

MaxDB Internals Performance Analysis

Werner Thesing

THE BEST-RUN BUSINESSES RUN SAP™ 



x_cons

- shows current DB activity (snapshot)

Database Analyzer

- detects possible bottlenecks
- collects and stores data at given intervals

Diagnostic Monitor (Command Monitor)

- Lists single long running SQL commands

Diagnose Analyze (Resource Monitor)

- Displays accumulated data for all SQL commands

Performance Analysis Tools

MaxDB provides various tools and methods for the analysis of performance bottlenecks and monitoring current database activities. Some of these tools were originally developed only for testing and analysis in MaxDB development, but can also be used by experienced database administrators for performance analysis.

The following are of particular importance for performance analysis:

- The **x_cons** console for monitoring current operations
- The **Database Analyzer** program for analyzing performance bottlenecks
- The diagnostic function **DIAGNOSE MONITOR** for identifying long-running or poorly-processed SQL statements
- The diagnostic function **DIAGNOSE ANALYZE** for displaying information about all current SQL statements

x_cons and Database Analyzer are stand-alone programs and are called from the operating system command line. DIAGNOSE MONITOR is a part of the core functions of MaxDB.

In SAP WebAS, all functions and results can be controlled and analyzed using transaction DB50 => Current Status or DB50 => Problem Analysis. Required parameter settings, if any, are menu-driven.



Database console x_cons features:

- process overview
- configuration overview
- observing session activities and wait states
- watching I/O activities and wait queues
- measuring of detailed task specific times

Call:

- **x_cons <serverdb> <command> [<interval>] [<repeat>]**

e.g. x_cons E30 show active 10 6

advantage: delta information using ,interval' and ,repeat'

- **dbmcli -d ... -u ... [-n <node>] db_cons <command>**

advantage: works per remote connection to database host

DB Console x_cons

The database console **x_cons** gives you a quick overview of the operating system resources that the database system is using, the distribution of the database session among the operating system threads, and the status of the active database sessions. You can also use other functions that are intended mainly for support employees and developers.

Start on shell level: `x_cons <dbname > <command> [<interval>] [<repeat>]`

`x_cons <dbname> help` returns a complete overview of all available command functions.

The database console can also be addressed remotely via the DBM server.



```

x_cons <dbname> <Command> [<interval>] [<repeat>]
<Command> (choice):
show io statistics/states
show move info (load balancing)
UKT sleep statistic
suspend reasons
show task counts
show tasks move info
show task queues
show task regions
show task statistics
Thread time usage
cancels the command of task
displays help file
time measurement
kills the session of task

SHOW ACTIVE          [ DW | SV | US | GC]
SHOW ALL
SHOW AIO (backup only)
SHOW IO
SHOW DEV_IO
SHOW IOPENDING
SHOW CPORT
SHOW MOVEINFO
SHOW QUEUES
SHOW REGIONS
SHOW RTE
SHOW RUNNABLE       [ DW | SV | US | GC]
SHOW SLEEP
SHOW STATE
SHOW STORAGE
SHOW SUSPENDS
SHOW T_CNT          [ DW | SV | US | T<taskindex>]
SHOW T_MOVE
SHOW T_QUEUE
SHOW T_REG
SHOW T_STAT
SHOW TASKS
SHOW THRD_TIMES
SHOW SLEEP
SHOW VERSIONS
CANCEL <taskindex>
HELP
TIME <ENABLE | DISABLE>
KILL <taskindex>
    
```

CANCEL	index	cancels the command executed by task <index>
KILL	index	kills the session of task <index>
SHOW [LONG COMPRESS] (Unix)		
DEBUGLEV	level	set debug level for the kernel
DEBUGTASK	index	writes back trace of task to knldiag
RESET T_CNT REGIONS (ALL)	obj_cnt	resets counter about the following objects: IO incl. local counters of any task
ERRIOR	devno	forces error at next read to devno
ERRIOW	devno	forces error at next write to devno
TIME	enable	enables time measurements
! command		execute shell command (Unix only)
QUIT		exit console driver (Unix only)

x_cons Process Configuration (1)



```
x_cons <dbname> show rte
```

Kernel Threads:

Thread Name	UNIX Tid	State	Sleep Time
TIMER	16660	Sleeping	2
COORDINATOR	16639	Sleeping	
CLOCK	16654	Sleeping	
CONSOLE	16655	Sleeping	
REQUESTOR	16658	Sleeping	

User Kernel Threads:

Thread Name	UNIX Tid	State	Dispatch Counter	TaskSwitch Counter	Active Tasks	Total Tasks	Task Cluster
UKT1	16669	Sleeping	6	0	1	1	TW
UKT2	16670	Sleeping	1687	0	1	1	LW
UKT3	16671	Sleeping	2	0	1	1	UT
UKT4	16672	Sleeping	1281	1206	406	406	406*SV
UKT5	16673	Sleeping	7825	837	11	18	10*GC,8*FS
UKT6	16674	Sleeping	71738	6527	65	65	TI,64*PG
UKT7	16675	Sleeping	142908	57	6	50	50*US,IDL
UKT8	16676	Sleeping	93433	9	5	52	50*US,IDL

Kernel parameters (please do not change directly):

```
TaskCluster01 tw;lw;ut;2000*sv;10*fs,10*gc;
TaskCluster02 ti,100*pg;1*bup,50*us;
TaskCluster03 equalize
```

Processor information:

```
Processors : 4
Processor cores: 2
```

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 5

x_cons <dbname> show rte

This shows the distribution of the MaxDB threads among the operating system processes.

The DB threads coordinator, console, timer, requestor and Dev0 each have their own operating system threads. The entire database kernel runs in a single process.

However, multiple database tasks (user task, log writer, utility task, and so on) can be located together in an operating system thread, which is called a UKT (user kernel thread). The MaxDB runtime environment uses internal tasking to administer these database tasks. Internal MaxDB administration takes up less operating system time, and gives you more control over the scheduling and prioritization of individual database sessions.

The database parameter MAXCPU is normally used to distribute the tasks automatically to the UKTs; the (support) database parameter TASKCLUSTER (requires change in the control file cserv.pcf) can also be used for this purpose, but only in consultation with SAP support.

x_cons Process Configuration (2)



I/O via UKTs and I/O Threads:

Thread Name	UNIX Tid	Volume Name	Devs. No.	Read Count	Write Count	Queue Len.	Max.
UKT1	16669	knltrace	1	0	174	--	(--)
UKT2	16670	/sapdb/A...DISKL001	11	0	1745	--	(--)
I/O0	0	knltrace	1	0	1	0	(1)
I/O0	0	/sapdb/A...ISKD0001	2	0	0	0	(0)
I/O1	0		2	171775	66	1	(1)
I/O2	0		2	65178	16	0	(1)
I/O3	0		2	23567	6	0	(1)
I/O4	0		2	6663	3	0	(1)
I/O5	0		2	1270	3	0	(1)
I/O6	0		2	173	2	0	(1)
I/O7	0		2	7	0	0	(1)
I/O0	0	/sapdb/A...ISKD0002	3	0	0	0	(0)
I/O1	0		3	154935	57	0	(1)
I/O2	0		3	59352	21	0	(1)
I/O3	0		3	21569	8	0	(1)
I/O4	0		3	6099	4	0	(1)
I/O5	0		3	1085	3	0	(1)
I/O6	0		3	128	1	0	(1)
I/O7	0		3	17	0	0	(1)
I/O0	0	/sapdb/A...ISKD0003	4	0	0	0	(0)

...

Kernparameter:

VolumeIOQueuesFor_____Priority: number of I/O-Threads per volume, see chapter Kernel parameters

Abbreviations of the Database Tasks in TASKCLUSTER:

Abbreviation

- tw Trace writer, writes kernel traces and dumps
- ti Task for timeout monitoring
- al Log writer
- dw Tasks for cache monitoring and asynchronous cache displacement as well as savepoint I/O
- ut Utility task for administration tasks (start backup, recovery, and so on).
- sv Server processes for backup I/O and special operations such as parallel index generation
- us User tasks for executing SQL statements
- gc Garbage collector
- ev Event task
- fs Floating service for load balancing

x_cons Task Activity



```
x_cons <dbname> show active [<interval>] [<repeat>]
```

```
x_cons E70 show active 10 6
```

ID	UKT	UNIX tid	TASK type	APPL pid	Current state	Timeout priority	Region cnt try	Wait item
T146	7	-1	User	28069	Running		0 220 99	741131 (r)
T147	7	-1	User	28072*	Runnable	48	0 111	741131 (r)
T152	8	-1	User	28071*	Runnable	56	0 76	424309 (r)
T154	8	-1	User	28070	Running		0 55	424309 (r)
.....								
T2	2	-1	Logwr	-1	IO Wait (W)		0 1 5	1978 (s)
T152	8	-1	User	28069	LogIOwait(234)		0 0	424800 (s)
.....								
T66	6	-1	Pager	-1	Vvectorio		0 0	3258 (s)
T67	6	-1	Pager	-1	IO Wait (W)		0 0 1	3258 (s)
T87	4	-1	Savepnt	-1	PagerWaitWritr		0 0	234617 (s)
.....								
T75	4	-1	BUPvol	-1	AsynWaitRead		0 0	11368 (s)
T76	4	-1	BUPvol	-1	AsynWaitWrite		0 0	11368 (s)
T159	8	-1	User	28072	IO Wait (R)		0 0 2	429215 (s)
.....								
T152	8	-1	User	28069	InvRootExcl		0 0 74573	24185 (r)
T154	8	-1	User	28070	Running		0 55	438561 (r)
.....								
T142	7	-1	User	0*	Vwait		0 0	745843 (s)
T157	8	-1	User	0*	IO Wait (R)		0 0 1	852579 (s)

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 7

```
x_cons <serverdb> show active [<interval>] [<repeat>]
```

Presents an overview of the states of all active tasks.

Appl pid

- Process ID of the application program linked to the task. An asterisk (*) before the PID indicates that the process ID is on a separate computer and is being accessed remotely.

Region

- cnt: Displays the number of times the region has been accessed since the task has been running.
- try: The number of the queried or held region

UKTsleap

- Number of semaphore waits per UKT



AsynClose	closes an I/O port after backup or recovery
Asyncntl	determines parameter or initialises a backup device
AsynIO	asynchronous I/O (during backup oder recovery)
AsynOpen	opens an I/O port for backup or recovery
AsynWaitRead	waits for an I/O operation to end, then read (backup or recovery)
AsynWaitWrite	waits for an I/O operation to end, then write (backup or recovery)
Command reply	delivers a result to the application
Command wait	task is waiting for a new request
Connect wait	task is free for a new session
DcomObjCalled	a DB-procedure or a COM-object is currently executed
Diaginit	initialises the datenbase internal trace files
Inactive	task is in initial state and has no resources yet
InsertEvent	creates an event
IO Wait (R)	waiting for I/O (R=read)
IO Wait (W)	waiting for I/O (W=write)
IO2 Wait (R)	waiting for I/O for mirrored disk (log only)
IO2 Wait (W)	waiting for I/O for mirrored disk (log only)
Locked	task is locked during kernel shutdown (to prevent rescheduling)

In a system with one CPU, only one task can be running at a given time. If x_cons nevertheless shows two tasks running, this is due to unprotected access.



