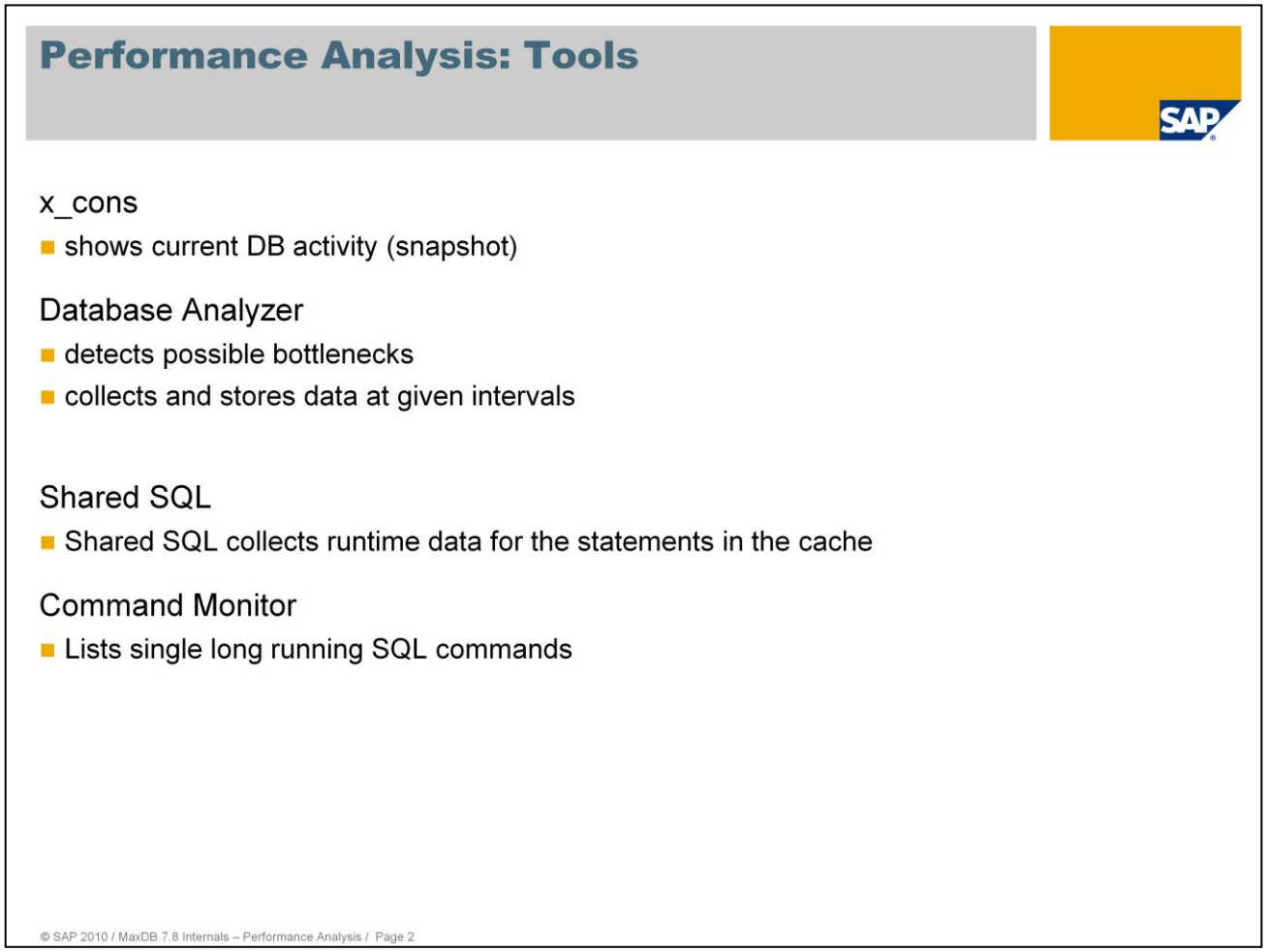
MaxDB Internals
Performance Analysis
Version 7.8

Christiane Hienger

THE BEST-RUN BUSINESSES RUN SAP™



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
- As of version 7.8 Shared SQL collects performance data with each statement in the cache. This substitutes the former Resource Monitor in older versions which had to be enabled explicitly.
- The Command Monitor provides a list of long-running or poorly-processed SQL statements

x_cons and Database Analyzer are stand-alone programs and are called from the operating system command line. The Command Monitor and Shared SQL are part of the core functions of the MaxDB kernel.

In SAP WebAS, all functions and results can be controlled and analyzed using transaction DBACockpit => Current Status or DBACockpit => Performance.

The former transactions DB50 and ST04 have been substituted by transaction DBACockpit in newer SAP WebAS versions.

DB-Console: x_cons

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 WB5 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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 3

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.

DB Console x_cons (II)

```
x_cons <dbname> <Command> [<interval>] [<repeat>]
<Command> (choice):
                                    SHOW ACTIVE          [ DW | SV | US | GC]
                                    SHOW ALL
                                    SHOW AIO  (backup only)
show io statistics/states           SHOW IO
                                    SHOW DEV_IO
                                    SHOW IOPENDING
                                    SHOW CPORT
show move info (load balancing)     SHOW MOVEINFO
                                    SHOW QUEUES
                                    SHOW REGIONS
                                    SHOW RTE
                                    SHOW RUNNABLE        [ DW | SV | US | GC]
UKT sleep statistic                 SHOW SLEEP
                                    SHOW STATE
                                    SHOW STORAGE
suspend reasons                     SHOW SUSPENDS
show task counts                    SHOW T_CNT           [ DW | SV | US | T<taskindex>]
show tasks move info                SHOW T_MOVE
show task queues                    SHOW T_QUEUE
show task regions                   SHOW T_REG
show task statistics                SHOW T_STAT
                                    SHOW TASKS
Thread time usage                   SHOW THRD_TIMES
                                    SHOW SLEEP
                                    SHOW VERSIONS
cancels the command of task         CANCEL <taskindex>
displays help file                  HELP
time measurement                    TIME   <ENABLE | DISABLE>
kills the session of task           KILL <taskindex>
```

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 4

CANCEL	index	cancels the command executed by task <index>
KILL	index	kills the session of task <index>
SHOW		
DEBUGLEV	level	set debug level for the kernel
DEBUGTASK	index	writes back trace of task to knldiag
RESET	obj_cnt	resets counter about the following objects: IO T_CNT REGIONS (ALL) incl. local counters of any task
TIME	enable	enables time measurements
QUIT		exit console driver

The option –e before a SHOW command shows an extended output for tasks lists.

x_cons Process Configuration (1)

```
x_cons <dbname> show rte
Kernel Threads:
 Thread             UNIX  State         Sleep
 Name                Tid                Time
 TIMER             16660  Sleeping      2
 COORDINATOR       16639  Sleeping
 CLOCK             16654  Sleeping
 CONSOLE           16655  Sleeping
 REQUESTOR         16658  Sleeping
 …


User Kernel Threads:
 Thread        UNIX  State       Dispatch  TaskSwitch Active Total Task
 Name           Tid              Counter     Counter  Tasks Tasks Cluster
 UKT1         19067  Sleeping          3           0      1     1 TW
 UKT2         19068  Sleeping      18210           0      1     1 LW
 UKT3         19069  Sleeping          2           0      0     1 UT
 UKT4         19070  Sleeping      14525         393     43    43 43*SV
 UKT5         19071  Sleeping          1           0      0     8 8*FS
 UKT6         19072  Sleeping       3537        3533     10    10 10*GC
 UKT7         19073  Sleeping      70395       52103     65    65 TI,64*PG
 UKT8         19074  Sleeping     239871       18583      9    47 46*US,IDL
 UKT9         19075  Sleeping     457240       15572     10    55 54*US,IDL
```

Kernel parameters (don‘t 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 : **8**
Processor cores: **2**

© SAP 2010 / MaxDB 7.8 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 parameters MAXCPU and UseableCPUs are 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.	Queue 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-Queues per volume, see chapter Kernel parameters

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 6

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> [-e] 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	Region 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 2010 / MaxDB 7.8 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

UKTsleep

- Number of semaphore waits per UKT

x_cons: Task States (1)

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)

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 8

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.

x_cons: Task States (2)

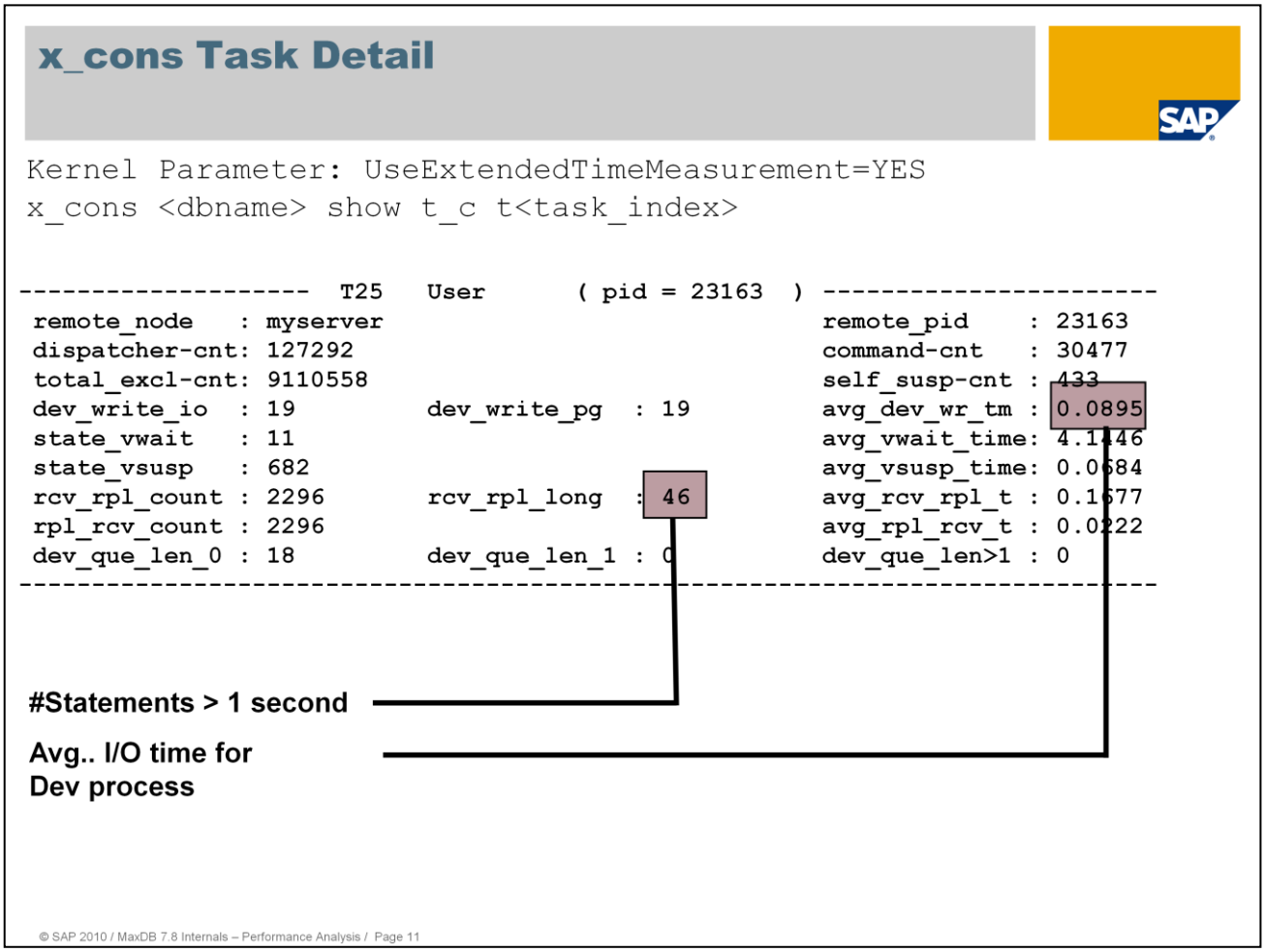
Not Connected	
ReschedulеMsec	brief wait, continues automatically
Runnable	immediately runnable
Running	running, using CPU time
Stopped	suspended by kernel and waiting to proceed running
Terminated	task or database session has been canceled
UNKNOWN	task state unknown
Vacknowledge	
Vattach	opens I/O ports (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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 9

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /



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).

