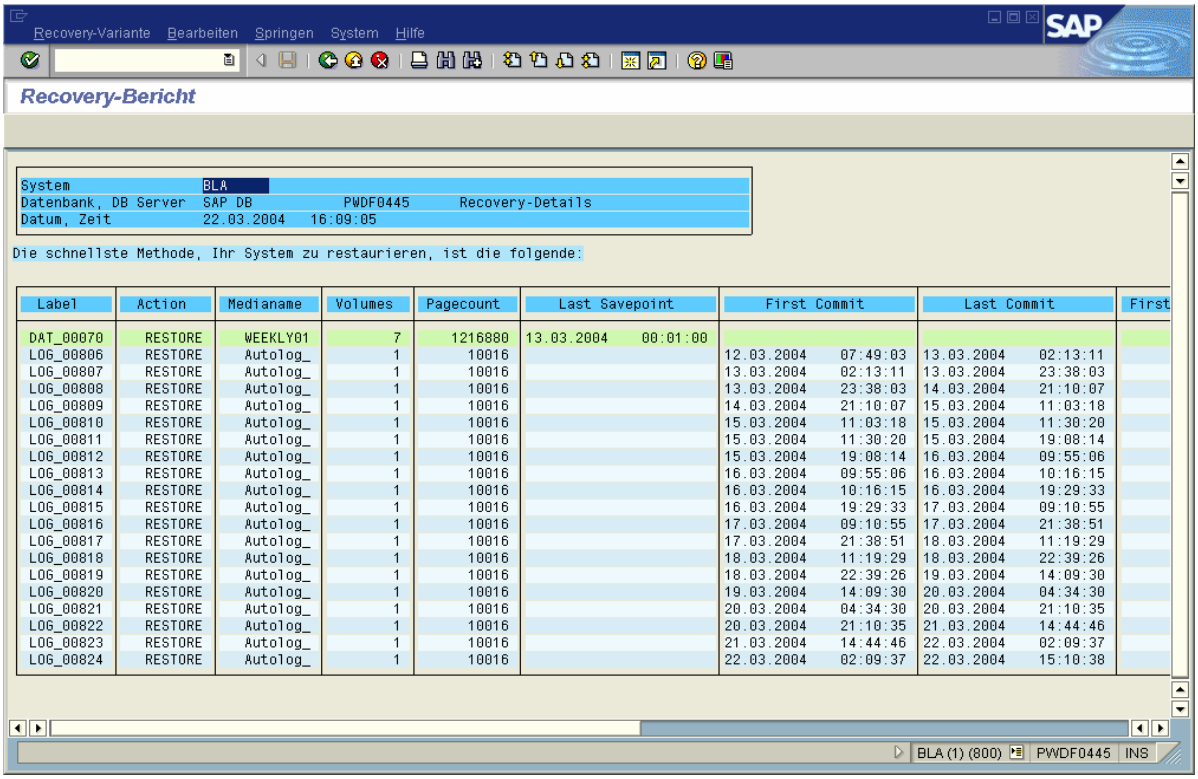
DB12 – Recovery Report (1)



The recovery report presents an ordered list of backups.

It is generated when the DBM parameter DBATL is set to 1. For further information, see the Note 431508.

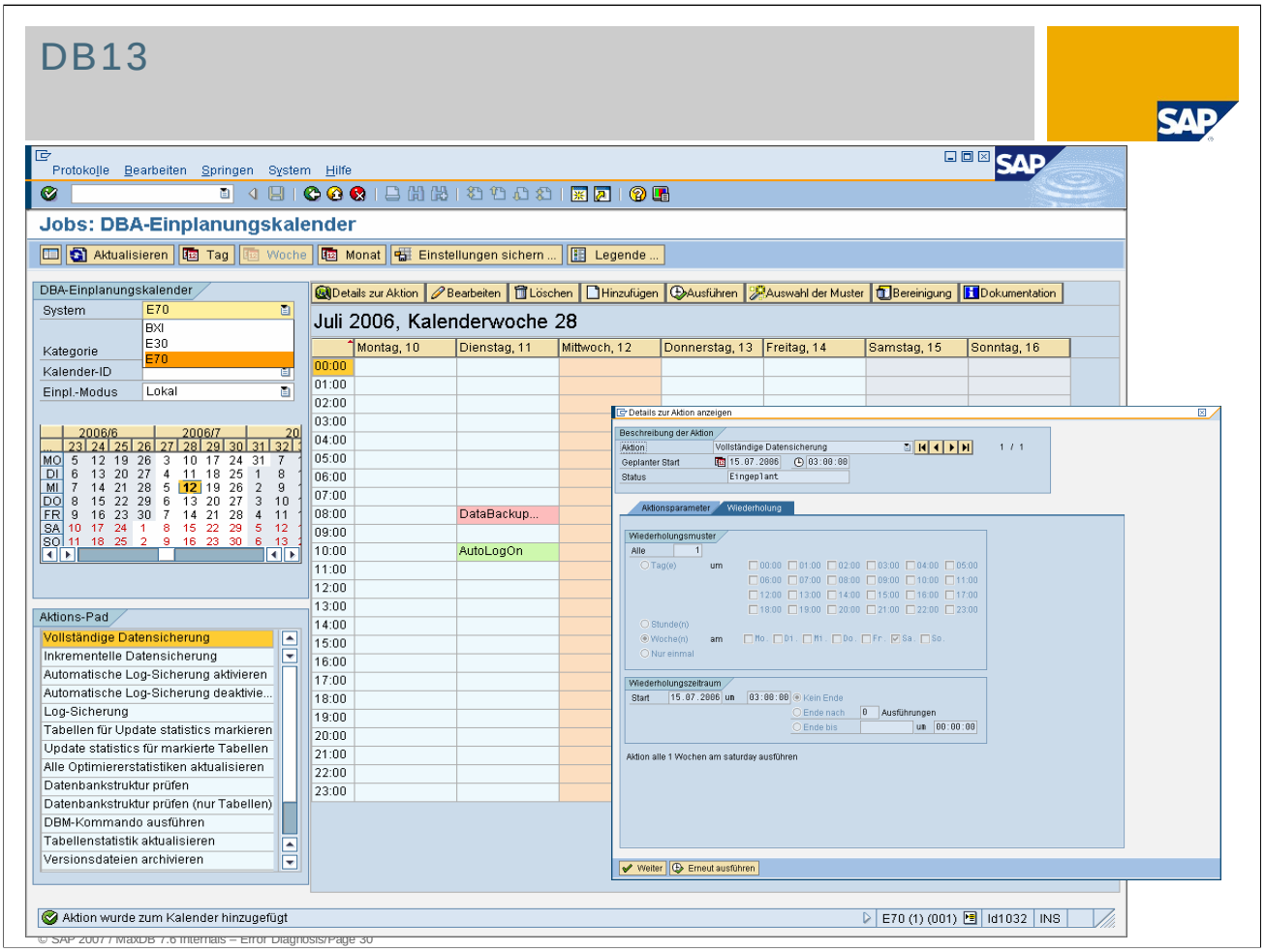
DB12 – Recovery Report (2)



If different database backups are available, the system can automatically determine the corresponding log backup.

If, alongside the log backups, incremental data backups are also performed, the recovery variant likely to be fastest can be proposed here.

Alternatives can also be useful if, for example, one of the tapes returns an error because it has been rendered unreadable due to external influences.



db13 is the scheduling calendar for backups, Update Statistics runs and consistency checks. A weekly schedule can be used to plan the regular execution of activities.

Transaction DB13C is no longer required with WebAS 7.0 since transaction DB13 allows scheduling of activities for various instances. Integrate an instance with transaction DB59. Double-click to go to the database monitoring and via Tools -> DBA Planning Calendar to transaction DB13. Now the new instance will henceforth be known in transaction DB13.

If an error occurs during an action, it is displayed with a red background.

The causes of errors can be determined with the familiar diagnosis files. The job logs may also contain information that is useful in this regard.

dbm.ebp

External Backup Protocol (dbm.ebp)

- Is created by each action using a supported backup tool.
- Is overwritten with every start of the DBM server, if it communicates with an external backup tool.

Contents

- Configuration values
- Commands of the database kernel
- Call of the backup tools
- Returncodes of the backup tools and of the database kernel
- Output of the backup tools

For diagnosing problems with backups using external backup tools, the log file ***dbm.ebp*** plays a decisive role.

It is stored in the run directory of the database (default: <indepdatapath>/wrk/<SID>.

Using the DBMGUI, you can access the file as follows:

Check -> Diagnosis Files -> External Backup Protocol

Zugriff via DB50: ***Eigenschaften -> Dateien -> BACKEBP***

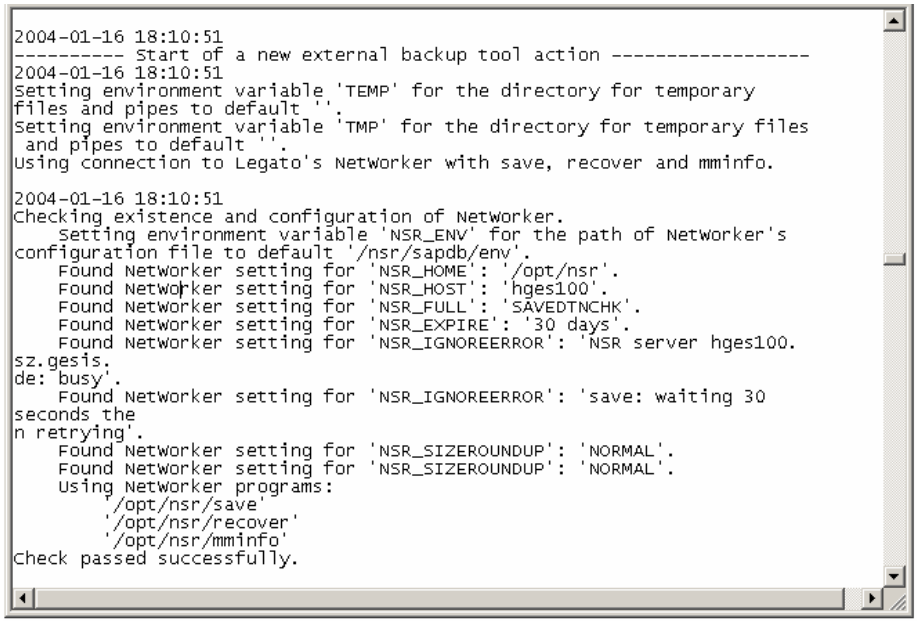
Note that this file is overwritten after each start of the DBM server when it communicates with the external backup tool. A new DBM server is started with each dbmcli call, to name one example.

In addition to information about the configuration parameter of the tool, *dbm.ebp* contains information about the commands sent to the database kernel as well as the backup tool call. The error position makes it possible to identify who was responsible for the problem.

dbm.ebl (1)

External Backup Log

- Contains a summary of the last <n> dbm.ebp files.
- The number is configurable.



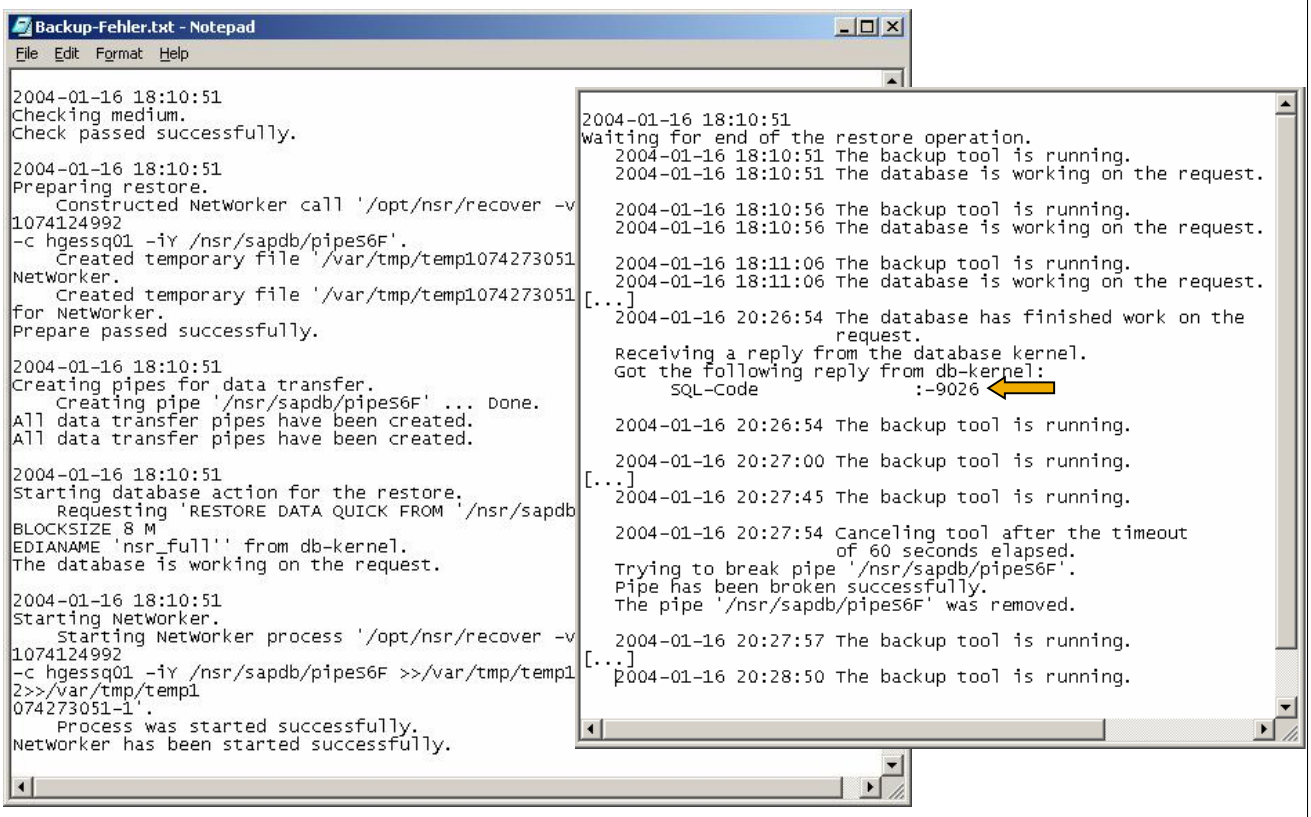
```
2004-01-16 18:10:51
---------- Start of a new external backup tool action ------------------
2004-01-16 18:10:51
Setting environment variable 'TEMP' for the directory for temporary
files and pipes to default ''.
Setting environment variable 'TMP' for the directory for temporary files
 and pipes to default ''.
Using connection to Legato's NetWorker with save, recover and mminfo.

2004-01-16 18:10:51
Checking existence and configuration of NetWorker.
    Setting environment variable 'NSR_ENV' for the path of NetWorker's
configuration file to default '/nsr/sapdb/env'.
    Found NetWorker setting for 'NSR_HOME': '/opt/nsr'.
    Found NetWorker setting for 'NSR_HOST': 'hges100'.
    Found NetWorker setting for 'NSR_FULL': 'SAVEDTNCHK'.
    Found NetWorker setting for 'NSR_EXPIRE': '30 days'.
    Found NetWorker setting for 'NSR_IGNOREERROR': 'NSR server hges100.
sz.gesis.
de: busy'.
    Found NetWorker setting for 'NSR_IGNOREERROR': 'save: waiting 30
seconds the
n retrying'.
    Found NetWorker setting for 'NSR_SIZEROUNDUP': 'NORMAL'.
    Found NetWorker setting for 'NSR_SIZEROUNDUP': 'NORMAL'.
    Using NetWorker programs:
        '/opt/nsr/save'
        '/opt/nsr/recover'
        '/opt/nsr/mminfo'
Check passed successfully.
```

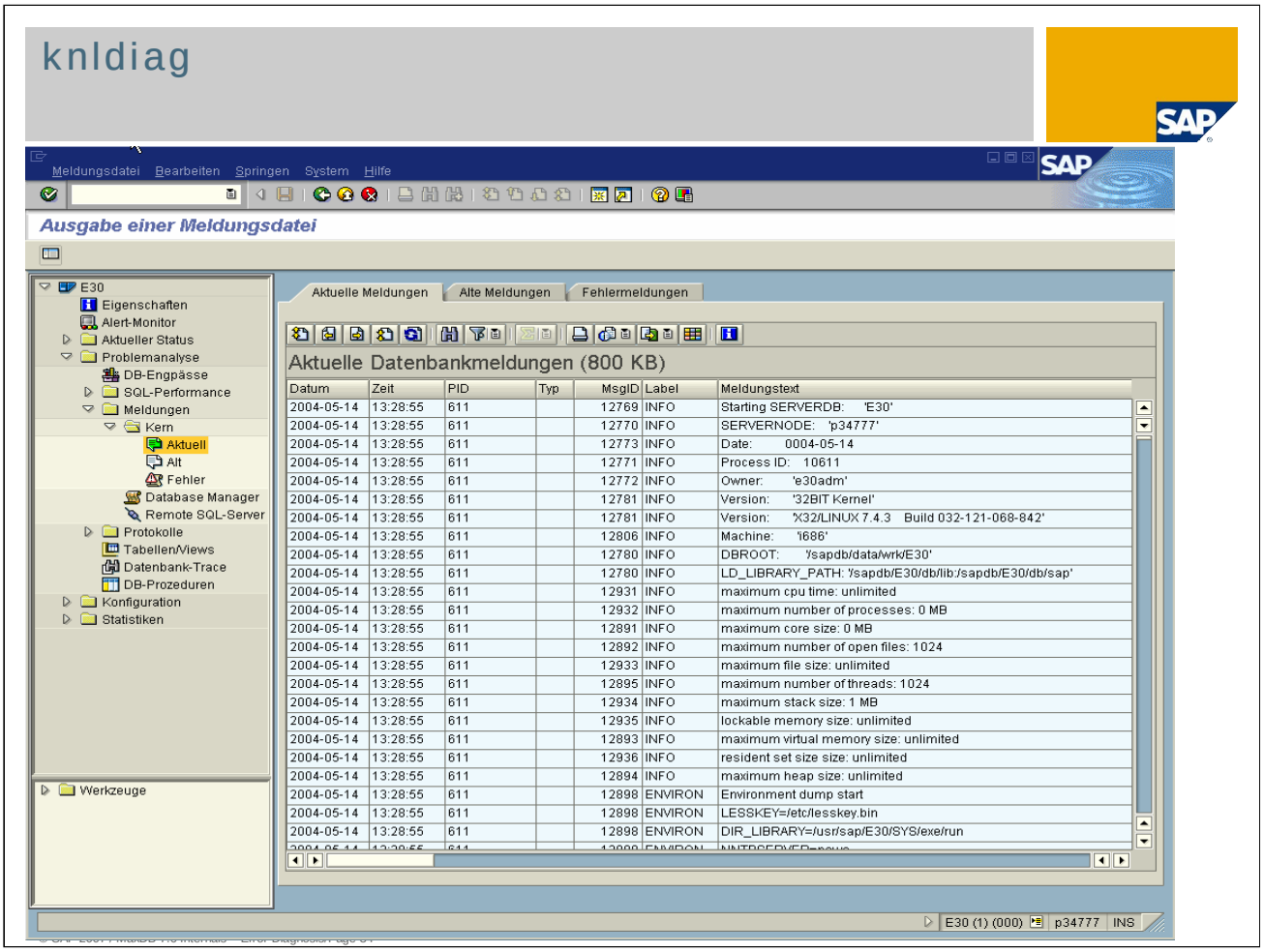

Because the file *dbm.ebp* is promptly overwritten, there is a summary of it called ***dbm.ebl***. This log file contains the last <n> logs, the number of which can be configured with the DBM parameter DBM_EBLSIZE. The file ***dbm.ebl*** is stored in the run directory of the database ((default: <indepdatapath>/wrk/<SID>).

Access via DBMGUI: ***Check -> Diagnosis Files -> External Backup Log***

Access via DB50: ***Properties -> Files -> DBMEBL***

dbm.ebl (2)





The ***knldiag*** file contains messages that occur during communications between the MaxDB kernel and the runtime environment. This file is recreated each time the database instance is started. The previous file is renamed ***knldiag.old***. The messages - apart from the header (start messages) - are overwritten cyclically.Error messages are recorded in ***knldiag*** but also - due to the risk that they will be overwritten there - in the file ***knldiag.err***. This file is written continuously.

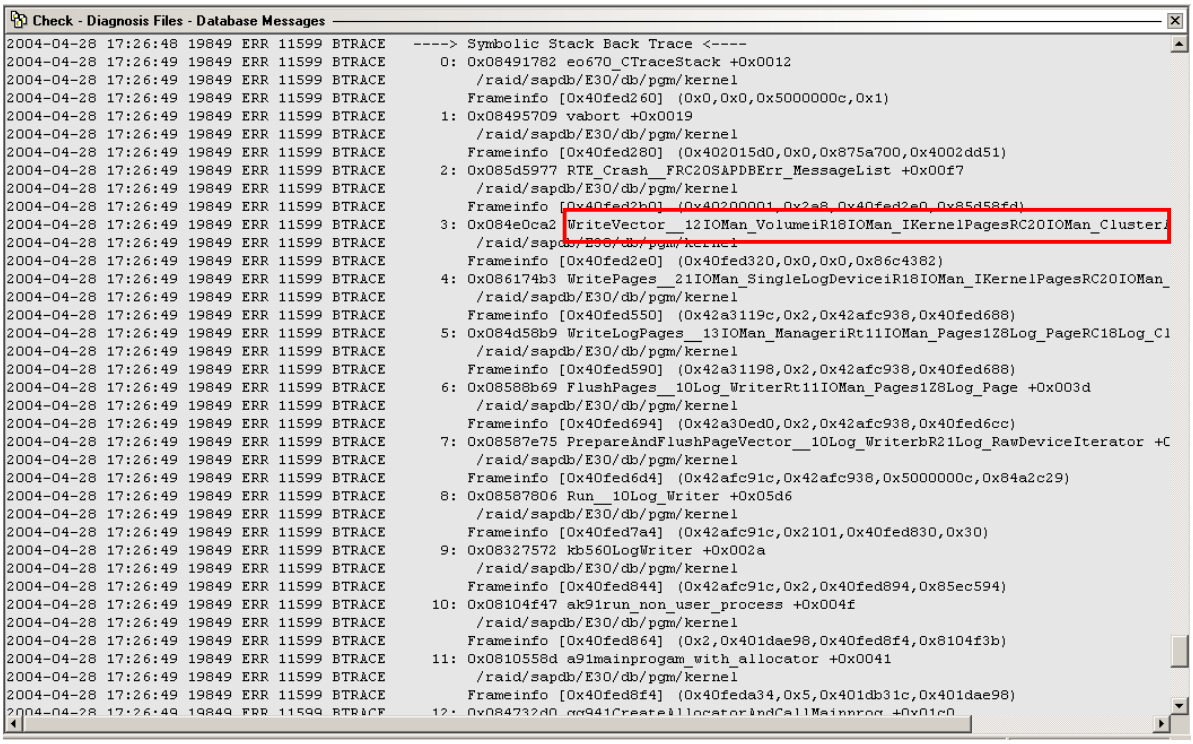
The file ***knldiag*** is stored in the run directory of the database ((default: <indepdatapath>/wrk/<SID>).

Location, size and name of the file can be changed with the setting for the parameters _KERNELDIAGFILE and KERNELDIAGSIZE.

Access via DBMGUI: ***Check -> Diagnosis Files -> Database Messages***

Access via DB50: ***Problem Analysis -> Logs -> Kernel Administration***

Stack Backtrace in knldiag



When the database crashes, support often needs to know at which point in the source code the database was when the crash occurred.

On Unix/Linux, this information is usually generated from a core dump with a debugger. On Windows, this information is found in the file drwtsn32.log, but only if Dr Watson is registered as the system debugger.

Core dumps can be very large. Writing a core dump delays the crash of the process.

For that reason, when a crash occurs the MaxDB kernel automatically writes the backtrace stack and values of the CPU register to the knldiag file.

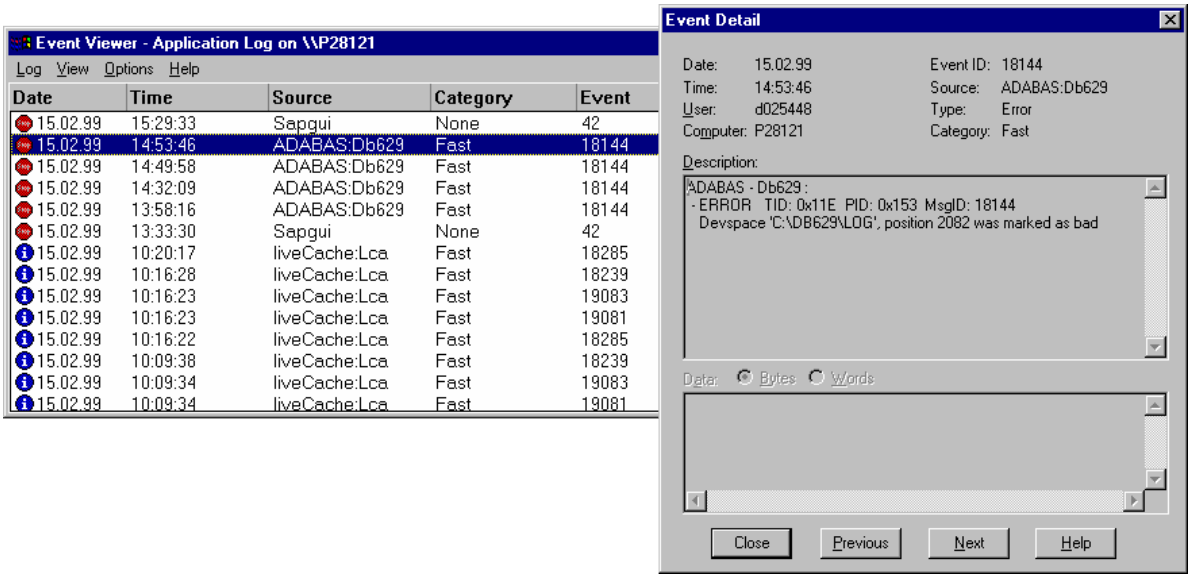
If the problem is due to an error in the database software, the cause can usually be found using this information.

In the present example we see a simulation of an I/O error during writing to a log volume. It is not a software error.

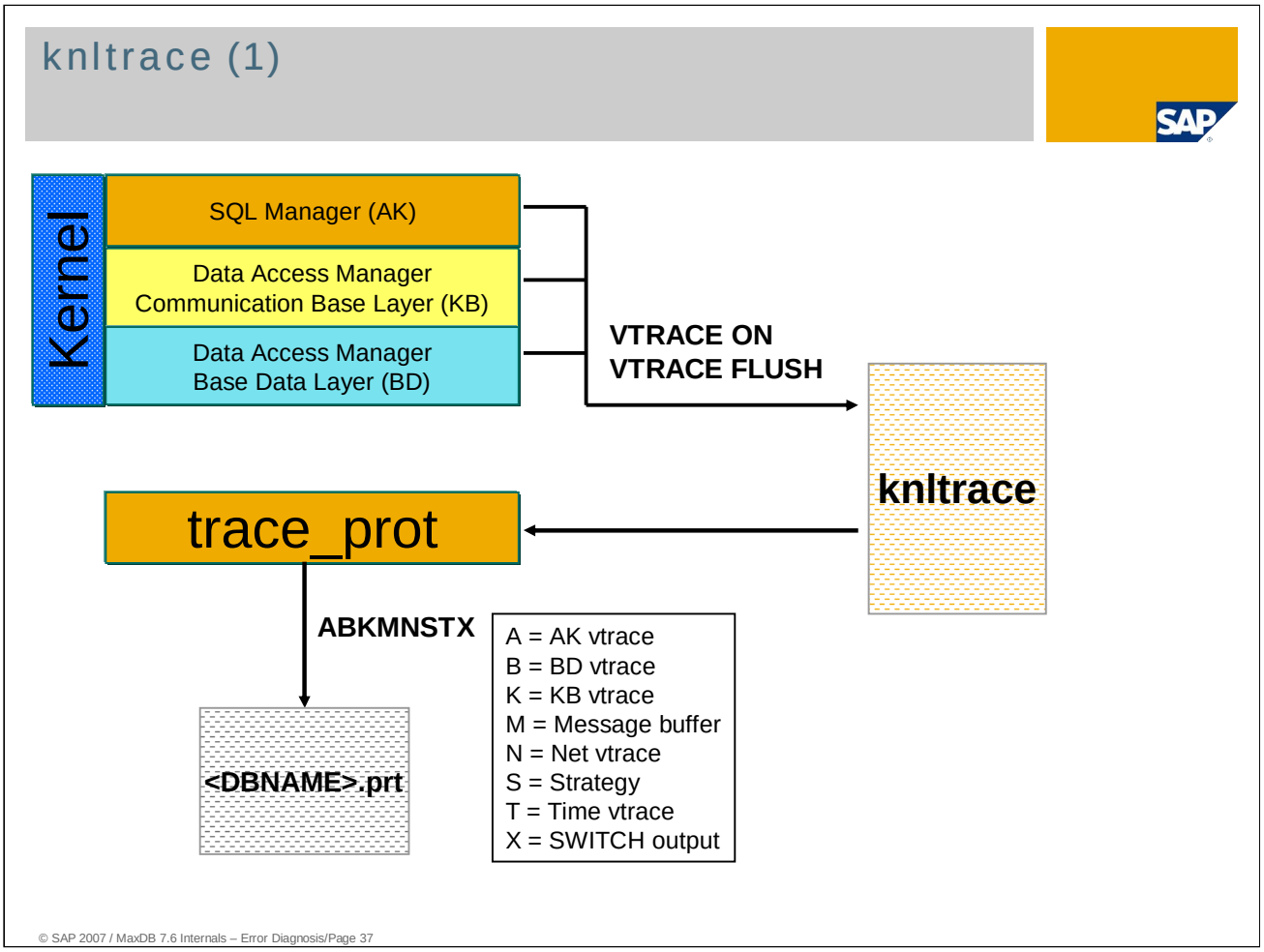
Event Viewer

Under WINDOWS NT important messages are additionally written to the event log.