Not Connected	
RescheduleMsec	brief wait, continues automatically
Runnable	immediately runnable
Running	running, using CPU time
Stopped	suspended by kernel and waiting to proceed running
Terminated	task or database session has been canceled
UNKNOWN	task state unknown
Vacknowledge	
Vattach	opens I/O ports (volumes, normal operation)
Vbegexcl	waiting for protected memory access
Vblockio	runnable after protected memory access
Vdetach	closes I/O-ports (volumes, normal operation)
Dual Vector I/O	performs a vector-I/O-operation on two volumes in parallel
Vendexcl	leaving a protected area
Enter ExclLock	waiting to access a protected region with an exclusive lock
Enter ShareLck	waiting to access a protected region with a share lock

x_cons: Task States (3)



Leave ExclLock	leaves a protected region
Leave ShareLck	leaves a protected region
ShutDown	database is shut down (changing state from ONLINE to ADMIN)
Connect close	ends the database session
Vsleep	brief wait, continues automatically
Vsuspend	suspended and waiting to be explicitly activated by another task (e.g. for B*-Tree locks (very brief) or log I/O)
Vvectorio	performs a vector-I/O-operation (reading or writing)
Vwait	waiting to be explicitly activated by another task (e.g. waiting for an SQL-lock)
WaitForEvent	waiting for an event
Yielding	Briefly cedes control of CPUs during Busy Waiting

x_cons Task Detail



```
x_cons <dbname> time enable
x_cons <dbname> show t_c t<task_index>
```

```
----- T25   User      ( pid = 23163 ) -----
remote_node   : myserver                remote_pid    : 23163
dispatcher-cnt: 127292                  command-cnt   : 30477
total_excl-cnt: 9110558                self_susp-cnt : 433
dev_write_io  : 19                      dev_write_pg  : 19      avg_dev_wr_tm : 0.0895
state_vwait   : 11                      avg_vwait_time: 4.1446
state_vsusp   : 682                    avg_vsusp_time: 0.0584
rcv_rpl_count : 2296                    rcv_rpl_long  : 46      avg_rcv_rpl_t : 0.1577
rpl_rcv_count : 2296                    avg_rpl_rcv_t : 0.0222
dev_que_len_0 : 18                      dev_que_len_1 : 0      dev_que_len>1 : 0
```

#Statements > 1 second

Avg.. I/O time for
Dev process

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 11

`x_cons <dbname> show t_c t<task_index>` displays highly- detailed measurement values for individual database tasks. In this way, you can, for example, monitor the DB activity of an application while it remains connected to a database task (no permanent release/connect).

Furthermore, with `x_cons <dbname> time enable` you can activate precise time measurement of the different database states. Depending on the operating system, this time measurement costs between 1% and 5% performance.

Much of the output of the '`show t_c`' function was developed exclusively for developers, however, some of the values are of more general interest in special situations.

dispatcher-cnt	Count of how often the task passed control to the UKT dispatcher, because it could not run, its time slot had expired, or another task was prioritized.
total_excl-cnt	Number of region accesses
command-cnt	Communication count between application and kernel
self_suspend-cnt	Number of task suspensions in which the task remained executable but still gave up control
<dev/self>_<read/write>_io	Number of I/Os via UKT (self) or DEV threads (dev)
<dev/self>_<read/write>_tm	Duration of an I/O via UKT (self) or DEV threads (dev)
state_vwait	Number of waits on SQL locks
avg_vwait_time	Average wait time for an SQL lock
avg_rcv_rpl_t	Average processing time of an SQL statement in the database kernel

rcv_rpl_long
second

Number of SQL statements with a processing time of more than one

x_cons I/O Activities



```
x_cons <dbname> show io
```

Volume	No.	Read(s)	RPages	Write(s)	WPages
/sapdb/E70/sapdata/DISKD0001	1	10539	10539	11	12
/sapdb/E70/sapdata/DISKD0002	2	10525	10525	23	27
/sapdb/E70/sapdata/DISKD0003	3	10338	10338	22	22
/sapdb/E70/sapdata/DISKD0004	4	10000	10000	25	25
/sapdb/E70/saplog/DISKL001	5	0	0	36	36
total I/O		41402	41402	117	122

```
x_cons <dbname> show dev_io
```

I/O via I/O Threads only:

Volume Name	Devs. No.	Read(s) Count	Read Pages	AvgRead Time(ms)	Write(s) Count	Write Pages	AvgWrite Time(ms)
/sapdb/....ISKD0001	2	3	3	0.838	1	1	0.929
/sapdb/....ISKD0001	2	603392	603392	3.660	622	633	0.702
/sapdb/....ISKD0001	2	266987	266987	4.589	28	35	3.576
..							
/sapdb/....ISKD0009	10	304632	304632	3.522	622	740	0.658
/sapdb/....ISKD0009	10	13074	13074	4.018	13	95	9.436
/sapdb/....ISKD0009	10	196	196	8.681	13	78	8.379
total I/O:		8613812	8613843		6247	7095	

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 12

The command *show io* displays the number of read and write operations per volume as well as the number of 8 KB pages read. These numbers are independent of whether the I/O are synchronous or asynchronous.

Show dev_io displays the number of read and write operations of the I/O threads and the average I/O times.



SYSMON_SPECIAL_THREAD

- Shows special threads
- Similar to „x_cons <serverdb> show rte“
(see x_cons process configuration (2))

SYSMON_UKTHREAD

- Displays all threads containing tasks
- Analog to „x_cons <serverdb> show rte“
(see x_cons process configuration (1))

SYSMON_IOTHRD

- Shows I/O threads which performed I/O
- analog to „x_cons <serverdb> show rte“
(see x_cons process configuration (2))

Performance Tables/ Database Console

Much of the data generated with x_cons is also accessible through tables. Thus this performance data can also be displayed by other tools (SQLStudio, SAP WebAS->DB50).

The columns of the respective tables largely correspond to those of the database console.



SYSMON_TASK

- Shows all tasks
- analog to „x_cons <DBNAME> show tasks“

SYSMON_US

- shows all User Tasks
- analog to „x_cons <DBNAME> show tasks us“

SYSMON_DW

- shows all DataWriter Tasks
- analog to „x_cons <DBNAME> show tasks dw“

SYSMON_SV

- shows all Server Tasks
- analog to „x_cons <DBNAME> show tasks sv“



SYSMON_ACTIVE_TASK / SYSMON_RUNNABLE

- shows all active tasks
- analog to „x_cons <serverdb> show [active|runnable]“

SYSMON_US_ACTIVE / SYSMON_US_RUNNABLE

- shows all active User Tasks
- analog to „x_cons <serverdb> show [active|runnable] us“

SYSMON_DW_ACTIVE / SYSMON_DW_RUNNABLE

- shows all active DataWriter Tasks
- analog to „x_cons <serverdb> show [active|runnable] dw“

SYSMON_SV_ACTIVE / SYSMON_SV_RUNNABLE

- shows all active Server Tasks
- analog to „x_cons <serverdb> show [active|runnable] sv“

SYSMON_BACKUIOACCESS

- contains all volumes / backup-media, which have been used for a save / restore operation
- analog to „x_cons <serverdb> show aio“

SYSMON_IOACCESS

- all volumes, which have been accessed during MaxDB operation (but no SAVE / RESTORE)
- analog to „x_cons <serverdb> show io“



SYSMON_REGION

- all synchronized objects necessary to sync „critical areas“.
- analog to „x_cons <serverdb> show region“

SYSMON_STORAGE

- shows the tasks memory requirements
- analog to „x_cons <serverdb> show storage“

SYSMON_TASK_DETAIL

- detailed data for all [or a single] tasks
- analog to „x_cons <serverdb> show t_c [task_ID]“

SYSMON_CONNECTION

- displays additional connect-Information e.g. the Vservers (x-server) PID when using remote communication
- analog to „x_cons <serverdb> show connection“

SYSMON_TOTALCOUNT

- sums all counters
- analog to „x_cons <serverdb> show total“

Task Manager - Active Tasks Table:

ID	Thread-ID	Task-Typ	A	Task-Zustand	Zustandsbeschreibung	Wart.	Warten	Applikation	Applikationsserver
37	1582	User		Running				2887	p34777
61	1607	User		IO Wait (R)	/sapdb/E30/sapdata/D			2885	p34777
63	1607	User		Running				2884	p34777

Prozessübersicht Table:

Nr	Typ	Pid	Status	Grund	Start	Err	Sem	CPU	Zeit	Report	Man	Benutzer	Aktion	Tabelle
0	DIA	2884	läuft		ja				486	ZFBAD	000	E30	Sequentielles Lesen	ZZVIEW
1	DIA	2885	läuft		ja					SAPLTHFB	000	E30		
2	DIA	2886	wartet		ja									
3	DIA	2887	wartet		ja									
4	DIA	2888	wartet		ja									

The task manager in transaction DB50 displays all database tasks and their current status. The system displays an overview of the database tasks and information about the current state of each individual task.

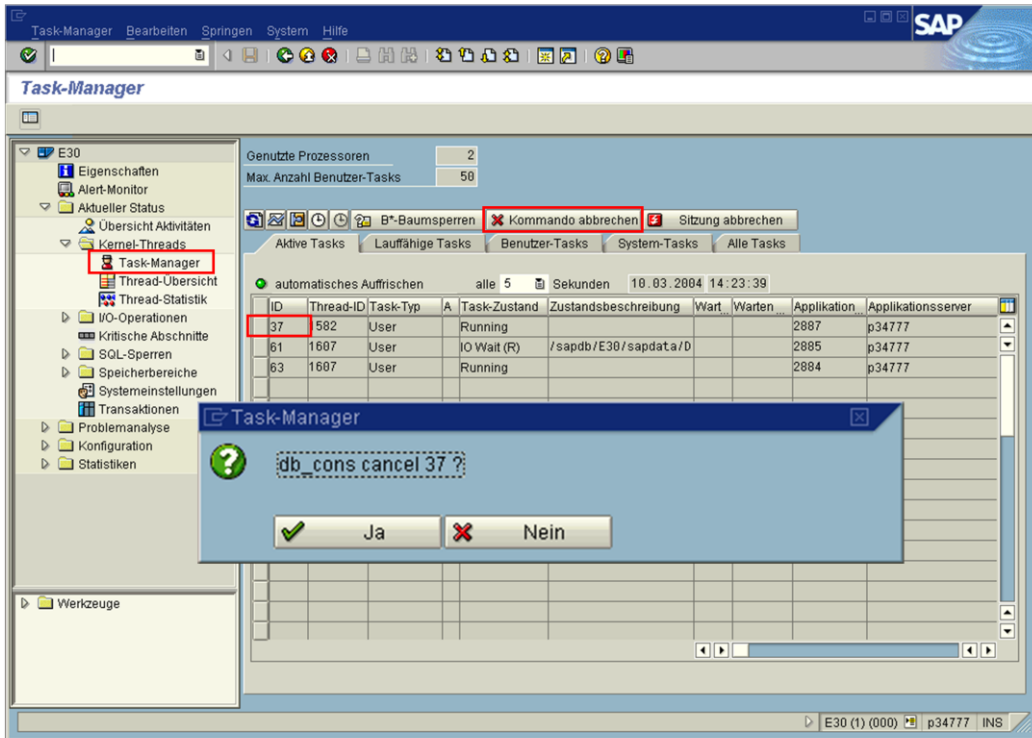
The following views are available: *Active Tasks*, *ExecutableTasks*, *User Tasks (task type User)*, *System Tasks*, *All Tasks*.

We use the task manager to analyze the following:

- For this database MAXCPU is set to 2. Thus the database can use 2 CPUs in parallel. Task T37 is running in another UKT (see Thread Overview, thread ID:1582) as task T61 and T63 (thread ID:1607). Tasks T37 and T63 can both have the *Running* status.
- We see a command (T61) that reads data from the disk to the cache - IO-WAIT (R).

In the Application column we see the process ID of the work process and via the Application Server column we see the SAP application server.

With transaction SM50, we can identify the application that caused the long-running command using the application PID (2887).



With the task manager it is possible to terminate the respective task (T37) directly on the database level.

The information in the process overview can then be used to examine the application for possible programming errors or missing indexes.

Customer

... is reporting performance issues he thinks are database related

Support

... analyses the situation

- configuration? (caches, MAXCPU...)
- collisions? (SQL/BD locks, regions ...)
- strategies? (used strategies, bad indices, current statistics.,...)
- I/O system? (log / data accesses?)
- ...

... gathers data from system tables / x_cons

- tedious work
- time consuming

Gathering relevant performance data with one tool

- Replaces `x_wizard` / `x_wiztrc`
- Flexible and upgradable through new rule sets
- Release and instance independent
- Remote access possible

CSN **note 530394** describes bottleneck analysis with the Database Analyzer.