As of version 7.8 the database should run with the default setting UseExtendedTimeMeasurement=YES. The kernel collects time values of most wait situations. Shared SQL stores time values belonging to SQL statements.

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 2010 / MaxDB 7.8 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 asynschonous.

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

Performance tables: Process Configuration

TASKGROUPSTATISTICS

- shows all threads, dealing with tasks
- analog to „`x_cons <dbname> show rte`“

BACKUPTHREADS

- shows all volumes and backup media, used during a backup or restore
- analog to„`x_cons <dbname> show aio`“

IOTHREADSTATISTICS

- Shows I/O figures per volume
- analog zu „`x_cons <dbname> show dev_io`“

IOJOBS

- Current state of the I/O orders in the I/O Queues
- analog to„`x_cons <dbname> show iopending`“

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 13

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->DBACockpit.

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

Performance tables: Task Activities (1)

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“

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Performance tables: Task Activities (2)

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“

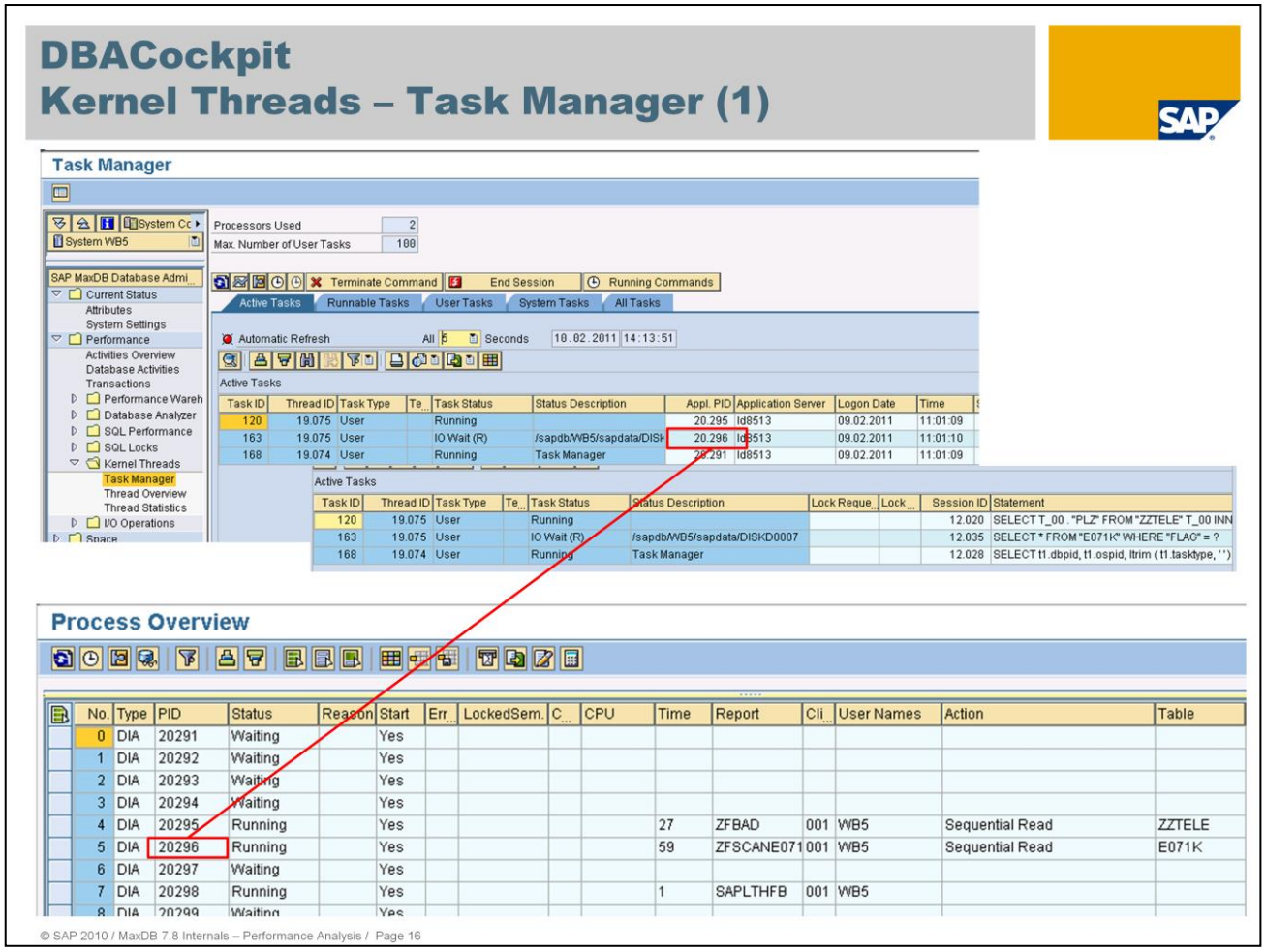
SYSMON_DW_ACTIVE / SYSMON_DW_RUNNABLE

- 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“

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /



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:

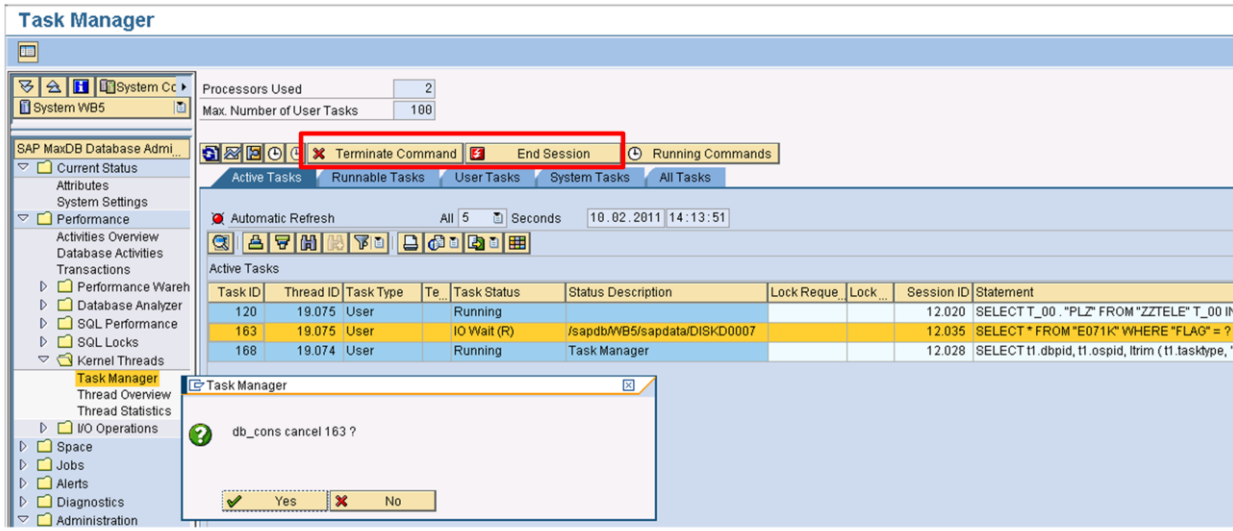
- For this database MAXCPU is set to 2. Thus the database can use 2 CPUs in parallel. Task T168 is running in another UKT (see Thread Overview, thread ID:19074) than task T120 and T163 (thread ID:19075). Tasks T120 and T168 can both have the *Running* status.
- We see a command (T163) that reads data from the disk to the cache - IO-WAIT (R).
- The task manager show the SQL statements executed by the sessions as of MaxDB version 7.8.

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 (20296).

DB50: Kernel Threads – Task Manager (2)

- The task manager can stop running SQL commands and terminate sessions



With the task manager it is possible to terminate the respective task (T163) 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.

Database Analyzer
Consider a Typical Problem ...

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

Database Analyzer Objectives

Gathering relevant performance data with one tool

- Flexible and upgradable through new rule sets
- Release and instance independent
- Remote access possible

Parameter setting recommendations

- Check current parameter values
- Show parameter recommendations according to the used version
- SAP note 1111426 provides configuration files

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 19

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

The DBAnalyzer is available as of version 7.3.00 Build 25.

Enhanceability

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 cahnge 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

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).

Database Analyzer Properties

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 20

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. Shared SQL and the command monitor are 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.

Database Analyzer Technical Details

Executable dbanalyzer

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

Configuration file dbanalyzer.cfg

- <Global Data>/env/dbanalyzer76.cfg

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 21

Configuration File: dbanalyzer.cfg

- Describes the data to be collected or calculated (***parameters***). These ***parameters*** are either taken from the database (system tables) or calculated from the ***data*** 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.