Example:



If the file *knldiag* has already been overwritten, you may still find useful information here.



The kernel trace, or Vtrace, is used for analyzing executed SQL statements.

When you activate Vtrace, you specify which areas of the kernel the file *knltrace* is written from. Generally a default setting is taken.

The kernel trace is not active by default. The default trace has a minimal effect on system performance. Each UKT writes to its own main memory buffer, precluding collisions during trace writing. If you select other options, however, writing the trace can be resource intensive and should be done only where needed for problem analysis.

For the trace output, you specify the levels or modules of the kernel for which the logs are to be extracted.

Data concerning strategies and times is only output if the options OPTIMIZER or TIME, respectively, are active for the Vtrace.

The SWITCH output contains data from the trace of a so-called slow kernel. A slow kernel is a special MaxDB debugging kernel. It is only used upon the special request of development or support.

The Vtrace can be activated for a single user session (FOR SESSION).

To prevent cyclical overwriting, the writing of the trace can be switched off automatically when a specified error code occurs (STOP ON ERROR).

knltrace (2) - Procedure

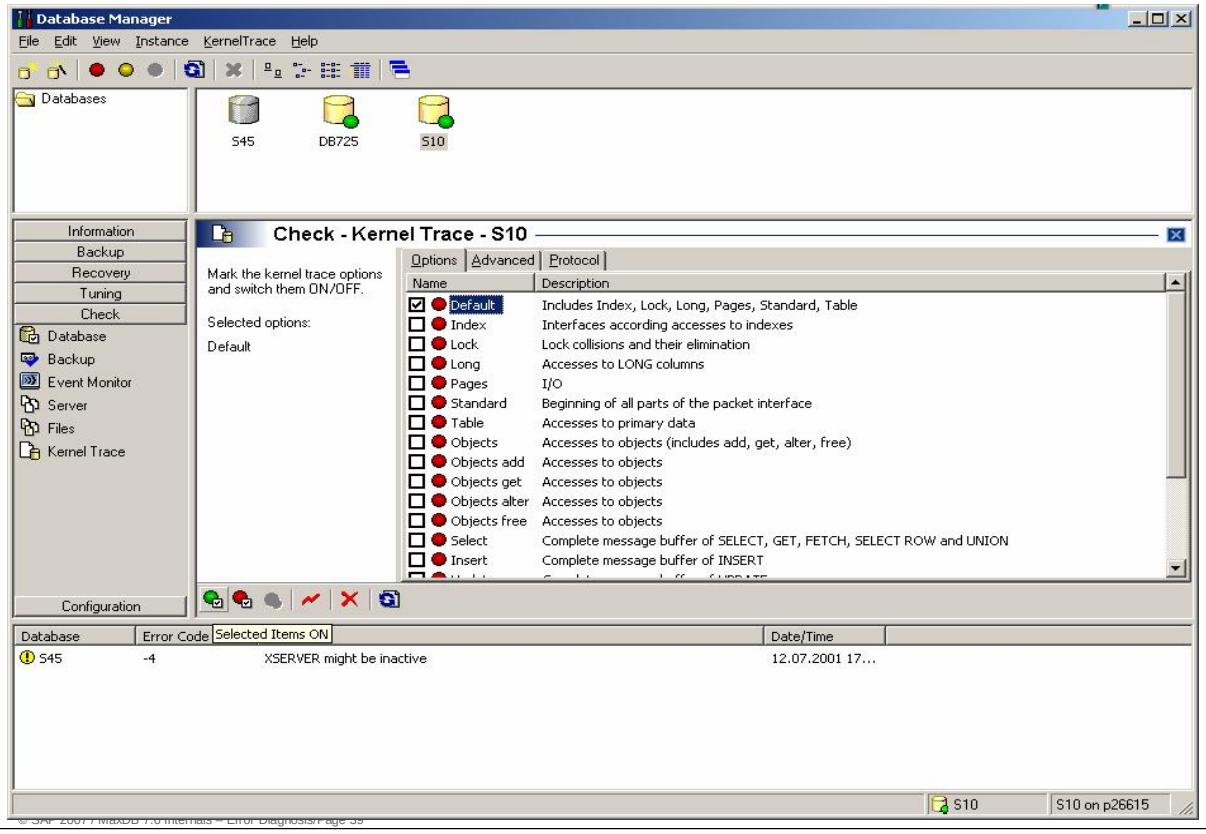
- Switch on the VTRACE
- Execute database action
  (as single database user, if possible)
- VTRACE FLUSH
- Switch off the VTRACE
- Convert the file to ASCII format

Switching the Vtrace on or off as well as flushing it can be done with the dbmcli, the DBMGUI or with transaction db50. Flushing the Vtrace can also be done with the SQLSTUDIO.

Required 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>

DBMGUI Kernel Trace (1)



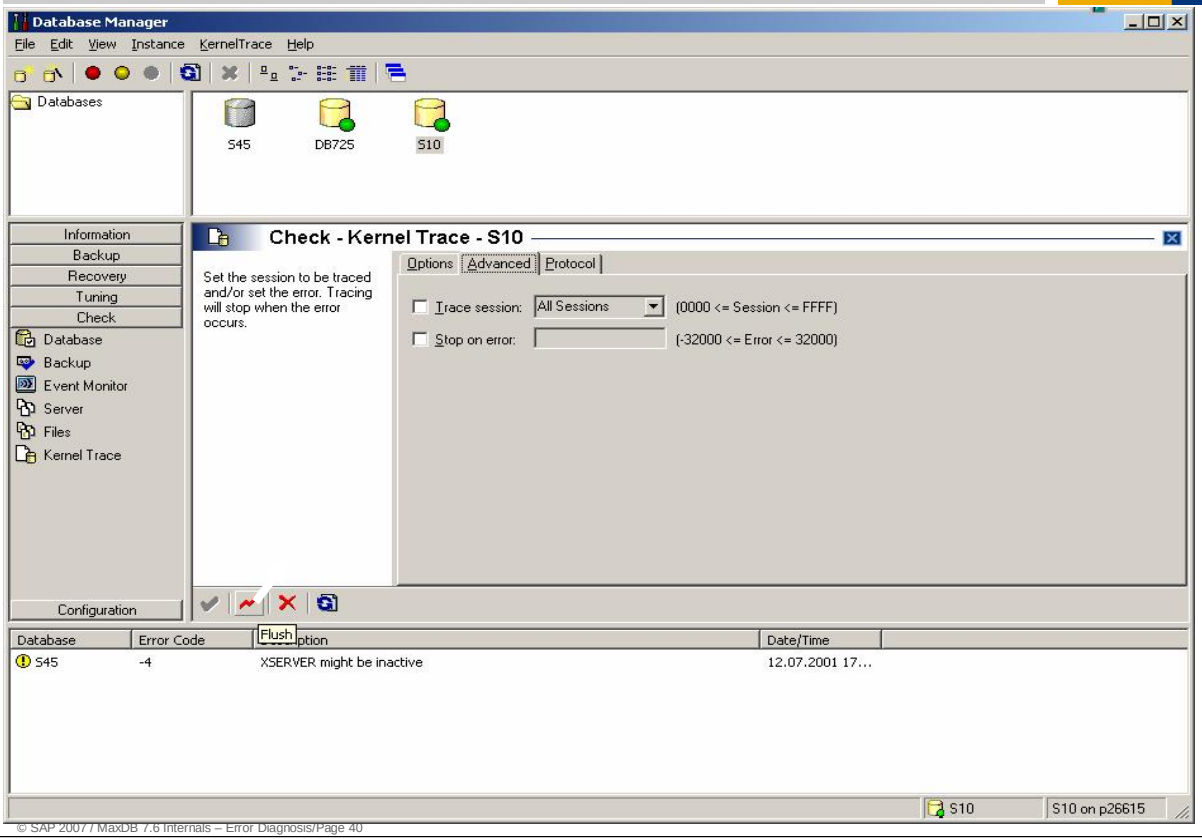
You can administer the database kernel trace with the DBMGUI.

Unless otherwise specified by development or support, the default Vtrace is sufficient.

You can also activate information about DELETE, INSERT, UPDATE, SELECT and Optimizer operations.

The Vtrace can be activated and deactivated, flushed, initialized and displayed using the buttons. During initialization, all information in the trace buffer is deleted.

Kernel Trace (2)

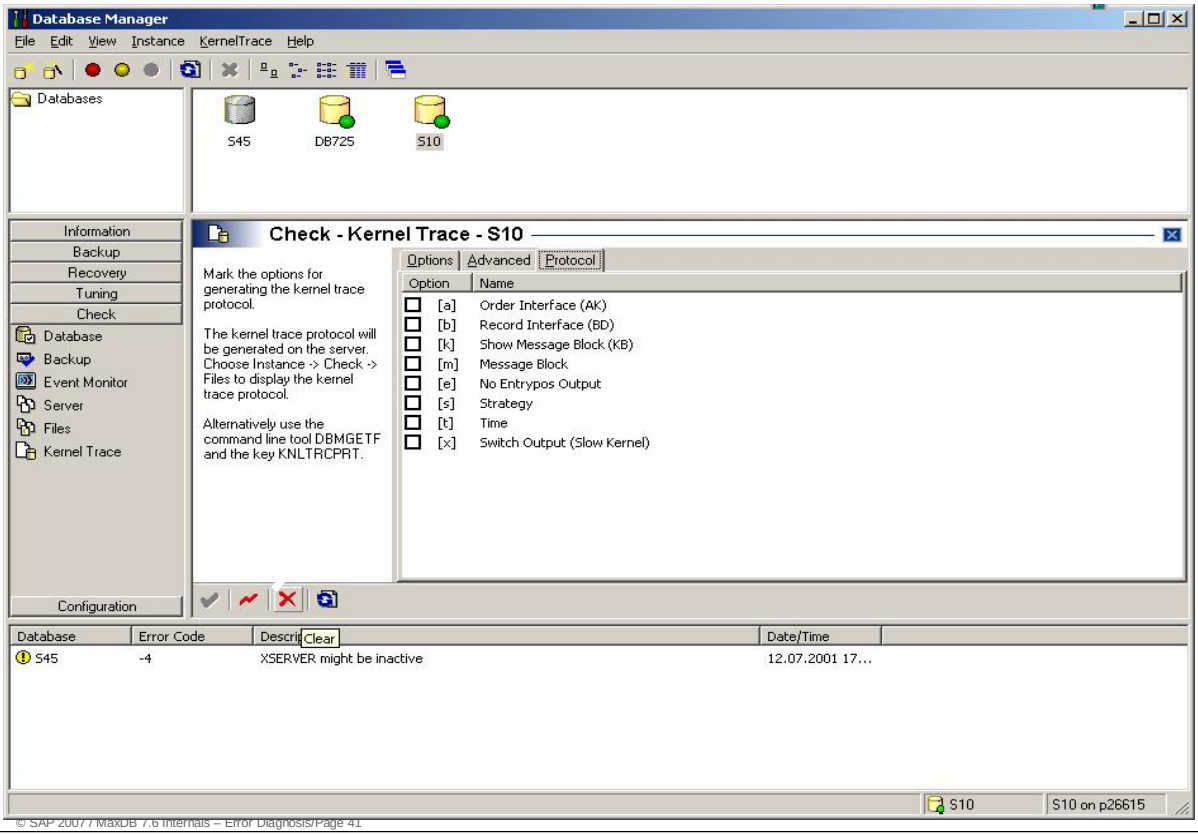


TRACE SESSION
The Vtrace can be activated for particular database sessions. To do so, however, the database session must be known.

The ouputs of
`x_cons <SID> show active` and
`SELECT * FROM TRANSACTIONS`
are helpful in this regard.

STOP ON ERROR
You can set the Vtrace so that it is automatically switched off when a certain error occurs. This is useful when you want to reproduce a particular problem and know which error will occur. This function prevents relevant information from being overwritten.

Kernel Trace (3)



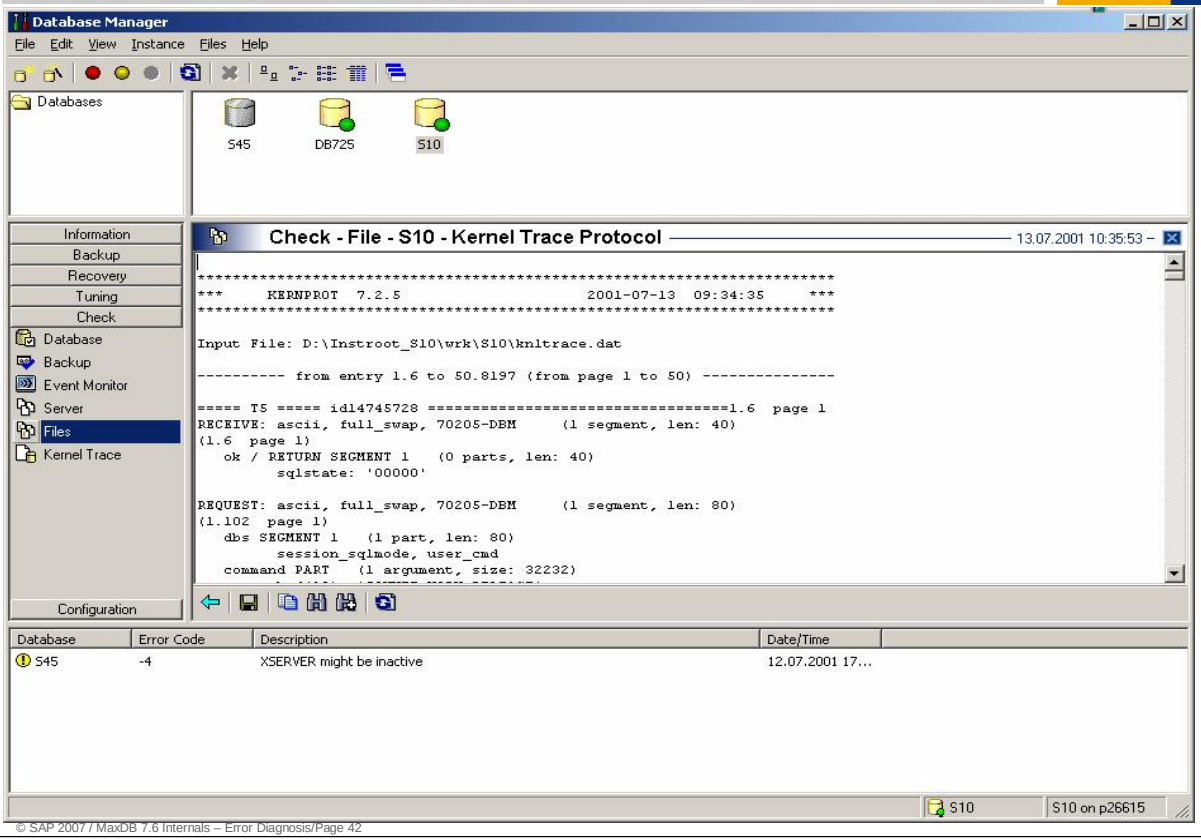
On the Log tab, you can sort the information from the *knltrace* file and extract desired areas to an ASCII file.

You specify the layers or modules of the kernel for which you want to extract the trace outputs; DEFAULT: abkmx.

Data concerning strategies and times is only output if the options OPTIMIZER or TIME, respectively, are active for the Vtrace.

The SWITCH output contains data from the trace of a so-called slow kernel. A slow kernel is a special MaxDB debugging kernel. It is only used upon the special request of development or support.

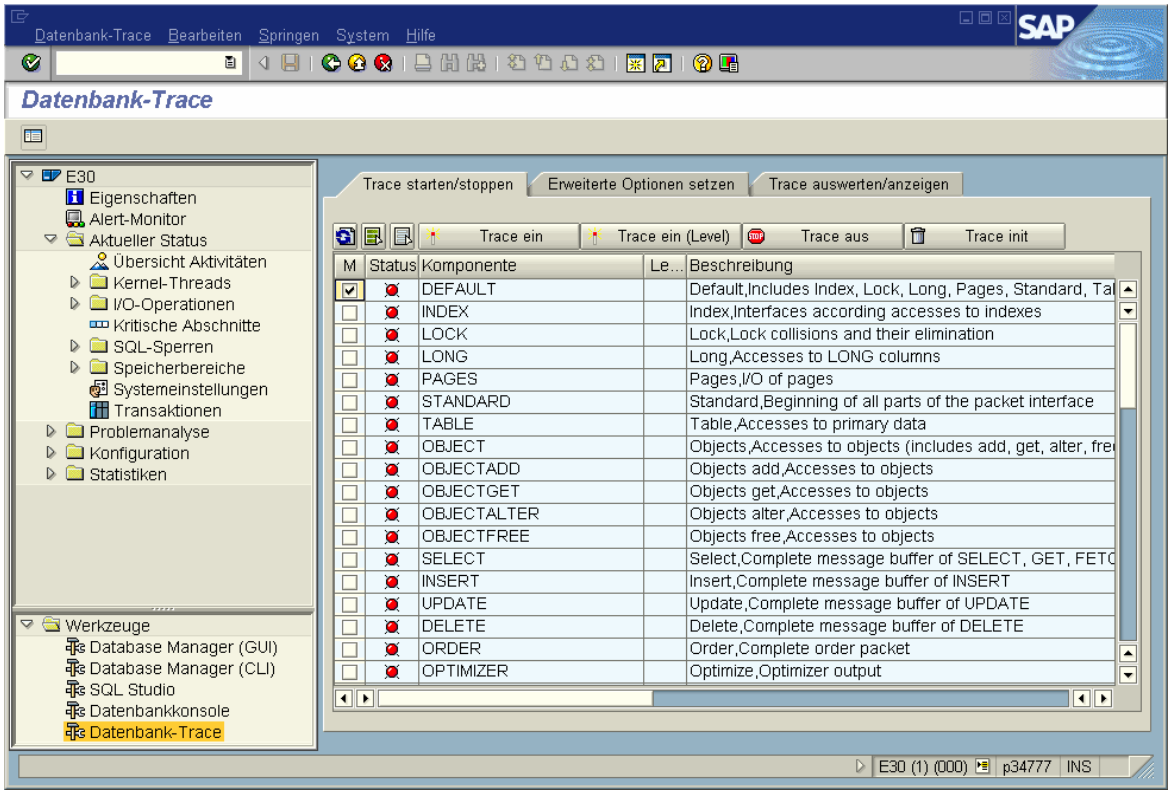
Kernel Trace (4)



You can display the contents of the Vtrace via the menu path ***‚Check -> Files -> Kernel Trace Protocol‘***

Even if you can find the evaluated error using the search function, it is all but impossible for a customer to form an independent interpretation of this trace. Errors can be found here only with knowledge of the source code. Thus the trace file should be provided to development.

DB50: Problem Analysis – Kernel Trace (1)



Administering the kernel trace (vtrace) can also be done with transaction db50.

Initialize Trace: If you want to be sure that only subsequent database actions are logged, choose *Initialize Trace*.

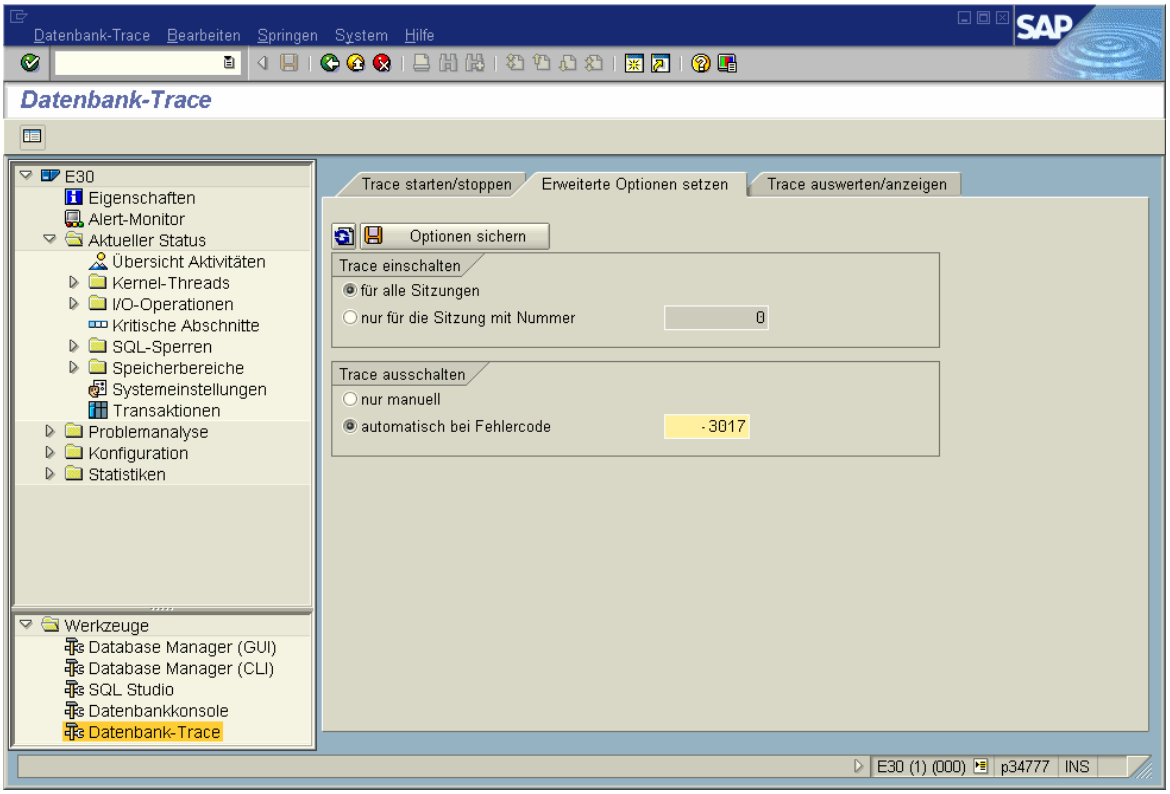
Activate Trace: To activate the trace, first choose your trace options, (usually default options) and then *Activate Trace*.

You can activate more trace options while the trace is running by selecting them and choosing *Activate Trace* again.

Then the program that received the short dump, for example, is restarted.

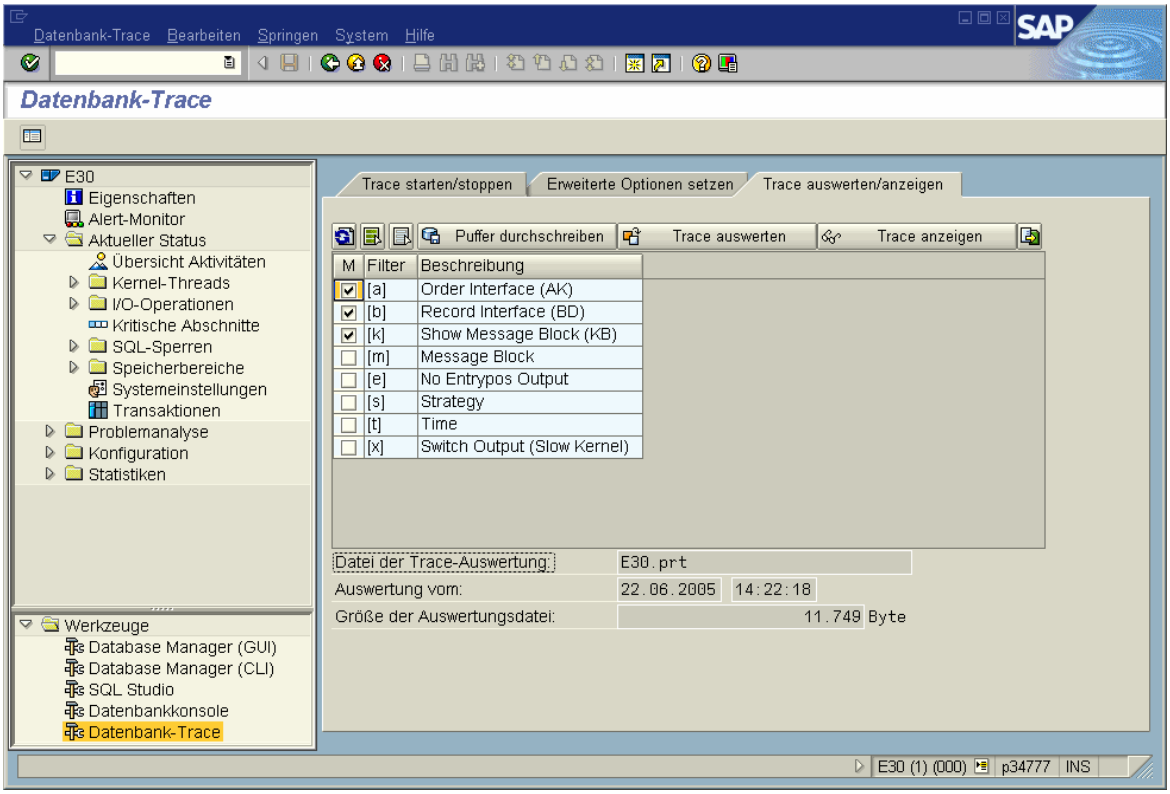
The "Status" column shows whether the trace is currently activated, and with which options. The activated options are displayed in green.

DB50: Problem Analysis – Kernel Trace (2)



On the ***'Set Adabced Options'*** tab, you can determine whether the kernel trace should be written only for a selected session and whether it should be stopped automatically in case of a selected error code in order to prevent overwriting.

DB50: Problem Analysis – Kernel Trace (3)



When the program you want to check has been terminated, the Vtrace has to be flushed so that the information in the buffer is written to the disk.

Flush Trace Buffer : To analyze the trace, choose ***‚Fluch Trace Buffer'*.**

Format and Display Trace : To format and display the trace in a legible form, first select the desired layers and then ***Analyze Trace'***

Display Trace : Immediate display can be effectuated with ***Display Trace***.

As the resulting file <SID>.prt can attain a considerable size, you can use the right-hand button to save to a local file.

The extracted trace is then read and analyzed by support and development.

knldump

Contains the global memory, e.g.:

- Lock lists, data cache, catalog cache, ...
- Administration structures of these caches

The file is created in the following ways:

- DIAGNOSE: by a user with DBA rights
- dbmcli: db_stop –dump
- when the database crashes

The file ‘knldump’ might become very huge. It contains binary data which can be transferred to a readable form with DIAGNOSE.

In case of crash or hanger situations due to manual interventions, the database generates a dump that contains the information from the global memory.

UNIX: No dump is written if the database crashes due to a UNIX signal.

The file ***knldump*** is stored in the run directory of the database ((default: <indepdatapath>/wrk/<SID>).

If there's not enough memory here, for example, you can change the location and name of the file with the parameter _KERNELDUMPFILE.