The DBAnalyzer is available as of version 7.3.00 Build 25.

The tools **`x_wizard`** and **`x_wiztrc`** were discontinued as of release 7.4.

Enhanceability

The tools **`x_wizard`** and **`x_wiztrc`** were not configurable. The Executables had to be recreated for each platform every time an enhancement was carried out.

The logic and rules for monitoring with the **Database Analyzer** are defined by way of a **configuration file** (ASCII text). In case of changes or enhancements, you only have to change the configuration file in the directory INSTROOT/env.

Release independence

As accesses to the system tables are defined in the **configuration file**, adjustments for new releases only require adjusting the configuration file. Consequently, this is release-independent, but the **Database Analyzer** itself is not. The configuration file takes account of the instance type (OTLP/LVC).

Remote capability

In addition to system tables, the **`x_wizard`** used output from “`x_cons`” that could only be generated on the database server.

The **Database Analyzer** uses only system tables. The data generated by “`x_cons`” can be queried via the `SYSMON_...`, system tables, which means they can be called remotely (e.g. via OSS).

Reporting weak spots in database configuration per given time intervals

Automatically classifies messages by color indicator (info, light to severe performance problem)

Collecting monitor data each time interval

Database Analyzer

For routine monitoring of database operation in the production system, an interval of 15 minutes (-t900) is adequate. For short-term monitoring of database operation, a measuring interval of 10-30 seconds is recommended.

If class W3 warnings occur frequently, you should certainly try to remove the bottleneck. W3 warnings generally indicate that the runtime behavior of the database is severely compromised. If you suspect poor search strategies (rows read/rows qualified), a more precise analysis is unavoidable. The command monitor (DIAGNOSE MONITOR) is available for this purpose.

Not all *Database Analyzer* outputs are necessarily caused by actual bottlenecks. For example table scans can be useful in certain situations, long runtimes of statements can automatically occur with large datasets etc.

Executable dbanalyzer

- collects, assesses and stores data
- has (almost) no hard coded knowledge about system tables
- only rule based infrastructure

Configuration file dbanalyzer.cfg

- <IndepDataPath>/env/dbanalyzer76.cfg

All changes concerning rules and assessments can be made in the configuration file without need to touch the software executable.

Configuration File: dbanalyzer.cfg

- Describes ("where and why") the data to be collected or calculated (**parameters**). These **parameter** are either taken from the database (system tables) or calculated from the **parameter** taken from the database. As the manual evaluation of parameters is time-consuming, the Database Analyzer formats the logged data.
- Describes the evaluation rules (**monitors**) for the parameters. The **monitors** have up to four conditions (**Information** and **Warnings 1** through **3**) and are logged in a way that takes account of the conditions. For logging the monitors, in the configuration file you can store a verbal assessment or even concrete instructions for the user.

e.g. monitor for "DC_Hit" of /sapdb/programs/env/dbanalyzer77.cfg:

```
[Monitor]
ID          = DC_HIT
Label       = "Data cache hitrate (SQL Pages) " + DC_Hit + "%, " + DC_Fails + " of
             "+ DC_Acc + " accesses failed"
Class       = Caches
Description = For a running database application the data cache hitrate \
             should not be less than 99%, otherwise too much data has \
             to be read physically. Data cache hitrates less than 99% \
             for intervals of 15 minutes or more must be avoided.
Warning3    = DC_Hit < 96 \
             && ( PReads ) > MAX_IDLE_IO_ALL_DEVS
Warning2    = DC_Hit < 98 \
             && ( PReads ) > MAX_IDLE_IO_ALL_DEVS
Warning1    = DC_Hit < 99 \
             && ( PReads ) > MAX_IDLE_IO_ALL_DEVS
Information = DC_Hit < 99 \
             && ( PReads ) < MAX_IDLE_IO_ALL_DEVS
UserAction  = In addition to enlarging the data cache (note the paging risk of the
             operating system), search for the cause of the high read activity.
             Frequently, individual SQL statements cause...
```

Up to four conditions for triggering the monitor. Conditions are boolean expressions that refer to *parameters*.

The top-level message is stored in the *label*. The *label* is an expression that is calculated when the *monitor* is activated. This enables references to the *parameters*.

User-selected texts for *Description* and *UserAction*.



General warnings on

- low cache hitrates (data-/catalog-cache)
- high I/O rate
- low hitrates on Selects, Updates und Deletes (ratio found/read rows; optimizer strategy)
- log queue filling level too high / overflows
- lock list escalations



Task specific warnings on

- poor I/O-times
- high lock waits (vwait/vsuspend)
- long command runtimes (receive/reply)
- high read activity (reads)
- a Usertask blockades in a certain state (e.g. Vwait, Vbegexcl...)



calling the Database Analyzer

- from a UNIX- or DOS-Shell
 - start: dbanalyzer
 - -n <server>
 - -d <database>
 - -u <user,pwd>
 - -f <configfile>
 - -t <interval>,<number>
 - -o <outputdir> -c <level>
 - stop: dbanalyzer
 - -n <server> -d <database> -u <user,pwd> -f <configfile> -o <outputdir> -stop
- with the DBMCLI command dban_start
- per WebAS
 - manually via DB50 -> problem analysis -> DB bottlenecks
 - implicit start with SAP WebAS 6.20 Basis SP 37
- using the SAP CSS Support connection (SAP DB Connection → SAPDBCON)
 - Enables SAP support to collect and store data on a host of their choice

```
Command Prompt - dbanalyzer
D:\>dbanalyzer
MaxDB Database Analyzer, The Performance Analysis Tool, Version 7.5
Copyright 2000-2004 by SAP AG

Enter database name: tz75
Enter user name: dba
Enter password:
Used protocol directory: d:\sdb\data\wrk\TZ75\analyzer
Used configuration file: d:\sdb\programs\env\dbanalyzer75.cfg
```

You can also call the Database Analyzers with the DBMCLI command dban_start . The Database Analyzer is then implicitly started in the background. The Database Analyzer call can be supplemented with various options.

- n <server>
Name of the computer on which the database instance is running. If you enter this argument, you have to specify a directory for logging with the -o switch.
- d <database>
Name of the database instance that is to be examined.
- u <user,pwd>
User name and password for authorization on the database server.
- f <configfile>
Indicates the name of the configuration file to be used. The standard setting specifies the file **dbanalyzer.cfg** in the directory **\$INSTROOT/env**.
- t <interval>,<number>
Defines the time interval (in seconds) between two evaluations. If <number> is specified, the Database Analyzer ends automatically when it has reached the specified number.
- o <outputdir>
Specifies the directory in which the log files of the Database Analyzer are written. If you specify -n <server> at the time of the call, you also have to specify a log directory. If you fail to specify a log directory, logging is done in the **RUNDIRECTORY** of the database instance in the subdirectory **analyzer**.
- c <outputlevel>
Specifies that Database Analyzer output also be written to the console. In the standard setting, no output is written to the console. With <outputlevel> you can specify how much is to be output. The possible values are **1, 2, 3** and **4**.
- i
Deletes (initializes) any pre-existing log files. This enables the logging of data from different databases in the same directory, which is otherwise prohibited. The data of the previously analyzed database are deleted in the process.



short term analysis: -t 10

- time interval 10 seconds
- evaluating data online

long term analysis: -t 900 (default)

- time interval 15 Minuten
- If necessary start with “nohup Database Analyzer... &” and option -s in background (nur UNIX)
- All time data saved (ca. 1MByte/day)

in both cases

- creating and saving the protocol files

For routine monitoring of database operation in the production system, an interval of 15 minutes (-t900) is adequate. Logging should be activated with -p to obtain a retrospective overview of DB activities. For short-term monitoring of database operation, a measuring interval of 10-30 seconds is recommended.

Database Analyzer: Bottleneck Analysis (1)



Analysis Day / Measure	SQL Statements	Message
07:51:55 #654	1 4 257.585	SQL commands executed: 257585, avg. 283.96 per second Server task 261 is waiting in state No-Work (255) for page 14888, locked from task 547 Deletes selectivity 0.56%: 1463 deletes, 2555986 rows read, 14246 rows qualified 42355 table scans, selectivity 1.01%: 10492471 rows read, 106104 rows qualified 365 isolated index scans, selectivity 0%: 7539681 rows read, 16510 rows qualified LOGQUE1: collision rate 7.5%, 11172 collisions, 109 waits (0.07%), 148907 accesses on re Garbage collector tasks activated 530 times, currently active: 0 server tasks activity but not for backup. Dispatches: 3758 JobTicketQueuePrio_01: collision rate 0.61%, 153 collisions, 2335097 spin loops, 1067...
08:07:03 #655	1 9 471.040	SQL commands executed: 471040, avg. 519.61 per second Server task 261 is waiting in state No-Work (255) for page 14888, locked from task 547 Avg read time 38.45 ms for 1 reads on volume /sapdb/Q2F/sapdata1/DISKD0001 Avg read time 22.99 ms for 1 reads on volume /sapdb/Q2F/sapdata1/DISKD0005 Avg read time 29.23 ms for 1 reads on volume /sapdb/Q2F/sapdata1/DISKD0007 Avg read time 26.47 ms for 1 reads on volume /sapdb/Q2F/sapdata1/DISKD0016 Avg write time 22.86 ms for 7 writes on volume /sapdb/Q2F/sapdata1/DISKD0019 42763 table scans, selectivity 2.08%: 8441854 rows read, 175402 rows qualified 352 isolated index scans, selectivity 0%: 4369348 rows read, 12396 rows qualified TRANS1: collision rate 5.03%, 6902 collisions, 11 waits (0.01%), 137163 accesses on re

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 29

As of support packages 6.20 SP37, the Database Analyzer starts automatically when the SAP WebAS system is started.

You can call the Database Analyzer from transaction DB50 -> Problem Analysis-> Bottlenecks. You can also stop and restart it from there.

The default time interval for determining measurement data is 15 minutes. You can override this configuration stopping and restarting the Database Analyzer.

Each time the Database Analyzer is started, information about the configuration and performance-relevant data from system tables is output, including, for example, the number of tables that require an Update Statistics. You can determine the table names with a Select on the system table *sysupdstatwanted*.

Detected bottlenecks are output in text form to rapidly provide database administrators with an overview of the possible causes of performance problems. The analysis can be performed just once or at regular intervals.

Database Analyzer: Bottleneck Analysis (2)



View of aggregates

Analysis Day	Monitoring Class	File Name	Size	Time
21.07.2008	ALERTS	DBAN_prt	262.744	08:37:23
	BACKUP	DBAN_BACKUP.csv	2.043	08:37:23
	CACHES	DBAN_CACHES.csv	4.160	08:37:23
	CACHE_OCCUPANCY	DBAN_CACHE_OCCUPANCY.csv	1.965	08:37:23
	CPU_UTILIZATION	DBAN_CPU_UTILIZATION.csv	3.692	08:37:23
	FILLING	DBAN_FILLING.csv	2.947	08:37:23
	GC	DBAN_GC.csv	2.443	08:37:23
	IO	DBAN_IO.csv	4.912	08:37:23
	IOTHREADS	DBAN_IOTHREADS.csv	2.674	08:37:23
	LOAD	DBAN_LOAD.csv	4.366	08:37:23
	LOGGING	DBAN_LOGGING.csv	3.121	08:37:23
	OVERVIEW	DBAN_OVERVIEW.csv	3.172	08:37:23
	REGIONS	DBAN_REGIONS.csv	2.401	08:37:23
	RW_LOCKS	DBAN_RW_LOCKS.csv	2.452	08:37:23
	SHARED_SQL	DBAN_SHARED_SQL.csv	3.960	08:37:23
	SPINLOCKS	DBAN_SPINLOCKS.csv	2.239	08:37:23
	STRATEGY_INDEX	DBAN_STRATEGY_INDEX.csv	4.763	08:37:23
	STRATEGY_PRIMKEY	DBAN_STRATEGY_PRIMKEY.csv	2.901	08:37:23
	STRATEGY_SCANS	DBAN_STRATEGY_SCANS.csv	2.918	08:37:23
	TASK_ACTIVITIES	DBAN_TASK_ACTIVITIES.csv	4.467	08:37:23
	TASK_JO	DBAN_TASK_JO.csv	3.445	08:37:23
	TASK_STATES	DBAN_TASK_STATES.csv	2.746	08:37:23
	TRANSACTIONS	DBAN_TRANSACTIONS.csv	3.203	08:37:23

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 30

Under *Expert Analysis* you can view into the logs of a particular day.

