SAP® MaxDB™
Expert Session 4
Performance Optimization with SAP MaxDB

MaxDB/liveCache Development Support
Christiane Hienger
Melanie Handreck
December, 8 2009

THE BEST-RUN BUSINESSES RUN SAP™

Agenda

1. **Introduction**

2. **Database Analyzer**
   - 2.1. Parameter Check
   - 2.2. Bottleneck Analysis
3. **SQL Command Analysis**
   - 3.1. Command Monitor
   - 3.2. Update Statistics
   - 3.3. Resource Monitor
4. **Questions & Answers**

This is the forth SAP MaxDB Expert Session and this session covers the topic database performance analysis.

Analyzing database performance is a complex subject. This session gives an overview about the SAP MaxDB performance analysis tools. Further Expert Sessions in, which more detailed information about database performance analysis will be provided, will follow.

In today's session we show you with some examples how to execute a performance analysis using the MaxDB tools. The usage of the SAP application performance tools like transaction ST03 or ST03N will not be part of this session.

The presentation is based on database EXPERTDB with MaxDB version 7.7.06, which was created during the previous Expert Sessions.

Our focus will be the usage of *Database Analyzer* including the *Parameter Checker* and the SQL performance analysis using the *SAP MaxDB Command Monitor*. We will also discuss the impact of update statistics (including the file directory counters and the eval functionality).

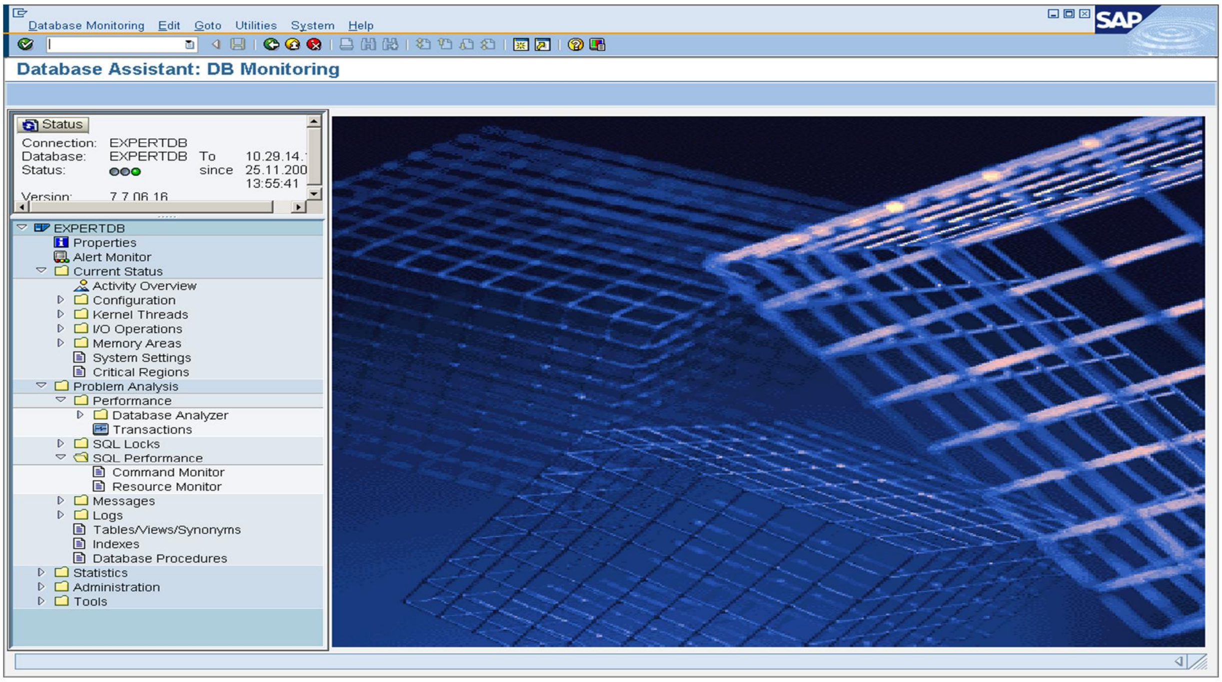
Objectives

After this presentation, you will be able to:

- Use Database Analyzer for database performance analysis
- Check parameter settings with DB-Analyzer Parameter check tool
- Use Command Monitor to analyze long running SQL commands
- Know the importance of Update Statistics
- Use Resource Monitor to analyze the workload of SQL commands

In this chapter, you will learn how to use the database performance analysis tools *Database Analyzer*, *Command Monitor* and *Resource Monitor*. You will know the importance of update statistics.