As this is a binary file, displaying it with the DBMGUI or transaction db50 is not useful.

rtedump (1)

Status of the runtime environment in case of crash

- x_cons <SID> show all
- helps to identify previously active tasks
- detailed information about the individual tasks
- region statuses, suspend reasons, counter statistics

Additional analysis for knldiag output

No formatting necessary (legible file)

Storage in run directory

If a crash occurs, the status of the runtime environment is recorded in an ***rtedump***.

The data corresponds to that in the output of *x_cons <SID> show all*

The file can be viewed directly in a system editor; no further formatting with a tool is required.

The file ***rtedump*** is stored in the run directory of the database ((default: <indepdatapath>/wrk/<SID>). Location and name of the file can be changed with the setting for the parameter _RTEDUMPFILE.

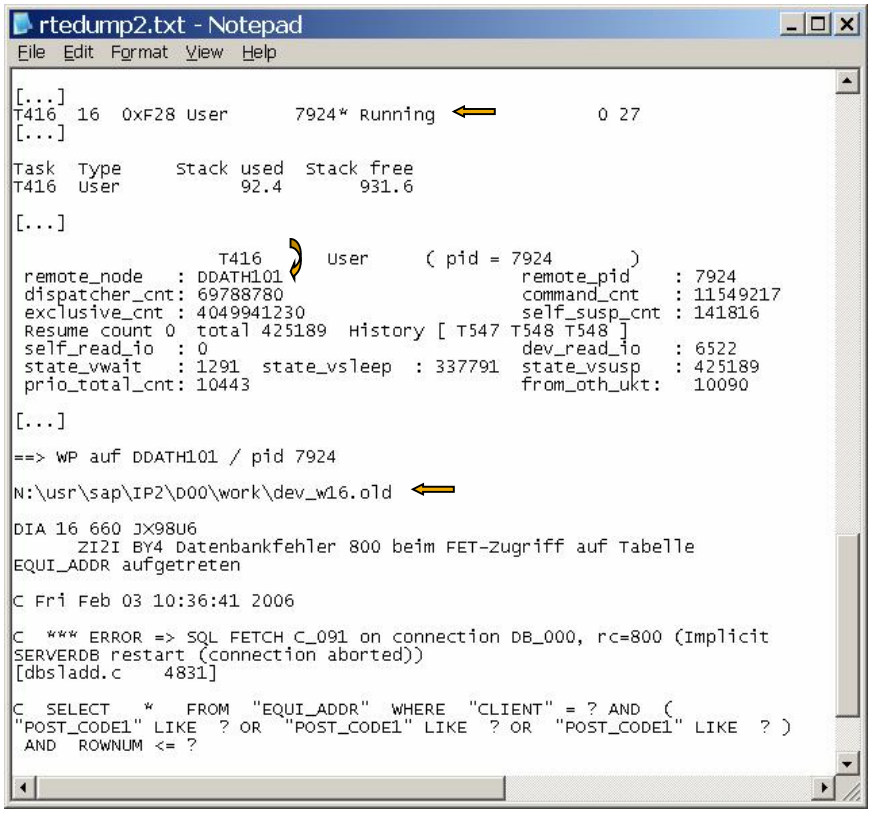
Access via DBMGUI: ***Check -> Diagnosis Files -> Runtime Environment Dump (RTEDUMP)***

Access via DB50: ***Properties -> Files -> RTEDUMP***

rtedump (2)

Example:

- Identification of the active task (status "Running")
- Detailed information about this task
- Identification of the affected application server
- If necessary, additional information in the system log as well as in the dev log tell you more about the triggering command.



In addition to the information from ***knldiag*** , the output of ***rtedump*** can be of use in analyzing crashes. This can be the case for a variety of reasons as this dump contains a plethora of information from the runtime environment. But these special cases will not be discussed further here.

An example is shown on the slide. ***rtedump*** can help identify the command that caused a crash by determining the tasks that were active at the time. They are in the *x_cons <SID> show task* part of the output and marked "Running". In the detailed information for each individual task you'll find the application server under “remote_node”. In the system log or the dev logs of this application server, commands are logged that led to some problem. Even if it cannot be guaranteed that the identified command was solely responsible for the crash, it is still worthwhile to try to reproduce the crash and (for instance with activated traces) determine the cause of the error.

*.bad, *.cor

Dump of corrupt pages

- Checksum error: *.bad
- Problem with page content identified: *.cor

Storage in run directory

Formatting done with x_diagnose

If corrupt pages are identified, they are written to the file system so they can be subjected to further analysis.

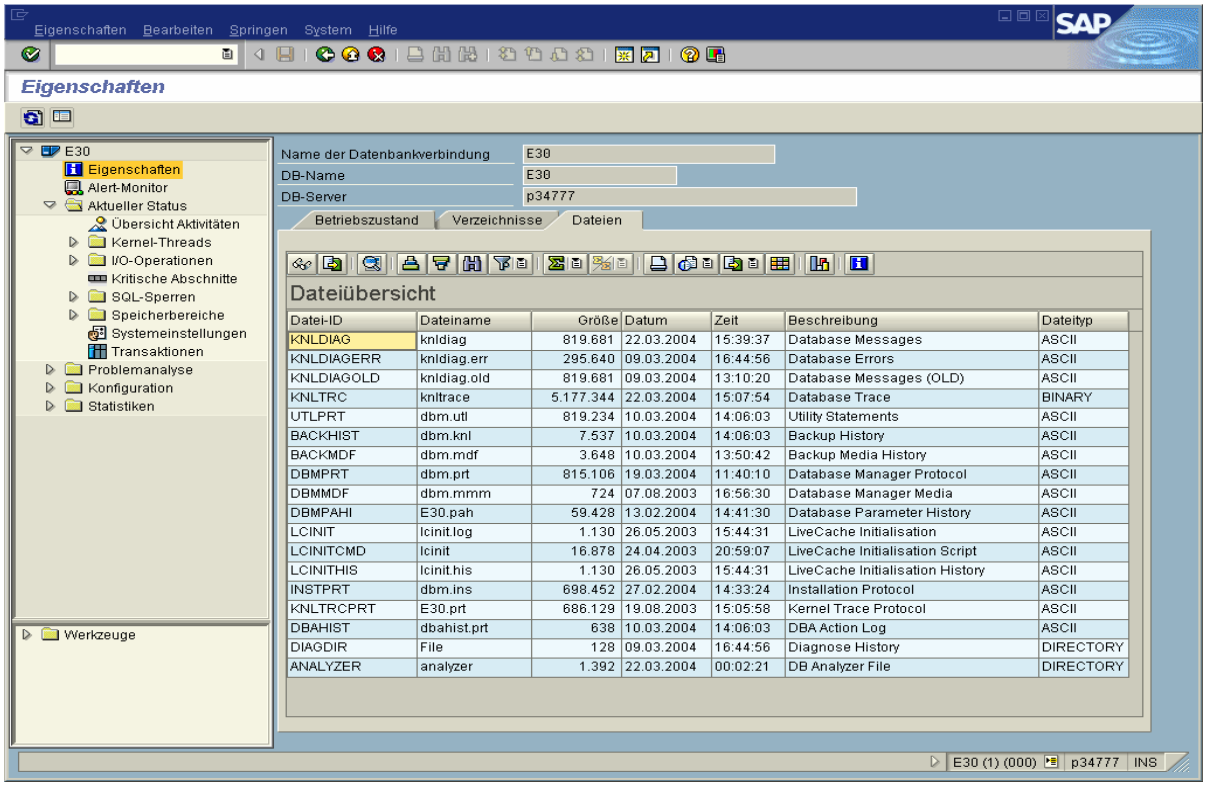
A corrupt page is dumped as a *.bad file if the I/O check found an error while importing a page.

A *.cor file is generated if a content problem is identified with the available context knowledge while working with a page in the cache.

The files are generated in the run directory of the database ((default: <indepdatapath>/wrk/<SID>.

As these are binary files, display with DBMGUI or transaction db50 is not useful. Evaluation is done with the tool x_diagnose.

DB50 – Access to Diagnosis Files



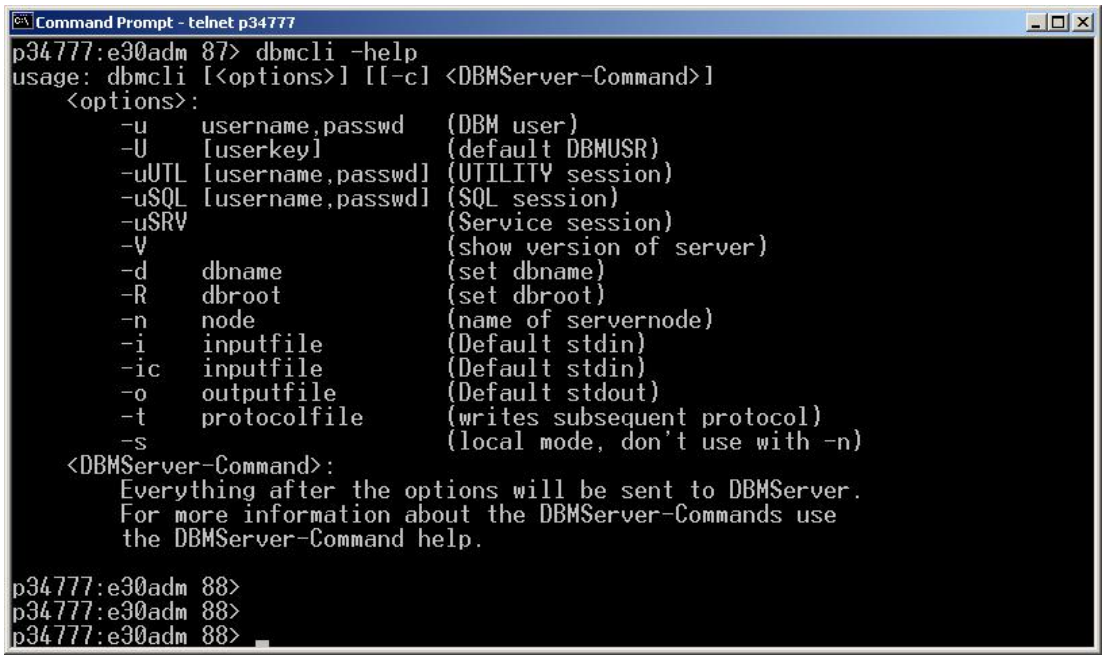
Datei-ID	Dateiname	Größe	Datum	Zeit	Beschreibung	Dateityp
KNLDIAG	knldiag	819.681	22.03.2004	15:39:37	Database Messages	ASCII
KNLDIAGERR	knldiag.err	295.640	09.03.2004	16:44:56	Database Errors	ASCII
KNLDIAGOLD	knldiag.old	819.681	09.03.2004	13:10:20	Database Messages (OLD)	ASCII
KNLTRC	knltrace	5.177.344	22.03.2004	15:07:54	Database Trace	BINARY
UTLPRT	dbm.utl	819.234	10.03.2004	14:06:03	Utility Statements	ASCII
BACKHIST	dbm.knl	7.537	10.03.2004	14:06:03	Backup History	ASCII
BACKMDF	dbm.mdf	3.648	10.03.2004	13:50:42	Backup Media History	ASCII
DBMPRT	dbm.prt	815.106	19.03.2004	11:40:10	Database Manager Protocol	ASCII
DBMMDF	dbm.mmm	724	07.08.2003	16:56:30	Database Manager Media	ASCII
DBMPAHI	E30.pah	59.428	13.02.2004	14:41:30	Database Parameter History	ASCII
LCINIT	lcinit.log	1.130	26.05.2003	15:44:31	LiveCache Initialisation	ASCII
LCINITCMD	lcinit	16.878	24.04.2003	20:59:07	LiveCache Initialisation Script	ASCII
LCINITHIS	lcinit.his	1.130	26.05.2003	15:44:31	LiveCache Initialisation History	ASCII
INSTPRT	dbm.ins	698.452	27.02.2004	14:33:24	Installation Protocol	ASCII
KNLTRCPRT	E30.prt	686.129	19.08.2003	15:05:58	Kernel Trace Protocol	ASCII
DBAHIST	dbahist.prt	638	10.03.2004	14:06:03	DBA Action Log	ASCII
DIAGDIR	File	128	09.03.2004	16:44:56	Diagnose History	DIRECTORY
ANALYZER	analyzer	1.392	22.03.2004	00:02:21	DB Analyzer File	DIRECTORY

With SAP transaction db50, error diagnosis can be performed for a running (online) database using the SAPGUI. Which tool you use is a matter of personal preference; however, this redundance is often useful, for instance if only certain activity types or not all passwords for the various access types are available to you.

db50, then, also allows simple access to all diagnosis files of the database via the menu option *Properties* and the *Files* tab. Here you see an unarranged list; the actual contents of the most important diagnosis files are still located on the various menu paths.

Using transaction db59, you can administer multiple MaxDB and liveCache instances from a SAP WebAS.

dbmcli

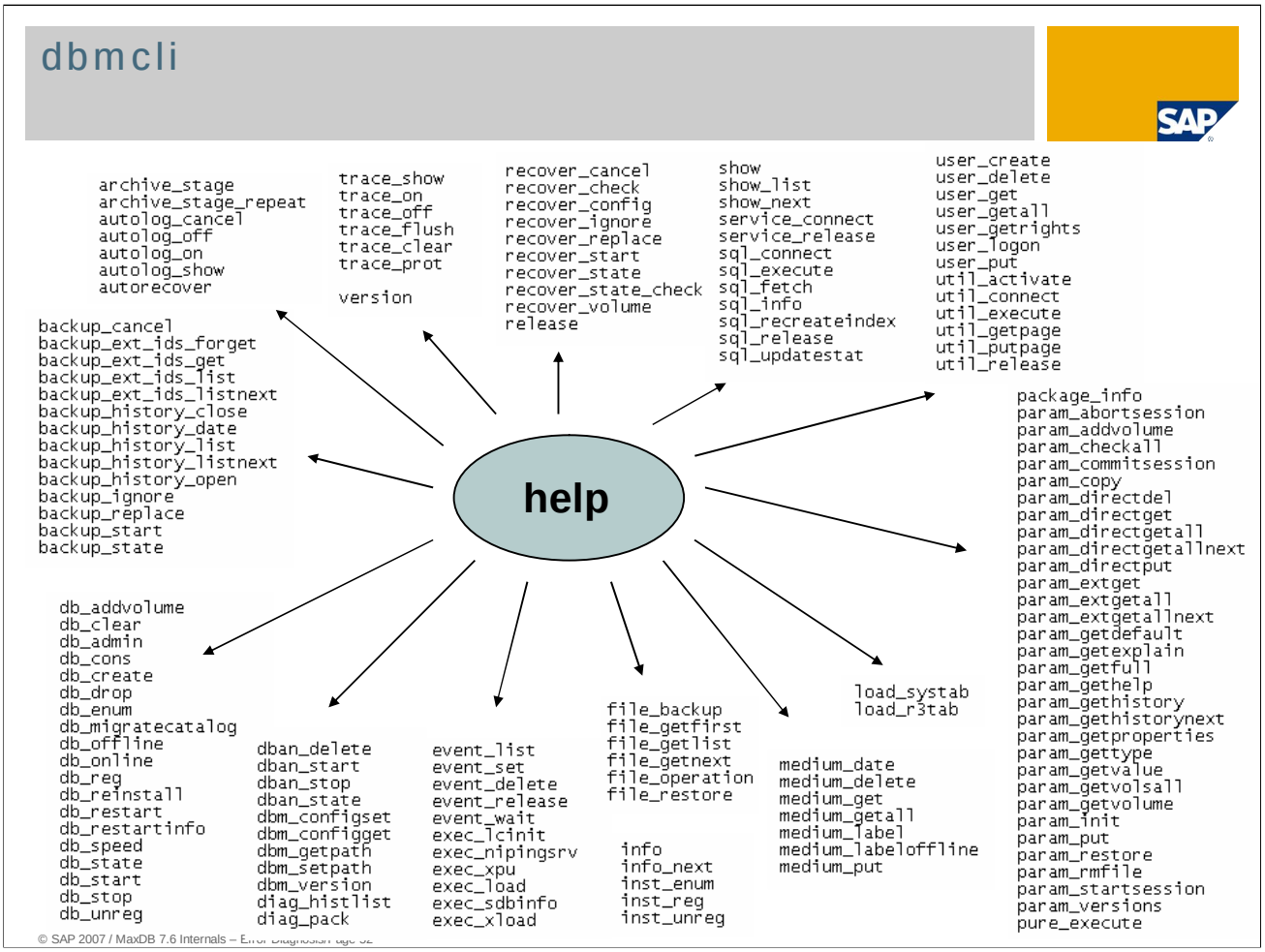


dbmcli is used for line-based database administration work; the name is an acronym for Database Manager Command Line Interface.

It can be useful for short ad hoc queries in a telnet session or for use in scripts. For more extensive administration tasks, the DBMGUI is preferable as it initiates the action and does not require precise knowledge of the command sequences, which can be very complex.

Commands are sent to the DBM server, which processes the requests; the commands that have been sent are logged in the file ***dbm.prt*** .

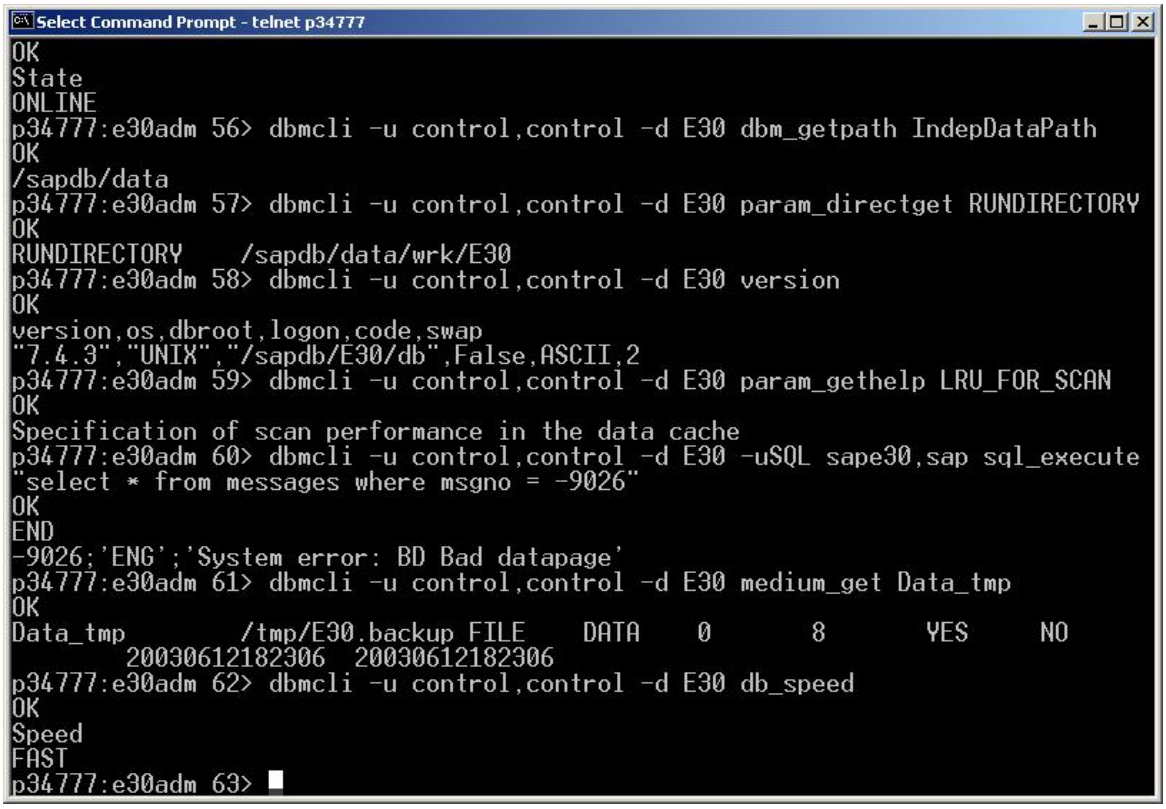
The ***dbmcli*** allows you to open a utility or an SQL session, which means that SQL queries can be sent to a database in the online operational state.



The ***dbmcli***, as the illustration makes clear, has an extensive range of functions. You can display the list of possible commands in a dbmcli session with ***help***. The help information contains additional information about which parameters have to be entered and what type of logon is required.

Some commands cannot be used alone, but only make sense as part of a command sequence.

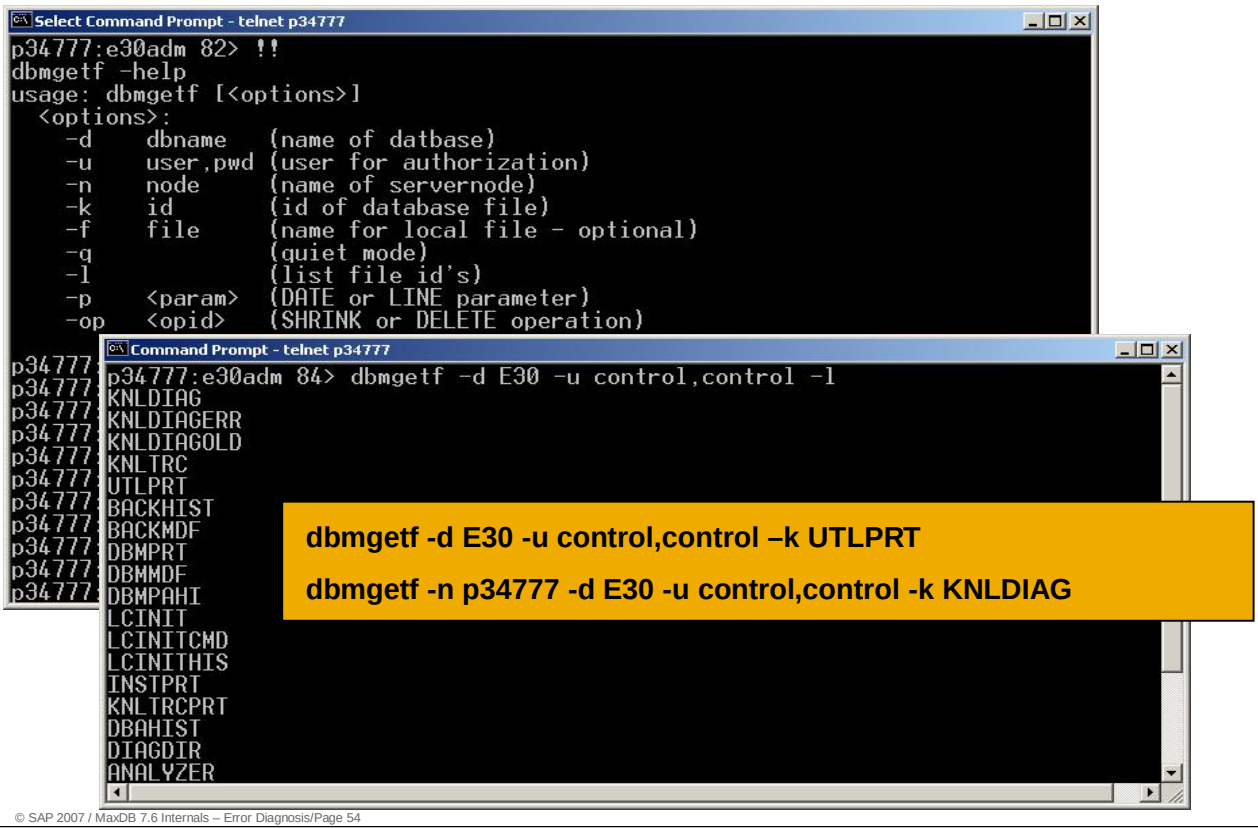
dbmcli - Examples



The examples show some commands that are useful for diagnosis; these are stand-alone commands that can provide an initial overview of the situation.

In command 60, in addition to logging on with the DBM operator, you must also specify a user authorized to access database objects.

dbmgetf

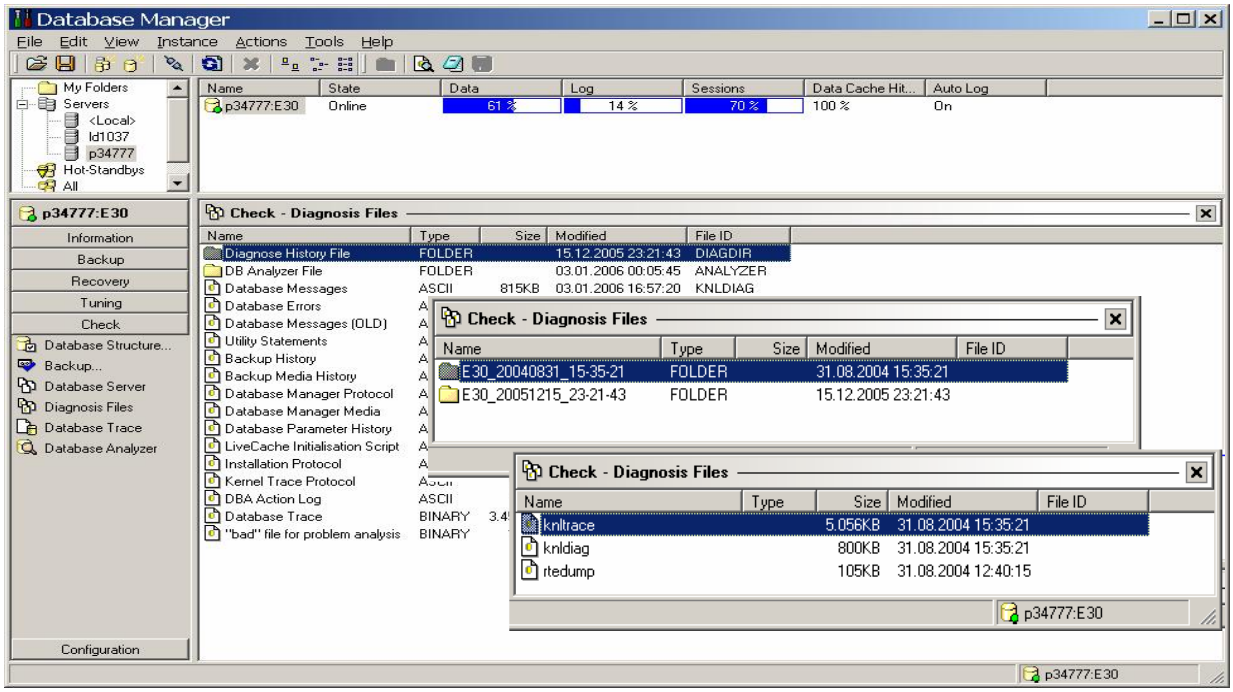


dbmgetf is a tool that enables quick access to log files, for instance in a telnet session. It is mainly used internally since, in general, the GUI-supported display options are more convenient.

With the -n option, you can specify a computer on which you want to enable remote access.

The log files are not addressed by the names stored in the operating system, but rather by abbreviations, which can be displayed using the -l option.

DIAG_HISTORY



There is an automatic procedure for receiving important information about crash situations.

The following files do not have to be explicitly backed up after a crash since they are automatically copied to a backup directory:
knldiag, knltrace, knldump, rtedump, *.dmp, *.buf, *.stm

If the database recognizes that it is being restarted after a crash, then the necessary files are backed up to a directory with the following naming convention:

- *<DB-NAME>_<DATUM>_<ZEIT>*, e.g.: *S10_20001114_12-09-45*

The backed up diagnosis files are deleted from the original directory.

The backup directory is under the directory **DIAG_HISTORY_PATH** (which must be configured) and is referred to as the history in the following.

You can also configures the number of histories (DIAG_HISTORY_NUM). If you exceed this number of histories, then the oldest history is deleted when a new backup is made.

The database can still be restarted if a backup cannot be made correctly.

CHECK DATA / CHECK TABLE

CHECK DATA [Options]

- Checks structural consistency of the whole database. If no errors are found, „bad flags“ in the so-called filedirectory and the root page are reset.

CHECK TABLE <OWNER>.<TABLENAME> [Options]

- Checks all pointers within the specified table tree.

Mirroring

- If data volumes are mirrored by means of the operating system or by hardware, the database cannot influence which disk is used for reading pages. CHECK TABLE may not find any errors.

Errors

- If CHECK TABLE delivers an error, hardware problems must be solved and a backup must be restored.

Check Data (previously Verify) checks the structural consistency of the entire database. It considers tables as well as indexes and Long columns.

The semantics of the data model is not taken into account. Logical errors are not found, but only errors caused by hardware defects.