Logs are implicitly deleted periodically via the program *RSDBANCONTROL*. You can configure how long logs are kept using transaction DB59 in the integration data for the respective system. (6.20 as of basis SP 37).

Database Analyzer: View of aggregates



Load History

Aggregated Data From 03.04.2008 To 17.07.2008

Aggregation Level / Monitor Classes / Period

- ▷ Daily Aggregates
- ▷ Weekly Aggregates
 - ▷ BACKUP
 - ▷ CACHES
 - ▷ CACHE_OCCUPANCY
 - ▷ CPU_UTILIZATION
 - ▷ 31.03.2008 - 13.07.2008
 - ▷ FILLING
 - ▷ GC
 - ▷ IO
 - ▷ IOTHEADS
 - ▷ LOAD
 - ▷ LOGGING
 - ▷ OVERVIEW
 - ▷ REGIONS
 - ▷ RW_LOCKS
 - ▷ SHARED_SQL
 - ▷ SPINLOCKS
 - ▷ STRATEGY_INDEX
 - ▷ STRATEGY_PRIMKEY
 - ▷ STRATEGY_SCANS
 - ▷ TASK_ACTIVITIES
 - ▷ TASK_IO
 - ▷ TASK_STATES
 - ▷ TRANSACTIONS
- ▷ Monthly Aggregates
 - ▷ 3 Monthly Aggregates

Q2F (1) 001 | Idai1q2f | INS

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 31

You can build aggregates on a daily, weekly, monthly or quarterly basis for the journalized data. Data can be prepared furthermore by the list viewer building sums, min, max and average values, can be loaded to the local desktop or graphically displayed.

Database Analyzer: Log Files (1)



The screenshot shows a Windows Explorer window titled "D:\sdb\data\wrk\TZ75\analyzer\20050110". The left pane shows a folder tree with "data" expanded to "wrk" and then "20050110". The right pane lists 16 CSV files, including DBAN.prt, DBAN_BACKUP.csv, DBAN_CACHES.csv, DBAN_FILLING.csv, DBAN_IO.csv, DBAN_LOAD.csv, DBAN_LOGGING.csv, DBAN_OVERVIEW.csv, DBAN_REGIONS.csv, DBAN_SPINLOCKS.csv, DBAN_STRATEGY_INDEX.csv, DBAN_STRATEGY_PRIMKEY.csv, DBAN_STRATEGY_SCANS.csv, DBAN_TASK_ACTIVITIES.csv, DBAN_TASK_IO.csv, DBAN_TASK_STATES.csv, and DBAN_TRANSACTIONS.csv.

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

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 32

Storing performance data in the logs is useful when checking runtime behavior later.

The collected data is stored as "csv" files in the directory/YYYYMMDD specified with "-o".

If you start the Database Analyzer on the DB server, you can omit the "-o" entry.

In that case, logging is done in the run directory/YYYYMMDD

A directory contains the data from one day.

The data is grouped by contents and stored in different files. You can display the day in a table with MS Excel and from the WebAS.



DBAN.prt

- quick overview; records monitor data including all rule based values

DBAN_BACKUP.csc

- physical reads/writes for backup, read/write time (ms) for backup

DBAN_CACHES.csv

- accesses, successful, failed and hit rates of all caches (DATA, CATALOG,...)

DBAN_FILLING.csv

- database filling level (size, permanently/temporarily occupied...)

DBAN_IO.csv

- virtual/physical reads/writes (common, permanent, temporary, long)

DBAN_LOAD.csv

- accesses / selektivty of selects and fetches, inserts, updates, deletes



DBAN_LOGGING.csv

- number of actual log writes, log queue overflows, max log queue used

DBAN_OVERVIEW.csv

- summarizing the other protocols key points

DBAN_REGIONS.csv

- Region accesses, collisions, waits and dispatches

DBAN_SPINLOCKS

- spinlock collisions, read/write locks

DBAN_STRATEGY_INDEX.csv

- accesses / selectivity of index, index ranges and isolated index / index ranges

DBAN_STRATEGY_PRIMKEY.csv

- accesses / selectivity of primary key and primary key ranges



DBAN_STRATEGY_SCANS.csv

- accesses / selectivity of table and isolated index scans

DBAN_TASK_ACTIVITIES.csv

- SQL commands, task statistics (active, running, runnable...)

DBAN_TASK_IO.csv

- I/O number / duration for logwriter, user und datawriter Tasks

DBAN_TASK_STATES.csv

- number and elapsed time of processed commands
- number and used time in task states Vsuspend, Vwait, Vsleep

DBAN_TRANSACTIONS.csv

- number commands, prepares, executes, commits, rollbacks, subtrans, lock request timeouts and lock request escalations



Maintain Database Integration

Database Connection Information

Name of Database Connection	Q2F
Database name	Q2F
Database Server	lddbq2f
Description	

User Data Automatic Monitoring

CCMS Monitor Sets (RZ20)

- Activate Alert Monitor
- Report Critical Alerts When the Database Assistant Is Called

Bottleneck Analysis (Database Analyzer)

- Activate when starting the SAP system

Collect. Interval: 900 Seconds (Min: 60, Max: 3600)

DBA Logs

Automatic Deletion of the DBA Logs

- On the Database Host and in the DB After 53 Weeks (>= 4)

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 36

Via transaction DB59 -> *Integration Data*-> *Automatic Monitoring*, you can define the time interval at which Database Analyzer logs are deleted.

By default, the logs are stored for 93 days.

The corresponding information in the database table SDBCCMS, however, is kept for 15 weeks. For more information, see **note 530394**.

You can make your own personal settings by choosing Display/Change.

low data cache hitrate:

<percentage> % <number> accesses, <number> successful, <count> unsuccessful

Cause:

- data cache too small
- SQL statements creating a lot of page reads (unselective commands, missing indices)

Action:

- Finding cause, e.g. with the Diagnose Monitor and pay attention to further Database Analyzer messages
- If nothing indicates an application or design problem: increase cache size to reduce risk of I/O sequentialization

Database Analyzer: Data Cache

Low data cache hit rate : <percentage> %

<number of> accesses, <number> successful, <number> unsuccessful

Explanations

The hit rate is too low when accessing the database cache. The data cache hit rate for a running database application should not be less than 98%; otherwise, too much data has to be read physically. For a short time, lower hit rates may occur; e.g., when reading tables for the first time, or when the table does not fit into 10% of the data cache with repeated table scans (only with

`UseDataCacheScanOptimization/LRU_FOR_SCAN = NO`). Data cache hit rates under 98% for intervals of 15 minutes or more must be avoided.

User response

In addition to enlarging the data cache (note the paging risk in the operating system), search for the cause of the high read activity. Frequently, individual SQL statements cause a high percentage of the total logical and physical read activities. Enlarging the cache only transfers the load from the disk to the CPU although an additional index, for example, could transform a read-intensive table scan into a cheap direct access.



User task physical writes <number of phys. writes>

Causes:

- write transactions changing data pages in the cache
- data cache full, no more space for new pages
- before reading a new page, an already modified page has to be displaced

Action:

- increase cache size
- activate pager

Database Analyzer: cache displacements

Cache displacements: <number of> pages/second

Explanations

Modified pages are displaced from the data cache to disk because the data used by the applications cannot be completely kept in the data cache. If the size of the data cache were sufficient, the physical write would be delayed until the next SAVEPOINT and then be done asynchronously. Cache displacements result in synchronous I/O and should be avoided, if possible.

User response

Enlargement of the data cache. Particularly with larger data imports, the so-called *paggers* should be activated for regular asynchronous buffer flushes between the SAVEPOINTS database parameter DataCacheIOAreaSize, DataCacheIOAreaFlushThreshold, DataCacheLRUAreaFlushThreshold or in earlier versions _DW_IO_AREA_SIZE, _DW_IO_AREA_FLUSH, _DW_LRU_TAIL_FLUSH).



low access hitrates via <Optimizer Strategy>:

<percentage> % <number> accesses, <number> rows read, <number> rows qualified

Causes:

- disadvantageous execution of SQL commands. Too many reads necessary to fetch just a few results
- unfavourable SQL syntax/statement
- missing indices

Action:

- update statistics
- Find the responsible SQL commands with the help of Diagnose Monitor, analyse them and - if necessary - rewrite SQL or create index

Database Analyzer: selectivity

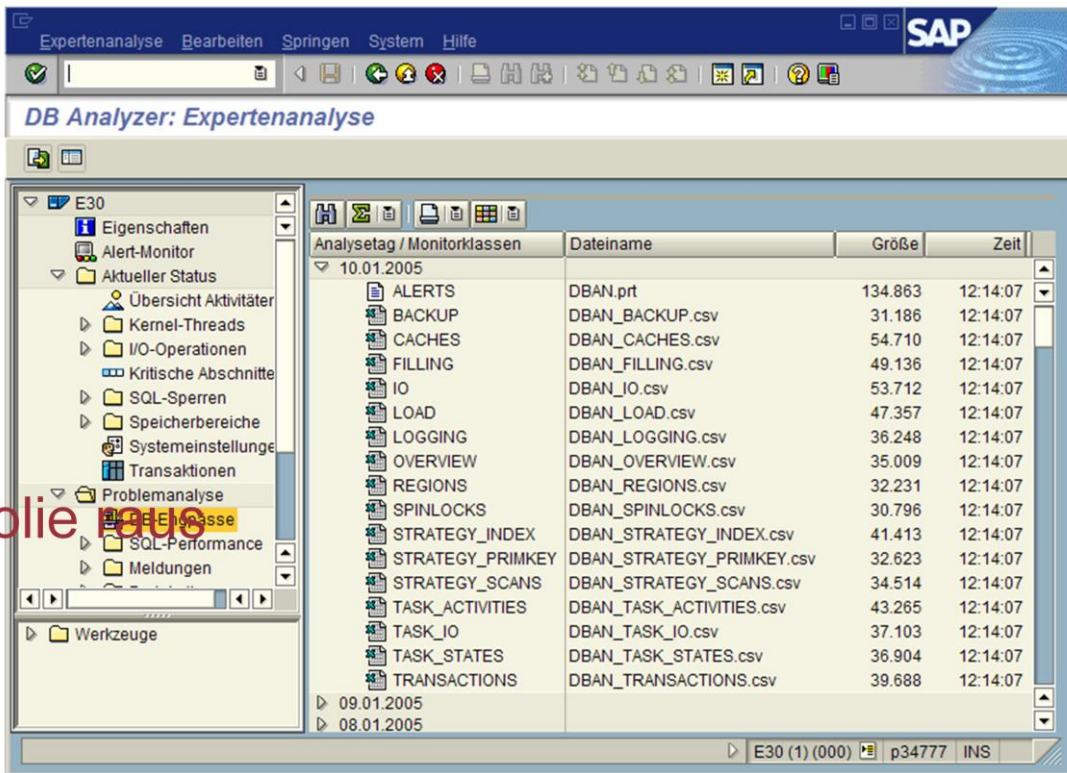
Explanations

The relationship between read and found (qualified) rows is poor for a certain access strategy applied by the MaxDB Optimizer. This indicates a poor search strategy, caused either by the application (missing or insufficient indexes) or by poor formulation of SQL statements. Searching large quantities of data can seriously compromise the performance of the system as a whole due to the numerous negative effects (I/O, CPU load, etc.).

User response

First of all, see if MaxDB Optimizer is able to find a more suitable strategy after updating the internal database statistics. The update should be done directly from the SAP system with transaction DB13.