Introduction: Transaction DB50

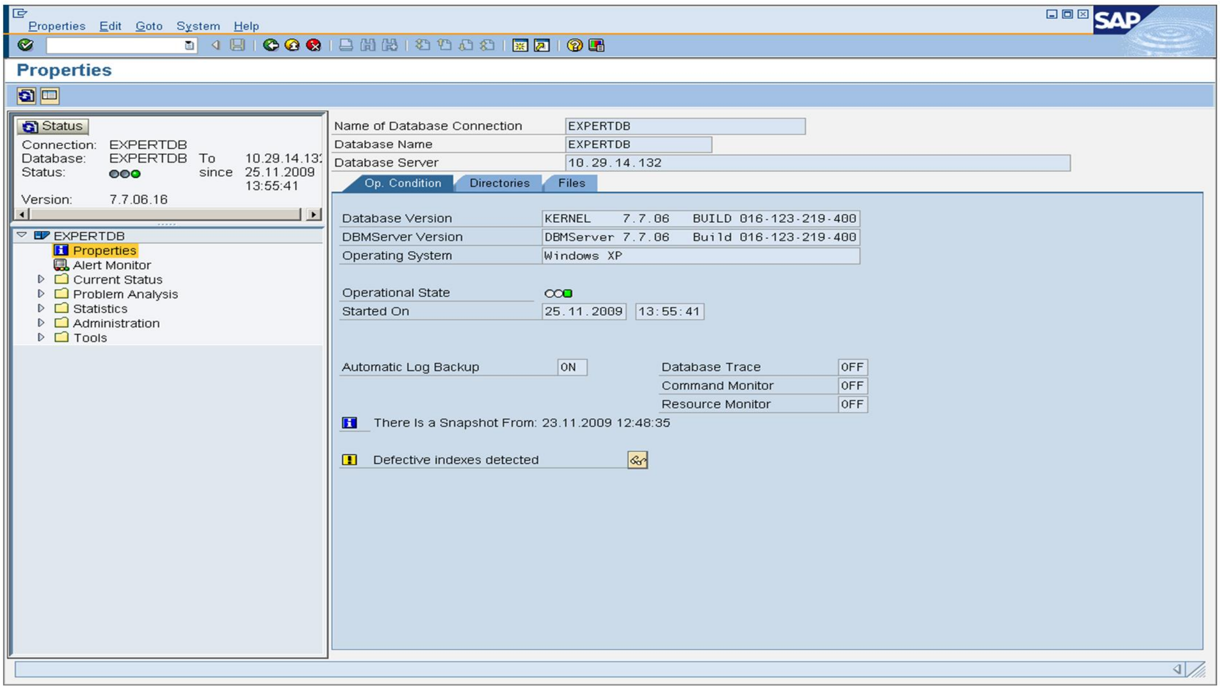


The CCMS transactions DBACockpit or DB50 are used for MaxDB database performance analysis. If you want to analyse a remote database from your SAP system you must use transaction DB59 to integrate that remote database, before you can use transaction DB50 for the analysis.

Starting with the *Database Assistant* (transaction DB50) you can use the MaxDB performance analysis tools *Database Analyzer, Command Monitor* and *Resource Monitor* .

Only the Database Analyzer parameter check function cannot be executed directly in the SAP system using transaction DB50. You need to run this tool on the command line of the operating system.

DB50: Properties



When you start a performance analysis you should first have a short look at the *Properties* of your system. Here you can check when the database was restarted.

To carry out a global performance analysis, the system needs to be balanced. It doesn't make sense to perform a performance analysis directly after the database was restarted and all data still needs to be read from the hard disks.
During the restart of an OLTP or BW database data pages are not loaded into the IO Buffer Cache implicitly like in a liveCache environment.