Every page contains a check number. This is calculated with each read-I/O and compared with the value stored on the page. If the values are different, there is an error.

One typical error that may be detected is BAD DATA PAGE.

Check Table checks all dependencies and links within the specified table tree. Indexes are not taken into account.

CHECK DATA - Options

EXTENDED

- Extended check of the key sequence

WITH LONG CHECK (former WITH SHARE LOCK)

- Additional check of LONG columns
- Share lock is set

EXCEPT INDEX

- Indexes are not checked

WITH UPDATE

- Execution in DB mode ADMIN
- Additional maintenance of the converter: page numbers with no references are removed.

Another diagnosis option is calling

CHECK DATA EXTENDED.

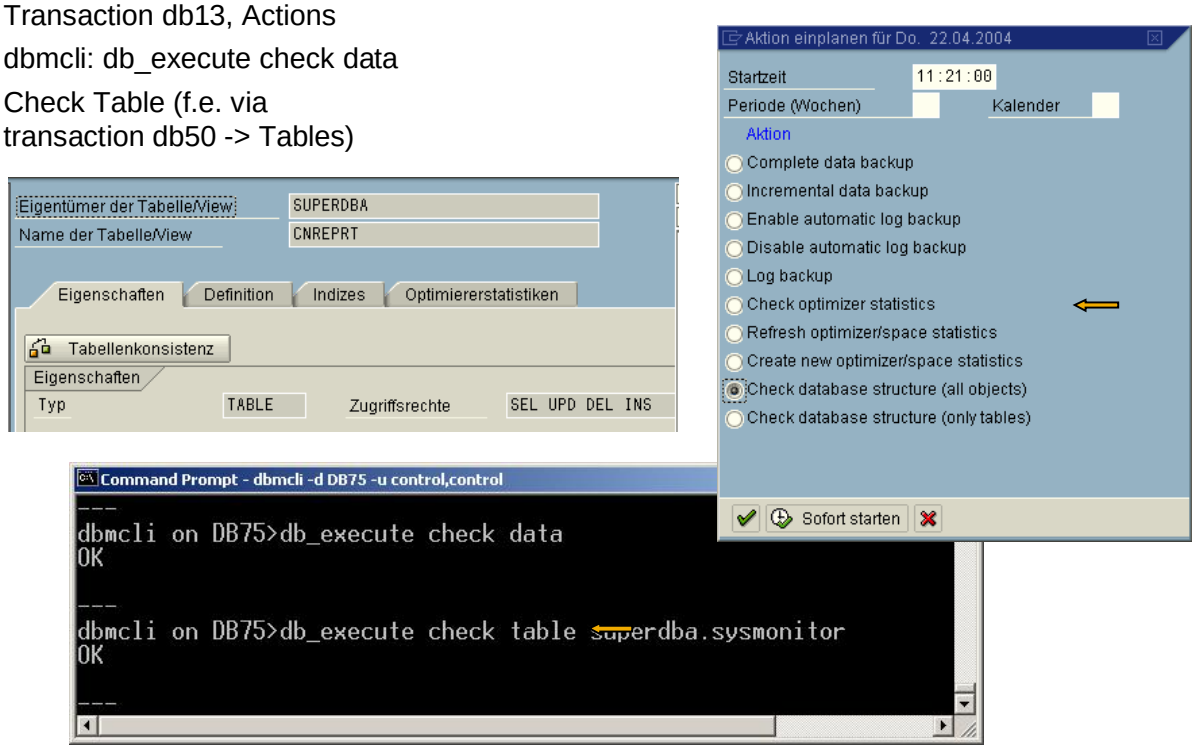
This performs a more precise check of the key lengths and checks the sequence of the primary keys on all levels of the B* tree. Because this option is CPU-intensive, execution is not standard. As of version 7.6.01 it is standard behaviour for CHECK DATA and CHECK TABLE, because CPU load can now be neglected due to the performance of modern CPUs.

The option WITH LONG CHECK makes an additional check of BLOBs. As the name in older releases suggests, a lock is set on tables while the command is executed.

To save time when checking the database, you can use the option EXCEPT INDEX. Secondary indexes are not checked in that case.

Consistency Checks (1)

- Transaction db13, Actions
- dbmcli: db_execute check data
- Check Table (f.e. via transaction db50 -> Tables)



The structural consistency of the database can be checked in different ways.

If you choose '*Check database structure (all objects)*' transaction db13, all B* tress, including indexes, are checked. *‚Check database structure (only tables)'* checks only the tables.

You can also start consistency checks with the dbmcli:

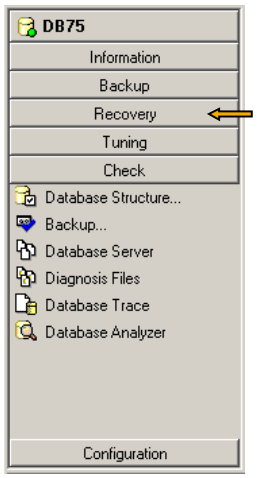
dbmcli > db_execute check data (checks all tables and indexes)

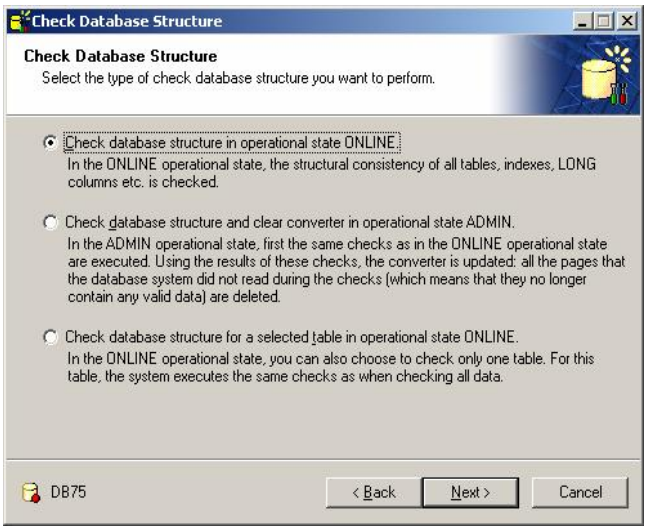
dbmcli > db_execute check table <owner>.<tablename> (selection of a table)

Transaction db50 enables you to select a table for which ‚*Check Table*' (see next slide) is then initiated.

Consistency Checks (2)

- DBMGUI: Check -> Database Structure





In the DBMGUI, choose *Check -> Database Structure*. There are several options.

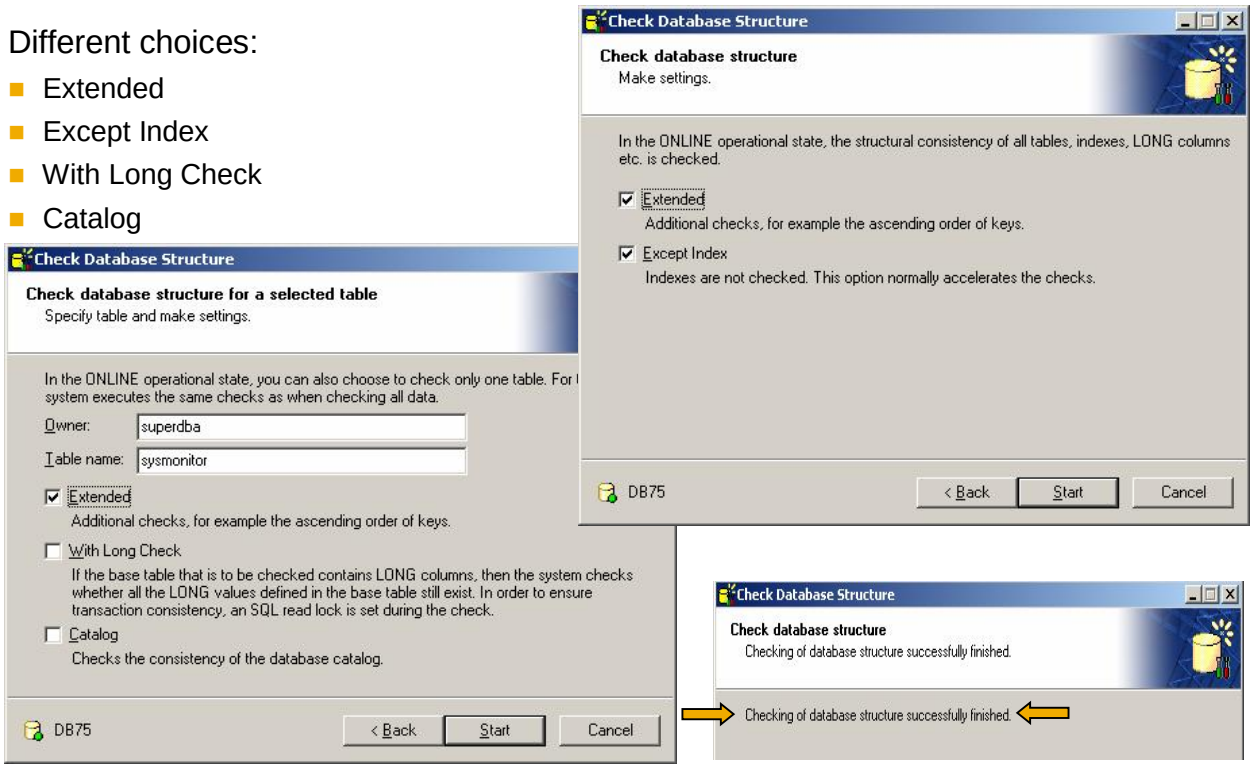
A consistency check can be done in various operational states. In the ONLINE operational state, the structural consistency of all tables, indexes, and Long columns is checked. In the ADMIN operational state, the converter is also updated; pages that are no longer referenced are deleted.

The check can be restricted to a single table.

Consistency Checks (3)

Different choices:

- Extended
- Except Index
- With Long Check
- Catalog



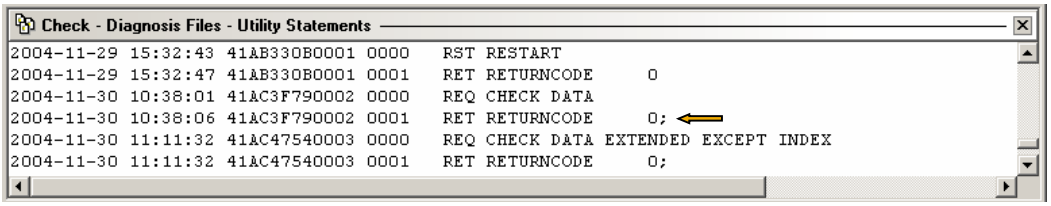
The selection options EXTENDED, EXCEPT INDEX, WITH LONG CHECK have already been explained.

CHECK CATALOG enables you to check the catalog information of a selected table.

A database structure is very time-consuming and CPU-intensive. In your production system, plan a check only for times in which the database system has a light load (for instance on weekends), or perform the check on a separate system copy.

Consistency Checks (4) - Returncodes

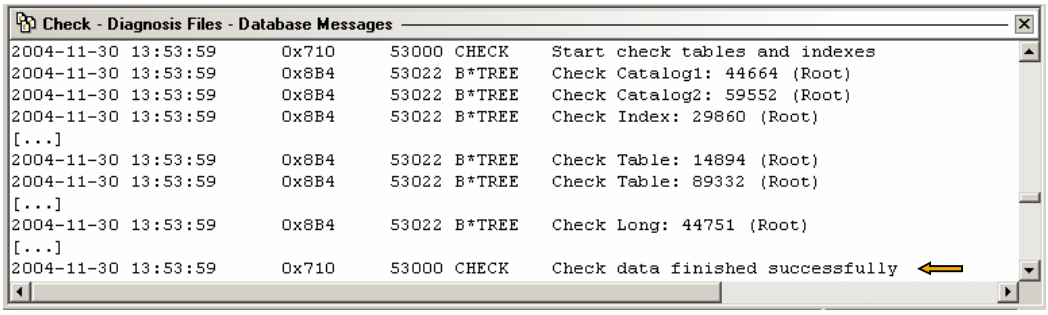
dbm.utl



dbm.prt

2004-11-30 11:11:31 0x000009c0 0 DBM command db_execute CHECK DATA
2004-11-30 11:17:37 0x0000079c 0 DBM command db_execute CHECK TABLE

knldiag



You can check whether CHECK DATA was successful in the files ***dbm.utl, dbm.prt*** or in ***knldiag***.

If a non-zero return code is returned, an error has occurred and the defective data object must be determined. The roots of the defective B* trees are listed in ***knldiag***.

Check Backup (1)

Backups are checked using a service database.

- No data is written to the disks.
- Service database merely occupies disk space.
- Check of parallel backups is possible.

Check if the backup

- is complete
- has valid contents

Before you overwrite the backups of one generation, you should make sure that you have an intact backup.

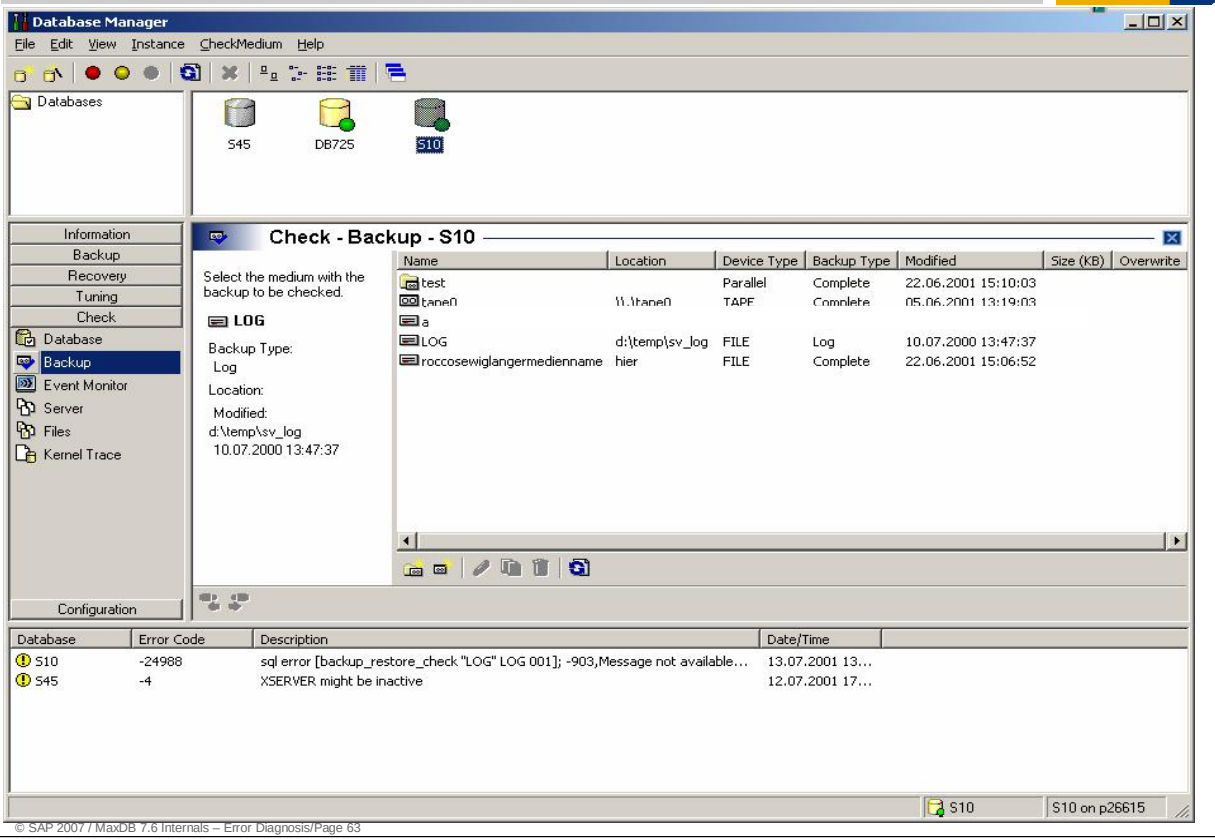
Since the check of a backup is executed on a special service database which merely uses disk space for log files, no resource bottleneck occurs.

The service database is automatically registered when a database instance is created and is stored under the name .M<version> (e.g.: .M750019).

(In older releases, the Name _SAPDB<SID> was used, though the name was shortened to 8 characters, so part of <SID> was lost.)

For a restore, the processes are logged in ***knldiag*** and the I/O can be monitored with x_cons.

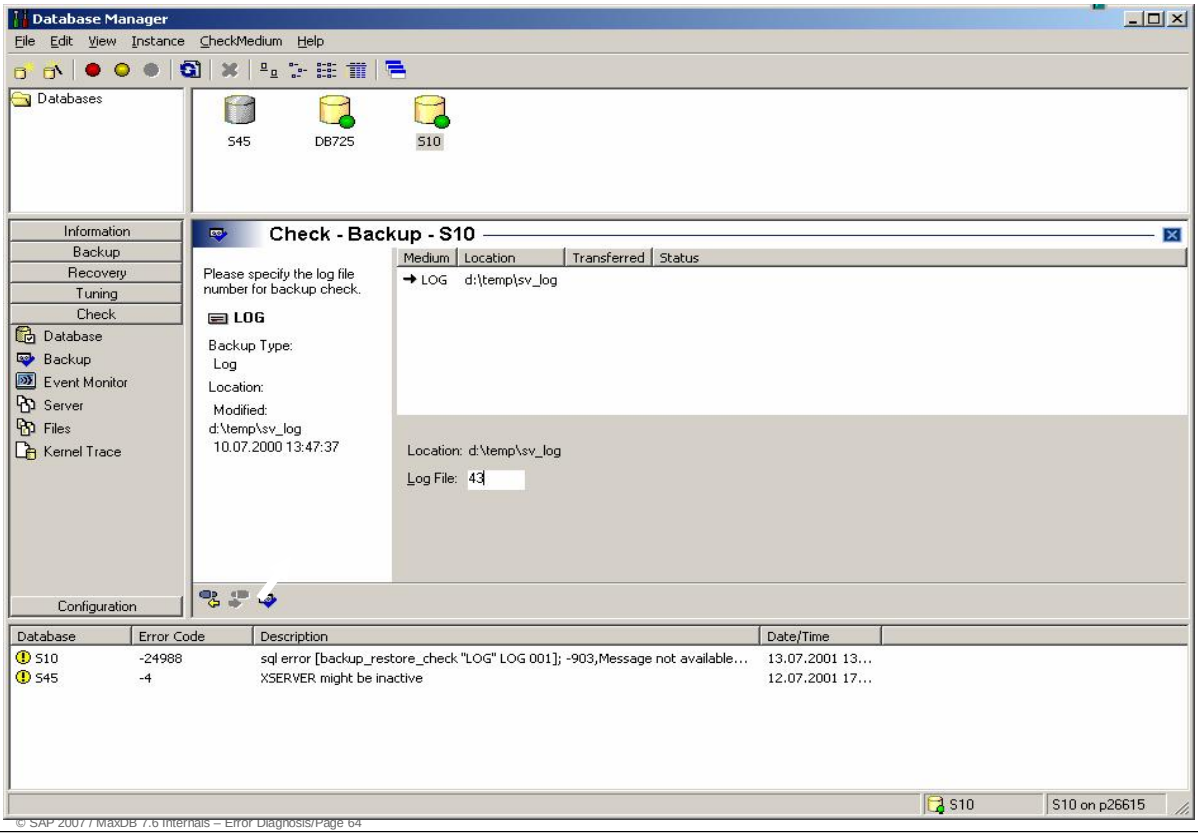
Check Backup (2)



In the DBMGUI, you can execute a check of a backup by choosing *'Check -> Backup'*.

Then you have to select the appropriate backup medium.

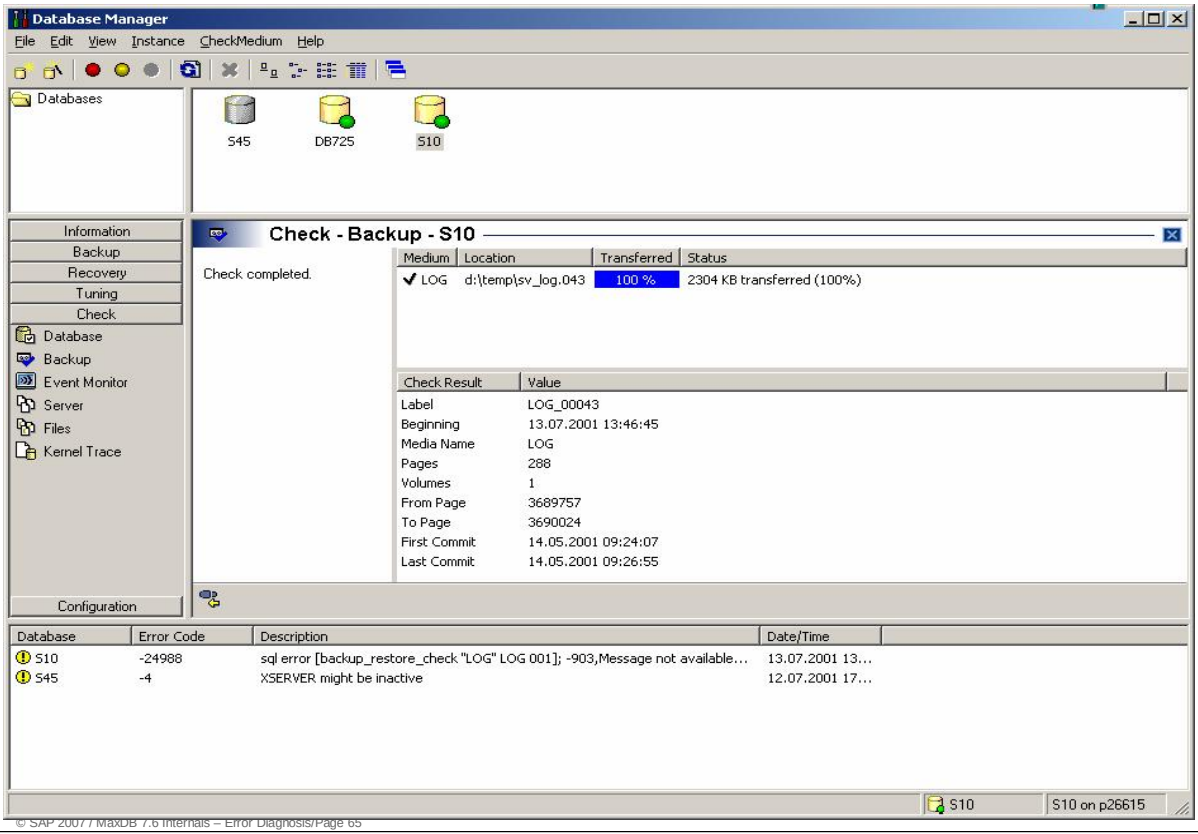
Check Backup (3)



Then, with log backups, you can also specify the version number.

You start the check with the start button.

Check Backup (4)



When the check is finished, the result is displayed in the DBMGUI.

Examples of Problem Situations

Log full

DB full

Crash, Emergency Shutdown

System hanger

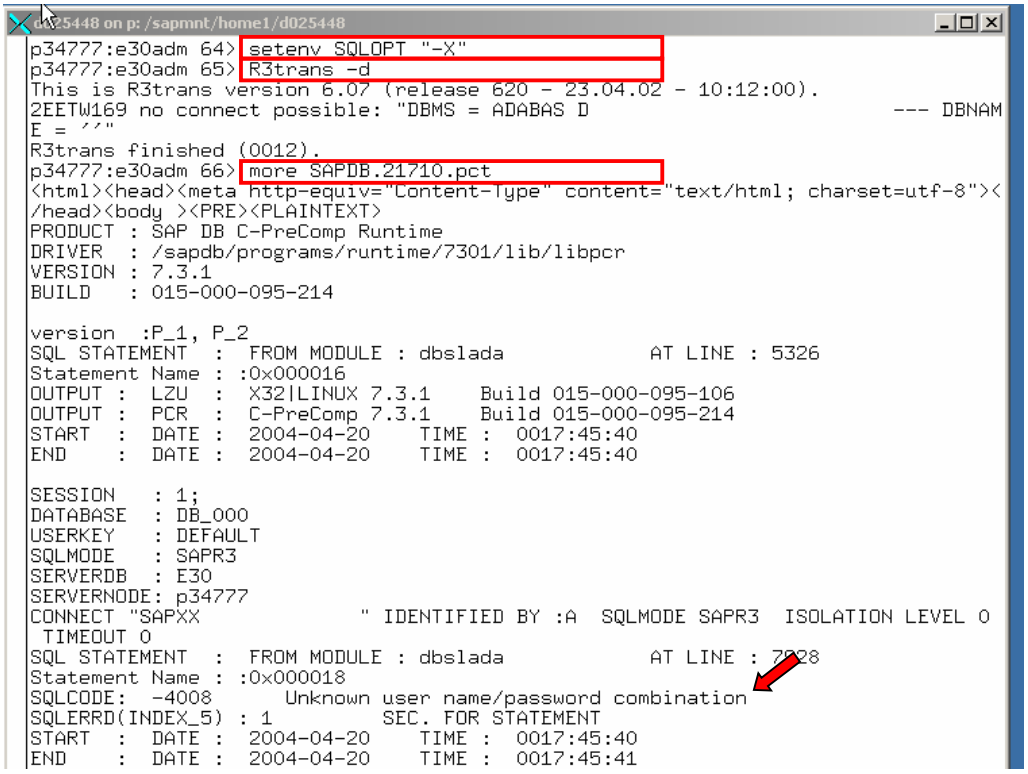
Restart problem

System copy

I/O problems

System errors –9026, -9028

Connect Problems: Check with R3trans



"Connect" problems can usually be reproduced quite easily with R/3trans. Call R3trans with option –d or –x.

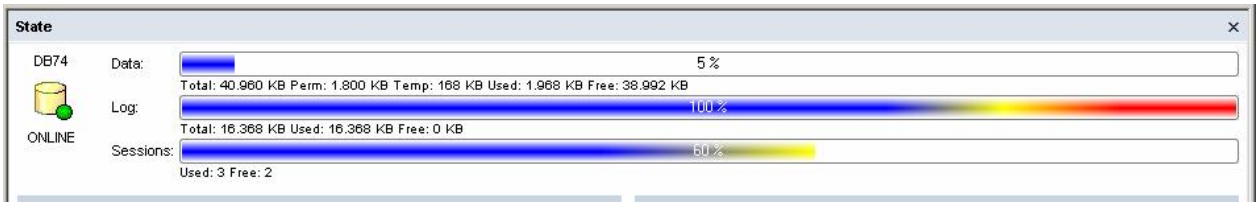
The Precompiler Runtime of the database creates a trace if the variable SQLOPT contains the value "-X". The trace is written to the file SAPDB.<PID of the client process>.pct.

In this example either the user name or the password is incorrect. The user SAPXX is probably not correct.

Check the xuser specifications with the command "*xuser list*". Maintain the xuser data as described in note 39439.

With "Connect" problems, it is often helpful to have a look in the dev logs (the ***dev_w**** files from the work directory in the SAP system).