If this does not produce the desired result, search for the statement that triggers the unfavorable search strategy. The easiest way to do this is with DIAGNOSE MONITOR.



For detailed information on the monitor classes, choose *Expert Analysis*.

You get an overview of all available logs generated by the Database Analyzer. The significance of the individual logs is described on the following slides

To display the content of a file, double-click the relevant monitor class.



Automatic tracking of problematic commands

criteria:

- page accesses
 - registers SQL commands exceeding a set limit for page accesses
- runtime
 - records SQL commands exceeding a supplied runtime (in seconds)
- selectivity
 - Tracks SQL commands falling below a given ratio of qualified and read records

THE tool for identifying long running SQL commands

Logging can be switched on directly in SAP WebAS using transaction DB50 or with the following dbmcli command:

```
dbmcli -n <dbserver> -d <dbname> -u ... -uSQL sapr3,sap sql_execute
```

```
diagnose monitor selectivity <number>
```

```
| read <number>  
| time <number>  
| rowno <number>  
| off
```

Selectivity:	Ratio of qualified to total records read is below the specified threshold value
Read (page accesses):	Specified number of virtual reads exceeded
Time (runtime):	Runtime of the command exceeds the specified time in milliseconds
Rowno:	Number of statements to be stored as specified in SYSMONITOR (default 255)
Off:	Deactivation



Commands and measured data are stored in tables SYSPARSEID, SYSMONITOR and SYSMONDATA:

SYSPARSEID

- stores Parse-ID and SQL command string

SYSMONITOR

- Periodically overwritten, max. 3000 commands
- contains Parse-ID and performance data

SYSMONDATA

- stores command data

Performance tables

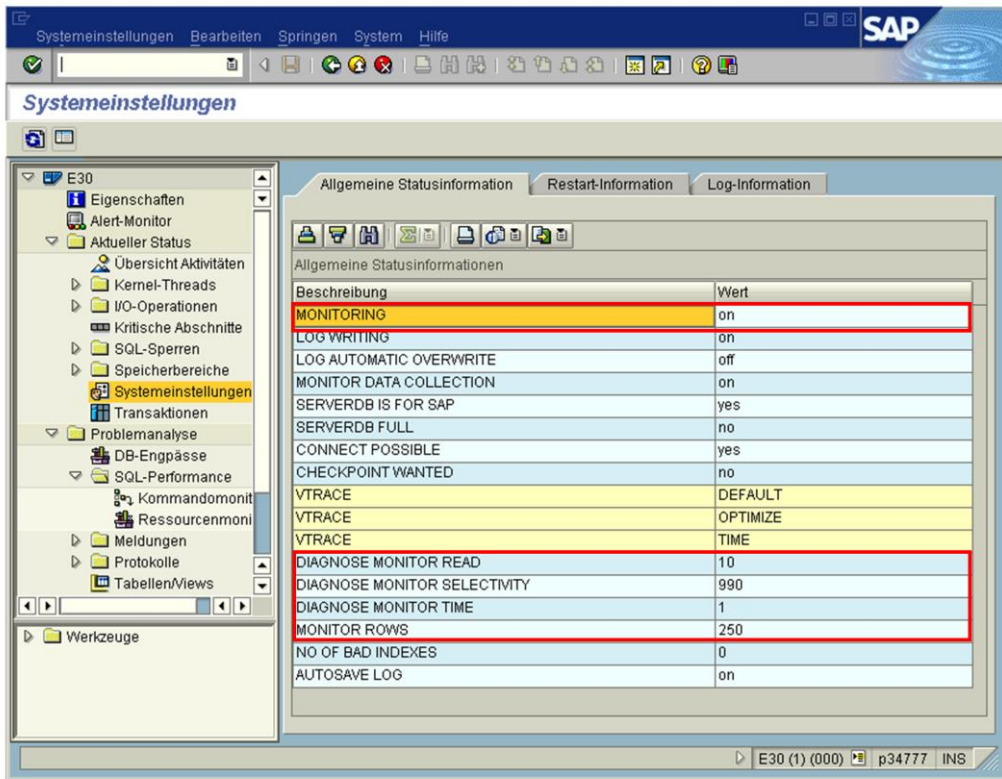
The tables SYSMONITOR and SYSPARSEID are filled following activation of the DIAGNOSE MONITOR. SYSPARSEID can grow to any size, SYSMONITOR is overwritten cyclically.

The table SYSPARSEID contains the parse ID PARSEID for every parsed statement and the command string SQL_STATEMENT.

COLUMN NAME		MOD	DATA TYPE	CODE	LEN
PARSEID		KEY	CHAR	BYTE	12
LINKAGE		KEY	FIXED		1
SELECT_PARSEID	OPT	CHAR	BYTE		12
OWNER		OPT	CHAR	ASCII	18
SQL_STATEMENT	OPT	CHAR	ASCII		3900

Tables are joined for analysis

```
SELECT rows_read, rows_qual, strategy, runtime,  
physical_io, sql_statement, substr(data,1,25)  
FROM sysmonitor, sysparseid, sysmondata  
WHERE sysmonitor.parseid = sysparseid.parseid  
AND sysmonitor.sysk = sysmondata.sysk  
ORDER BY runtime DESC
```

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 44

When you do a performance analysis, check the system settings to ensure that the performance analysis tools are working.

Monitoring: If monitoring is active, general performance-relevant data is stored in the system tables. Monitoring is automatically activated by the start scripts when you start the SAP WebAS and the database.

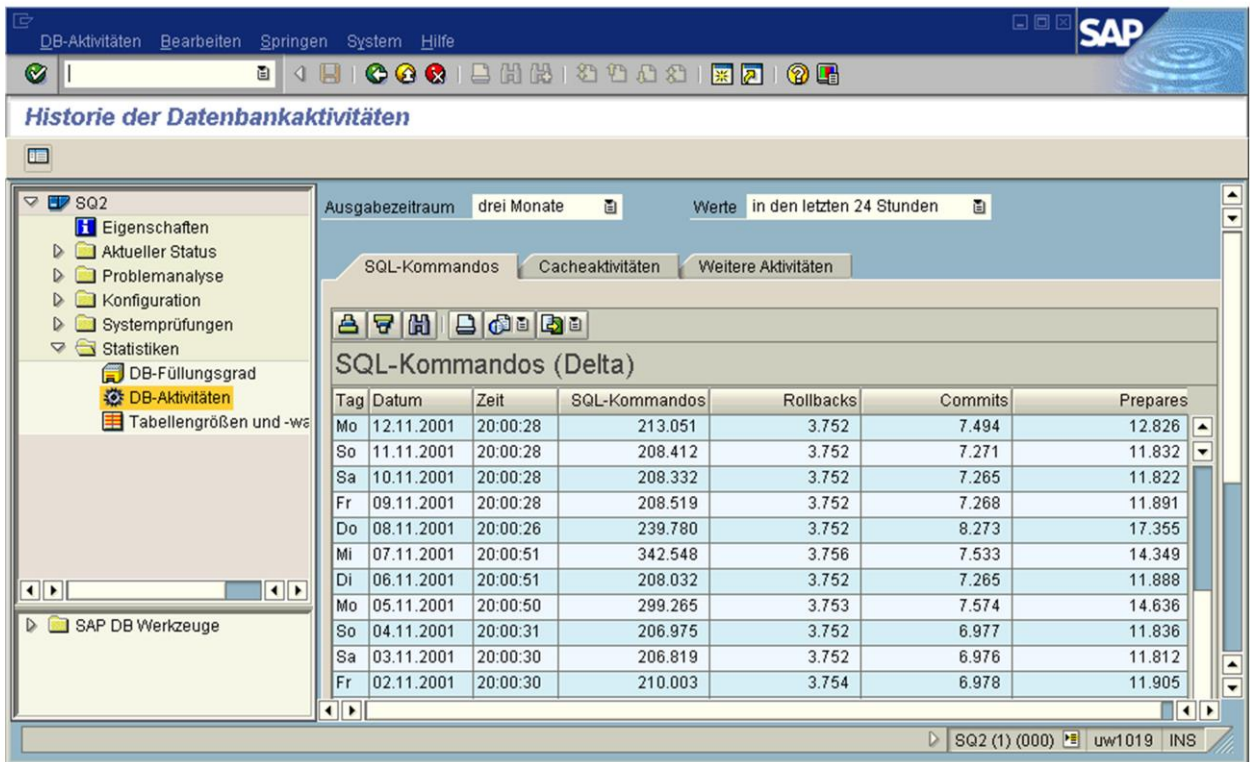
Monitor Data Collection: activates logging of transfer values in the Where condition of each SQL statement in the command monitor.

Diagnose Monitor Read / Selectivity: display the settings with which the command monitor (DIAGNOSE MONITOR) was started. If the command monitor is not active, the corresponding entries in the system settings are missing.

Diagnose Analyze and Diagnose Analyze Filter: Resource monitor (see below)

Monitor Rows: Number of SQL statements

No of bad Indexes: indicates if there are defective indexes in the database. If an index is defective, it cannot be used for access.



The screenshot shows the SAP DB50 interface. The title bar reads 'Historie der Datenbankaktivitäten'. The left sidebar contains a tree view with 'SQ2' expanded, showing 'Eigenschaften', 'Aktueller Status', 'Problemanalyse', 'Konfiguration', 'Systemprüfungen', 'Statistiken', 'DB-Füllungsgrad', 'DB-Aktivitäten', and 'Tabellengrößen und -w'. The main area has a toolbar and a table titled 'SQL-Kommandos (Delta)'. The table columns are 'Tag', 'Datum', 'Zeit', 'SQL-Kommandos', 'Rollbacks', 'Commits', and 'Prepares'. The data shows a peak in SQL commands on Wednesday, 07.11.2001.

Tag	Datum	Zeit	SQL-Kommandos	Rollbacks	Commits	Prepares
Mo	12.11.2001	20:00:28	213.051	3.752	7.494	12.826
So	11.11.2001	20:00:28	208.412	3.752	7.271	11.832
Sa	10.11.2001	20:00:28	208.332	3.752	7.265	11.822
Fr	09.11.2001	20:00:28	208.519	3.752	7.268	11.891
Do	08.11.2001	20:00:26	239.780	3.752	8.273	17.355
Mi	07.11.2001	20:00:51	342.548	3.756	7.533	14.349
Di	06.11.2001	20:00:51	208.032	3.752	7.265	11.888
Mo	05.11.2001	20:00:50	299.265	3.753	7.574	14.636
So	04.11.2001	20:00:31	206.975	3.752	6.977	11.836
Sa	03.11.2001	20:00:30	206.819	3.752	6.976	11.812
Fr	02.11.2001	20:00:30	210.003	3.754	6.978	11.905

You start the detailed performance analysis by checking the statistics to see if there is anything unusual in the functioning of the system today.

Display the database activities for a certain point in time. Let's have a look at Wednesday, 7 November, 2001.

The overview of database activities is logged daily by a collector (COLLECTOR_FOR_PERFORMANCEMONITOR).

SQL Commands: Total number of SQL statements, commits, rollbacks, prepares

We see that on Wednesday, 7 November, a somewhat higher number of SQL statements was executed than on other days.

DB-Aktivitäten Bearbeiten Springen System Hilfe

Historie der Datenbankaktivitäten

Ausgabezeitraum drei Monate Werte in den letzten 24 Stunden

SQL-Kommandos Cacheaktivitäten Weitere Aktivitäten

Cacheaktivitäten (Delta)

Tag	Datum	Zeit	Data	Conv.	Catlg.	Phys. Lesezugriffe	Phys. Schreibzugriffe
Mi	14.11.2001	20:00:28	100	100	85	13.744	2.932
Di	13.11.2001	20:00:28	100	100	85	15.147	2.435
Mo	12.11.2001	20:00:28	100	100	85	85.221	96.989
So	11.11.2001	20:00:28	100	100	85	298	2.601
Sa	10.11.2001	20:00:28	100	100	85	245	2.210
Fr	09.11.2001	20:00:28	100	100	85	12.028	2.745
Do	08.11.2001	20:00:26	100	100	85	26.337	2.445
Mi	07.11.2001	20:00:51	100	100	85	152.207	162.530
Di	06.11.2001	20:00:51	100	100	85	11.831	2.557
Mo	05.11.2001	20:00:50	100	100	86	20.211	3.893
So	04.11.2001	20:00:31	100	100	86	33	2.153

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 46

Cache activities: Cache hit rates, number of reads and writes to the database

The cache statistics for 7 November show that the cache hit rates were good, but the number of physical read and write accesses was significantly higher than on the other days.