On the properties screen you get also the information if there exist any corrupted indexes in your database. Those indexes are blocked (we call it *marked as bad*). They cannot be used anymore by the optimizer. This can be one reason for bad response times of individual transactions which appear without changing anything in the system.

In this overview you will also find the information if the command and resource monitor tools are active.

Parameter Check with Database Analyzer: Prerequisites

- SAP Note 1111426 Parameter check for liveCache/MaxDB instances
  - Attachments
    - DbanalyzerParamCheck.SAR
- Download the attachment into a temporary directory, e.g. /tmp
- Unpack DbanalyzerParamCheck.SAR
  - sapcar –xvf DbanalyzerParamCheck.SAR

→ dbanalyzer_InstanceParameterCheck.cfg

As bad performance could be caused by wrong parameter settings, you should check the database configuration first.

SAP MaxDB offers a check tool for MaxDB kernel parameter settings. This check is embedded into the *Database Analyzer*. The parameter check tool is used to check whether the configuration of your liveCache, MaxDB, OneDB or BW system corresponds to the current SAP recommendations.
In general the parameter recommendations which are described in the MaxDB parameter notes (MaxDB Version 7.7: 1004886, MaxDB Version 7.8: 1308217) are checked.

The parameter check should be executed after each upgrade to a new liveCache/MaxDB version. Different recommendations may be relevant for different database versions.

The parameter check tool uses a special *Database Analyzer* configuration file. This special configuration file is attached to note 1111426. As this file is regulary updated, you must download it again before each check. This file can be stored in a temporary directory – e.g. /tmp

Use *sapcar –xvf DbanalyzerParamCheck.sar* to extract the configuration file *dbanalyzer_instanceParameterCheck.cfg*

Do not replace the original database analyzer config file with the new one!

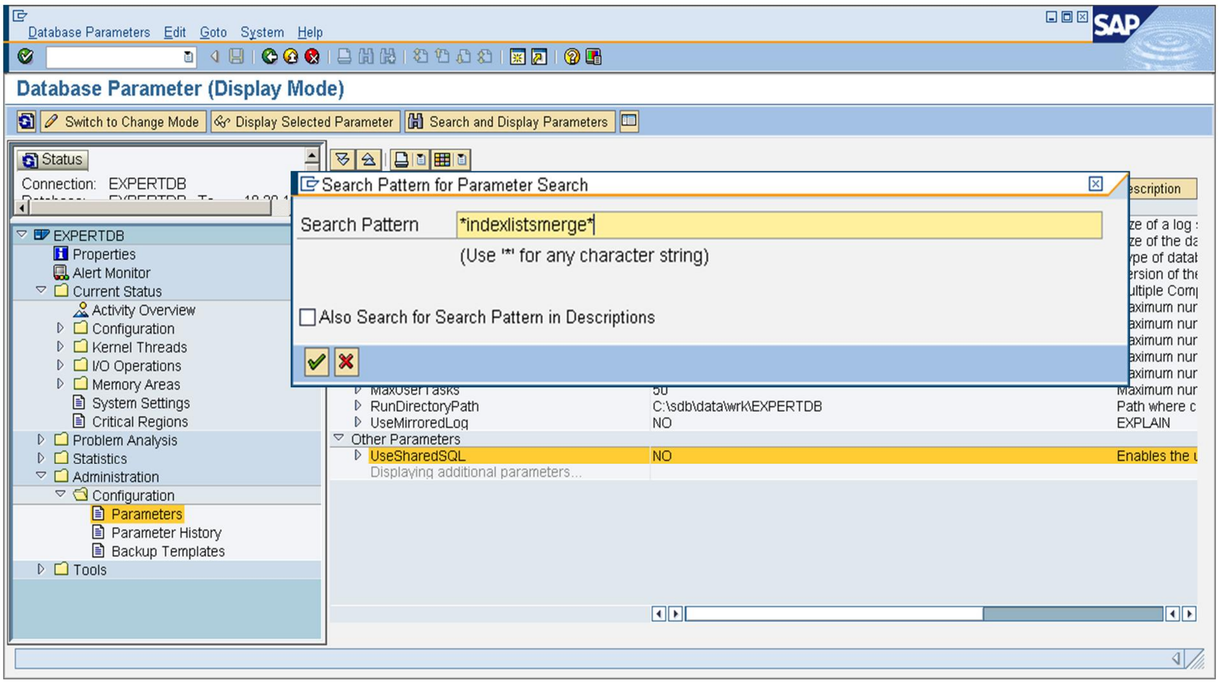
Parameter Check with Database Analyzer

```
C:\WINDOWS\system32\cmd.exe
(0.13 MB)
*  I  Number of data volumes 1, usable size 9998 pages (0.08 GB), used size 597
pages (0 GB), filling level 5%
*  I
*  I  General checks:
*  I  ---------------
* W1  Recommended value for parameter 'IndexlistsMergeThreshold' is 0, current v
alue is 500
*  I
*  I  If instance EXPERTDB is used for Data Warehouse applications, the followin
g recommendations are of interest:
*  I  ------------------------------------------------------------------------
--------------------------------
* W1  Recommended value for parameter 'HashJoinTotalMemorySize' is 24000, curren
t value is 5120
* W1  Recommended value for parameter 'HashJoinSingleTableMemorySize' is 4000, c
urrent value is 512
* W1  Recommended value for parameter 'UseDataCacheScanOptimization' is YES, cur
rent value is NO

===== #1          at 2009-10-21 13:47:56
* OK


C:\tmp\param_check>_
```