Log Full - Identification




```
D:\sapdb\indep_data\wrk\DB74>x_cons DB74 sh act

SERVERDB: DB74

ID   UKT  Win   TASK       APPL Current         Timeout Region     Wait
          tid   type        pid state         priority cnt try    item
T13    7  0x890 User       1344 LOG FULL (246)        0 0              35609(s)
```

knldiag - Notepad

File Edit Format Help

```
2004-04-05 11:23:37       0x890      19864 DCOM     Release instance IID:{FC2F8867-6983-11D2-A97F-00A0C94311A5},
2004-04-05 11:23:37       0x890      19651 CONNECT  Connection released, T13
2004-04-05 11:23:37       0x890      19633 CONNECT  Connect req. (T13, Node:'', PID:1344)
2004-04-05 11:23:37       0x890      19863 DCOM     Create instance IID:{FC2F8867-6983-11D2-A97F-00A0C94311A5},
2004-04-05 11:23:37       0x210          1 Log      Log is full.
2004-04-05 11:23:37       0x210          2 Log      Log is nearly full (99 percent).
2004-04-05 11:23:37       0x210          1 Log      Log is full.
2004-04-05 11:23:37       0x210          2 Log      Log is nearly full (99 percent).
2004-04-05 11:23:37       0x210          1 Log      Log is full.
2004-04-05 11:23:37       0x890 WRN     42 Log      LOG FULL: task 13 suspended
2004-04-05 11:23:46       0x740         44 Log      Savepoint requested by T1 reason 'Log_Writer' (started).
2004-04-05 11:23:46       0xA24          4 Pager    SVP(1) Start Write Data
2004-04-05 11:23:46       0xA24          5 Pager    SVP(1) Stop Data IO, Pages: 172 IO: 47
2004-04-05 11:23:46       0xA24          6 Pager    SVP(2) Wait for last split, TaskId: 34
2004-04-05 11:23:46       0xA24          7 Pager    SVP(2) Stop Wait for last split, Pages: 0 IO: 0
2004-04-05 11:23:46       0x210          2 Log      Log is nearly full (99 percent).
2004-04-05 11:23:46       0x210          1 Log      Log is full.
2004-04-05 11:23:46       0x210          2 Log      Log is nearly full (99 percent).
2004-04-05 11:23:46       0x210          1 Log      Log is full.
2004-04-05 11:23:46       0xA24      53070 SAVPOINT B20PREPARE_SVP: 4
2004-04-05 11:23:46       0xA24          8 Pager    SVP(3) Start Write Data
2004-04-05 11:23:46       0xA24          9 Pager    SVP(3) Stop Data IO, Pages: 1 IO: 1
2004-04-05 11:23:46       0xA24         10 Pager    SVP(3) Start Write Converter
2004-04-05 11:23:46       0xA24         11 Pager    SVP(3) Stop Converter IO, Pages: 8 IO: 8
2004-04-05 11:23:46       0xA24      53071 SAVPOINT B20SVP_COMPLETED: 4
------------------------------------------- current write position ----------------
```


A Log Full situation first manifests itself in that an hourglass is displayed for all dialog users who are performing change actions. This suggests that the database is at a standstill and the user tasks have been stopped.

A quick glance in the DBMGUI shows that the log is 100% full, both in a bar and in text form.

Alternatively, *knldiag* and x_cons offer the same information.

As a general rule, we recommend using automatic log backup, which usually keeps this situation from happening.

Log Full - Solution

Release the log full situation:
Perform a log backup

A Log Full situation can **ONLY** be resolved by executing a log backup.

The Backup Wizard guides you through the required steps.

Adding a new log volume is NOT a possible way of solving the problem. As log volumes are cyclically overwritten, the pointer is usually 'somewhere in the middle' of the device and cannot jump to a new volume.

DB Full - Identification

```
D:\sapdb\indep_data\wrk\DB74>x_cons DB74 sh act

SERVERDB: DB74

ID    UKT  Win    TASK        APPL Current         Timeout Region     Wait
           tid    type         pid state          priority cnt try    item
T14     7  0x890 User         2104 db-full    (197)         0 0             43554(s)
```

knldiag - Notepad

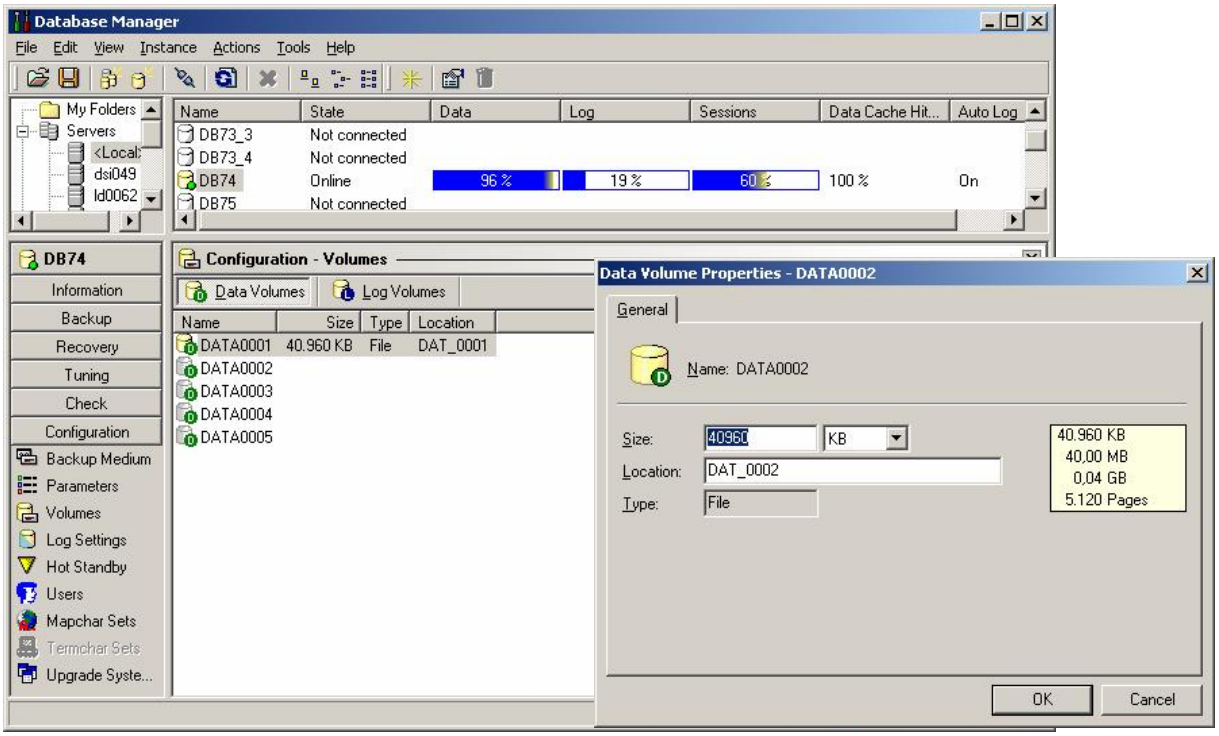
File Edit Format Help

```
2004-04-05 12:51:33         0xA24          8 Pager    SVP(3) Start Write Data
2004-04-05 12:51:33         0xA24          9 Pager    SVP(3) Stop Data IO, Pages: 1 IO: 1
2004-04-05 12:51:33         0xA24         10 Pager    SVP(3) Start Write Converter
2004-04-05 12:51:33         0x890      19633 CONNECT  Connect req. (T15, Node:'', PID:2532)
2004-04-05 12:51:33         0xA24         11 Pager    SVP(3) Stop Converter IO, Pages: 8 IO: 8
2004-04-05 12:51:33         0xA24      53071 SAVPOINT B20SVP_COMPLETED: 18
2004-04-05 12:51:33         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:33         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:33         0xA24      52024 AUTOSAVE 200 pages -> "d:\backup\db74.log.012"
2004-04-05 12:51:33         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:33         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:33         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0xA24      52024 AUTOSAVE 696 pages -> "d:\backup\db74.log.012"
2004-04-05 12:51:34         0x948      19626 IO       Async I/O thread stopped, 'd:\backup\db74.log.012'
2004-04-05 12:51:34         0xA24      52031 AUTOSAVE ready
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:34         0x890      19651 CONNECT  Connection released, T15
2004-04-05 12:51:34         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890      19633 CONNECT  Connect req. (T15, Node:'', PID:2532)
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:49         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:50         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:50         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:50         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
2004-04-05 12:51:50         0x890      19651 CONNECT  Connection released, T15
2004-04-05 12:51:50         0x890 WRN     39 Converte DB FULL(FBM): Task 14 suspended (45 pages changed
--------------------------------------------- current write position ---------------
```


A DB Full situation first manifests itself to the user exactly as it does with a Log Full. The user tasks are suspended and no further actions are possible.

Here too, the DBMGUI (without picture), *knldiag* and x_cons provide information about the hang situation.

DB Full - Solution



To resolve a DB Full situation, you have to add another data volume.

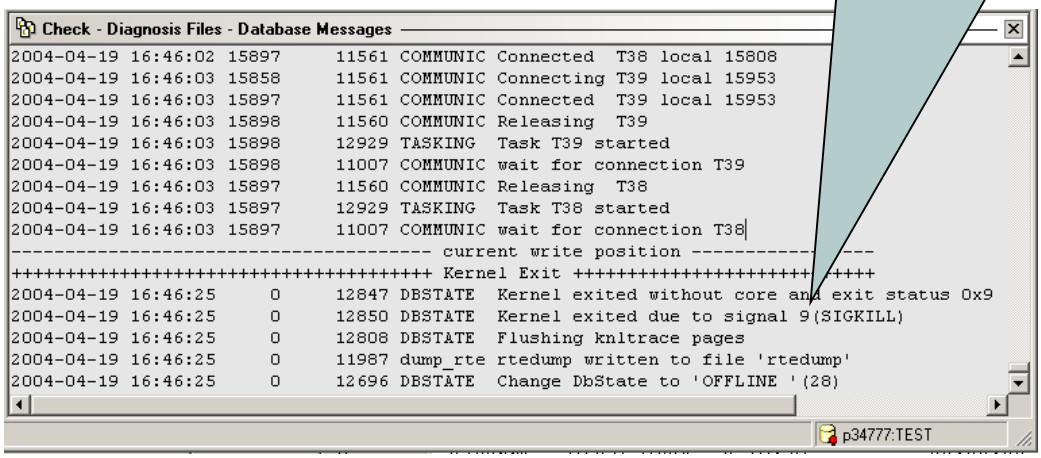
To do this, choose *Configuration -> Volumes'.*

The DBMGUI generates default values for the new volume and directs the rest of the process.

Crash of the Database: kill

knldiag

Signal 9: A thread of the database has been killed with „kill –9“.



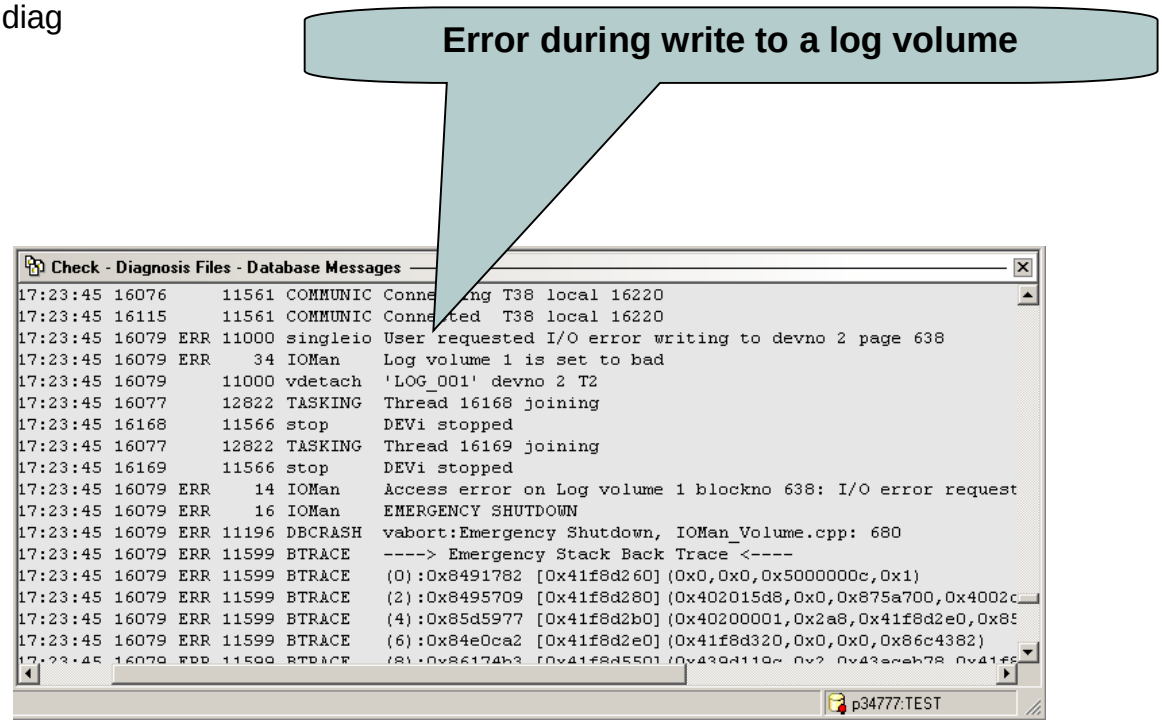
The first place to look after a database crash is *knldiag*. In this example, the database process on Unix/Linux received signal 9. Signal 9 comes from "outside" and is not caused by the database. On Unix you can find a short description of the signals in the file /usr/include/sys/signal.h. Linux stores these definitions in /usr/include/bits/signum.h.

Interesting signals:

- SIGILL 4 /* Illegal instruction (ANSI). */
  This signal comes from outside and implies a hardware problem.
- SIGABRT 6 /* Abort (ANSI). */
  Termination without further information.
- SIGKILL 9 /* Kill, unblockable (POSIX). */
  Process/thread was terminated with kill.
- SIGBUS 10 /* bus error */
  Error predominantly in the bus system; usually an error in the database software.
- SIGSEGV 11 /* Segmentation violation (ANSI). */
  Memory overwrite; usually an error in the database software.

Emergency Shutdown: Log I/O Error

knldiag



Errors while writing to the database log are very critical, in particular if the database is not being mirrored.

Determine the cause of the I/O error. For this example, the error has been simulated.

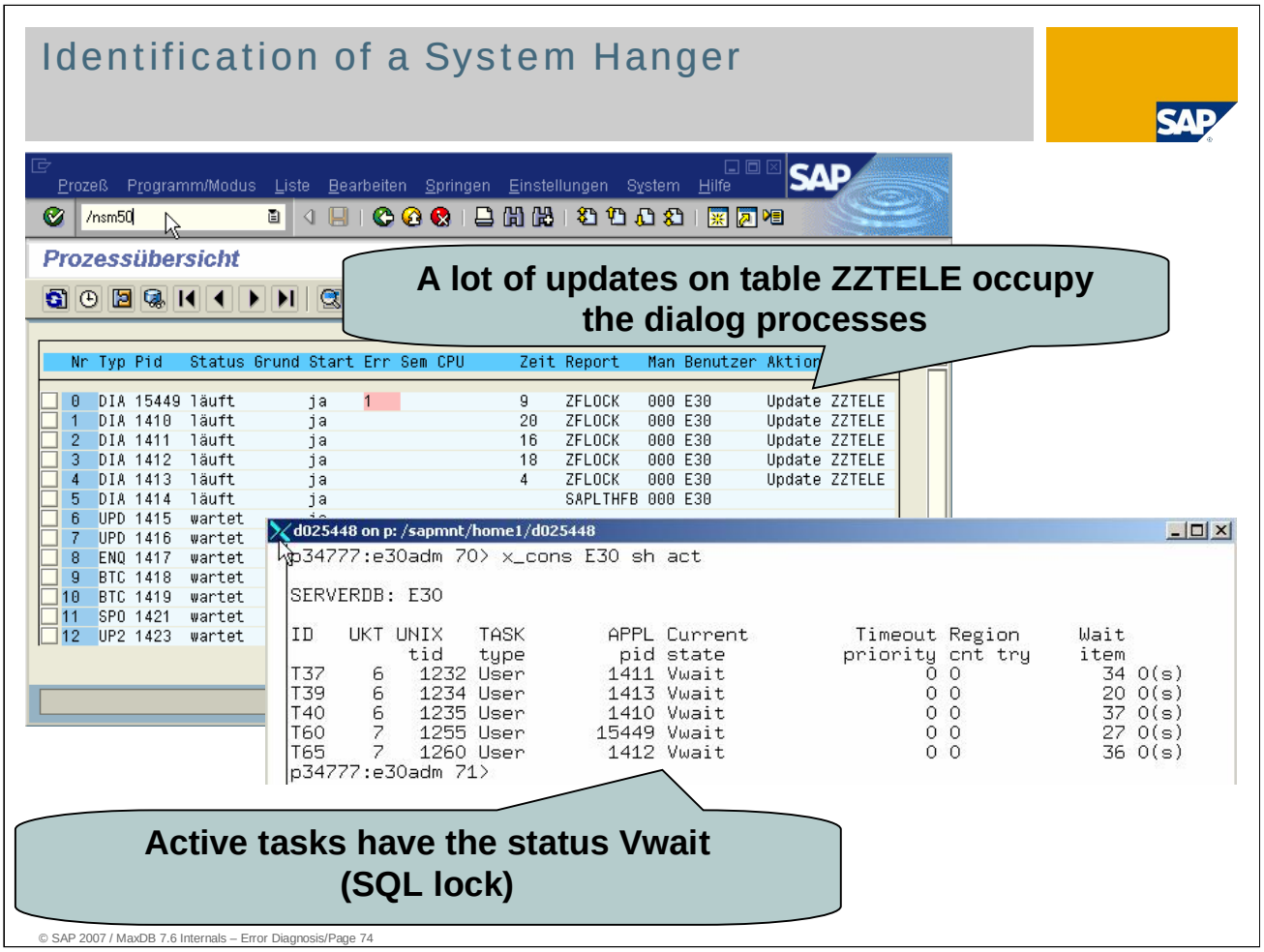
If the log is mirrored on the database side, then

- provide a new disk for the log volume,
- transfer the database to the ADMIN operational state and execute a restore for the volume using:
  dbmcli > util_execute restore log volume ‘<name des volumes>‘ ,
- start the database ONLINE.

If the log is mirrored in the system, check whether the error can be corrected in the system. If that is the case, start the database in the ONLINE operational state after the correction has been made.

If the log is irreperable, proceed as follows:

- Create a data backup. The backup is consistent on the basis of the last savepoint.
- Back up the current log area. If the log area cannot be backed up, you can use the generated data backup.
- Initialize the instance when a functional disk has been provided for the log volume. Import the data backup and the generated log backup.
  You use the Installation Wizard of the DBMGUI to initialize it.



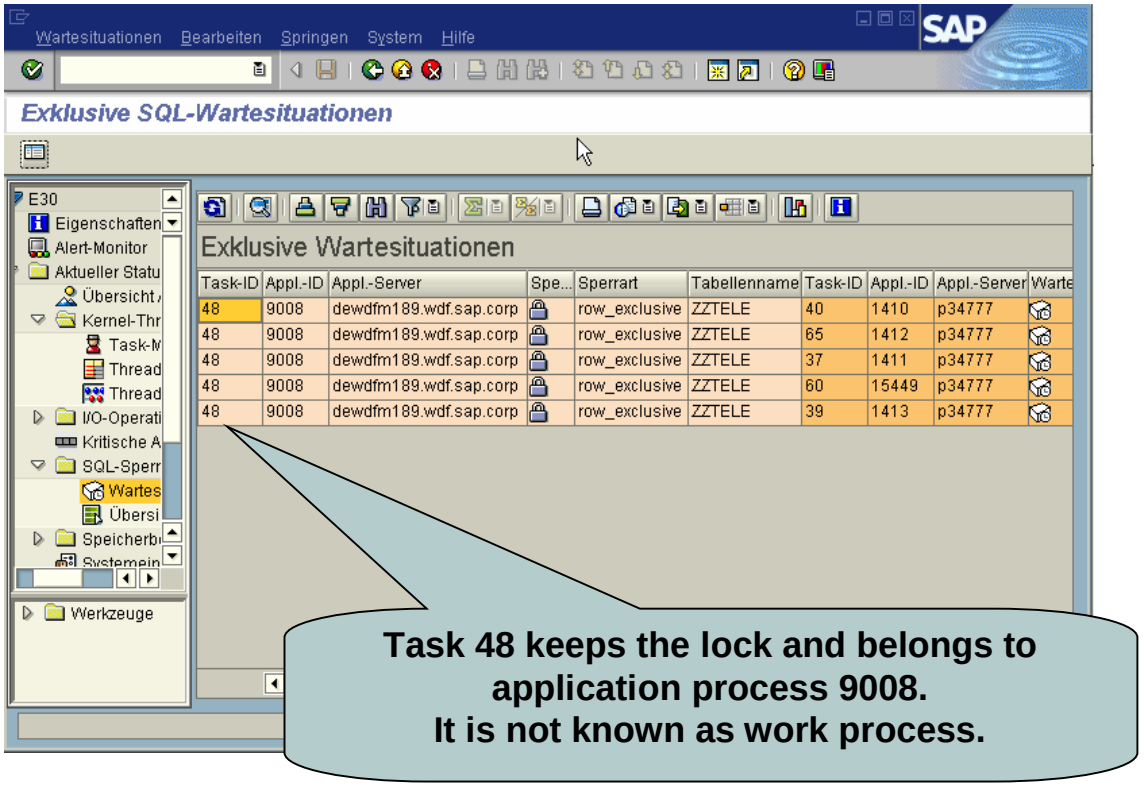
This example shows a system hanger situation.

Transaction sm50 or sm66 show numerous dialog processes that are executing updates on table ZZTELE. If transactions sm50 and sm66 are no longer usable because all dialog processes are occupied, call the program *dpmon* on the operating system level. In the 'Menu' there you'll see a comparable output.

The database console shows the respective tasks in the Vwait status. The tasks are waiting for the release of an SQL lock.

At present no other task is active in the database; that is, the lock holder is active in the application or waiting for user input.

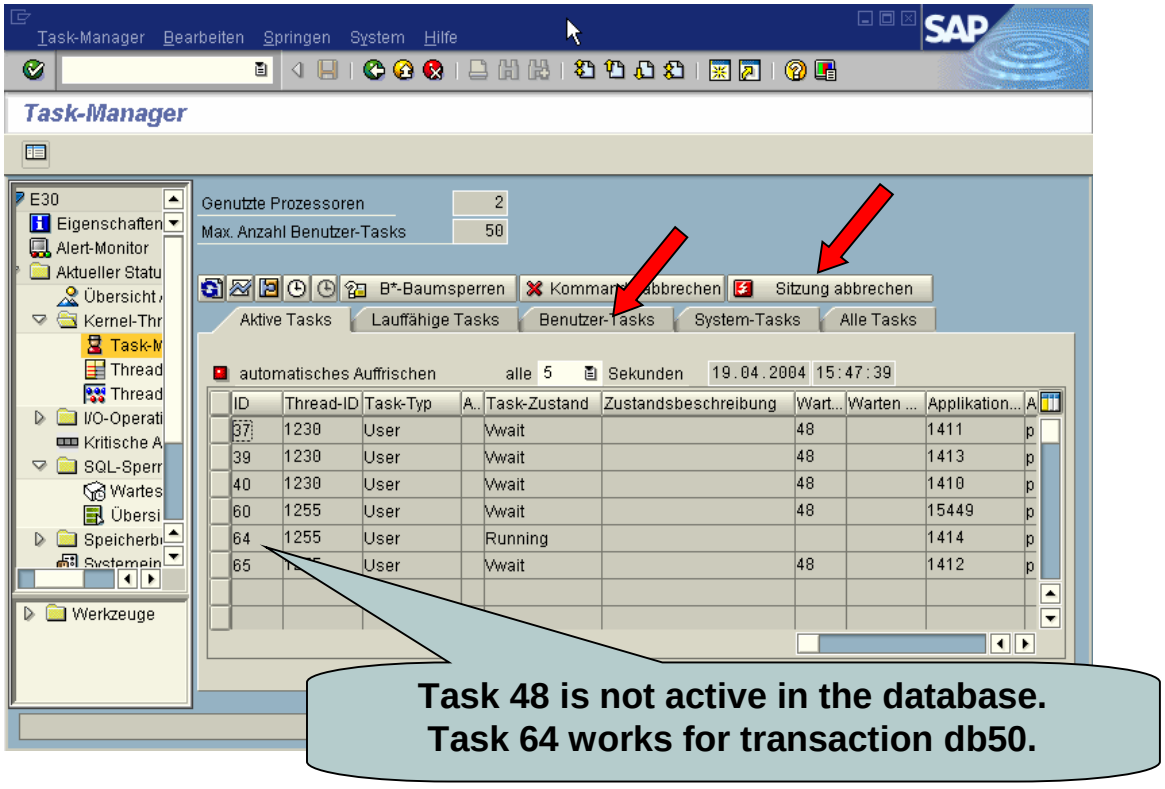
System Hanger: Who is the Lock Holder?



Transaction db50 provides more information under '*SQL Locks -> Wait Situations*'. All waiting tasks are waiting for task 48. This task belongs to application process 9008 on the server dewdfm189. The server is not a SAP application server.

User operations generally has priority. Task 48 should therefore be forced to release the lock.

System Hanger: What is Task 48 doing?

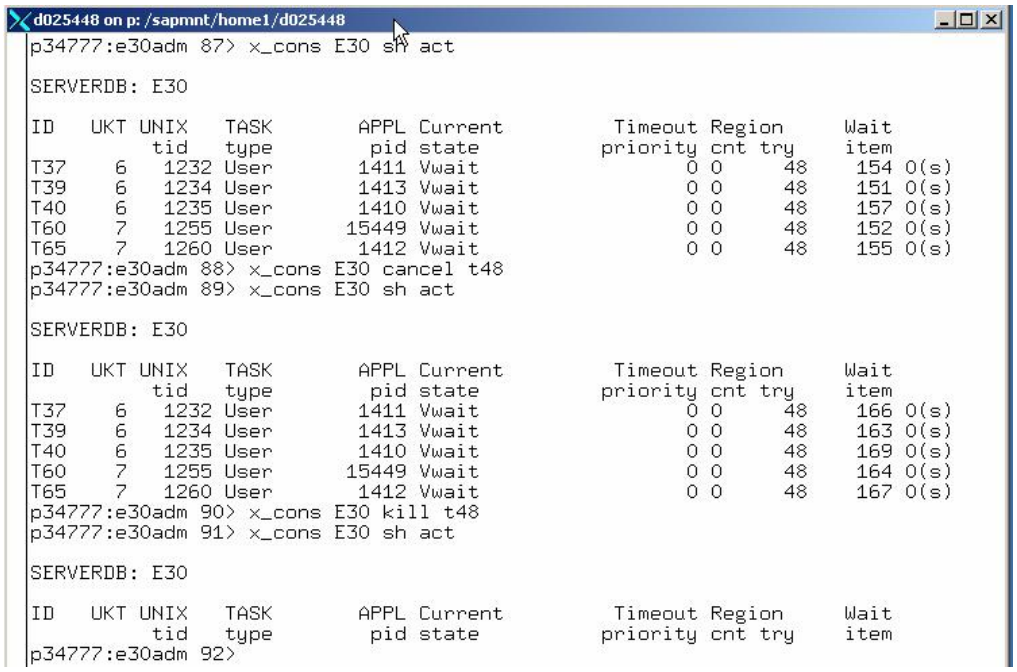


Under '*Current Status -> Kernel Threads -> Task Manager*', task db50 displays the task activities. Task 48 is not active. The running task 64 formats the information for db50 itself.

To terminate 48, display all user tasks. Select task 48 and choose 'Terminate Session'.

It is not possible to terminate a command for task 48 if task 48 is not active. The lock can only be released by terminating the transaction. If the locking transaction is not active in the database, its transaction can be terminated by closing the session.
If a work process which is holding locks is active on the database, the termination of the command leads to the termination of the transaction. When it receives return code -102, the SAP system rolls back the transaction and writes a short dump.