DB-Aktivitäten Bearbeiten Springen System Hilfe

Historie der Datenbankaktivitäten

Ausgabezeitraum drei Monate Werte in den letzten 24 Stunden

SQL-Kommandos Cacheaktivitäten Weitere Aktivitäten

Weitere Aktivitäten (seit Restart)

Tag	Datum	Zeit	Tabellenscans	Geschr. Logpages	Lockkollisionen
Mo	12.11.2001	20:00:28	4.969	11.566	33
So	11.11.2001	20:00:28	4.578	11.479	39
Sa	10.11.2001	20:00:28	4.568	11.476	29
Fr	09.11.2001	20:00:28	4.579	11.499	13
Do	08.11.2001	20:00:26	4.929	11.620	13
Mi	07.11.2001	20:00:51	7.914	11.683	25
Di	06.11.2001	20:00:51	4.583	11.509	23
Mo	05.11.2001	20:00:50	5.422	11.827	28
So	04.11.2001	20:00:31	4.578	11.482	43
Sa	03.11.2001	20:00:30	4.568	11.487	41
Fr	02.11.2001	20:00:30	4.576	11.503	25

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 47

Other Activities: Number of table scans, log activities, lock information

On 7 November, the number of table scans is notable. It, too, is very high compared to the other days. This all implies an application that is "problematic" in performance terms.

DB50: Problem Analysis – Command Monitor

(1)

SAP

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 48

The command monitor (Diagnose Monitor) allows you to identify long-running SQL statements. This tool is intended for short analyses, since the number of recorded SQL statements is limited. Specify criteria to restrict the volume and type of SQL statements that are recorded.

Within SAP WebAS, logging is activated via transaction DB50 ? Problem Analysis? SQL Performance ? Command Monitor. You can set up various criteria in succession. A statement is logged when at least one of the criteria is fulfilled.

Choose *Command Monitor* → *Change Monitor Settings* to determine the recording criteria by which SQL statements are logged in the command monitor tables.

Number of page accesses: An SQL statement is logged when the number of page accesses exceeds the specified value.

SQL statement runtime: An SQL statement is logged when the runtime exceeds the specified value (in seconds).

Selectivity: An SQL statement is logged in the command monitor tables if the ratio of qualified records to records read falls below the specified percentage.

Save parameter values: Select this field if you want to log the SQL statements with their parameters.

Max. number of monitor entries: This value determines the maximum number of entries that are held in the table SYSMONITOR before the table is overwritten.

DB50: Problem Analysis – Command Monitor (2)



The screenshot shows the SAP SQL Command Monitor interface. At the top, there's a menu bar with 'Kommandomonitor', 'Bearbeiten', 'Springen', 'System', and 'Hilfe'. Below that is a toolbar with various icons. The main area is titled 'SQL-Kommandomonitor' and contains 'Aktuelle Monitoreinstellungen' with options for 'Pagezugriffe', 'Laufzeit', 'Selektivität', and 'Parameterwerte speichern'. A table below shows monitoring data for various SQL statements. A red circle highlights the 'Layout ändern' button in the toolbar. An inset window titled 'Spaltenauswahl' is open, showing a list of columns that can be configured for the selected SQL statement.

Tabellen	Programm	Laufzeit	#P Zugrif...	#R Geles...	#R Qualif.	# Disk-I/O	Strategie	gekürzte SQL-Anweisung
"ZZTELE"	ZFBAD	0,442	2.277	114.199	2	0	SCAN	SELECT * FROM "ZZTELE" WHERE "ORT" = ? AND "STR" = ?
"ZZTELE"	ZFBAD	0,436	2.277	114.199	2			"STR" = ?
"ZZTELE"	ZFBAD	0,431	2.277	114.199	2			"STR" = ?
"ZZTELE"	ZFBAD	0,430	2.277	114.199	2			"STR" = ?
"ZZTELE"	ZFBAD	0,425	2.277	114.199	2			"STR" = ?
"ZZTELE"	ZFBAD	0,335	2.277	114.199	0			
"ZZTELE"	ZFBAD	0,324	2.277	114.199	0			
"ZZTELE"	ZFBAD	0,324	2.277	114.199	0			
"ZZTELE"	ZFBAD	0,322	2.277	114.199	0			
"ZZTELE"	ZFBAD	0,322	2.277	114.199	0			
"ZZTELE"	ZFBAD	0,306	2.277	114.199	7			

Description of columns that can be configured via the "Change Layout" button:

Table	Table on which the SQL statement was used
Program	ABAP program that executed the SQL statement
Runtime	Runtime of the SQL command in seconds
#P Accesses	Accesses to data pages (in the cache and on the disk)
#R Read	Table rows that were read while processing the statement
#R Qualif.	Table rows that met the selection condition
Selectivity	Ratio #R qualif / #R read in %
#P/R	Number of page accesses per qualif. row
#Fetched (#Abgeholt)	Number of rows fetched
#Disk I/O	I/O accesses to disk (reading and writing incl. converter)
Strategy	The select strategy chosen by the Optimizer (table scan, index, key etc.)
Parseid	Parse ID
SQL wait situations	Number of collisions on SQL locks
Task suspensions	Number of collisions on internal locks
Number fetch requests	Number of fetches while processing the Select statement
Result set	YES, if internal result set was generated (e.g. with sorting), otherwise NO
Date	of execution
Time	of execution

Subrequests

DB50: Problem Analysis – Command Monitor (3)



The screenshot shows the SAP Command Monitor interface. At the top, there is a menu bar with 'SQL-Anweisung', 'Bearbeiten', 'Springen', 'System', and 'Hilfe'. Below the menu bar is a toolbar with various icons. The main area is divided into two panes. The top pane, titled 'SQL-Anweisung', contains the following SQL query:

```
SELECT
FROM
  ZZTELE
WHERE
  ORT = 'Berlin' AND STR = 'Stromstr'#
```

The bottom pane, also titled 'SQL-Anweisung', displays the execution plan for the query. It is presented as a table with the following columns: OWNER, TABLENAME, COLUMN OR INDEX, STRATEGY, and PAGECOU.

OWNER	TABLENAME	COLUMN OR INDEX	STRATEGY	PAGECOU
SAPE30	ZZTELE	ZZTELE~2 STR	RANGE CONDITION FOR INDEX (USED INDEX COLUMN) RESULT IS NOT COPIED , COSTVALUE IS	1

At the bottom of the interface, there is a status bar showing 'E30 (1) (000) p34777 INS'. A red circle highlights a magnifying glass icon in the top-left corner of the SQL-Anweisung pane, with a red arrow pointing to the 'OWNER' column of the execution plan table.

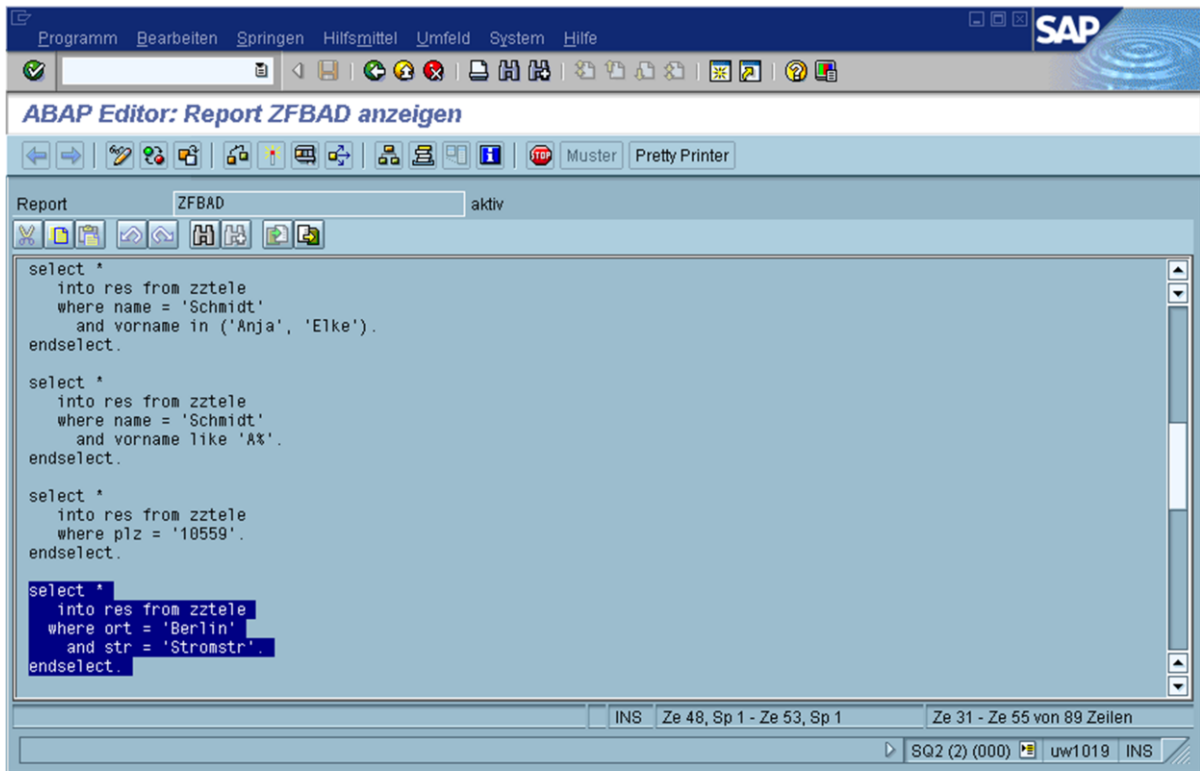
© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 50

To get the detail view of the command, select and double-click the command you want to examine.

If the *Save parameter value* criterion is active (data collection on), choose *Display Execution Plan for SQL Statement* to check which access strategy the SQL optimizer would choose to process this SQL statement.

If an error in the MaxDB Optimizer is responsible for an unsuitable strategy, development support may require a trace of the call of the Explain statement. To do this, choose *Trace Execution Plan for SQL Statement*. This generates a special Optimizer Vtrace that can be analyzed using transaction DB50 Problem Analysis-> Database Trace.

DB50: Problem Analysis – Command Monitor (4)



The screenshot shows the SAP ABAP Editor interface. The title bar reads "ABAP Editor: Report ZFBAD anzeigen". The main window displays the following SQL code:

```
select *
  into res from zztele
  where name = 'Schmidt'
        and vorname in ('Anja', 'Elke').
endselect.

select *
  into res from zztele
  where name = 'Schmidt'
        and vorname like 'A%'.
endselect.

select *
  into res from zztele
  where plz = '10559'.
endselect.

select *
  into res from zztele
  where ort = 'Berlin'
        and str = 'Stromstr'.
endselect.
```

The status bar at the bottom indicates "INS Ze 48, Sp 1 - Ze 53, Sp 1" and "Ze 31 - Ze 55 von 89 Zeilen". The bottom right corner shows "SQ2 (2) (000) uw1019 INS".

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 51

The *Calling point in the ABAP Program* can be used to determine the BAP program from which the statement was started. For this link to work with systems with WebAS versions below 6.40 dbssl patch 32, set the following parameter in the instance profile of the SAP WebAS: `dbsslada\register_appl_info = 1`.

In most cases, the processing of a statement can be accelerated by adjusting the ABAP code.