The database instance must be in operational state ONLINE when you start the parameter check tool. Perform the automatic check as SYSDBA user (e.g. dbadmin)
dbanalyzer –d EXPERTDB –u dbadmin,secret –f c:\tmp\dbanalyzer_instanceParametercheck.cfg -o c:\tmp\param_check -i –c 1 –t 1,1 –n <server>
-i the output directory will be cleaned up
-c output will be send to screen as well
-t only 1 snapshot in an interval of one second

Analyze the screen output or the file */tmp/param_check/<YYYYMMDD>/DBAN.prt*.
Important are all messages that are marked with "* W1 to * W3"
The following checks are executed:
- general parameters
- parameters which influence the I/O performance
- optimizer parameters
- special liveCache parameters
- additional checks
    - do corrupt indexes exist?
    - is the database kernel trace activated?
    - do tables exist which do not have any file directory counters?
    - is logging activated and autooverwrite deactivated?
    - does the size of the IO Buffer Cache correspond to the SAP recommendation, which is
      2% of the configured volume size for UNICODE systems and
      1% for NON-UNICODE systems?

DB50: Displaying Parameters

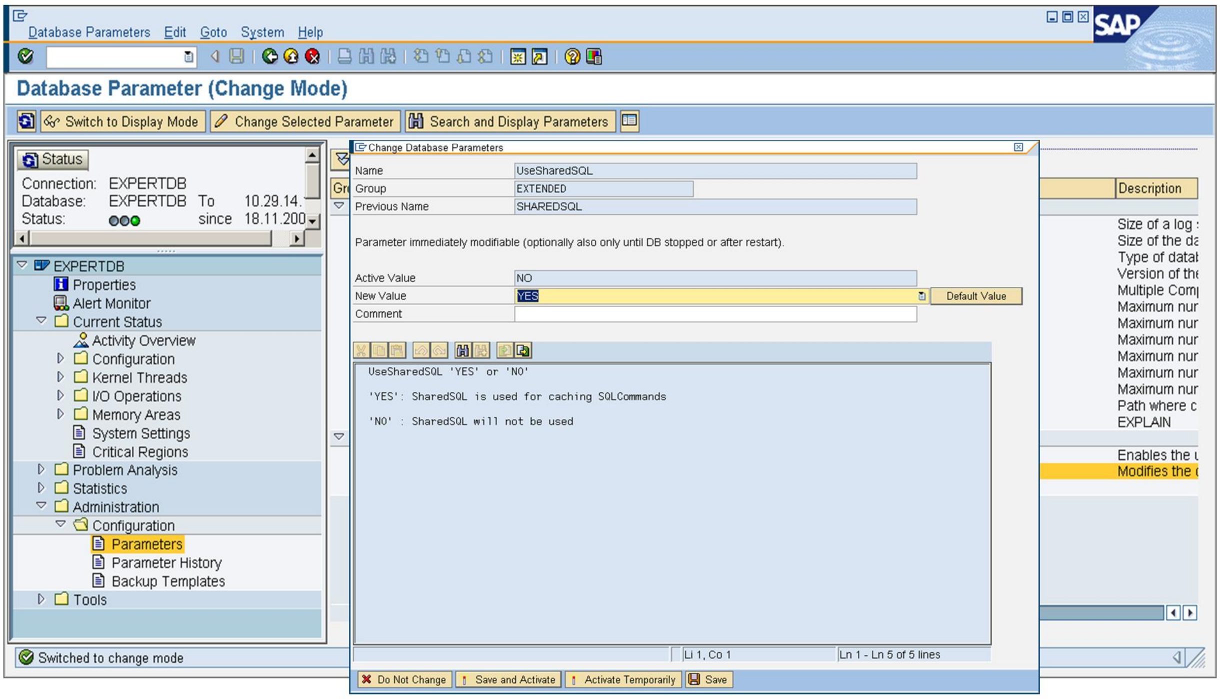


If there are parameter settings which differ from the recommendation you can use Database Studio, DBMCLI or transaction DB50 to change the parameter setting.

Change the parameter setting of Parameter UseSharedSQL from NO to YES, because this parameter is important for monitoring the system using the *Resource Monitor*.
Also change the parameter IndexlistsMergeThreshold to the recommended value 0.

A restart of the database is only necessary if the parameter is not online changeable.