Database Analyzer
Monitoring System Tables

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...
```

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 22

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.*

Database Analyzer Features (1)

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer Features (2)

Task specific warnings on

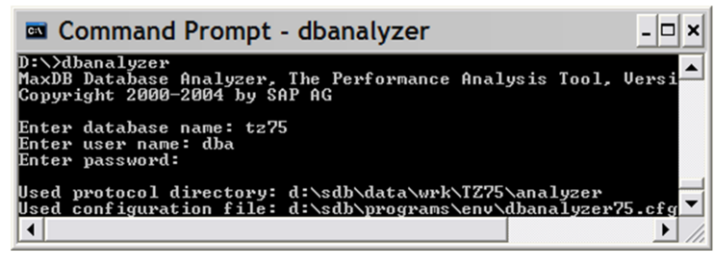
- poor I/O-times
- high lock waits (vwait/vsuspend)
- long command runtimes (receive/reply)
- high read activity (reads)
- a Usertask blockades in a certain state (e.g. Vwait, Vbegexcl…)

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer: Program Start

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 DBACockpit ->Performance->Database Analyzer
  - 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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 25

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.

Database Analyzer
Time Interval

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)

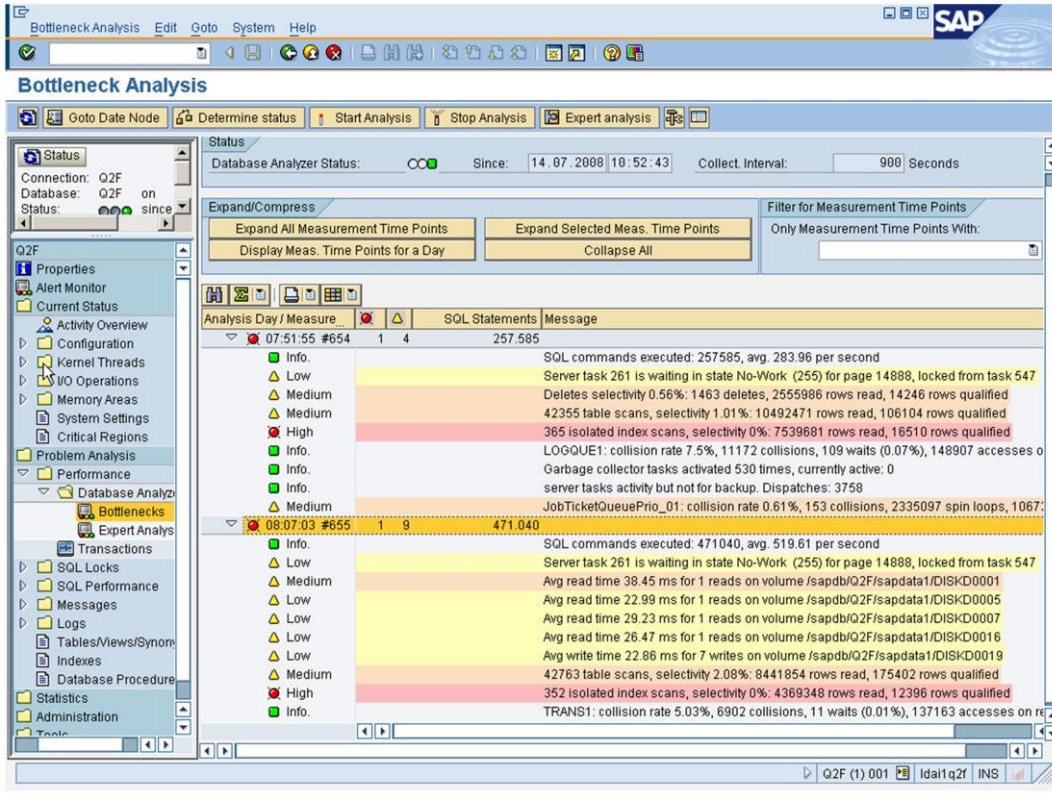
in both cases

- creating and saving the protocol files

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 26

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 Alerts



© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 27

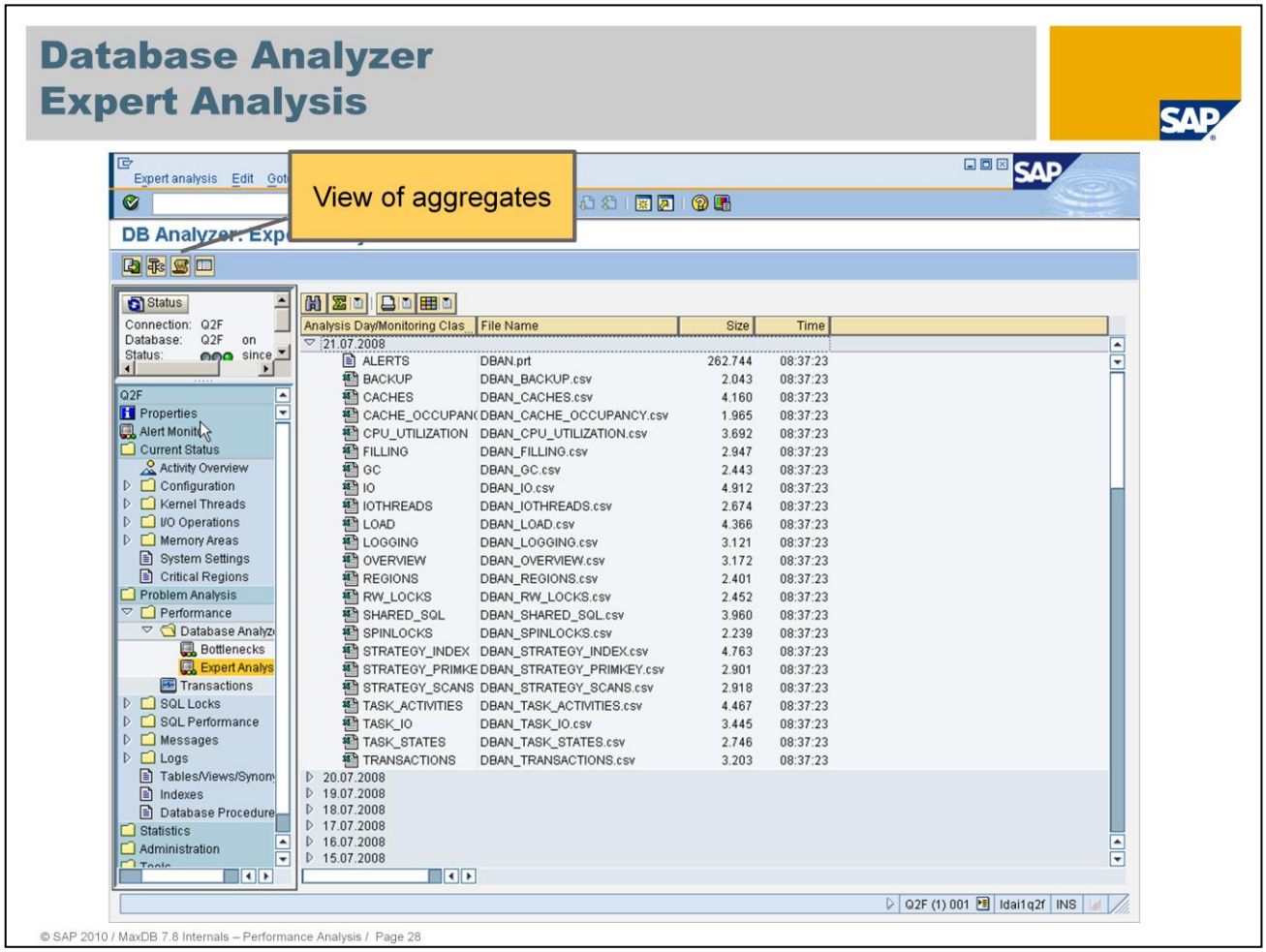
As of support packages 6.20 SP37, the Database Analyzer starts automatically when the SAP WebAS system is started.

You can call the Database Analyzer from transaction DBACockpit ->Performance->Database Analyzer. 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.*

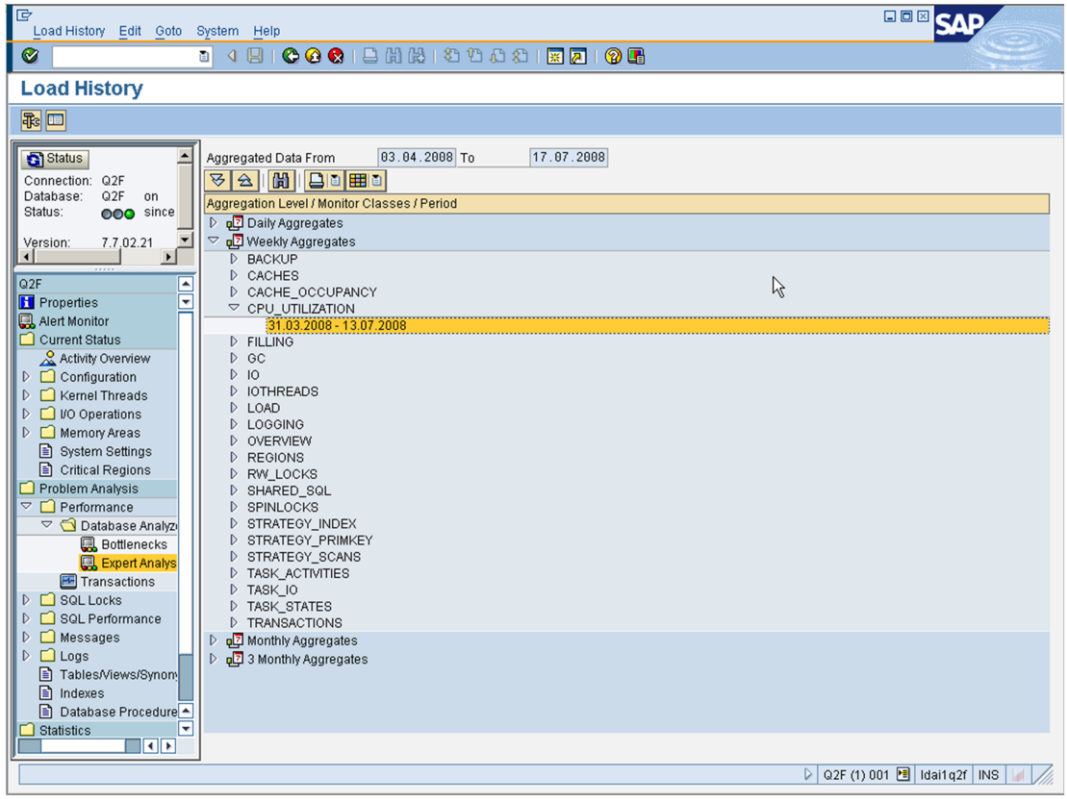
Detected bottlenecks are output in text form to rapidly provide database administrators with an overview of the possible causes of performance problems. The analysis can be performed just once or at regular intervals.



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

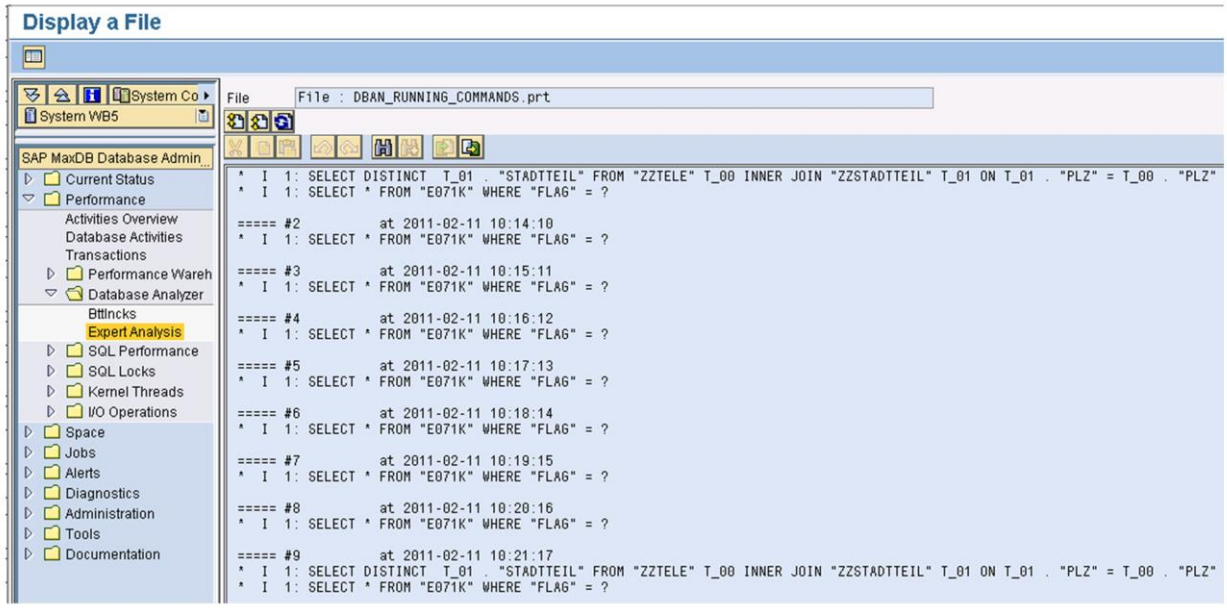
Logs are implicitly deleted periodically via the program *RSDBANCONTROL*. You can configure how long logs are kept using transaction DB59 in the integration data for the respective system. (6.20 as of basis SP 37).

Database Analyzer
View of aggregates



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.

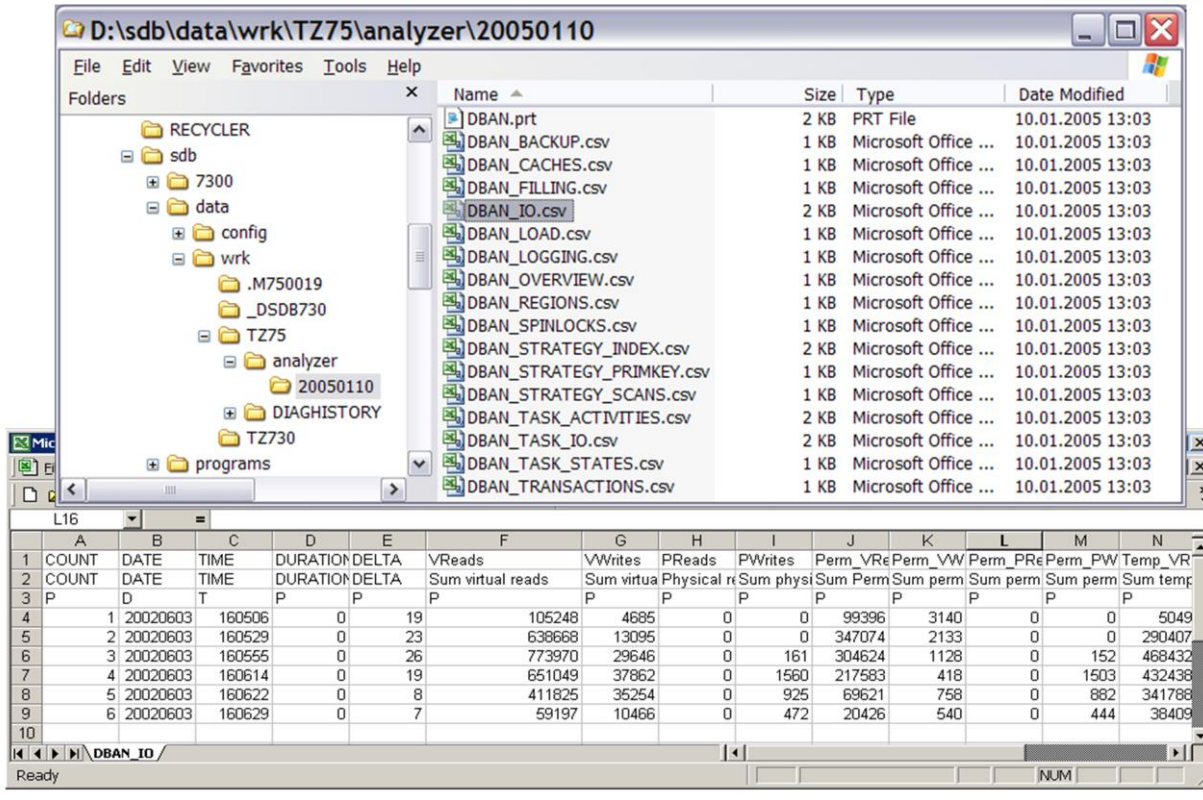
Database Analyzer
Example: Running Commands



© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 30

The Database Analyzer writes snapshot data like the running commands and the task activities (x_cons show active) into log files as of version 7.8.

Database Analyzer Log Files (1)



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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 31

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.

Database Analyzer
Log Files (2)

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 / selektivity of selects and fetches, inserts, updates, deletes

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer
Log Files (3)

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer
Log Files (4)

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 requst timeouts and lock request escalations

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer
Log Files (5)

RUNNING_COMMANDS.prt

- Running SQL commands at the time of the data collection

SHOW_ACTIVE.prt

- Active tasks at the time of the data collection

DBAN_UKT_CPU_UTILIZATION.prt

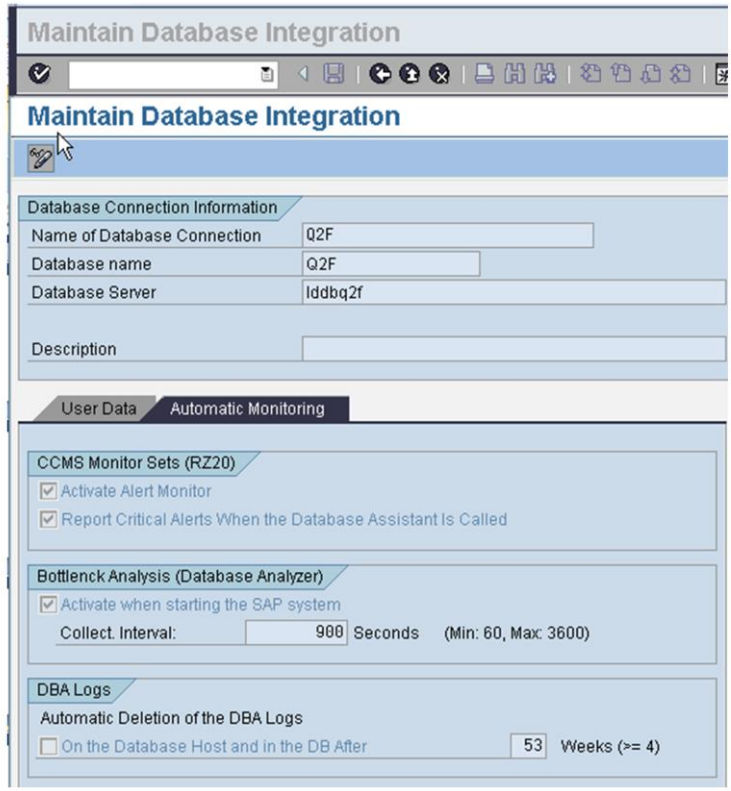
- CPU Usage of the User Kernel Threads at the time of the data colleciton

DBAN_USER_TASKS_CMDS_EXECUTED.prt

- Dispatcher Counts per Usertask serving a session

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis /

Database Analyzer Log Files (6)



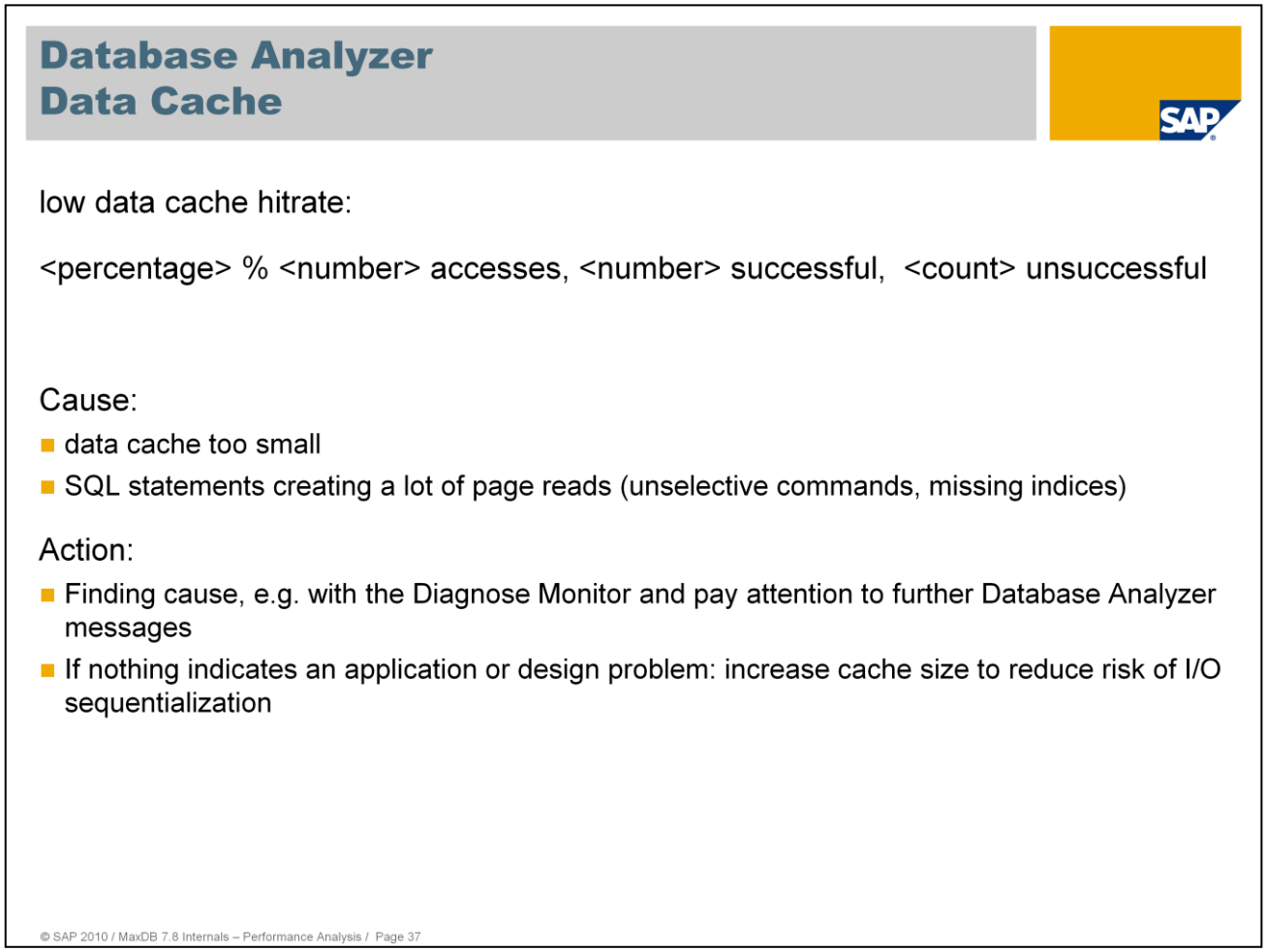
© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 36

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.



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.

Database Analyzer
Paging Cache Entries

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

© SAP 2010 / MaxDB 7.8 Internals – Performance Analysis / Page 38

Database Analyzer: cache displacements

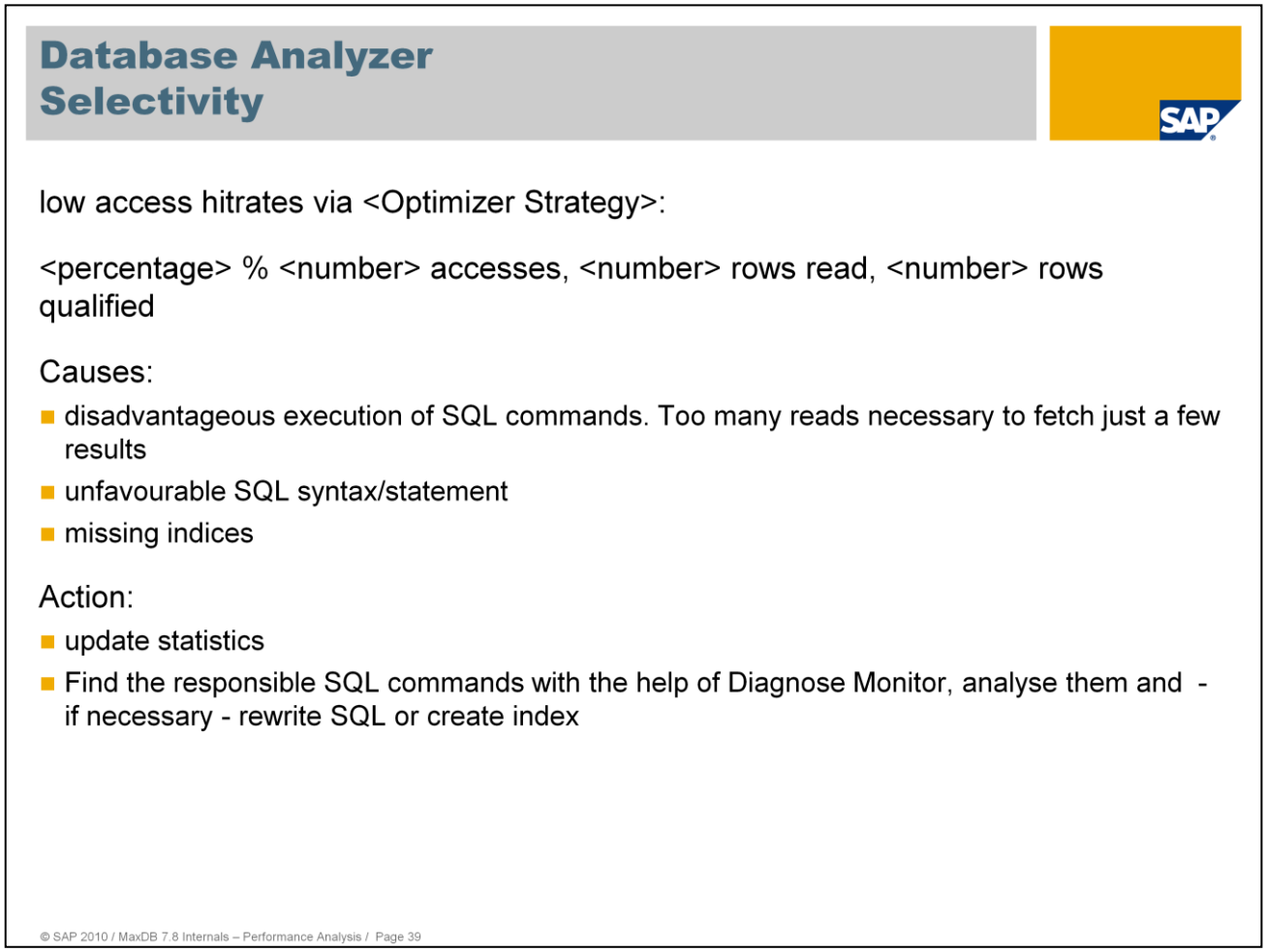
Cache displacements: <number of> pages/second

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 *pagers* 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).



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.).

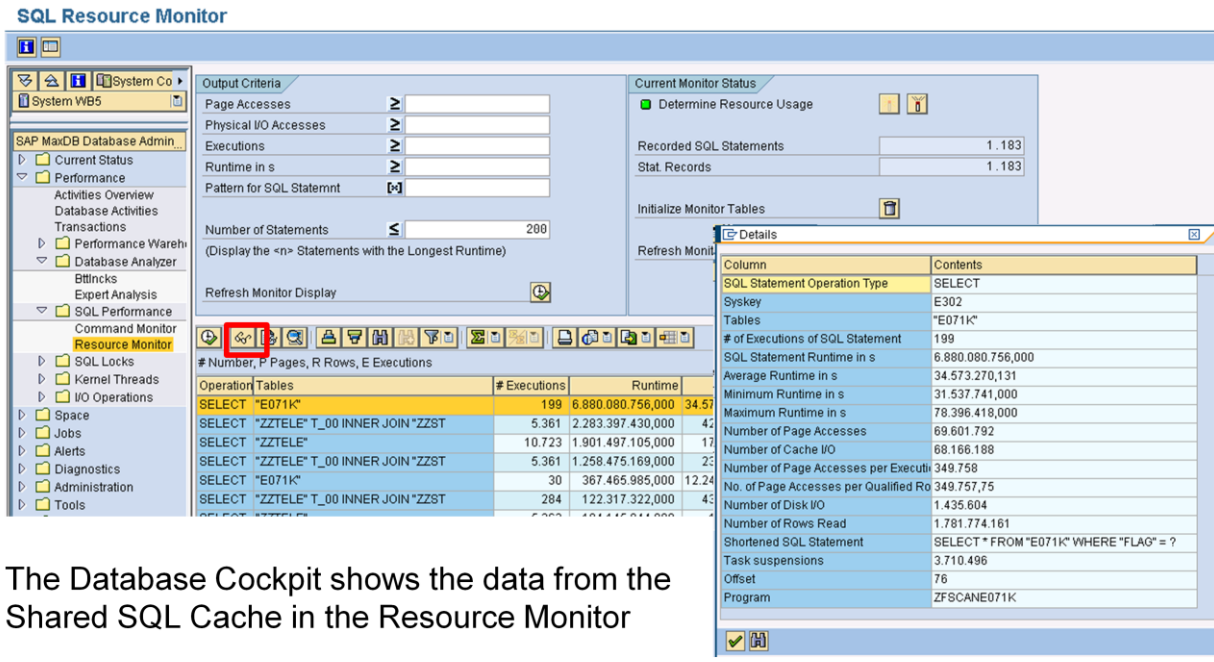
User response

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 the data in the Shared SQL Cache or with the Command Monitor.

Shared SQL
Display SQL statements

THE tool for identifying high load causing SQL commands



The Database Cockpit shows the data from the Shared SQL Cache in the Resource Monitor

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 40

As of version 7.8 MaxDB collects runtime data for the known SQL commands. These figures are available if Shared SQL is turned on an the parameter UseExtendedTimeMeasurement has the value YES.

The runtime data collection has a small impact on the performance of the system. The statement text is available in Shared SQL and doesn't need to be stored in specific monitoring tables.

The Resource Monitor in the DBACockpit works downward compatible. It shows the data from the diagnose analyze tool in older version and the data from Shared SQL with MaxDB 7.8 and newer versions.

The runtime data includes information about number of executions, overall – minimum – maximum and average execution time, number of page accesses in memory and on disk, number of read and qualified rows, wait situations and more.

Shared SQL collects the execution times in microseconds. The Resource Monitor shows the execution times as milliseconds which is incorrect in the current version of the DBACockpit.

The Shared SQL data helps analyzing the over all load in the systems. The Resource Monitor filters the commands to be displayed by the output criteria.

This example shows a select reading data from table E071K. It runs quite often with a high execution time. It reads all records but doesn't find any according to the WHERE condition. An optimization of the application or an index on table E071K for the field FLAG could reduce the load in the system significantly.

A double-click on the command guides you to the complete statement. The Resource Monitor can jump to the table definition and to the ABAP program. The DBACockpit supports the explain command in the Command Monitor. MaxDB needs the values of the input parameters for the explain. Shared SQL doesn't store input value parameters of the single executions.

Shared SQL
View COMMANDSTATISTICS

The system view COMMANDSTATISTICS shows all entries in Shared SQL

SQL | Result (1)