DB50: Problem Analysis – Command Monitor (5)



Ausführungsplan Bearbeiten Springen System Hilfe

Ausführungsplan der SQL-Anweisung(Explain)

Ausführungsplan des SQL-Optimierers

OWNER	TABLERNAME	COLUMN OR INDEX	STRATEGY	PAGECOUNT	O	D	T	M
SAPR3	ZZTELE		TABLE SCAN	2339				
SAPR3			RESULT IS NOT COPIED , COSTVALUE IS	2507				

SQL-Anweisung

```
SELECT
  *
FROM
  "ZZTELE"
WHERE
  "ORT" = ? AND "STR" = ?
```

Variablen

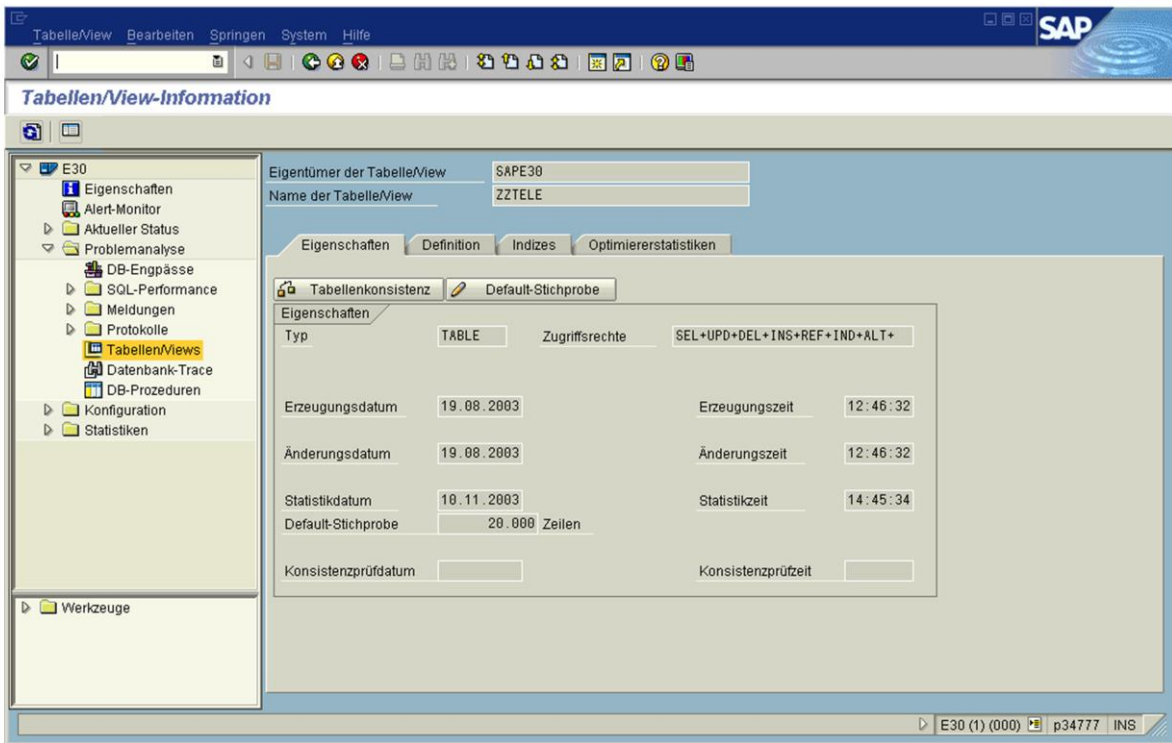
```
P1 (CH,6 ) = Berlin
P2 (CH,8 ) = Stromstr
```

SQL2 (2) (000) uw1019 INS

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 52

Using the Explain statement, we see that in our case the statement was executed with a table scan. That is, the whole table was read in order records that have *Berlin* in the *CITY* column and *Stromstr* in the *ST* column.

Here the question arises why an index wasn't used, or rather, why the SELECT was formulated in such a way.



© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 53

You can get information on a specified database view or table via DB50 or directly from the command monitor for the current table.

Attributes: Type, access rights, creation and change dates, the date of the last run for determining optimizer statistics for this table, as well as the date of the last Check Table on this table (show table).

Definition: Definition of the table in the database instance (this is not the table definition from the ABAP Dictionary but rather the table definition from the system tables of the database system)

Indices: Indices defined for this table (show index).

Optimizer statistics: Last values determined for the optimizer statistics (show optimize stat).

Table consistency: It is possible to start a CHECK TABLE directly from DB50.

Default sample: Using this function, in the system table domain.tables you change the sample value for this table when carrying out the UPDATE STATISTICS command. From then on, all following UPDATE STATISTICS are carried out using the new sample value.



Tabellen/View Bearbeiten Springen System Hilfe

Tabellen/View-Information

Eigentümer der Tabelle/View SAPE30
Name der Tabelle/View ZZTELE

Eigenschaften Definition Indizes Optimiererstatistiken

Tabellendefinition SAPE30 ZZTELE

Spaltenname	Typ	Datentyp	Codetyp	Län...	Dezi...	Zugriff...	Default	Position	Schlüssel...	Erzeugungsd...	Zeit	Änc
NAME	KEY	VARCHAR	ASCII	40		SEL+...		1	1	19.08.2003	12:46:32	19.0
VORNAME	KEY	VARCHAR	ASCII	20		SEL+...		2	2	19.08.2003	12:46:32	19.0
STR	KEY	VARCHAR	ASCII	40		SEL+...		3	3	19.08.2003	12:46:32	19.0
NR	OPT	NUMBER		10	0	SEL+...	0	4		19.08.2003	12:46:32	19.0
PLZ	OPT	VARCHAR	ASCII	5		SEL+...		5		19.08.2003	12:46:32	19.0
ORT	OPT	VARCHAR	ASCII	25		SEL+...		6		19.08.2003	12:46:32	19.0
CODE	OPT	VARCHAR	ASCII	31		SEL+...		7		19.08.2003	12:46:32	19.0
ADDINFO	OPT	VARCHAR	ASCII	31		SEL+...		8		19.08.2003	12:46:32	19.0

E30 (1) (000) p34777 INS

The table definition can provide information as to whether the command could have been processed using the primary key.

The key of table ZZTELE consists of the columns: *Name*, *First name* and *Street*.

The WHERE condition of the SQL statement consists of the columns *City* and *Street*.

Because neither *Name* nor *First name* are specified in the WHERE condition, the key cannot be used for optimization.



The screenshot shows the SAP DB50 Performance Analysis tool interface. The main window displays the 'Optimiererstatistiken' for table 'ZZTELE' in schema 'SAPE30'. A dialog box titled 'Statistiken aktualisieren' is overlaid on the table, allowing the user to update the statistics. The dialog shows two options: 'Zeilenanzahl' (Number of rows) set to 20.000 and 'Prozentsatz' (Percentage) set to 0%.

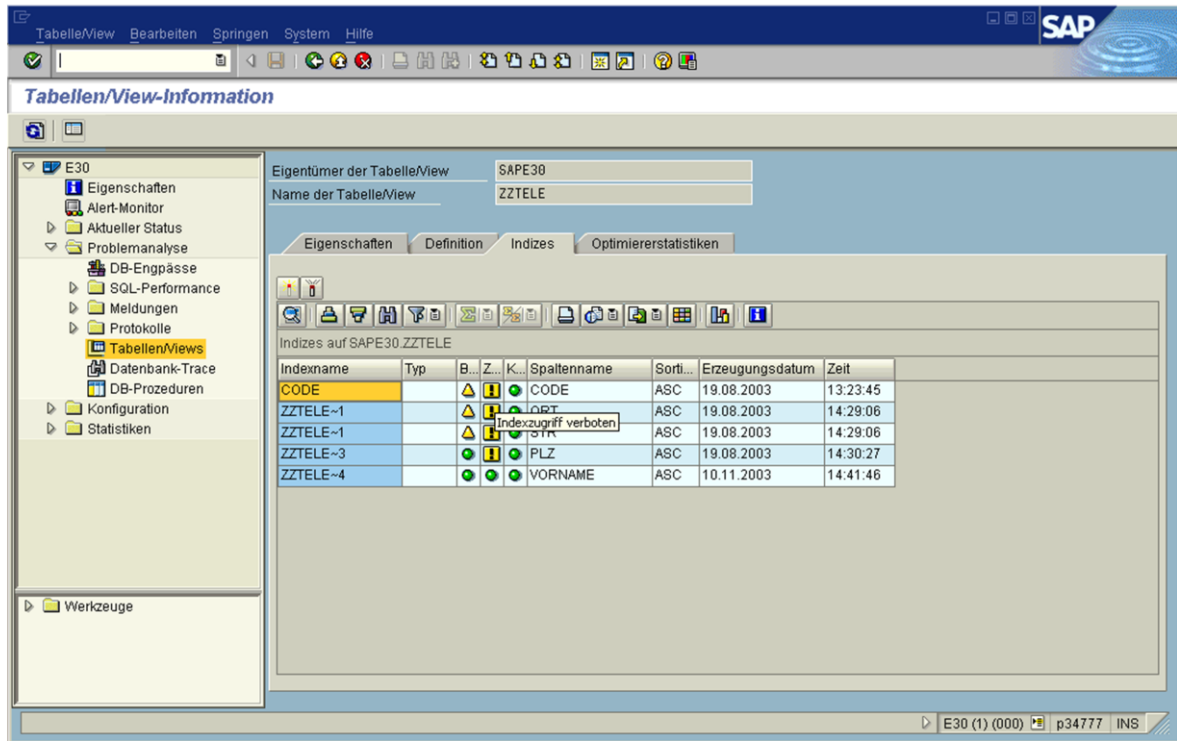
Spaltenname	Indexname	Anzahl unte...	Anzahl P
ADDINFO		1969	
CODE		2	
NAME		13363	
NR		255	
ORT		2	
PLZ		20001	
STR		8	
VORNAME		5156	
	CODE		1155
	ZZTELE~1		1165
	ZZTELE~3		1112
	ZZTELE~4		1334
TABLE STATISTICS		114199	1839

The optimizer statistics provide an overview about the selectivity of the individual columns.

The cost-based optimizer determines the best access strategy with the help of statistical information about the size of the table and values within the table columns.

A cost-benefit plan is created for the various access options.

The optimizer statistics are updated by an UPDATE STATISTICS. You have the option to specify sample values for this UPDATE STATISTICS run. There is no entry of the sample value in the system table *domain.tables*.



Using the *Indices* function you can check the following:

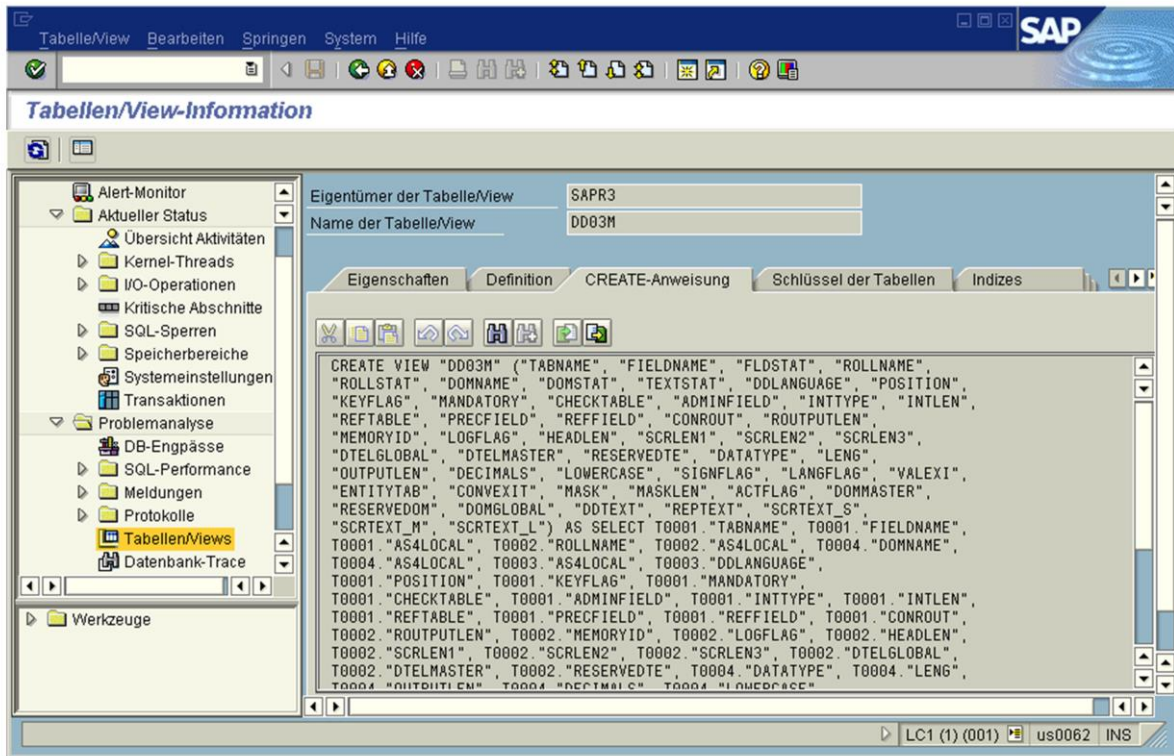
- whether indices exist for the table
- whether these indices were already used by the optimizer
- whether access to this index is allowed (enabled) or not allowed (disabled)
- whether the index is consistent or perhaps set to BAD
- via which table column the index has been created

There were 3 indices created for the table ZZTELE, and access to them is forbidden. In other words, the indices were disabled. This could have been necessary in order to test out how a command will be processed if the index did not exist. This feature is offered since this procedure can be performed more quickly than if the index is first deleted and then recreated later. Often this is not possible, especially for large table.