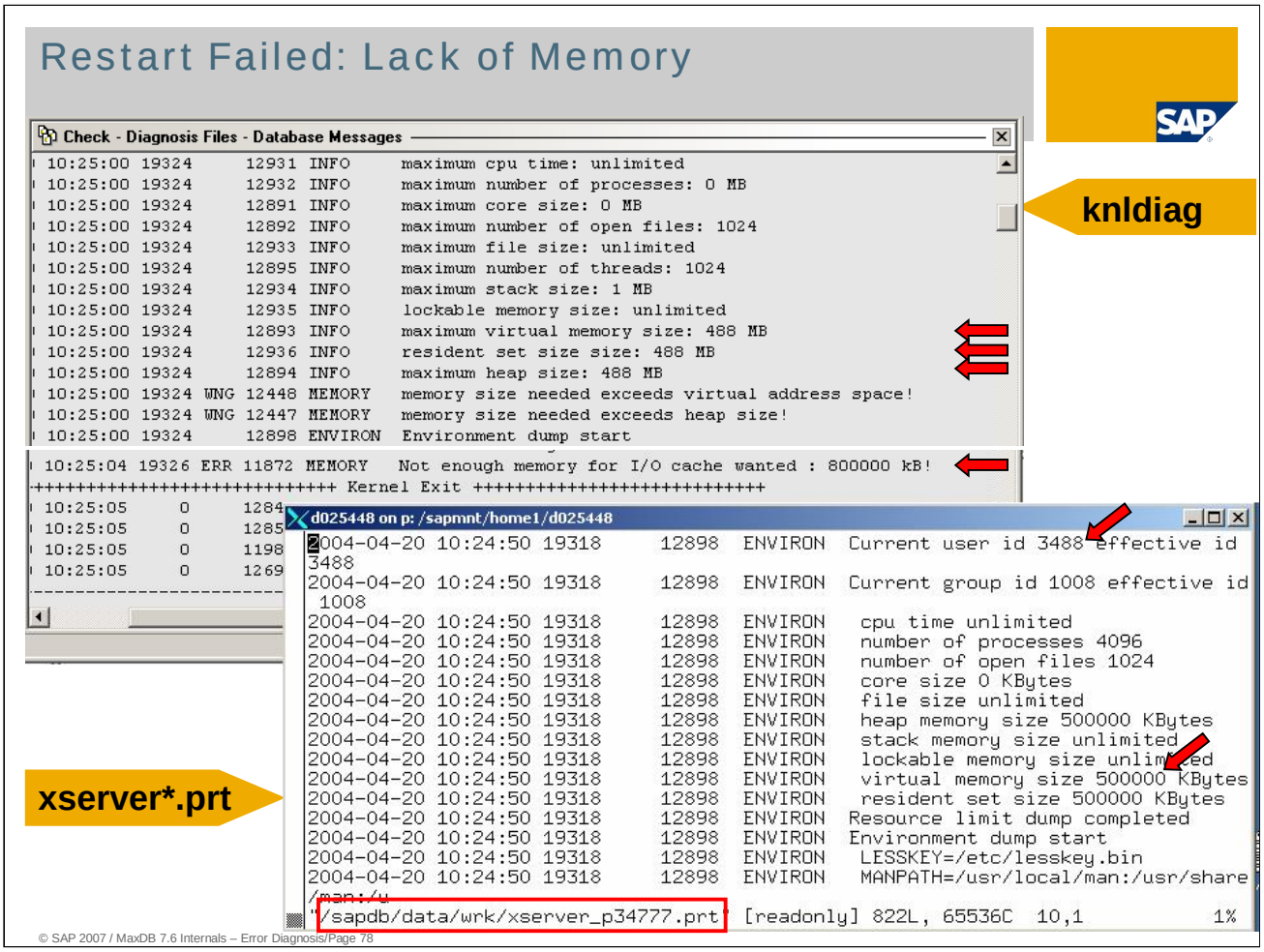
Standstill of the System: Eliminate Cause



The action '*Terminate Command*' in transaction db50 corresponds to the console command '*cancel <task>*'. You terminate user sessions with '*kill <task>*'.

Terminating the locking transaction can take some time. MaxDB works with cooperative multitasking. The tasks are not managed through a dispatcher instance.
Some actions only check whether the termination flag is set every 30 seconds.

In the console output, if the termination flag was set, this is indicated by an exclamation mark. If the task remains active (in particular in the Running and I/OWait statuses), it executes a rollback of the changes that have already been made.



In this example the database cannot transfer to the ADMIN operational state because the operating system cannot allocate enough memory.

The file *knldiag* shows an excerpt of the limitations for the user. These limitations are inherited from the owner of the x_server process.

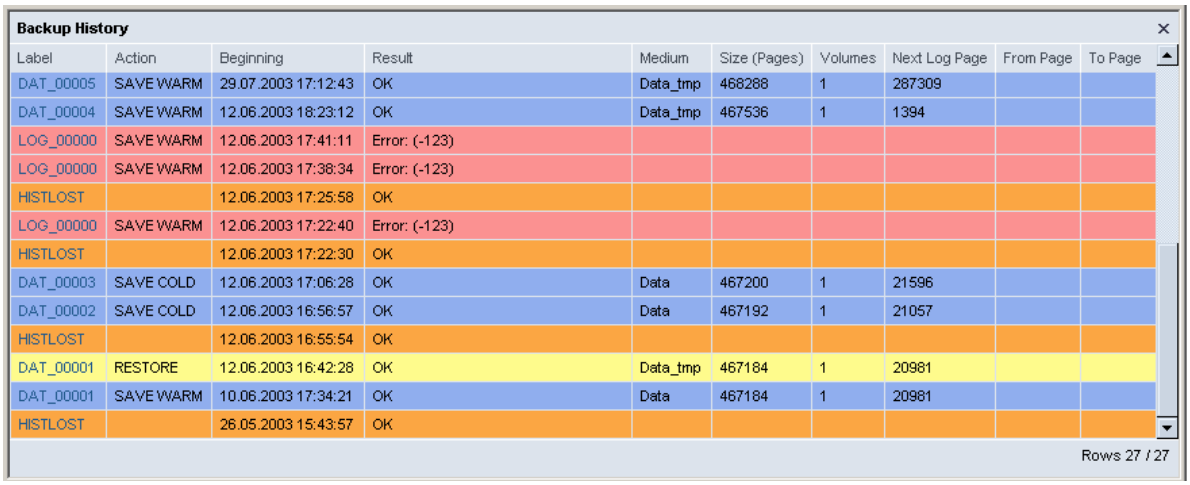
When you start the x_server, make sure that the user has set sufficient limitations. On Unix/Linux, you set limitations either with limit or ulimit, depending on the shell.
Check the limitations of the x_server process in the file <indepdatapath>/wrk/xserver*.prt.

This case can be resolved by setting the limitations correctly and restarting the x_server.

Backup / Restore

Overview of backup activities: dbm.knl

Formatted display in DBMGUI: Information -> Backup History



Backup History

Label	Action	Beginning	Result	Medium	Size (Pages)	Volumes	Next Log Page	From Page	To Page
DAT_00005	SAVE WARM	29.07.2003 17:12:43	OK	Data_tmp	468288	1	287309		
DAT_00004	SAVE WARM	12.06.2003 18:23:12	OK	Data_tmp	467536	1	1394		
LOG_00000	SAVE WARM	12.06.2003 17:41:11	Error: (-123)						
LOG_00000	SAVE WARM	12.06.2003 17:38:34	Error: (-123)						
HISTLOST		12.06.2003 17:25:58	OK						
LOG_00000	SAVE WARM	12.06.2003 17:22:40	Error: (-123)						
HISTLOST		12.06.2003 17:22:30	OK						
DAT_00003	SAVE COLD	12.06.2003 17:06:28	OK	Data	467200	1	21596		
DAT_00002	SAVE COLD	12.06.2003 16:56:57	OK	Data	467192	1	21057		
HISTLOST		12.06.2003 16:55:54	OK						
DAT_00001	RESTORE	12.06.2003 16:42:28	OK	Data_tmp	467184	1	20981		
DAT_00001	SAVE WARM	10.06.2003 17:34:21	OK	Data	467184	1	20981		
HISTLOST		26.05.2003 15:43:57	OK						

Rows 27 / 27

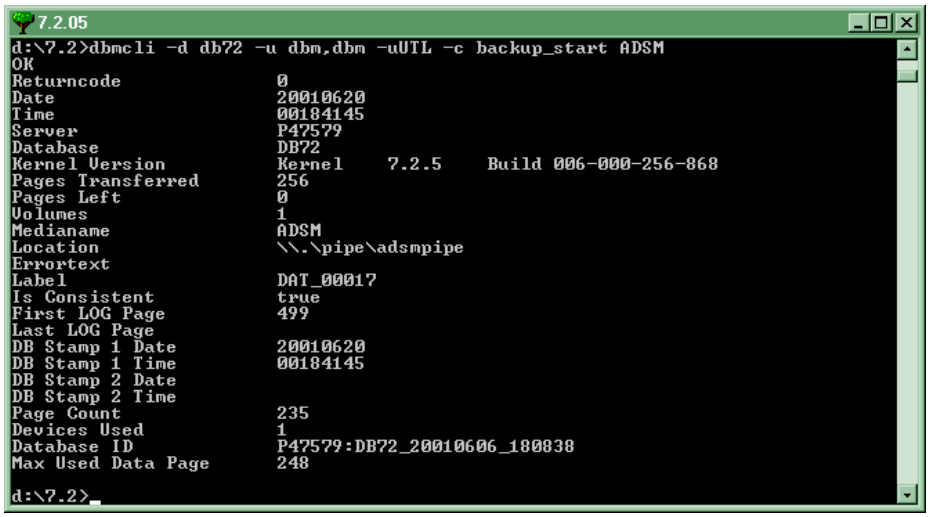
The file ***dbm.knl*** presents a first overview of which backups and which restore activities were successfully executed; or for a more orderly display in the form of a backup history in the DBMGUI, choose *Information->Backup History*.

If any errors have occurred, the causes are noted in brief. More precise information can be found in ***knldiag.***

The file ***dbm.utl*** provides information about backups.

If external backup tools (Networker, ADSM, Omniback, etc.) the backint interface are being used, you should also check their logs, which are described in the following.

Backup with External Backup Tool

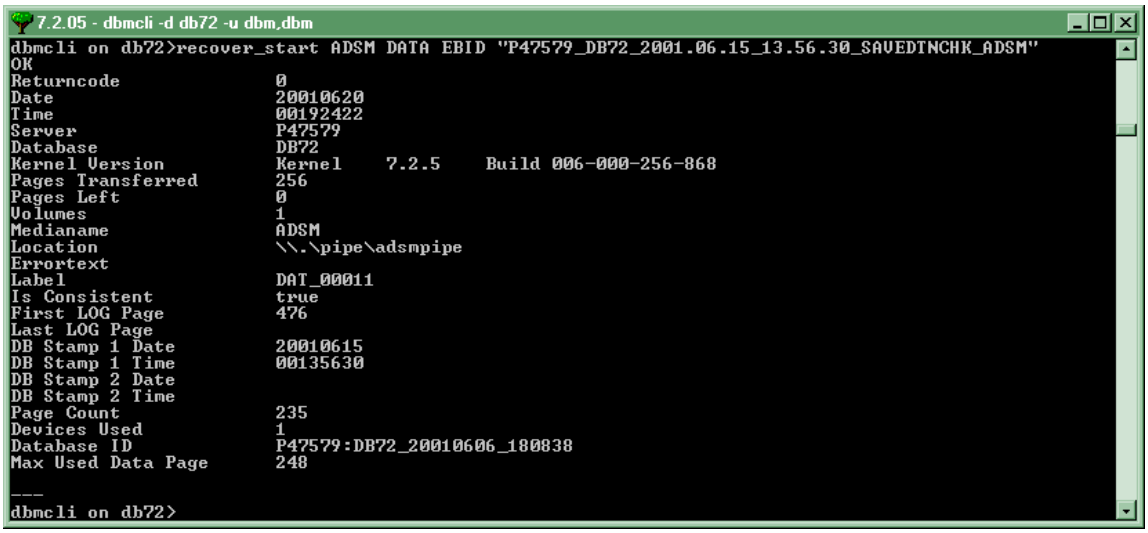


One example shows the execution of a backup via ADSM using the dbmcli.

You execute a backup of the database with the backup_start command.

As the DBM server can derive the desired backup tool from the names of the medium, there is, in this case, no difference to a backup without a tool.

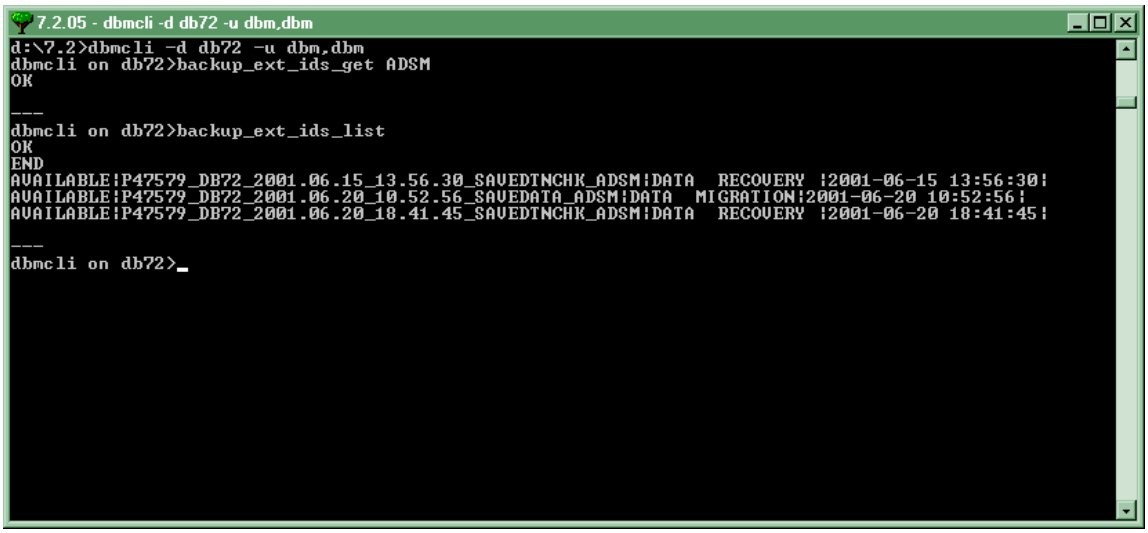
Restore with External Backup Tool



A restore is executed using the commands recover_start and recover_replace (for restoring multiple backups).

The keyword EBID (or ExternalBackupID) is followed by a list (separated by commas) of external backup IDs (only one in the present example). If the lists contains blanks, it must be written in quotation marks.

External Backup Identifier



As more than one DBM server command is required for displaying the external backup ID, you have to use an interactive dbmcli session.

The columns of the displayed list are separated by the pipe (|) character.

The list has the format:

<Availability>|<External Backup ID>|<backup typ>|<date_time>|

If an output of the backup_ext_ids_list commands contains a line with the keyword CONTINUE followed by OK, the next part can be queried with the backup_ext_ids_listnext command.

Example: Backup / Restore (1)

```
The restore from the system <SID> with backup from 15.01.01 was
killed with:
2004-01-16 20:25:53    13 ERR 54001 I/O      page 00C29008010D0200...008
2004-01-16 20:25:53    13 ERR 52015 RESTORE  write/check count mismatch
2004-01-16 20:25:53    12 ERR 52012 RESTORE  error occured, basis_err 30
2004-01-16 20:25:53    12 ERR 51080 SYSERROR -9026 Message not available
The restore with the backup from 16.01 was killed with:
2004-01-16 17:11:31    13 ERR 54001 I/O      page 00AA5684010D0200...203
2004-01-16 17:11:31    13 ERR 52015 RESTORE  write/check count mismatch
2004-01-16 17:11:31    12 ERR 52012 RESTORE  error occured, basis_err 30
2004-01-16 17:11:31    12 ERR 51080 SYSERROR -9026 Message not available
```


In the present example, the restoring of a backup terminated with a system error.

First, the system attempts to repeat the procedure with various backups. It turns out that several backups have already been affected and that a restore returns error -9026.

At this stage, the user should check the logs to see what they say about the backups, e.g. whether they were successful, etc.

Example: Backup / Restore (2)

```
UTLPRT:
=======
2004-01-19 01:00:11 400B1E0B0006 0000   SDB SAVE DATA QUICK TO '/nsr/sap
2004-01-19 03:48:08 400B1E0B0006 0001   RET RETURNCODE     0
[...]


DBMPRT:
=======
2004-01-19 01:00:11 0x00007227              0 DBM
  command backup_start nsr_full RECOVERY DATA AUTOIGNORE
2004-01-19 03:49:45 0x00007227 ERR     -24920 DBM
  ERR_BACKUPOP: backup operation was unsuccessful
                0x00007227 ERR     -24920 DBM
  The backup tool failed with 0 as sum of exit codes.
```


Looking at *dbm.utl* and *dbm.knl* is not sufficient! The return code 0 here shows only that the backup was successful from the database's point of view. In other words, it correctly delivered all database pages to the pipe of the external database tool.

In *dbm.prt* we see that the backup could not be completed successfully. The cause is not immediately visible in this file, but the tool has signaled that the backup failed from its point of view.

Backups that have the return code 0 in *dbm.knl* and *dbm.utl* (that is, on the kernel side) but failed according to *dbm.prt* are identified as having failed in the backup history (DBMCLI -> backup_history_list as well as in DBMGUI). The error code, then, is the error code of the backup/restore (generally -24920).

Example: Backup / Restore (3)

```
BACKEBL:
========

2004-01-19 03:49:44
Analyzing output of NetWorker.
    Have found a completion message in the NetWorker output file
    '/var/tmp/temp1074470411-1' in line:
    save: S6F level=full, 130 GB 02:43:44      5 files
    Have found a savetime message in the NetWorker output file
    '/var/tmp/temp1074470411-1' in line:
    completed savetime=1074470655
    NetWorker has saved only 140123308032 bytes
    (the database kernel has saved 159754027008 bytes).
The backup size of database and NetWorker are different.

2004-01-19 03:49:44
Filling reply buffer.
    Have encountered error -24920:
        The backup tool failed with 0 as sum of exit codes.

    Constructed the following reply:
        ERR
        -24920,ERR_BACKUPOP: backup operation was unsuccessful
        The backup tool failed with 0 as sum of exit codes.
Reply buffer filled.
```


The log of the backup tool provides information as to why the backup was not considered successful.

There is a big discrepancy between the number of bytes backed up by the database kernel and the number of bytes given by the Networker which cannot be explained by rounding errors.

Here we cannot identify the cause, which could only be determined in cooperation with the Networker manufacturer Legato. While a file system backup was in progress, the Networker failed to end the data backup correctly when it accessed the data backup pipe.

MaxDB now ensures that file system and database backups remain separate.

Example: Faulty Tape Devices

1st attempt: RESTORE DATA was successful

RESTORE LOG fails

```
The recovery of the system <SID> breaks with
2004-01-16 20:11:11    13 ERR 52015 RESTORE  bad log page 98897294
2004-01-16 20:11:25    13 ERR 52608 RESTART  LOCAL: failed
2004-01-16 20:11:29    12 ERR 52012 RESTORE  error occured, basis_err 90
2004-01-16 20:11:29    12 ERR 51080 SYSERROR -9030 Message not available
```

2nd attempt: RESTORE DATA fails, but is later successful with the identical backup

```
Excerpt from the Backup History:
4007CDD70001|DAT_00210|RESTORE  |2004-01-13 23:00:20|
2004-01-13 23:00:20|2004-01-16 12:46:10|2004-01-16 12:51:56|
98576129|          |YES|nsr_comp
10624|         0|     -9026|

4007D1BF0002|DAT_00210|RESTORE  |2004-01-13 23:00:20|
2004-01-13 23:00:20|2004-01-16 13:01:55|2004-01-16 19:39:16|
98576129|          |YES|nsr_comp
20023232|         1|         0|
```


In this example, the restore of a log backup terminates with system error -9030 (bad log page).

During the attempt to repeat the restore process completely, there was already a problem with the data backup. Subsequently, however, the data backup (with the same label) was successfully recovered.

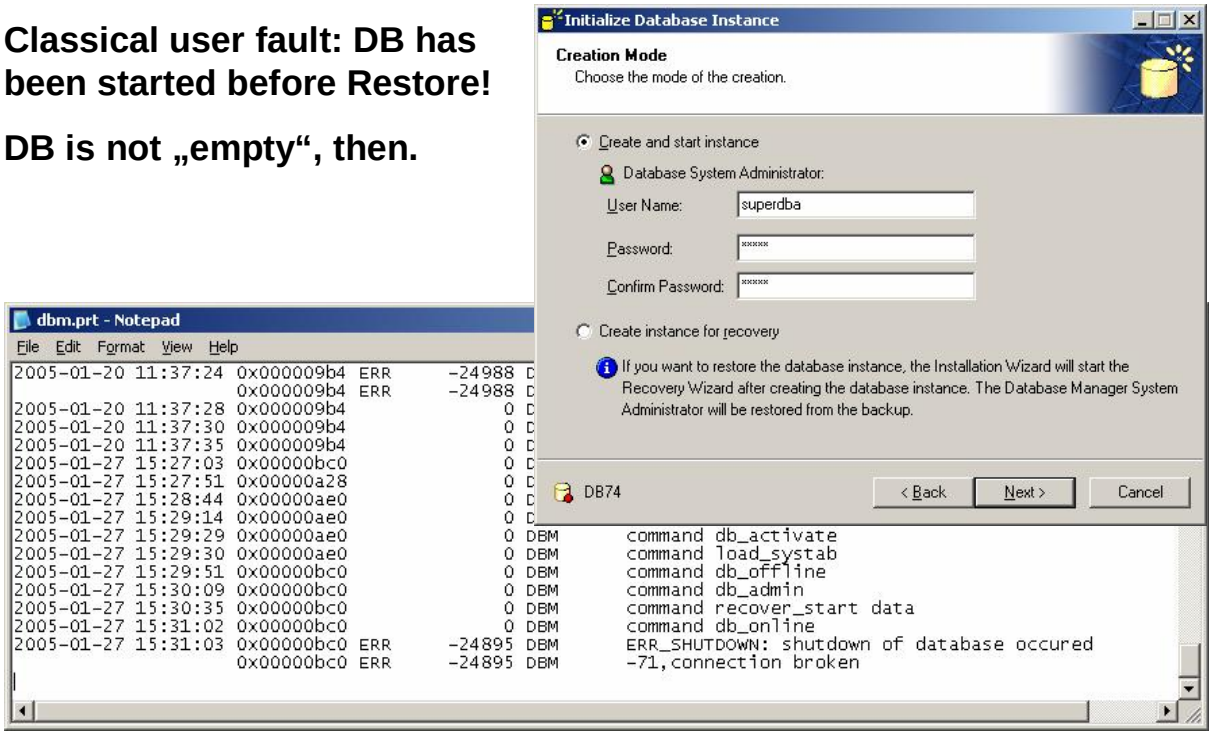
This non-deterministic behavior suggests a problem with the tape peripherals. In such cases, checking the tape drives and the controller and changing the defective device will solve the problem.

With luck, the problem will have been merely a read error; in the worst case scenario, the tapes will already have been incorrectly filled.

Error during System Copy (1)

Classical user fault: DB has been started before Restore!

DB is not „empty“, then.



One typical user error with system copies is to first completely install a database (including restart and loading the system tables) and only then import a backup. This often leads to confusion when it then turns out that it is no longer an "empty DB".

The user chose '***Create and start instance***' instead of '***Create instance for recovery***' in the DBMGUI.

The individual steps can be viewed in dbm.prt . We can see that db_activate was carried out, which represents the first restart of a DB, and that only afterwards a backup was imported.

The system's reaction to this error is somewhat different than in earlier versions (<= 7.3). The system no longer issues return code –8003 „Log and Data must be compatible“ because the database is immediately transferred to the OFFLINE operational state and the DBMGUI no longer receives a message about the exact cause of the error. This ensures that the memory areas can be completely cleared.

On the following slide, however, we can see that a similar message in ***knldiag*** provides more clarity.

Error during System Copy (2)

dbm.prt - Notepad

File Edit Format View Help

```
2005-01-20 11:37:24  0x000009b4 ERR      -24988 DBM       ERR_SQL: sql error
                     0x000009b4 ERR      -24988 DBM       100,Row not found
2005-01-20 11:37:28  0x000009b4                0 DBM       command trace_show
2005-01-20 11:37:30  0x000009b4                0 DBM       command trace_flush
2005-01-20 11:37:35  0x000009b4                0 DBM       command trace_prot
2005-01-27 15:27:03  0x00000bc0                0 DBM       command db_online -f
2005-01-27 15:27:51  0x00000a28                0 DBM       command backup_save "data" DATA RECOVERY
2005-01-27 15:28:44  0x00000ae0                0 DBM       command db_offline
2005-01-27 15:29:14  0x00000ae0                0 DBM       command db_admin -f
2005-01-27 15:29:29  0x00000ae0                0 DBM       command db_activate
2005-01-27 15:29:30  0x00000ae0                0 DBM       command load_systab
2005-01-27 15:29:51  0x00000bc0                0 DBM       command db_offline
2005-01-27 15:30:09  0x00000bc0                0 DBM       command db_admin
2005-01-27 15:30:35  0x00000bc0                0 DBM       command recover_start data
2005-01-27 15:31:02  0x00000bc0                0 DBM       command db_online
2005-01-27 15:31:03  0x00000bc0 ERR      -24895 DBM       ERR_SHUTDOWN: shutdown of database occured
                     0x00000bc0 ERR      -24895 DBM       -71,connection broken
```

knldiag - Notepad

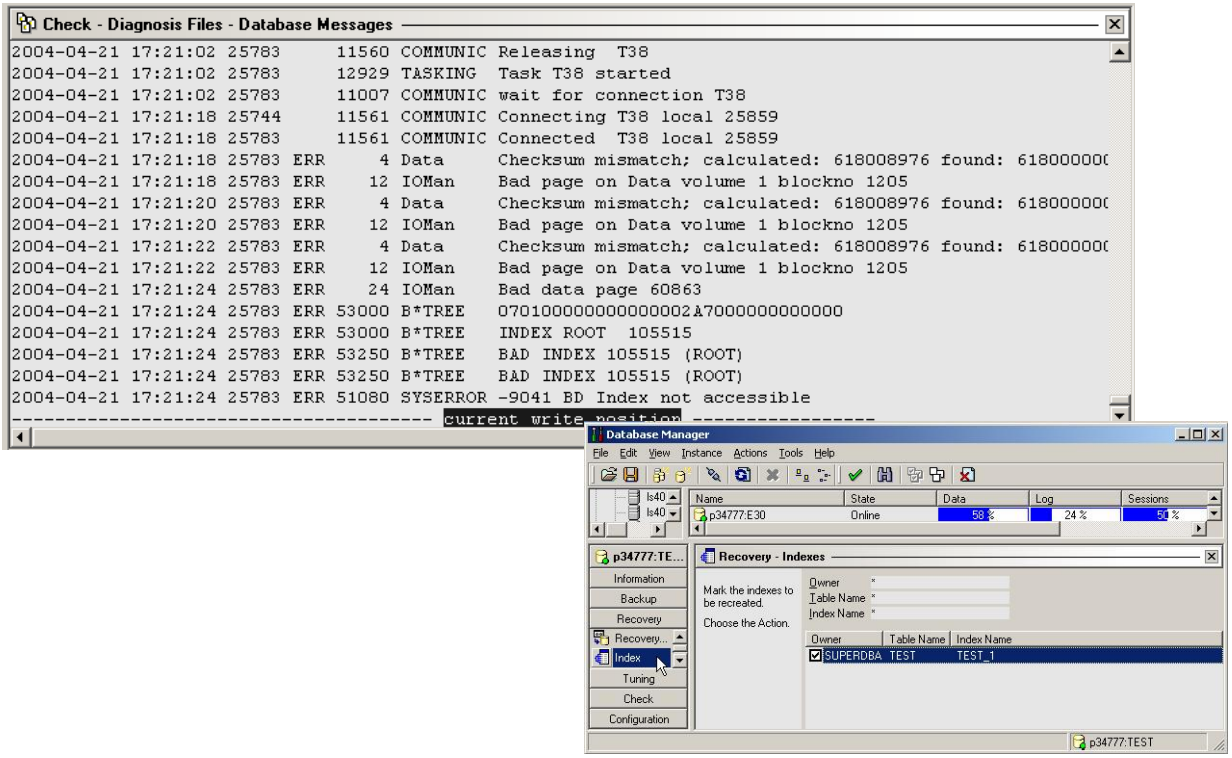
File Edit Format View Help

```
2005-01-27 15:31:02      0xDB4      54003 dynpool  LOCK SupplyPoolSegments:     75
2005-01-27 15:31:02      0xDB4      54003 dynpool  LOCK SupplyPoolSize    : 541504
2005-01-27 15:31:02      0xDB4      54003 dynpool  LOCK ObjColl_list size :   2816
2005-01-27 15:31:02      0xDB4      54003 dynpool  LOCK ObjReq_list size  :   1408
2005-01-27 15:31:02      0xDB4 ERR     29 Log      DBIdentifier of DataVolume (P124984.wdf.sap.corp:HOTELDB_20041223_105938)
                                                               and LogVolume (P124984.wdf.sap.corp:HOTELDB_20050127_152929)
2005-01-27 15:31:02      0xDB4      19616 VOLUMEIO Detaching volume 'DISKL001'
2005-01-27 15:31:02      0x140      19618 VOLUMEIO Single I/O detach, 'DISKL001', UKT:3
2005-01-27 15:31:02      0x2E8      19614 DBSTATE  I/O thread for 'DISKL001' stopped
2005-01-27 15:31:02      0xDB4 WRN     23 Admin    Initialization of log for 'restart' failed with 'LogAndDataIncompatible'
2005-01-27 15:31:02      0xDB4         37 Admin    Kernel state: 'OFFLINE' reached
2005-01-27 15:31:02      0xDB4      19620 DBSTATE  Shutdown normal requested
2005-01-27 15:31:02      0x13C      19617 VOLUMEIO Single I/O attach, 'knltrace', UKT:1
2005-01-27 15:31:02      0xDD4      19684 TASKING  Tracewriter termination timeout: 1200 sec
2005-01-27 15:31:02      0x13C      19619 TASKING  Releasing tracewriter
2005-01-27 15:31:02      0xDD4      19600 VERSION  'Kernel    7.5.0    Build 018-123-079-776'
2005-01-27 15:31:02      0xDD4      19654 DBSTATE  SERVERDB 'HOTELDB' has stopped
---------------------------------------- current write position ----------------
```


knldiag reports that the cause of the shutdown here again was LogAndDataIncompatible, albeit without the familiar return code.