```
select * from sysinfo.commandstatistics
order by executetime desc
```

	STATEMENT	APPLICATIONINFORMATION	APPLICATIONLINENUMBER	CURRENTEXECUTECOUNT	EXECUTECOUNT	EXECUTETIME	AVGEXECUTETIM
1	SELECT * FROM "E071K" WHERE "FLAG" = ?	ZFSCANE071K	76	0	199	6880080756	3457327
2	SELECT DISTINCT T_01 . "STADTTEIL" FROM "ZZTELE" T_00 INNER JOIN "ZZSTADTTEIL" T_01 ON...	ZFBAD	515	0	5361	2283397430	42592
3	SELECT "NAME" , "VORNAME" , "STR" , "NR" , "PLZ" , "ORT" , "CODE" , "ADDINFO" FROM "ZZTELE" ...	ZFBAD	333	0	10723	1901497105	17732
4	SELECT T_00 . "PLZ" FROM "ZZTELE" T_00 INNER JOIN "ZZSTADTTEIL" T_01 ON T_01 . "PLZ" = T_...	ZFBAD	542	0	5361	1258475169	23474
5	SELECT * FROM "E071K" WHERE "FLAG" = ?	ZFSCANE071K	76	0	30	367465985	1224886
6	SELECT DISTINCT T_01 . "STADTTEIL" FROM "ZZTELE" T_00 INNER JOIN "ZZSTADTTEIL" T_01 ON...	ZFBAD	515	0	284	122317322	43069
7	SELECT /*+ FIRST_ROWS (2000) */ * FROM "ZZTELE" WHERE "STR" = ? AND "NR" BETWEEN ? A...	ZFBAD	294	0	5362	104145944	1942
8	SELECT "NAME" , "VORNAME" , "STR" , "NR" , "PLZ" , "ORT" , "CODE" , "ADDINFO" FROM "ZZTELE" ...	ZFBAD	333	0	567	102704844	18113
9	SELECT T_00 . "PLZ" FROM "ZZTELE" T_00 INNER JOIN "ZZSTADTTEIL" T_01 ON T_01 . "PLZ" = T_...	ZFBAD	542	0	284	66394602	23378
10	SELECT T_00 . "NAME" , T_00 . "VORNAME" , T_00 . "STR" , T_00 . "NR" , T_00 . "PLZ" , T_00 . "O...	ZFBAD	442	0	5361	26335801	491
11	SELECT /*+ FIRST_ROWS (6000) */ "NAME" , "PLZ" FROM "ZZTELE" WHERE ROWNUM <= ?	ZFBAD	181	0	5362	25877704	482
12	SELECT "LDATA" FROM "REPOLOAD" WHERE "PROGNAME" = ? AND "R3STATE" = ? AND "MACH" =...	UNKNOWN	0	0	895	18417943	2057
13	SELECT VIRTUALKEY FROM DOMAIN.TABLES WHERE OWNER = USER AND TABLENAME = ?	SDB1FADA	2442	0	133806	16830985	12
14	SELECT TABNAME, BLOCKNR, FIELDSLG, FIELDS FROM "DDNTF" WHERE TABNAME = ? ORDER B...	UNKNOWN	0	0	2274	16348569	718
15	SELECT * FROM "ZZTELE" WHERE "NAME" IN (? , ?)	ZFBAD	135	0	5362	14282201	266
16	SELECT * FROM "ZZTELE" WHERE "STR" = ? AND "NR" BETWEEN ? AND ? ORDER BY "NAME" , "VO...	ZFBAD	270	0	5362	14229313	265
17	INSERT INTO "TBTCP" VALUES(? , ? ...	SAPMSSY2	5819	0	1877	9728088	518
18	SELECT DISTINCT tablename FROM domain.columns WHERE schemaname = CURRENT_SCHEMA A...	CL_SQL_STATEMENT====...	430	0	6	8966320	149438
19	SELECT /*+ FIRST_ROWS (2000) */ * FROM "ZZTELE" WHERE "STR" = ? AND "NR" BETWEEN ? A...	ZFBAD	294	0	284	7910098	2785
20	INSERT INTO "TBTCO" VALUES(? , ?...	SAPMSSY2	5089	0	975	7342280	753
21	SELECT "DEVCLASS" , "INTSYS" , "CONSYS" , "CTEXT" , "KORRFLAG" , "AS4USER" , "PDEVCLASS" , ...	SAPLSTAM	2302	0	78	6161601	7899
22	SELECT "LDATA" FROM "REPOLOAD" WHERE "PROGNAME" = ? AND "R3STATE" = ? AND "MACH" =...	UNKNOWN	0	0	199	5669796	2849
23	SELECT TABNAME, BLOCKNR, FIELDSLG, FIELDS FROM "DDNTF" WHERE TABNAME = ? ORDER B...	UNKNOWN	0	0	718	4636797	645
24	SELECT "JOBCOUNT" FROM "TBTCO" WHERE "JOBNAME" = ? AND "JOBCOUNT" LIKE ? ORDER BY "...	SAPLBTCH	67608	0	975	4188609	429
25	SELECT "NUM","NAME_PREFIX","NAME_POSTFIX" FROM "SVMCRT_RESOURCES" WHERE "MOD_NU...	UNKNOWN	0	0	14	3810216	27215
26	INSERT INTO "D010TAB" ("MASTER" , "TABNAME") VALUES (? , ?)	UNKNOWN	0	0	7	3580207	51145
27	SELECT NUM_TABS_FEW_SAMPLE_ROWS FROM DBAN_NUM_TABS_FEW_SAMPLE_ROWS	UNKNOWN	0	0	1	3536641	353664
28	SELECT * INTO ?,? FROM "TADIR" WHERE "PGMID" = ? AND "OBJECT" = ? AND "OBJ_NAME" = ? WITH LOCK ISOLATION LEVEL 1			0	2503	3360493	134

A select from this view with the condition „WHERE CURRENTEXECUTECOUT > 0“ shows all currently running commands.

A select from the system view SYSINFO.COMMANDSTATISTICS returns the runtime values stored in the Shared SQL Cache. The column CURRENTEXECUTECOUNT shows the number of sessions currently executing the command.

The view SYSINFO.COMMANDSTATISTICSRESET show the runtime values of commands executed after a reset. The SQL command “diagnose analyze clear all” performs the reset.

The following SQL statement shows all command executions after a reset:

```
select c.statement, r.* from commandstatisticsreset r, commandstatistics c
where c.commandid = r.commandid
order by r.executetime desc
```

Command Monitor

THE tool to find long running command executions and log them with the input parameter values

Logging of problematic SQL commands according to defined filters.

The Command Monitor logs the ID of a command with monitoring values if

- the number of **page accesses** in memory counted with an execution exceed the given value.
- the **runtime** of an execution exceeds the given value in milliseconds
- the **selectivity** (relation between read records and qualified records) falls below the given value

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 42

The Command Monitor finds long and/or expensive SQL command executions. Different input parameter values can lead to very different execution times. MaxDB can use different search strategies according to the input parameter values.

The Command Monitor stores the command ID, the monitoring values and the input parameter values when a command execution exceeds (page accesses and execution time) or falls below (selectivity) the given filter value.

The transaction DBACockpit and the Database Studio can run an Explain with the logged command using the input parameter values. Explain shows the search strategy of the statement.

The DBACockpit and the Database Studio use SQL statements to enable the Command Monitor and to display the results of the monitoring. Administrators can use those SQL statements manually, as well.

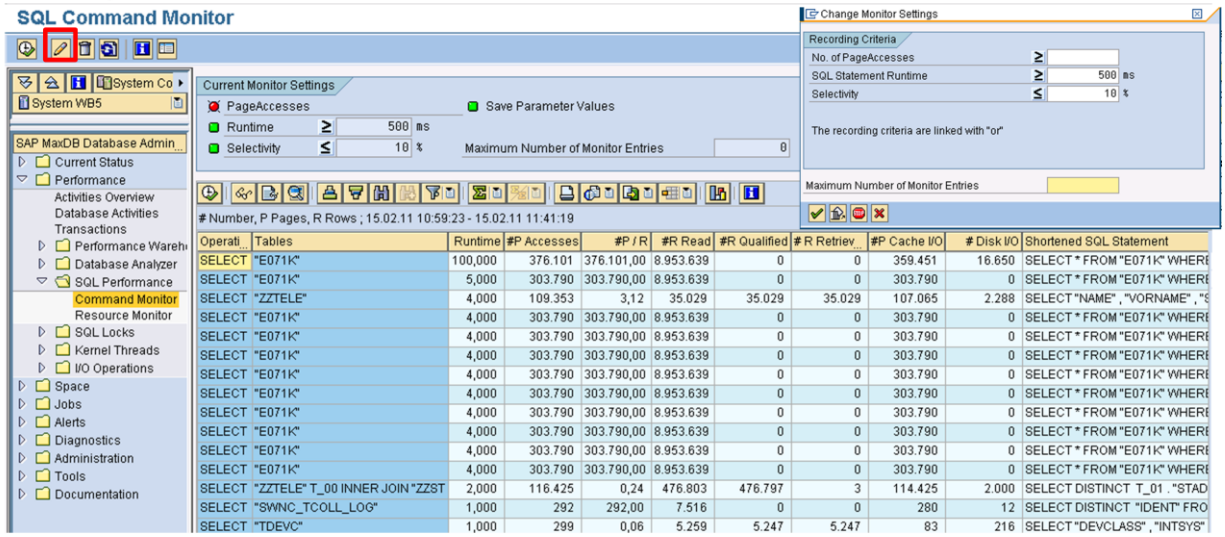
The Command Monitor has been re-implemented with MaxDB version 7.8. It uses commands stored in the Shared SQL Cache. It doesn't log the statement text anymore but only the command ID known by Shared SQL. This reduces the memory footprint of the Command Monitor significantly.

Furthermore the new implementation has a very small impact on the performance of the system as long as users define reasonable filter values.

Command Monitor DBACockpit

Turn on Command Monitor and show results

- Filter values combined with OR
- Choose reasonable filter values to prevent too much logging and performance impact
- Double click on row shows detailed information
- Settings remain after database restart



Current Monitor Settings

● PageAccesses
🟩 Runtime ≥ 500 ms
🟩 Selectivity ≤ 10 %
🟩 Save Parameter Values
Maximum Number of Monitor Entries 0

Change Monitor Settings

Recording Criteria
No. of PageAccesses ≥
SQL Statement Runtime ≥ 500 ms
Selectivity ≤ 10 %

The recording criteria are linked with "or"

Maximum Number of Monitor Entries

Number, P Pages, R Rows ; 15.02.11 10:59:23 - 15.02.11 11:41:19

Operati...	Tables	Runtime	#P Accesses	#P / R	#R Read	#R Qualified	# R Retriev...	#P Cache I/O	# Disk I/O	Shortened SQL Statement
SELECT	"E071K"	100,000	376.101	376.101,00	8.953.639	0	0	359.451	16.650	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	5,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"ZZTELE"	4,000	109.353	3,12	35.029	35.029	35.029	107.065	2.288	SELECT "NAME" , "VORNAME" , "
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"E071K"	4,000	303.790	303.790,00	8.953.639	0	0	303.790	0	SELECT * FROM "E071K" WHERE
SELECT	"ZZTELE" T_00 INNER JOIN "ZZST	2,000	116.425	0,24	476.803	476.797	3	114.425	2.000	SELECT DISTINCT T_01 . "STAD
SELECT	"SWNC_TCOLL_LOG"	1,000	292	292,00	7.516	0	0	280	12	SELECT DISTINCT "IDENT" FRO
SELECT	"TDEVC"	1,000	299	0,06	5.259	5.247	5.247	83	216	SELECT "DEVCLASS" , "INTSYS"

Users can turn on the Command Monitor with the desired filter value in the Performance section of the DBACockpit. The DBACockpit displays the logged SQL commands with the runtime values.

A double click on the line of a command shows more detail information about the command.

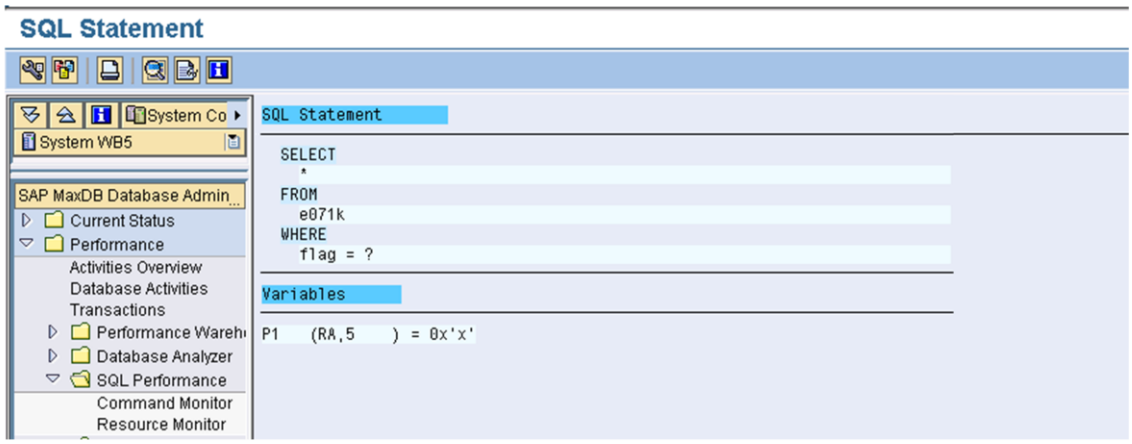
The layout definition allows the selection of user defined output columns. Important columns are:

Table	Table used by the statement
Program	ABAP Program calling the SQL command
Rutime	Runtime of the SQL command in seconds
#P Accesses	Number of page accesses in cache
#R Read	Number of read records
#R Qualified	Number of records matching the where clause
#P/R	Number of page accesses per qualified record
#Fetched	Number of records transported to the application
#Disk I/O	I/O accesses from and to disk
SQL Waits	Number of lock collisions
Task Suspends	Number of internal memory access collisions
No. Fetch Orders	Number of fetch order for record transportation
Result is Copied.	YES: an internal result set was created
Date	execution date
Time	execution time

Command Monitor
Display statement with input parameter values

Statement view supports:

- Explain
- Explain with trace
- Jump into ABAP program to the statement position
- Show table definitions



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 44

The Command Monitor collects the input parameter values belonging to the command executions. The DBACockpit inserts the values into the command text and executes an Explain.

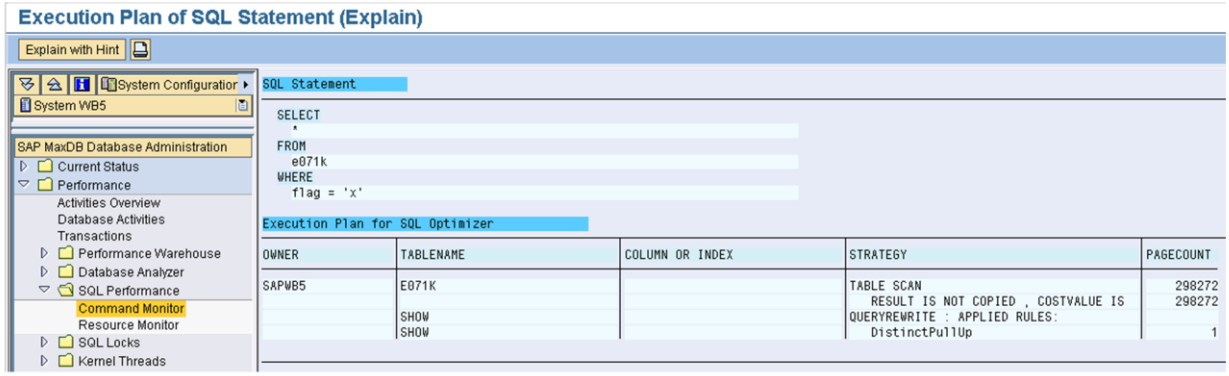
Explain show the search strategy found by the optimizer at the time of the explain execution.

The DBACockpit can jump into the ABAP program calling the SQL command. It can also jump into the table definition view with the tables referenced by the SQL command. The table definition view shows the definition in the database, not the data dictionary definition of the WebAS.

Command Monitor
Explain

EXPLAIN shows the search strategy of the command

- Search strategy at this time
- No SQL command execution
- Explain with kernel trace supported
- Explain with hints supported



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 45

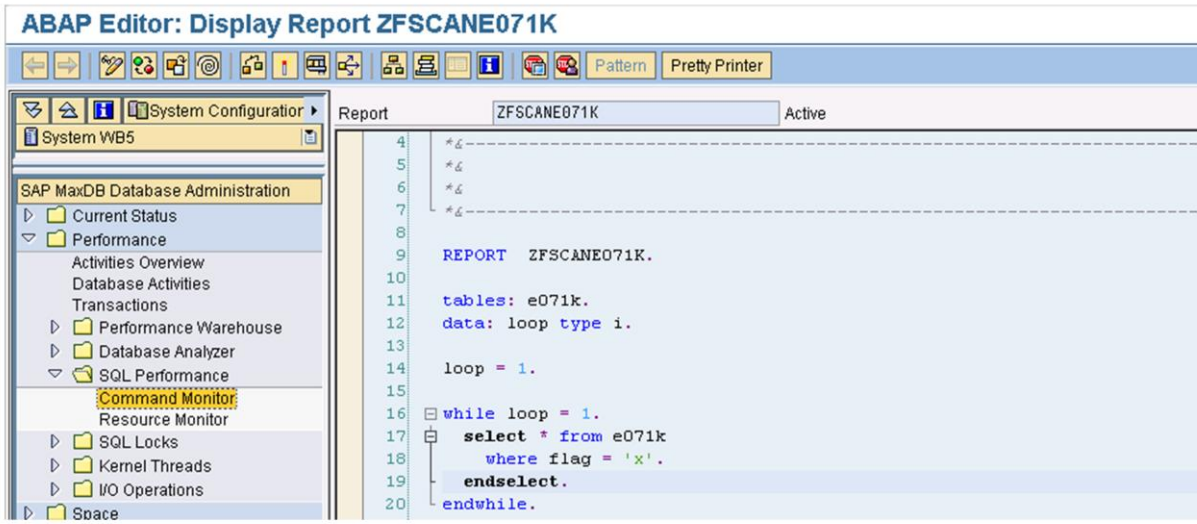
Explain show the search strategy of the command at the current time. The search strategy can change with modified table definitions or updated statistics. The search strategy at the Explain time can be different to the search strategy at the execution time of the statement.

The chapter Query Optimization provides more detailed information about the Explain command.

Command Monitor
Display Call position in ABAP Program

Display of the ABAP program with the call position of the SQL command

- Shared SQL stores the ABAP program and the call position
- The DBACockpit uses this information to jump into the ABAP program
- Supported in Command and Resource Monitor



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 46

The database interfaces transport the name of the ABAP program and the call position with a prepare of a statement to the database. Shared SQL stores this information with the command.

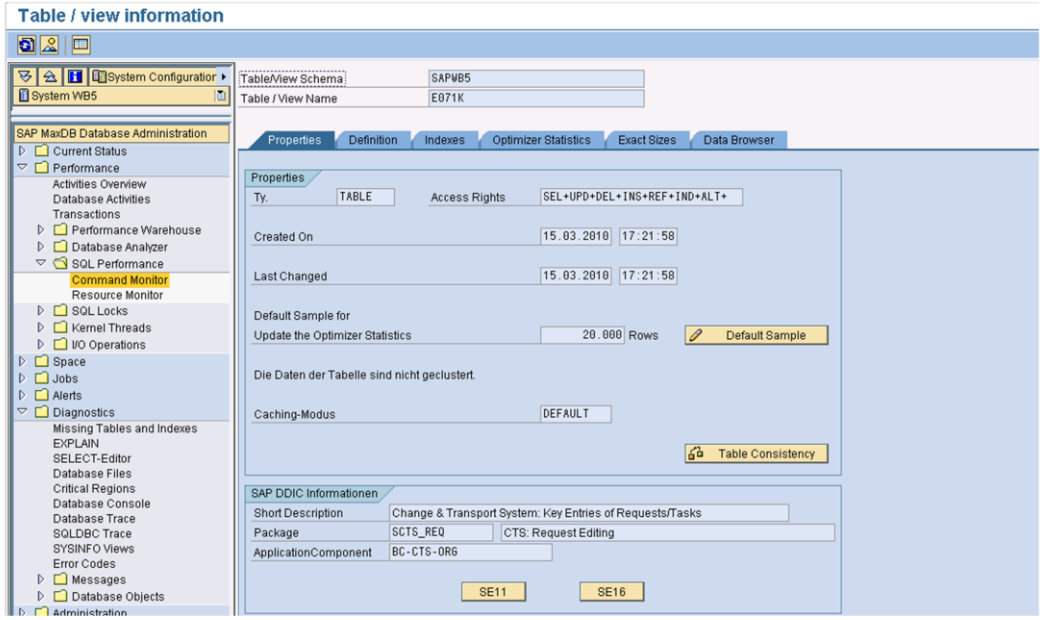
The DBACockpit uses this information to jump directly into the ABAP program to the call position of the statement. Both, the Command and the Resource Monitor support this functionality.

ABAP SQL statements often look different to the commands sent to the database. The ABAP database interfaces generate the SQL commands depending on profile settings and the used database management system.

Command Monitor
Table Definition

Display table definition

- Jump from statement view into table definition view
- Shows table definitions on database level (SE11 jumps into DDIC-Editor)



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 47

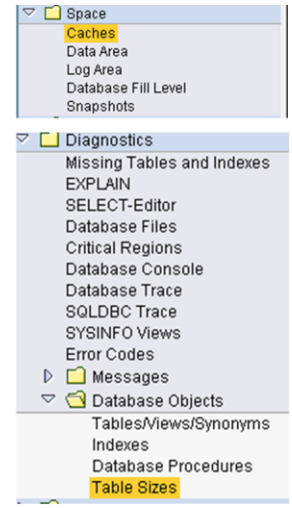
Resource and Command Monitor allow to jump from the statement to the table definition view. The

DBACockpit here shows the definition and more detailed information about the chosen table. The table size, optimizer statistics and the table data can be displayed in this view. Optimizer statistics can be updated as well.

DBACockpit
Further useful Diagnose Functions

The DBACockpit provides further diagnose functions supporting users when analyzing the performance of a system:

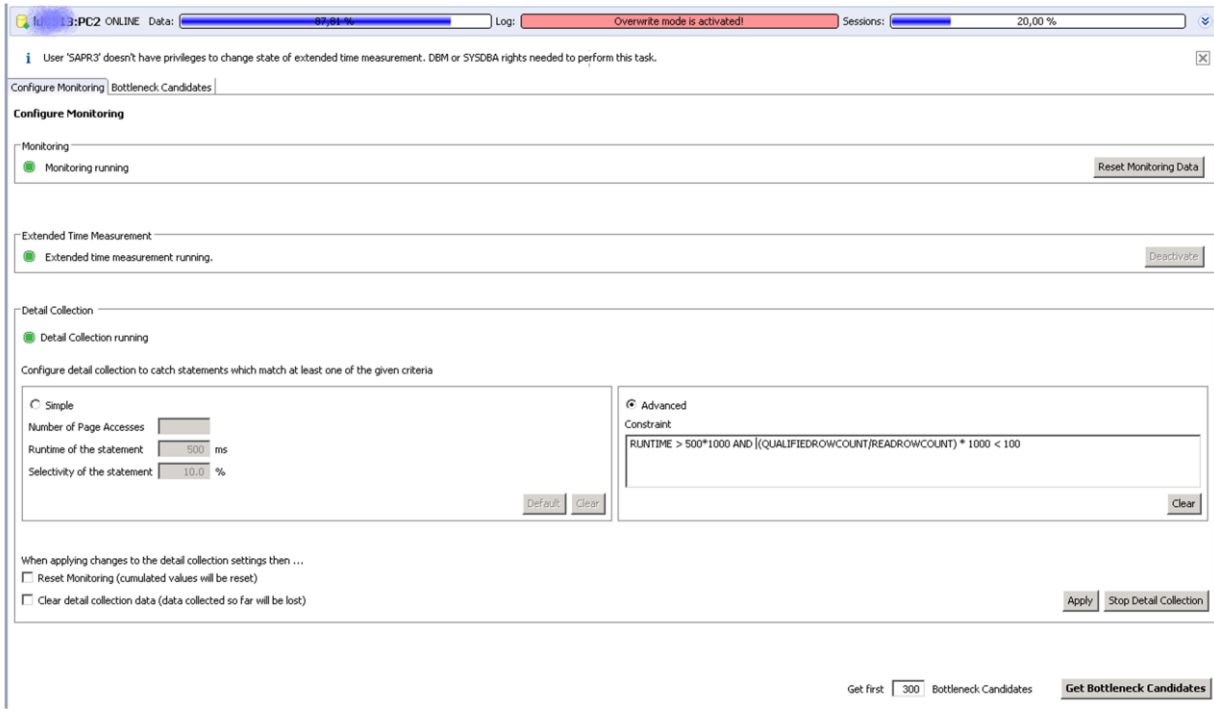
- Caches
  Cache sizes and usage, incl. hitrates
- Missing Tables and Indexes
  Jump into transaction DB02
- EXPLAIN
  Enter a command to get the search strategy
- Select-Editor
  Input and execution of SQL commands
- Critical Regions
  Accesses to synchronized memory areas (x_cons SHOW REGION)
- Database Console
  x_cons output
- SYSINFO Views
  Direct access to all views of the schema SYSINFO
- Database Objects
  Show database object definitions and observe table growth



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 48

Statement Analysis with Database Studio

The Database Studio supports extended filter values defined in a WHERE condition



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 49

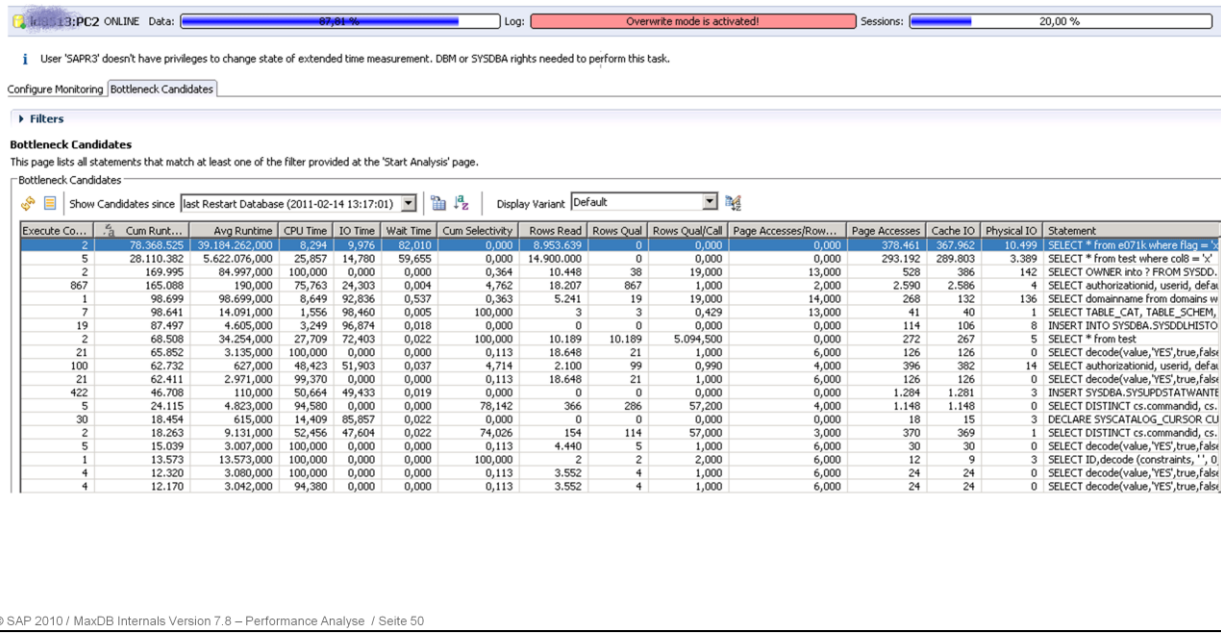
The DBACockpit can access to remote database. It can use the Command and Resource Monitor on databases without ABAP WebAS data.

The Database Studio supports SQL command analysis as well. This is very helpful if no DBACockpit is available. The Database Studio combines Resource and Command Monitor in one “SQL Performance Analysis” Editor. It allows the detailed definition of filter values for the Command Monitor in a WHERE condition .

Statement Analysis with Database Studio (Statement List)

Database Studio shows all commands in the Shared SQL Cache.

Display variants support column groups according to different topics



Execute Co...	Cum Runt...	Avg Runtime	CPU Time	IO Time	Wait Time	Cum Selectivity	Rows Read	Rows Qual	Rows Qual/Call	Page Accesses/Row...	Page Accesses	Cache IO	Physical IO	Statement
2	78.368.525	39.184.262,000	8,294	9,976	82,010	0,000	8.953.639	0	0,000	0,000	378.461	367.962	10.499	SELECT * from e071k where flag = 'x'
5	28.110.382	5.622.076,000	25,857	14,780	59,655	0,000	14.900.000	0	0,000	0,000	293.192	289.803	3.389	SELECT * from test where col8 = 'x'
2	169.995	84.997,000	100,000	0,000	0,000	0,364	10.448	38	19,000	13,000	528	386	142	SELECT OWNER into ? FROM SYSDD.
867	165.088	190,000	75,763	24,303	0,004	4,762	18.207	867	1,000	2,000	2.590	2.586	4	SELECT authorizationid, userid, defa
1	98.699	98.699,000	8,649	92,836	0,537	0,363	5.241	19	19,000	14,000	268	132	136	SELECT domainname from domains w
7	98.641	14.091,000	1,556	98,460	0,005	100,000	3	3	0,429	13,000	41	40	1	SELECT TABLE_CAT, TABLE_SCHEM,
19	87.497	4.605,000	3,249	96,874	0,018	0,000	0	0	0,000	0,000	114	106	8	INSERT INTO SYSDBA.SYSDDLHISTO
2	68.508	34.254,000	27,709	72,403	0,022	100,000	10.189	10.189	5.094,500	0,000	272	267	5	SELECT * from test
21	65.852	3.135,000	100,000	0,000	0,000	0,113	18.648	21	1,000	6,000	126	126	0	SELECT decode(value,'YES',true,fals
100	62.732	627,000	48,423	51,903	0,037	4,714	2.100	99	0,990	4,000	396	382	14	SELECT authorizationid, userid, defa
21	62.411	2.971,000	99,370	0,000	0,000	0,113	18.648	21	1,000	6,000	126	126	0	SELECT decode(value,'YES',true,fals
422	46.708	110,000	50,664	49,433	0,019	0,000	0	0	0,000	0,000	1.284	1.281	3	INSERT SYSDBA.SYSUPDSTATWANTE
5	24.115	4.823,000	94,580	0,000	0,000	78,142	366	286	57,200	4,000	1.148	1.148	0	SELECT DISTINCT cs.commandid, cs.
30	18.454	615,000	14,409	85,857	0,022	0,000	0	0	0,000	0,000	18	15	3	DECLARE SYSCATALOG_CURSOR CU
2	18.263	9.131,000	52,456	47,604	0,022	74,026	154	114	57,000	3,000	370	369	1	SELECT DISTINCT cs.commandid, cs.
5	15.039	3.007,000	100,000	0,000	0,000	0,113	4.440	5	1,000	6,000	30	30	0	SELECT decode(value,'YES',true,fals
1	13.573	13.573,000	100,000	0,000	0,000	100,000	2	2	2,000	6,000	12	9	3	SELECT ID,decode (constraints, '', 0
4	12.320	3.080,000	100,000	0,000	0,000	0,113	3.552	4	1,000	6,000	24	24	0	SELECT decode(value,'YES',true,fals
4	12.170	3.042,000	94,380	0,000	0,000	0,113	3.552	4	1,000	6,000	24	24	0	SELECT decode(value,'YES',true,fals

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 50

The Database Studio shows all command stored in the Shared SQL Cache. The commands are sorted by the execution time by default.

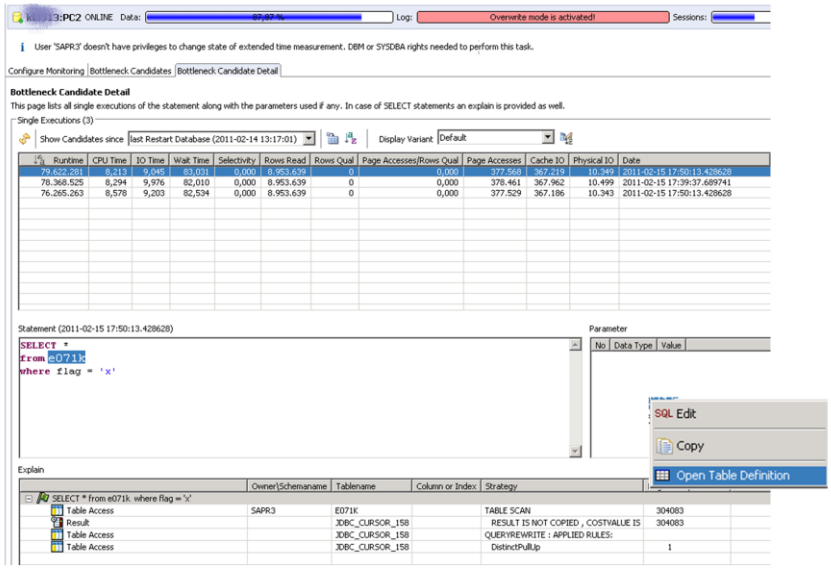
Different variants show groups of runtime figures. This insures a proper overview. The variant “Caller Details” show the commands logged by the Command Monitor only.

A double click on a statement line guides to the detailed statement view.

Statement-Analyse
Statement Details

The view „Bottleneck Candidate Details“ shows

- The monitoring data of the command execution collected by the Command Monitor
- The input parameter values of the selected command execution
- The explain output of the select command with it’s values



© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 51

The view “Bottleneck Candidate Details” shows all single command executions collected by the Command Monitor. It shows the input parameter values and the Explain output according to the chosen command execution.

A right click on the statement opens a new SQL editor with this statement. It can open the table definition editor for marked tables as well.

Command Monitor
Turn On

Database parameters (default values)

- UseSharedSQL = YES
- EnableCommandMonitor = YES
- UseExtendedTimeMeasurement = YES

Filters in table SYSINFO.COMMANDMONITORCONSTRAINTS

- The latest entry defines the current filter
- Old entries remain ➔ viewable history
- Example: Turned off
  <DISABLED>
- Example: Execution time>= 500 ms or selectivity <= 10%
  RUNTIME > 500*1000 OR VIRTUALREADCOUNT > 2147483647 OR
  (QUALIFIEDROWCOUNT/READROWCOUNT) * 1000 < 100