The command logged in the command monitor could therefore only be performed via a table scan because the index that could be used for the optimization is inactive.

The index can be activated directly from this menu. You can do this by selecting the index and choosing *Allow index access*. The column *Access* will be highlighted in green after performing this action.

After restarting the application, the analyzed command may no longer appear in the command monitor.



© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 57

As of version SAP WebAS 6.20 with Basis Support Package 39, the detail display is also expanded.

Properties: Type, access rights, creation and change date

Definition: Definition of the tables in the database instance that are involved in the view (this is not the table definition from the ABAP Dictionary but rather the table definition from the system tables of the database system)

Create Statement: displays the create statement with which the view was created.

Keys of the Tables: all key columns of all tables involved in the view.

Indexes: Indexes defined for this table

Optimizer statistics: Last values determined for the optimizer statistics (show optimize stat).

DB50: Problem Analysis – Resource Monitor (1)



Ressourcenmonitor Bearbeiten Springen System Hilfe

SQL-Ressourcenmonitor

Aktuelle Anzeigeeingrenzungen

Pagezugriffe	0
Physische I/O-Zugriffe	0
Ausführungen	0
Laufzeit in s	0,000
Muster für SQL-Kommando	
Anzahl der Anweisungen	220

Aktueller Monitorstatus

● Ressourcenverbrauch ermitteln

aufgezeichnete SQL-Anweisungen	1.125
Statistiksätze	3.234

Anzahl, P Pages, R Zeilen, E Ausführungen

Operati...	Tabell...	#Ausführung...	Laufzeit	durchschn. Laufzeit	min. Laufzeit	max. Laufzeit	#P Zugriffe	#P / E	#P / R	#R Abg...
SELECT	USER_...	6	1,072	0,179	0,167	0,203	6.078	1.013	0,05	3.300
SELECT		6	1,009	0,168	0,167	0,168	6.078	1.013	0,05	2.532
SELECT	USER_...	6	1,246	0,208	0,186	0,246	7.108	1.185	0,05	6.600
SELECT	USER_...	6	0,554	0,092	0,092	0,092	2.028	338	0,08	6.600
SELECT	ZZTELE	1	0,857	0,857	0,857	0,857	8.494	8.494	3,08	2.758

SQL2 (2) (000) uw1019 INS

© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 58

By analyzing the resource consumption, you can identify the most costly SQL statements. The resources used by an SQL statement are measured (runtime and I/O accesses, for example).

If an SQL statement is used more than once, the total cost is calculated. This enables you to recognize those SQL statements that have a relatively short runtime, but that generate a considerable database load due to the number of times they are executed.

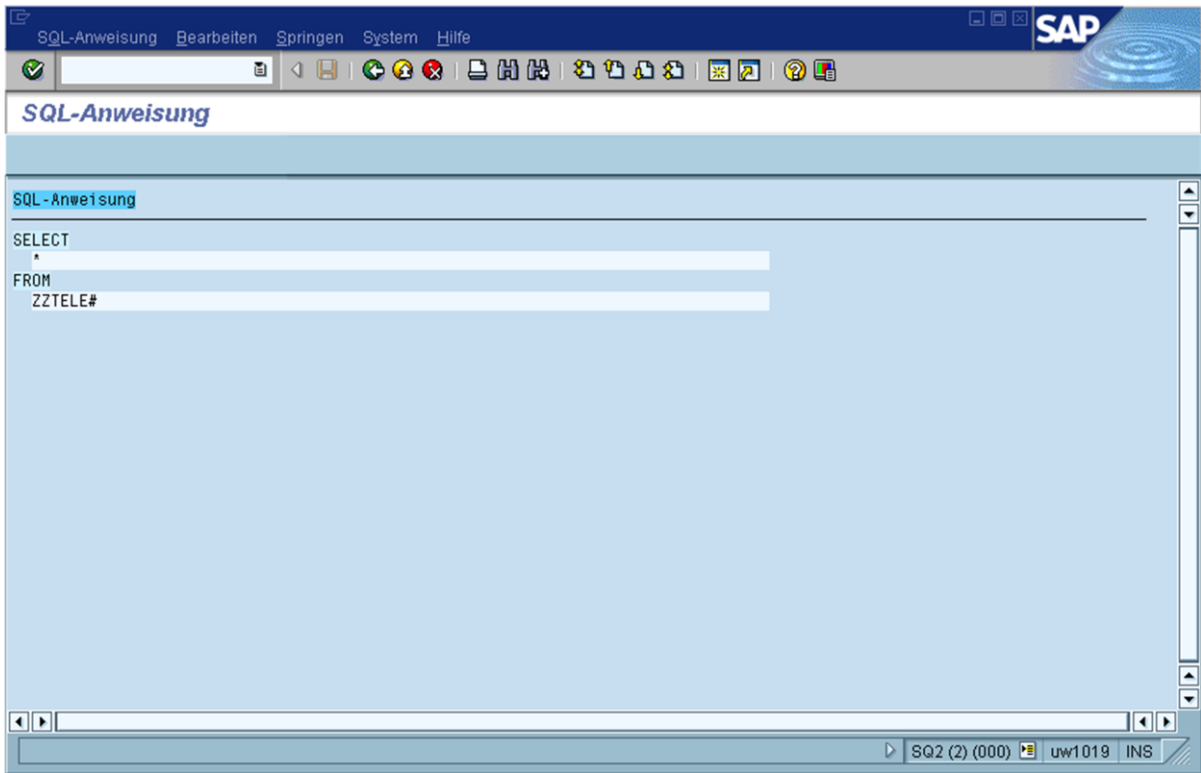
The resource monitor is therefore a monitoring tool that can be used for load analysis of one workday, for example.

You can restrict the statements to be displayed using additional definitions in the display limits.

With regard to the message in the bottleneck analysis *High Read rate physical*, it is page accesses that interest us here.

What is remarkable is that the last statement was only carried out once but displays over 8400 page accesses.

You can view the statement by double clicking on it.



© SAP 2007 / MaxDB 7.6 Internals – Performance Analysis / Page 59

Here we have a SELECT for table ZZTELE without a WHERE condition. Therefore all records of the table are read.

In such a case, the application should be examined more closely to see whether all records of the table are truly necessary for processing or whether using a WHERE condition as a limitation, the actual number of records to be processed can be decreased.



SYSCMD_ANALYZE

- shows command-ID and SQL command string

SYSDATA_ANALYZE

- entries never deleted or overwritten
- contains command-ID and measured data

every SQL command is recorded

- `select * from syscmd_analyze sc, sysdata_analyze sd where sql_statement like '%ZZTELE%' and sc.cmdid = sd.cmdid`

Performance Tables

The tables SYSCMD_ANALYZE and SYSDATA_ANALYZE are generated and subsequently filled after DIAGNOSE ANALYZE is activated.

Logging can be activated/deactivated with the following dbmcli command

```
dbmcli -n <SAP DB Hostname> -d <SID> -u control,control -uSQL sap<sid>,sap  
sql_execute diagnose analyze on | off
```

During parsing, the commands are entered in SYSCMD_ANALYZE and a unique command key is generated. Identical commands are stored only once for all concurrent sessions. Resource usage is not yet determined.

Logging of resource usage can be activated/deactivated with the following dbmcli command

```
dbmcli -n <SAP DB Hostname> -d <SID> -u control,control -uSQL sap<sid>,sap  
sql_execute diagnose analyze count on | off
```

Normal monitoring is required for this, that is, it is activated if necessary. The values are aggregated per session under the command key in the table SYSDATA_ANALYZE.

Aggregation over several sessions must be done by the application via the command key.

The generated data can be deleted with the following dbmcli command

```
dbmcli -n <SAP DB Hostname> -d <SID> -u control,control -uSQL sap<sid>,sap  
sql_execute diagnose analyze CLEAR COMMAND/DATA/ALL
```



Transaction DB13, DB20, DB50

- implemented through a work process with sampling
- SAP WebAS Alert table support

DBMCLI

- sql_updatestat
- sql_updatestat_per_systemtable

Database Manager (DBMGui)

- Instance -> Tuning -> Optimizer Statistics

As of version 7.5, MaxDB only requires statistics data for joins and selects with a restriction of the number of records in the result, such as "WHERE ROWNUM <= n".

For the table itself, Update Statistics only determines the data if the current size specifications are not in the file directory. This does not apply to tables that were created with databases of version < 7.6 and for which the size specifications in the file directory could not yet be determined.

Update Statistics determines statistics data for all columns that are primary keys or index columns. Additionally, it determines the statistics data for all columns beyond the primary key and index if statistics are already available.

If the optimizer discovers tables with unsuitable statistics data, it enters them in the table SYSUPDSTATWANTED. The DBM command sql_updatestat_per_systemtable executes an Update Statistics for all tables listed in SYSUPDSTATWANTED.

The DBM command sql_updatestat executes an Update Statistics for all tables in the database.

Update Statistics imports the data for a table from all data volumes in parallel. This makes it very speedy.

The programs "xpu" and "updcoll" are no longer available as of version 7.6..



Sample rates for Update Statistics can be configured as

Rows per table:

- UPDATE STATISTICS ... ESTIMATE SAMPLE <n> ROWS

Percentage per table

- UPDATE STATISTICS ... ESTIMATE SAMPLE <p> PERCENT

Advantage of sampling:

- shorter runtime of update statistic job

Disadvantage of sampling:

- Sample values are only estimated, if they do not resemble the actual data distribution, the optimizer might chose a suboptimal access strategy

Sampling with Update Statistics

Database statistics can be created on the basis of samples. The basis for the statistics can be either a number of rows of your choice or a percentage of the table. While the statistics are not exact, there are generally sufficient for a correct calculation of the SELECT strategy since this depends less on precision than on distinguishing between selective and non-selective columns.

Especially when creating an additional index for an inefficiently processed SQL command, the selectivity of all columns of a table can be determined relatively quickly using 'UPDATE STATISTICS COLUMN (*) ESTIMATE SAMPLE 20000 ROWS'. The selectivity of a column is an important criterion when selecting index columns.

The following values have proven adequate sampling quantities for column statistics: 20,000 rows or 5% for tables with more than 1,000,000 data records.

As of version 7.6, the sampling procedure in the standard uses a new algorithm for calculating the statistics data. You can determine the algorithm to be used with the parameter UPDATESTAT_SAMPLE_ALGO. The new algorithm generates more accurate statistics with fewer records read.

Thank you!