It is also noted that the DBIdentifiers of the data and log volumes do not match.

I/O Error: Verification of Checksum in Index Page



After a data page was read from a data volume, checksum 618008976 was calculated. Before writing the block, checksum 618000000 was calculated and written to the block. Apparently the block is not situated correctly on the disk.

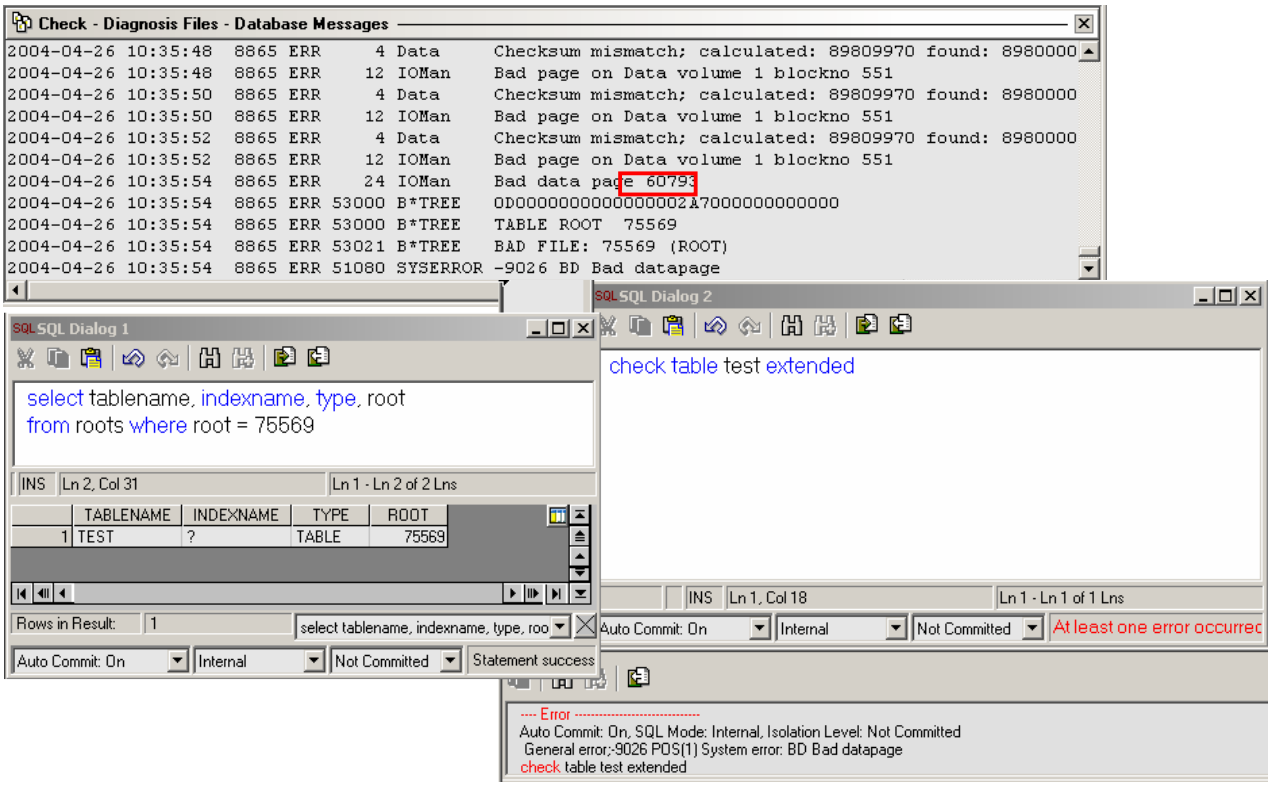
This read I/O is repeated twice. If the error occurs every time, the database assumes that the block is defective. This is a block for an index (secondary key tree). The index is marked as BAD.

Check the I/O system. If the damage to the I/O system can be repaired, you can delete the index and then regenerate it.

Under '*Recovery -> Index*' , the DBMGUI displays the indexes marked as BAD. You can select the index and recreate it.

Regardless of whether or not you are able to identify errors in the I/O system, it is a good idea to run a **CHECK DATA** in such a case.

I/O Error: Verification of Checksum in Data Page



In this case, too, a block was read whose checksum did not match with the calculated value. According to the ROOTS view, this tree belongs to the TEST table.

In such a case, check the table with the CHECK TABLE EXTENDED statement. With the option EXTENDED, the sequence of the primary keys is checked on all B* tree levels.

If CHECK TABLE does not return any errors, the table is intact. Note that in disk mirroring, depending on the disk used for the I/O, a correct block and then an incorrect block may be returned.

If CHECK TABLE continues to return the error, you have the following options:

- Restore the database
- Delete the tables and load the data from a sister system. This can lead to data loss. When tables are deleted, blocks that are no longer accessible remain occupied. In the ADMIN operational state, these blocks are transferred to free space administration with a **CHECK DATA WITH UPDATE**.
- Download the table without reading the records of the defective blocks, delete the table and load the downloaded records. The table data can be read in primary key sequence. The primary key values of the records in the defective block are not specified in the selection. This method is only possible if no index page of the B* tree has been affected. Data loss occurs.

System Error

Diagnosis of severe errors
(-10000 < Error number <= -9000)
Sometimes the database crashes.

- 1. After a crash, the start procedure copies the diagnosis files to a directory. In the standard, the parameter DIAG_HISTORY_PATH is set to <RUNDIRECTORY>/DIAGHISTORY. Only two versions of these files are kept (parameter DIAG_HISTORY_NUM)
- 2. After the crash:
  - Try to restart the database (with vtrace switched on)
- 3. Check, if the error can be reproduced (with vtrace switched on)
- 4. Inform the support group, if the cause could not be identified – open a customer message.

Diagnosis files only have to be explicitly saved if they are not automatically copied to the DIAGHISTORY.

Analysis Examples

Message -602 in Syslog

Zeit	Typ	Nr	Man	Benutzer	Tcod	MNr	Text	Datum: 17.02.05
10:48:07	DIA	1	000	E30	SE38	BYL	Datenbankfehler -602 erfordert Intervention durch Datenbankadministrator	
10:48:07	DIA	1	000	E30	SE38	BY4	Datenbankfehler -602 beim SEL-Zugriff auf Tabelle ZZTELE_902 aufgetreten	
10:48:20	DIA	1	000	E30	SE38	AB0	Laufzeitfehler "DBIF_RSQL_SQL_ERROR" aufgetreten.	
10:48:20	DIA	1	000	E30	SE38	AB1	> Kurzdump "050217 104820 p34777 E30 " erstellt.	

How can database problems be analysed using db50?

- Example 1: -9026 Bad Data Page
- Example 2: -9028 Bad File

How does MaxDB support do an analysis if the database can no longer be transferred to the Online operational state (DB state Admin)?

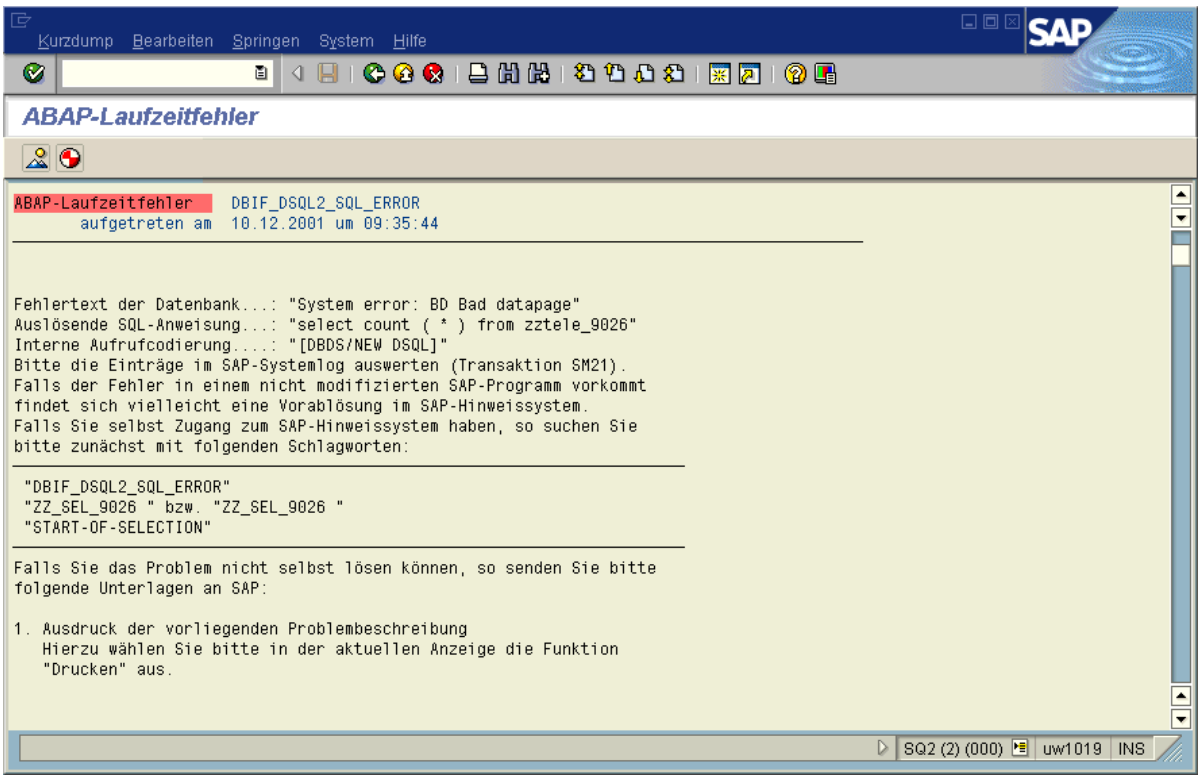
- Analyze pages
- Restart record

MaxDB system errors are "mapped" to the general error -602 in the WebAS System. So this error number does not tell you much.

If the database is still in the ONLINE operational state or has restarted it following a crash, the analysis can be continued with transaction db50.

If restarting the database is no longer possible, other measures are required.

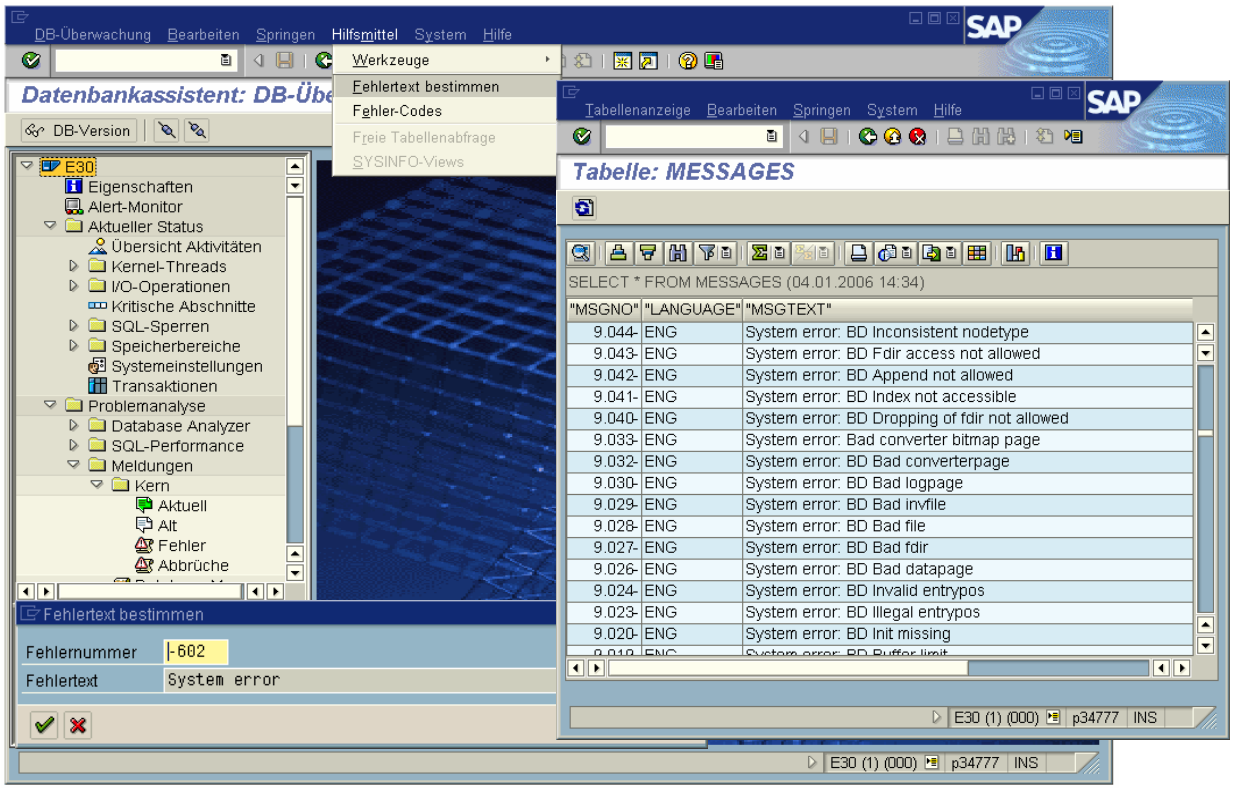
DB50: Problem Analysis -9026 Bad Data Page



A short dump with error -602 'BD Bad Data Page‘ occurred during execution of the ABAP report ZZ_SEL_9026.

The short dump thus provides more detailed information about the error than the system log and returns the corresponding text from the database.

DB50: Error Texts and Codes

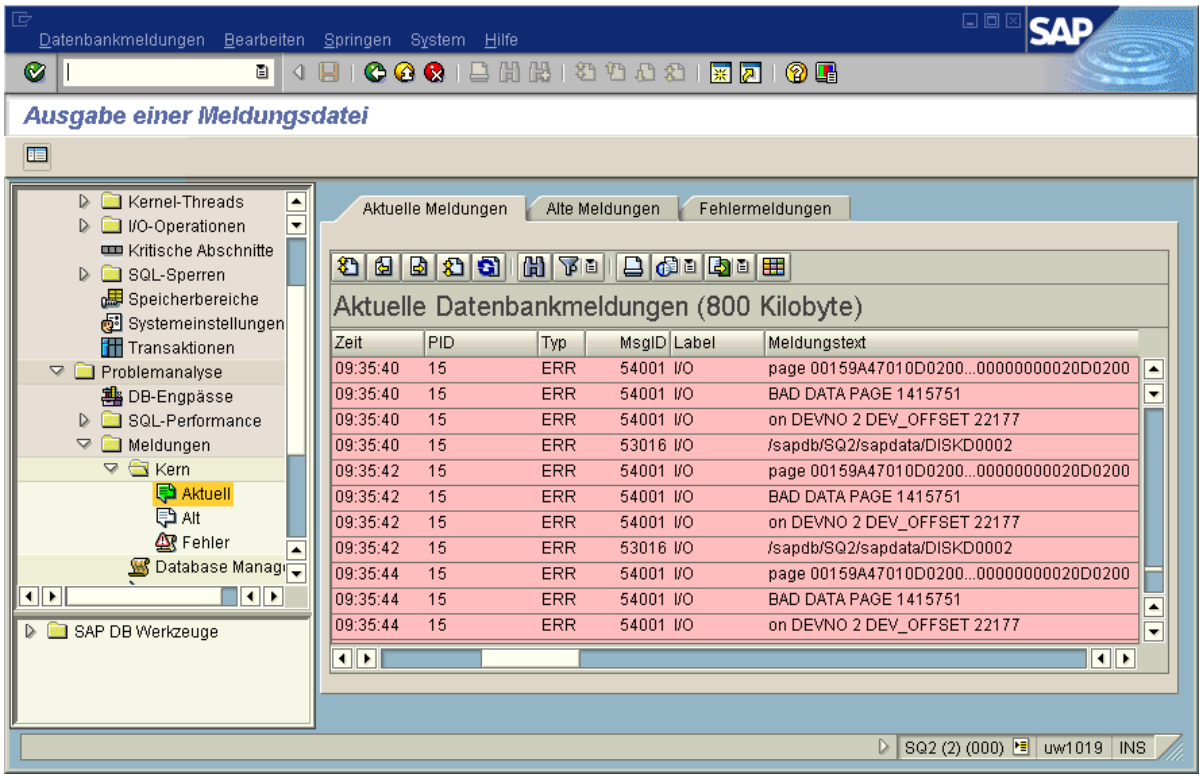


In the initial menu for transaction db50, you can find the text that corresponds to a particular number via ***Utilities -> Determine error text***. The text for error -602, however, is not terribly helpful as it is too general.

Utilities-> Error Codes provides information about (system) error numbers as well as their texts by displaying the Messages table.

IIn the present example, ***knldiag*** must be utilized for further analysis.

DB50: Problem Analysis - Messages



An short dump with error -602 'Bad Data Page‘ occurred during execution of the ABAP report ZZ_SEL_9026. The first step is to look in ***knldiag***.

To display the messages of the database system ***(knldiag)*** and the Database Manager, choose *Problem Analysis-> Messages.*

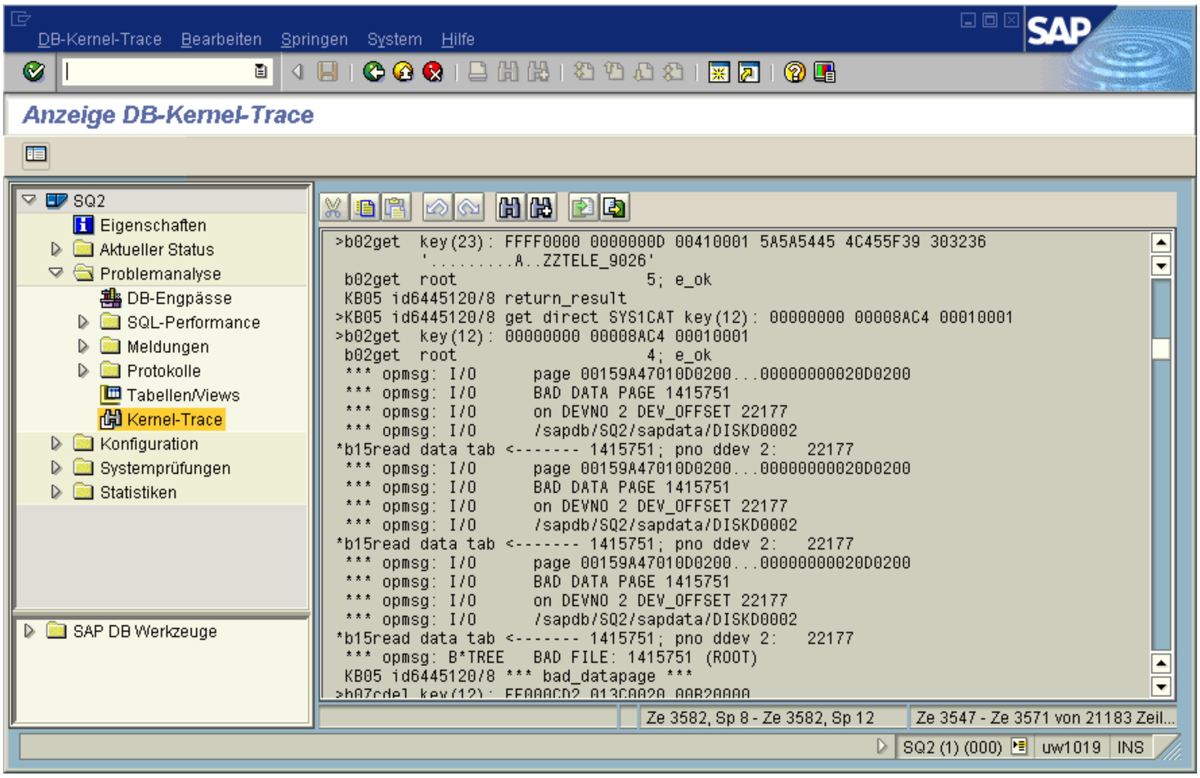
The error 'Bad Data Page‘ with error number -9026 is logged in ***knldiag***.

The root page of the affected object 1415751 is also recorded.

The position of the object is also logged. The defective object is located in Data Volume Number 2 at position 22177.

For a more precise analysis as to what is wrong with this object, the kernel trace (Vtrace) can be useful.

DB50: Problem Analysis – Kernel Trace



Using db50, a Vtrace has been created.

The root page of the affected object 141575, the volume (DevNo 2) and the position in the volume (22177) are logged.

In addition, an important section of the affected page is logged in the Vtrace which allows you to identify the cause of error -9026.

Each page has one so-called header and one trailer entry, consisting of 8 bytes each. Both entries are checked when the page is accessed.

Header entry: Page: 00 15 9A 47 01 0D 02 00

Trailer entry: Page: 00 00 00 00 02 0D 02 00

If inconsistencies appear when the header and trailer are compared, the 'Bad data page' error is sent to the application.

In this example we see that the first 5 bytes in the trailer differ from the header.

If the affected object is a database table, the database must be restored.

Example -9026: Solution

Check, if the affected object is a table.

Examine volume 2, if there are hardware problems.

Remove hardware problem.

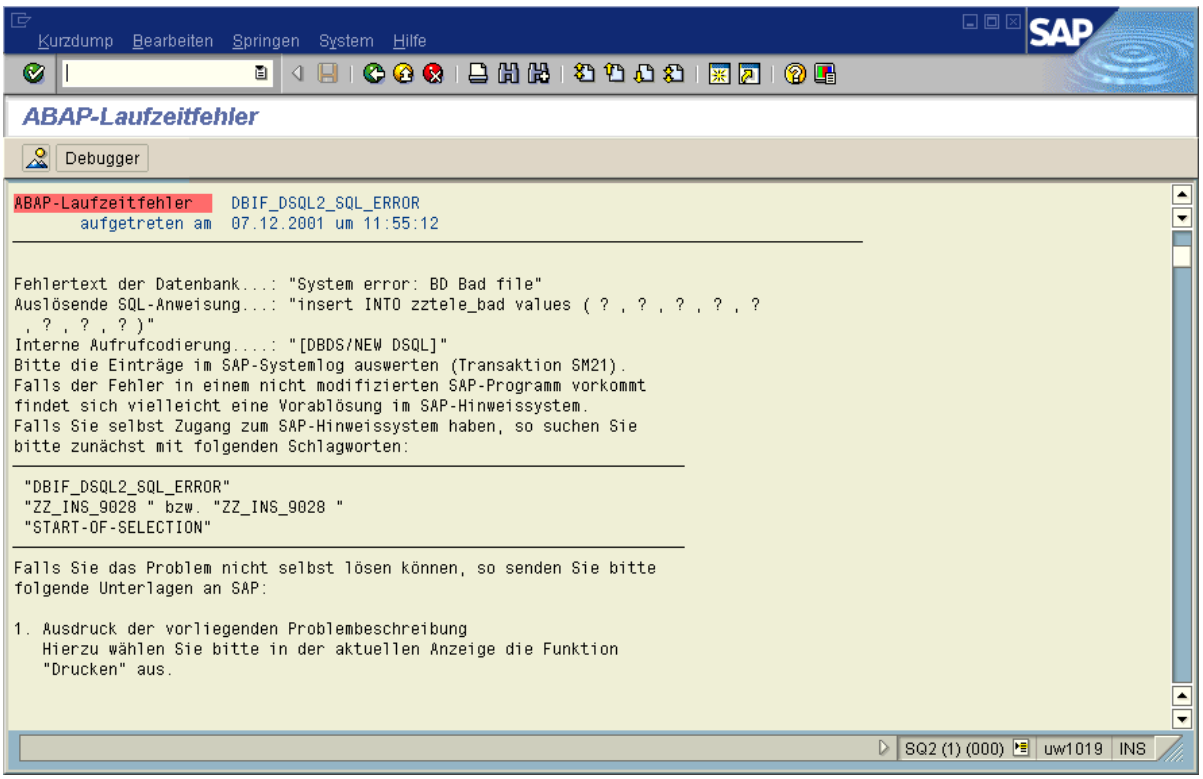
Restore of the database

CHECK DATA with transaction db13

If the affected object is an index, error -9026 could be remedied by simply deleting and recreating the index; the cause of the problem, however, would not be solved.

The hardware must be examined in any event as such cases (-9026) can be due to hardware errors.

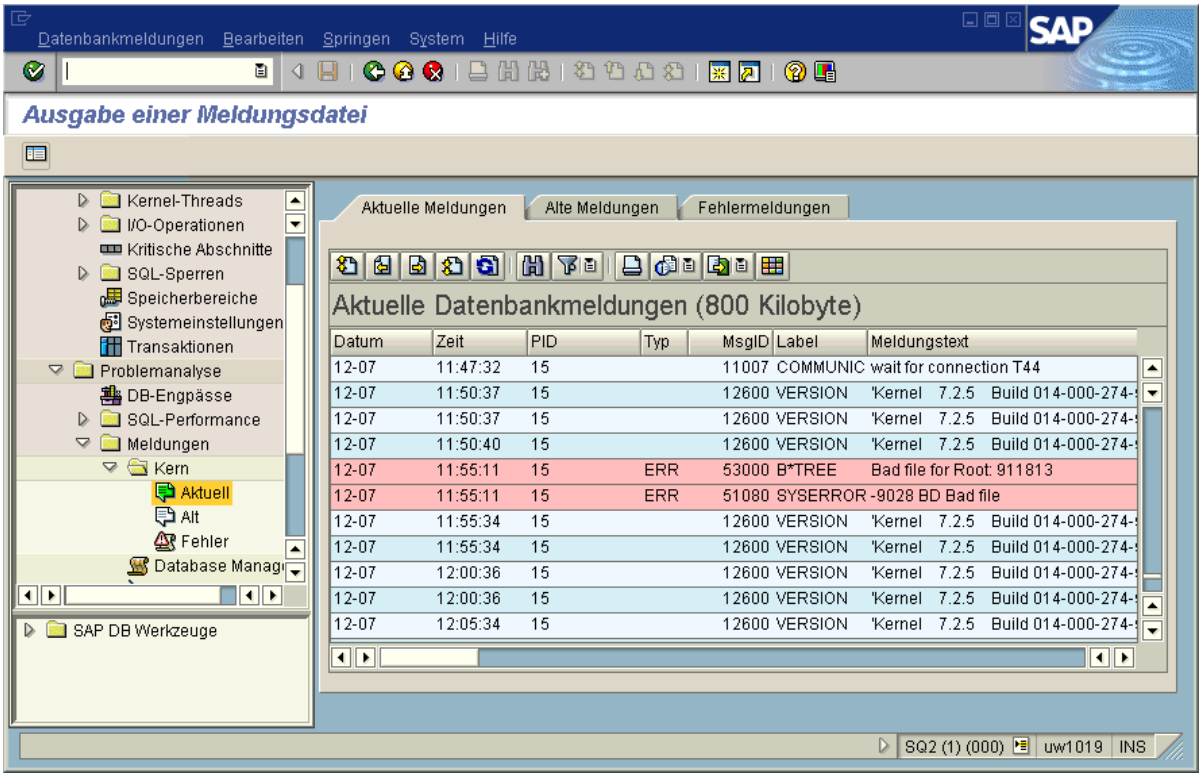
DB50: Problem Analysis -9028 BD BAD FILE



Report zz_ins_9028 terminated with a short dump.

The cause of the error, -602 'BD Bad File‘, can be seen in the short dump.