```
select * from sysinfo.commandmonitorconstraints
```

	ID	CREATEDATE	CONSTRAINTS
1	1	2011-01-20 18:16:42.847224	<DISABLED>
2	2	2011-02-15 17:07:07.289902	RUNTIME > 2147483647*1000 OR VIRTUALREADCOUNT > 2147483647 OR (QUALIFIEDROWCOUNT/READROWCOUNT) * 1000 < 100
3	3	2011-02-15 17:07:07.763846	RUNTIME > 500*1000 OR VIRTUALREADCOUNT > 2147483647 OR (QUALIFIEDROWCOUNT/READROWCOUNT) * 1000 < 100

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 52

The latest entry in the table SYSINFO.COMMANDMONITORCONSTRAINTS defines the filter of the Command Monitor. The WHERE condition uses any column of the table COMMANDMONITOR. It can use AND and OR conditions.

With this approach the Command Monitor as of version 7.8 can set more more detailed filters than the monitor in older versions.

The performance impact of the Command Monitor depends on the number of command executions to be logged. A high number of logged commands can lead to high memory consumptions.

The Command Monitor settings remain after a database restart as of version 7.8; i.e. the table COMMANDMONITORCONSTRAINTS stores persistent data.

Command Monitor Tables

Table SYSINFO.COMMANDMONITOR

- COMMANDID references to the ID in the Shared SQL Cache
- Runtime figures
- Execution timestamp
- Session-ID, application process

Table SYSINFO.COMMANDMONITORPARAMETERS

- COMMANDID and execution number reference to table COMMANDMONITOR
- Position, data type and value of input parameters