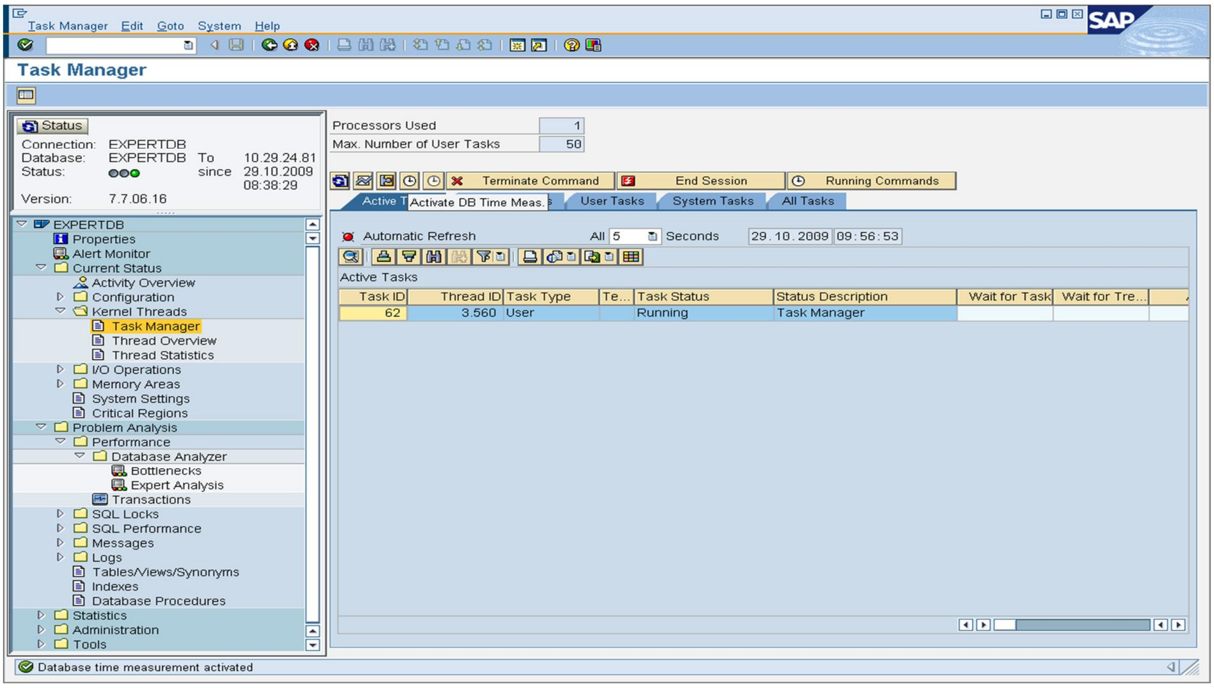
DB50: Changing Parameter Values



With *Search and Display* you can list the parameters you want to change. Choose *Switch to Change Mode* to be able to change the parameter value.
With *Save and Activate* the parameter will be changed online and the new value will be stored in the parameter file. So the new value will be active after the restart of the database as well. If you use *Activate Temporarily* the parameter will be changed online but the new value will not be stored in the parameter file. The parameter change will be lost after a restart of the database.

Adapt all parameters in this way, which have been logged by the parameter checker.

DB50: Activating the Time Measurement