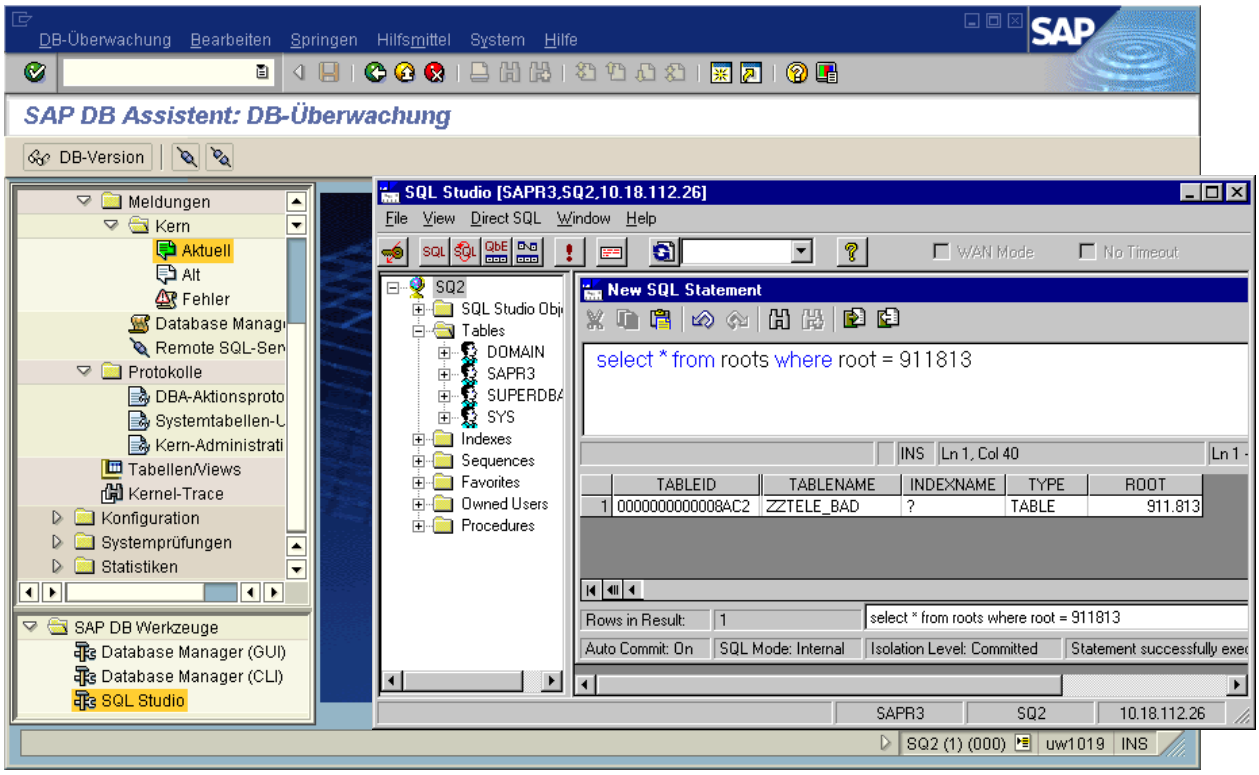
DB50: Problem Analysis – Messages



The current example has the error "-9028 Bad File“: access to the table has been blocked because a serious error (e.g. -9026) occurred.

The root page number is recorded in ***knldiag***. You can find out the table with the root page number.

DB50: MaxDB Tools



We have already seen how commands can be sent interactively to the database using the SQL Studio.

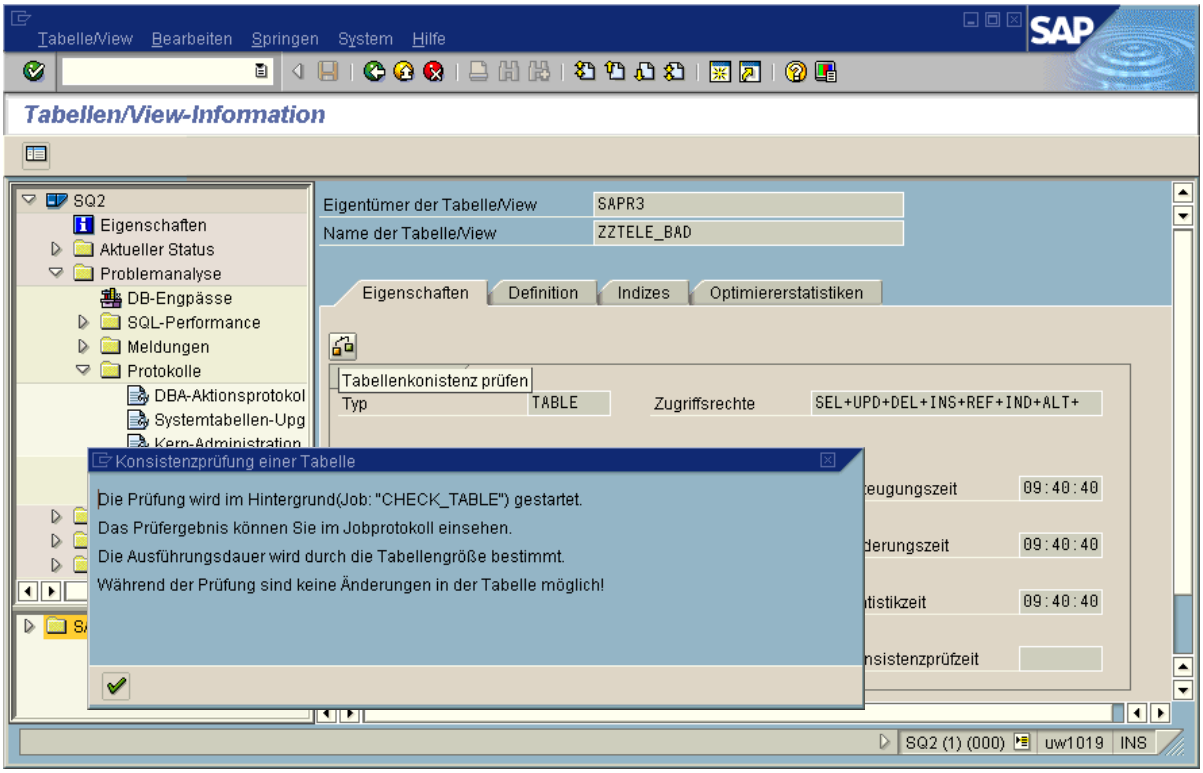
The SQL Studio can be started directly from transaction db50.

The Roots table contains the root page number, the object type and the name of the database object for all database objects.

Example – Field Types in the view roots

- SYS — System table (not accessible)
- NAMED INDEX — named index
- UNNAMED INDEX — unnamed index (for one column) (INDEXNAME=column name)
- TABLE Table
- SHORT STRING FILE — contains the short BLOB COLUMNs (exists for each table with BLOB COLUMNs)
- LONG COLUMN (OWNER, TABLENAME and INDEXNAME not specified)
- TEMP — temporary table

DB50: CHECK TABLE



In the present example, we know that table zztele_bad has caused a problem; a consistency check is triggered.

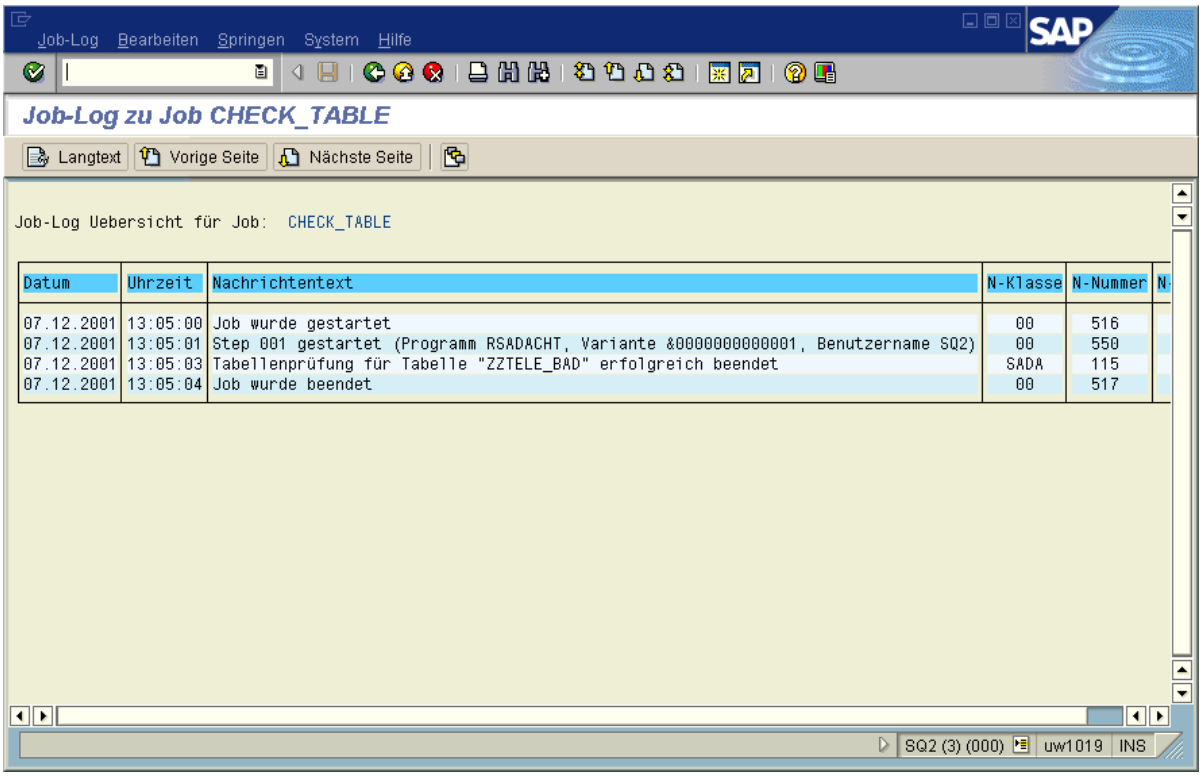
A Check Table is executed on the table.

Check Table checks the tree structure of the B* tree, header-trailer and so on.

If no inconsistencies are found, the BAD flag is retracted and access to the table is enabled.

This can happen if, for example, a Raid system reports an error but then corrects it immediately. Then the table is consistent, but has nevertheless been set to BAD.

SM37: Job Log Check_table



The '*Check Table*' executed without problems and reported no errors.

The program that had terminated with -9028 or -602 System Error can now be restarted.

x_diagnose

Direct access to database pages (data, converter, log)

Extraction of B* trees

Analysis of knldump

Command: <instroot>/bin/x_diagnose

The tool ***x_diagnose*** allows you to access log pages and data pages in the database directly. With ***x_diagnose***, you can can export configuration or restart information from the pages.

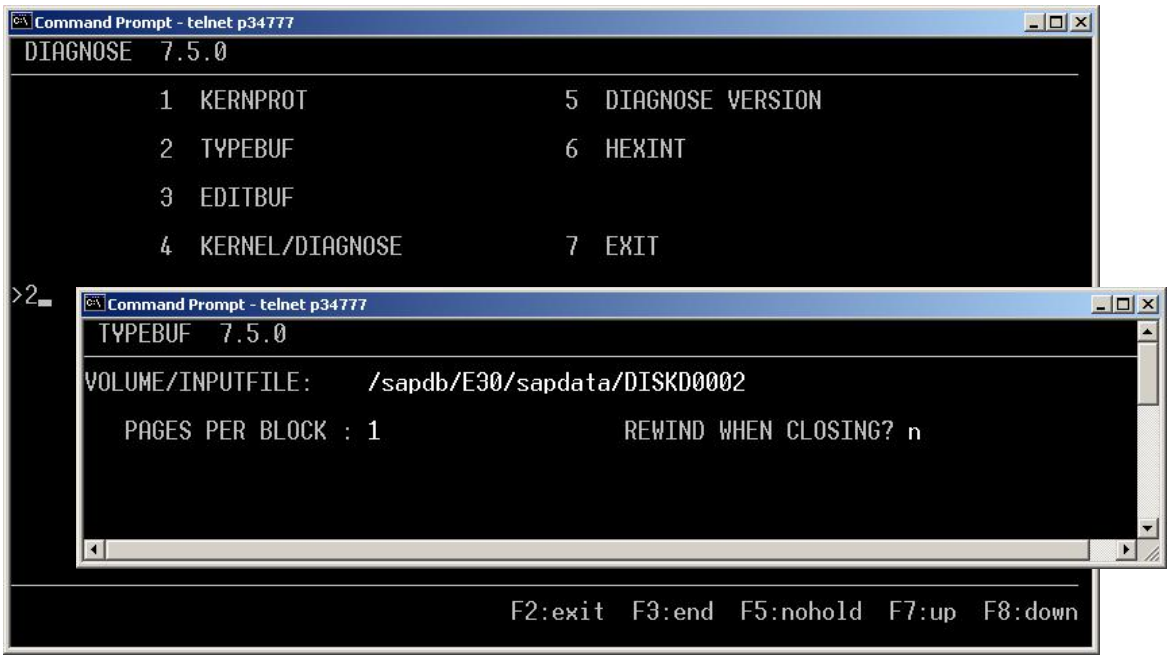
If necessary, you can extract an entire table tree.

x_diagnose is also used to evaluate ***knldumps***. Cache contents, converter information, lock list entries, etc., can be analyzed at a later time.

Because improper use of the tool can be dangerous, ***x_diagnose*** should only be used by development.

In exceptional cases, pages can be repaired directly using an editing function.

Diagnosis: Data Page (1)

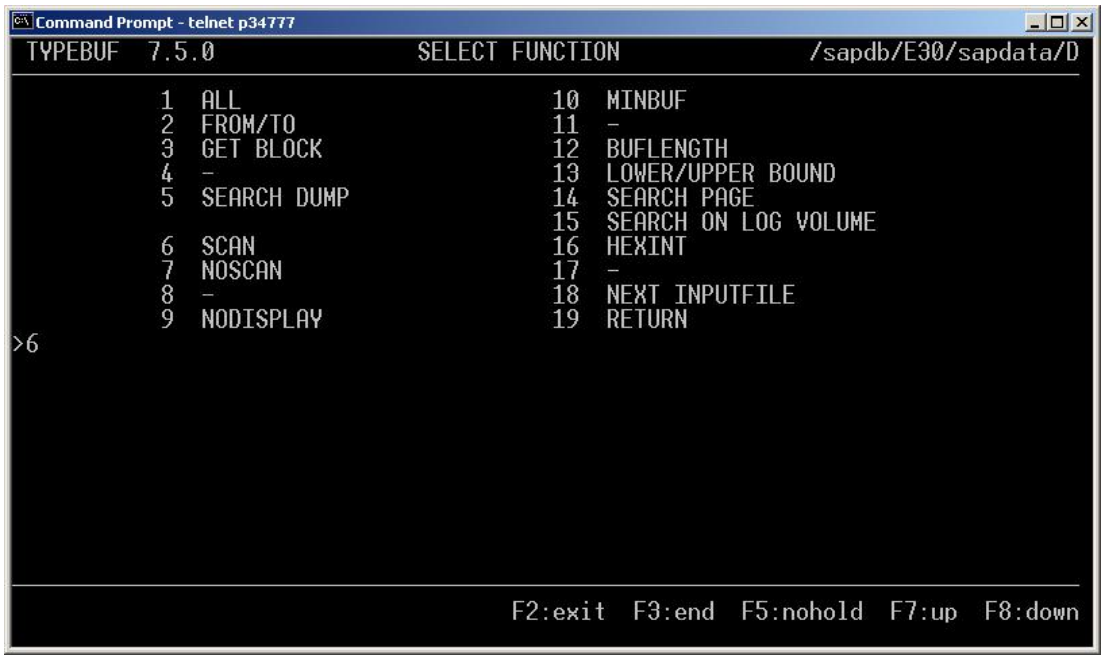


The following pages show how to extract a data page with ***Diagnose***.

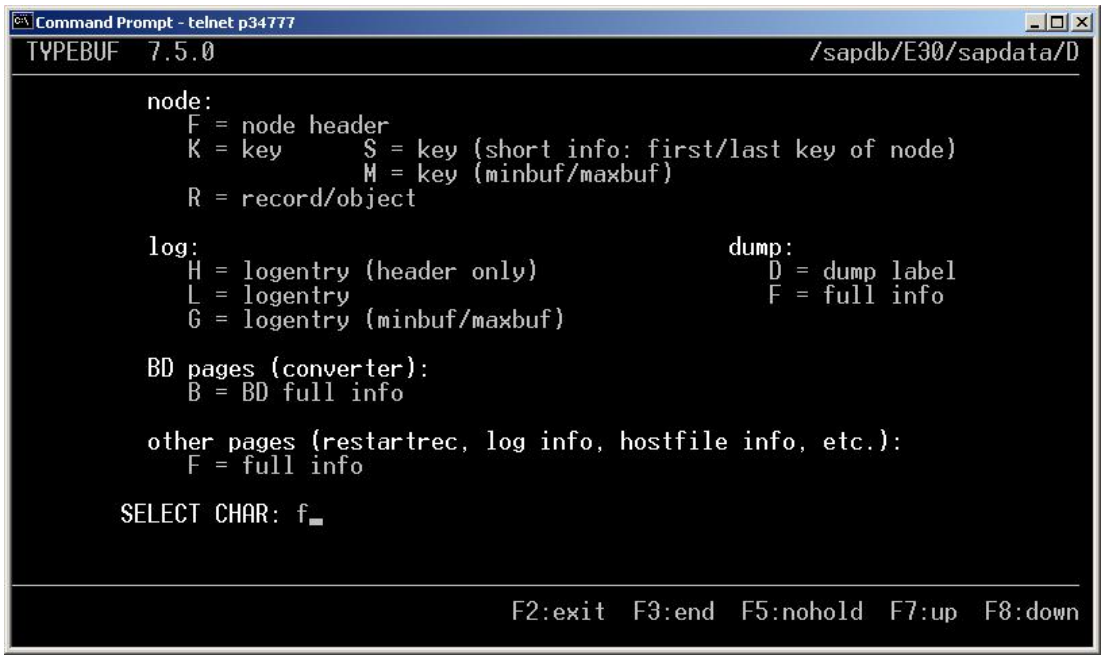
First you choose TYPEBUF.

Then you enter the volume name.

Data Page (2)



Data Page (3)

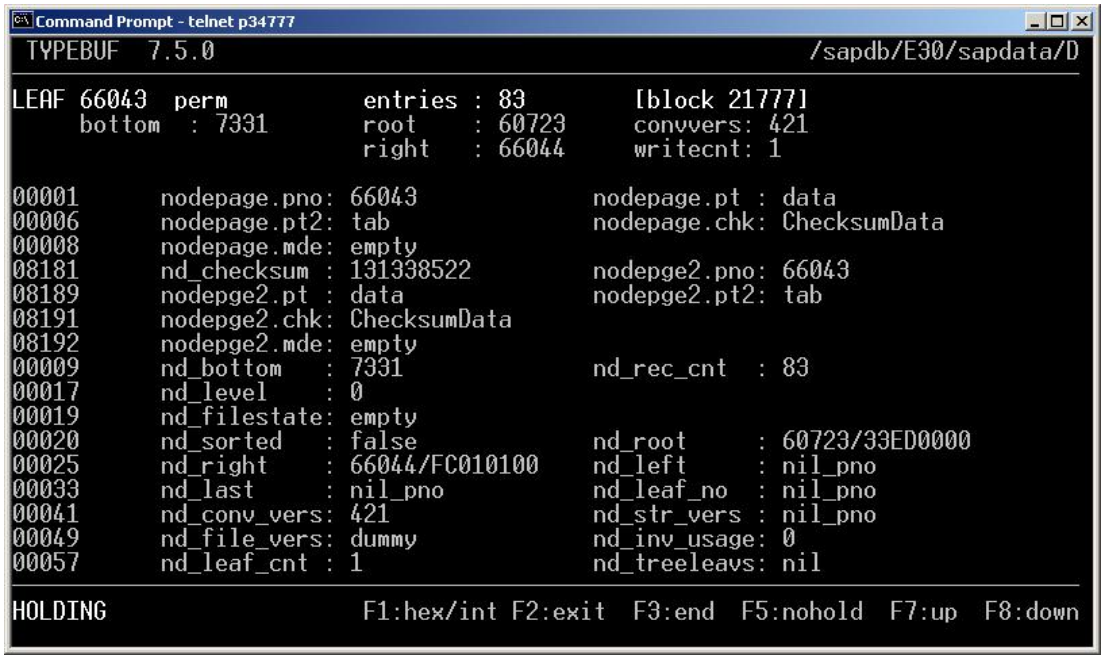


Using the SCAN menu, you can then specify what information you desire.

Data Page (4)

By specifying a block address - taken, for example, from ***knldiag*** - you come to the desired page.

Data Page (5)

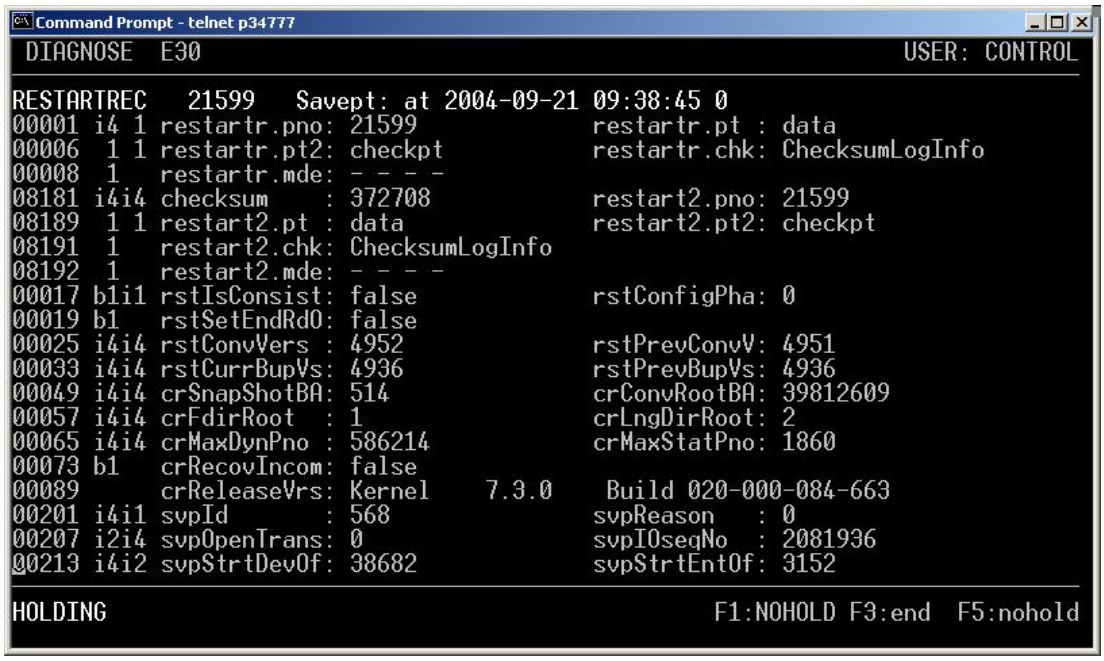


Diagnosis: Restart Record (1)

You want to check the restart record.

You can access the restart record in various ways. One way is is to choose KERNEL/DIAGNOSE, which brings you to the menus displayed here.

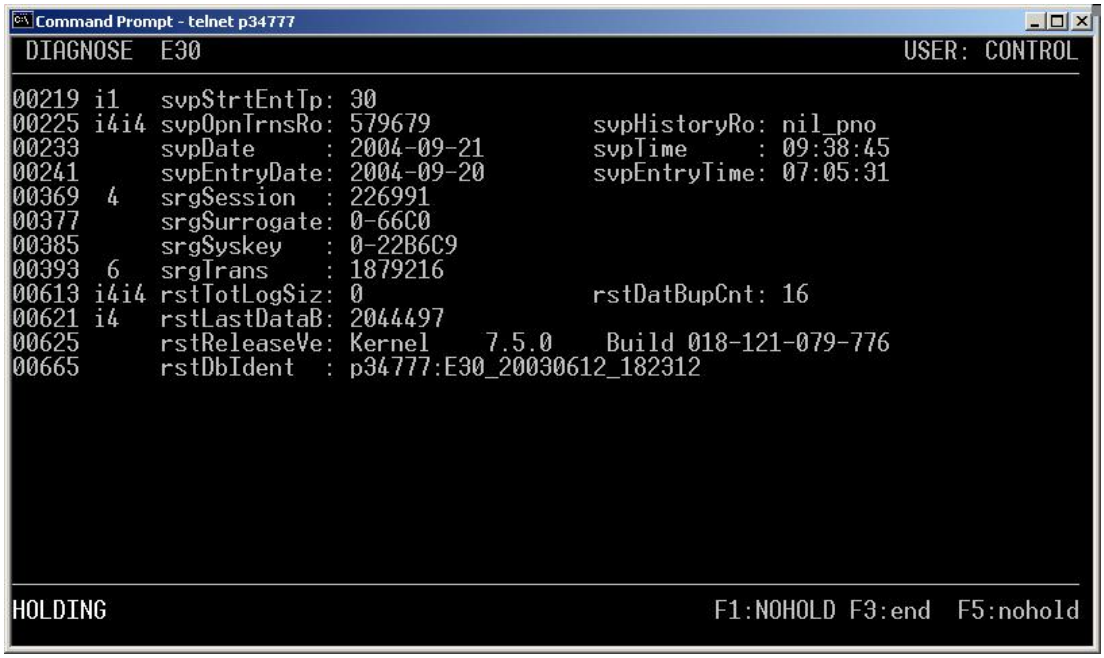
Restart Record (2)



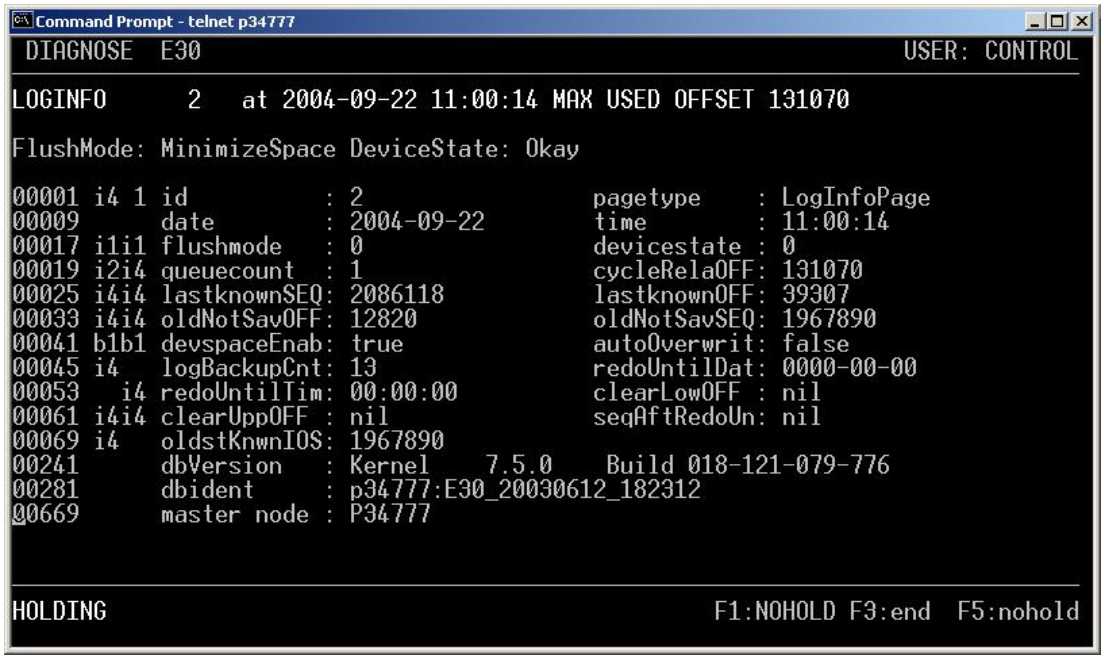
The last Savepoint was written on 21.09.2004.

The database is in an inconsistent state (rstIsConsist: false).

Restart Record (3)



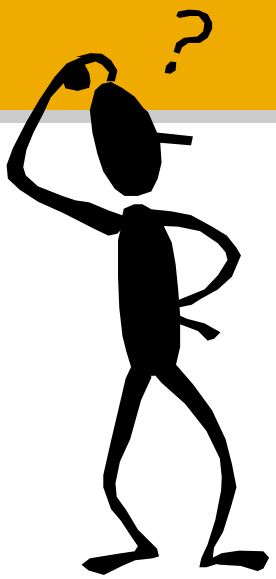
Diagnosis: Loginfo Page



You access Loginfo Page via the same menu (GET LOG INFO).

The DBIdent, among other things, can be determined here.

Any questions?



THE BEST-RUN BUSINESSES RUN SAP™