```
select m.executecount, c.statement, p.value, m.*
from sysinfo.commandstatistics c, sysinfo.commandmonitor m, sysinfo.commandmonitorparameters p
where c.commandid = m.commandid
 and m.commandid = p.commandid
 and m.executecount = p.executecount
 order by m.runtime desc, m.executecount, p.inputparameternumber asc
```

	EXECUTECOUNT	STATEMENT	VALUE	ROWNUMBER	SESSIONID	CONSTRAINTID	EXECUTIONENDDATE	RUNTIME	READROWCOUNT	QUALIFIEDROWCOUNT	PHYSICALREADCOU
1	1	SELECT * from e071k where flag = :input	X	1	432985	3	2011-02-16 09:04:31.748921	55319723	25355287	1745	807
2	2	SELECT * from e071k□□where flag = ?	X	1	428941	3	2011-02-16 08:46:59.853073	514564405	25355287	1745	807
3	1	SELECT * from e071k□□where flag = ?	X	1	428941	3	2011-02-16 08:35:13.000396	511485741	25355287	1745	807
4	1	SELECT * from e071k□□where viewname = ?	abcd	1	432985	3	2011-02-16 09:20:26.29558	79547759	8953639	0	10
5	1	SELECT * from e071k□□where flag = ?	'x'	1	432985	3	2011-02-16 09:36:49.263991	79162263	8953639	0	10
6	3	SELECT * from e071k□□where flag = ?	x	1	432985	3	2011-02-16 08:53:20.571511	78785429	8953639	0	10
7	2	SELECT * from e071k where flag = :input	5	1	432985	3	2011-02-16 09:08:56.978587	78503573	8953639	0	10
8	3	SELECT * from e071k□□where viewname = ?	'abcd'	1	432985	3	2011-02-16 09:26:50.145277	77492510	8953639	0	10
9	2	SELECT * from e071k□□where viewname = ?	abcd	1	432985	3	2011-02-16 09:26:01.31631	77423436	8953639	0	10
10	21	SELECT cs.applicationinformation, cs.appli...	5500000000000000	1	433102	3	2011-02-16 09:25:50.821648	480937	5382	56	
11	3	SELECT cs.applicationinformation, cs.appli...	4700000000000000	1	431097	3	2011-02-15 17:49:04.36199	228328	5372	56	
12	4	SELECT cs.applicationinformation, cs.appli...	4700000000000000	1	431098	3	2011-02-15 17:49:30.295756	157626	5372	56	

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 53

The Command Monitor stores all runtime figures, the timestamp and the SESSIONID of a command execution matching the filter criteria in the table SYSINFO.COMMANDMONITOR. The COMMANDID references to the COMMANDID in the view SYSINFO.COMMANDSTATISTICS.

The table SYSINFO.COMMANMONITORPARAMETERS stores the input parameter values belonging to the command executions together with the COMMANDID, execution number and the parameter position. This table can become big with many logged command executions specially if the commands have many input parameters. An SQL DELETE command can remove the records from the table.

This select joins the data from the relevant system tables and shows the command executions with their parameter values sorted by the execution runtime:

```
select m.executecount, c.statement, p.value, m.*
  from sysinfo.commandstatistics c, sysinfo.commandmonitor m, sysinfo.commandmonitorparameters p
  where c.commandid = m.commandid
    and m.commandid = p.commandid
    and m.executecount = p.executecount
  order by m.runtime desc, m.executecount, p.inputparameternumber asc
```

Optimizer Statistics
Tools Executing Update Statistics

Transaction DB13, DB20, DB50

- Execution via ABAP programs with sampling
- Update of SAP WebAS alert tables

Determine tables with outdated statistics

- CALL SYSDBA.SYSCHECKSTATISTICS (<schemaname>, <threshold>)
  Example: CALL SYSDBA.SYSCHECKSTATISTICS (‚SAPWB5‘, 40)
- The threshold defines the difference between actual size and statistic in percent
- The procedure inserts relevant table names into SYSINFO.SYSUPDSTATWANTED

DBMCLI

- sql_updatestat
- sql_updatestat_per_systemtable
- auto_update_statistics

Database Studio

- Automatic Update Statistics

© SAP 2010 / MaxDB Internals Version 7.8 – Performance Analyse / Seite 54

As of version 7.5 MaxDB uses statistics data only for joins and single table selects with a result set limit like “WHERE ROWNUM <= n”.

The Update Statistics collects the table size and number of rows only if they are not available in the File Directory. This can happen with tables created by older database version than 7.6.

Update Statistics collects and calculates statistic values for primary- and secondary key columns It collects the statistic values for other columns if statistics already exists.

The database kernel inserts table names into the table SYSUPDSTATWANTED during command execution if it recognizes outdated statistics. The DBM command sql_updatestat_per_systemtable executes an Update Statistics for all tables logged in SYSUPDATESTATWANTED.

The pre-definied database procedure SYSDBA.SYSCHECKSTATISTICS checks the size of all tables in the given schema. It inserts table names into the table SYSUPDSTATWANTED if the actual table size differs to the size in the statistics by more percentage than the threshold.

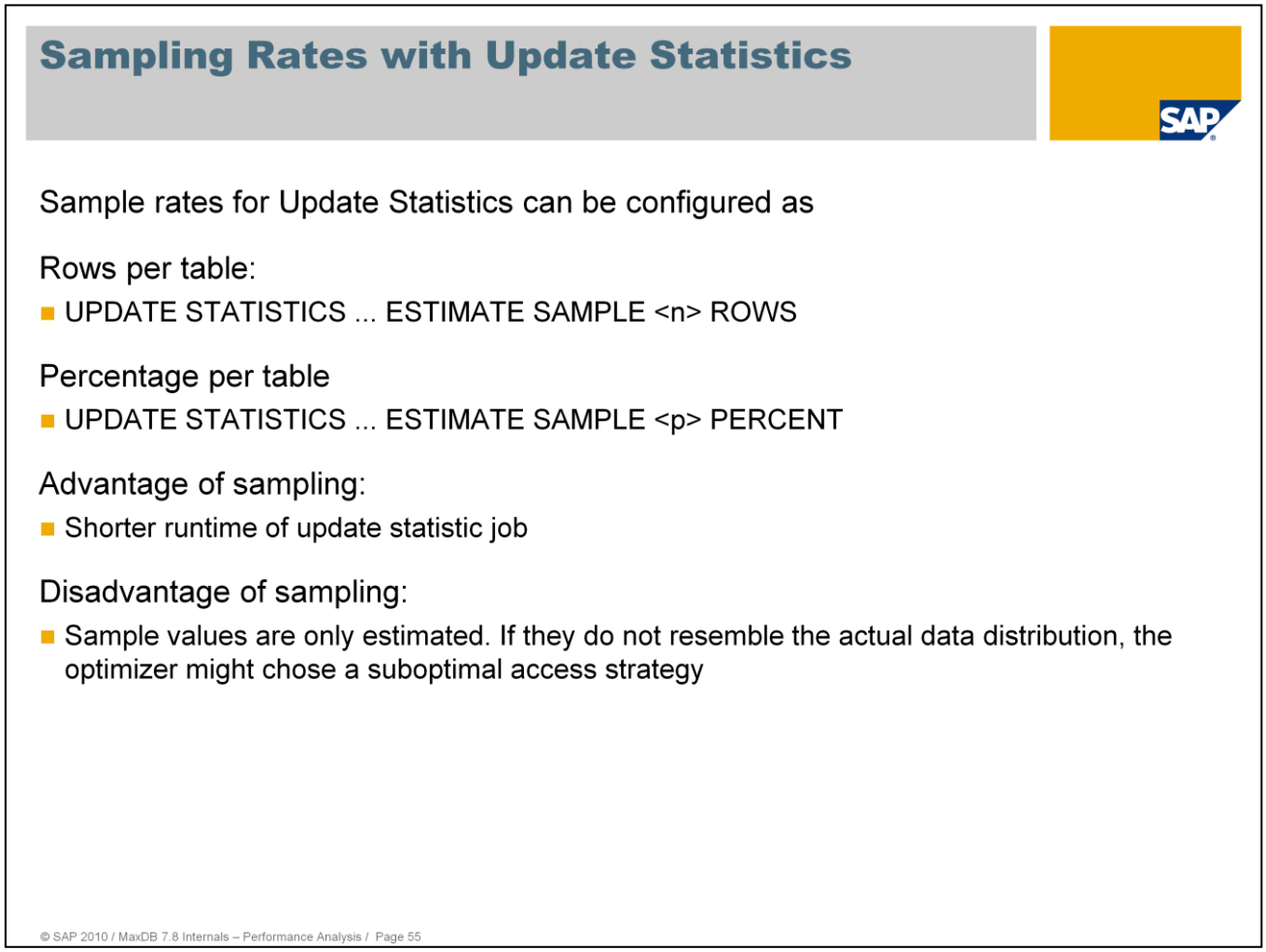
The database procedure SYSCHECKSTATISTICS checks the size of all tables in the given schema. It inserts table names into the table SYSUPDSTATWANTED if the current table size differs from the statistics by more then the given threshold in percent.

The automatic update statistics function starts a DBM event process. The database kernel sends an event to the DBM process when it inserts a table name into the table SYSUPDSTATWANTED. The DBM process than starts the command sql_updatestat_per_systemtable.

A DBM event process receives an event from the database kernel about outdated statistics

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

The Update Statistics speeds up the read of the table records by using parallel I/O orders.



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.

50,000 rows have been proven as adequate sampling quantities for column statistics.

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!