The database analysis tools need to have detailed information about the runtime several activities need - e.g. how long does an I/O to/from disk take or how long are wait times.

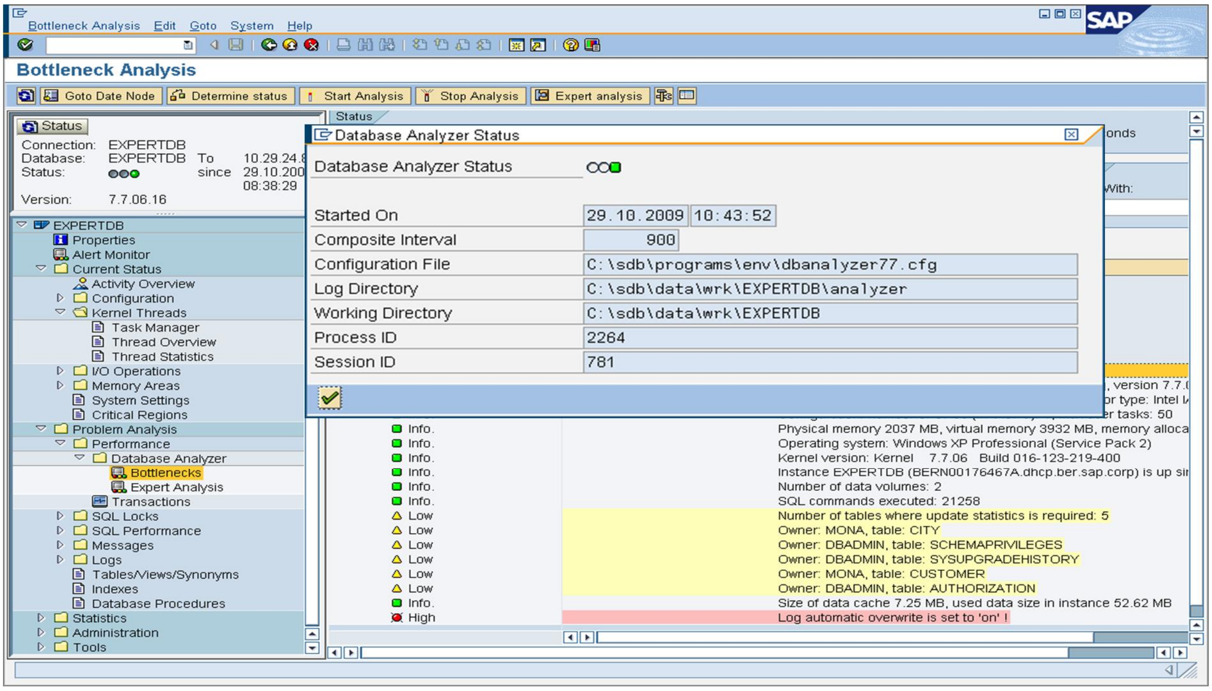
To get detailed information about used time the database internal time measurement has to be activated explicitly by using the *Task Manager* in transaction DB50. To do this go to *Current Status –> Kernel Threads –> Task Manager* and choose *Activate DBTime Measurement*.

It is recommended to switch on the time measurement only during a performance analysis and not by default. When the database is restarted, the time measurement is switched off implicitely. It can also be deactivated explicitely in transaction DB50 *Current Status –> Kernel Threads –> Task Manager -> Deactivate DB Time Measurement*.

As of MaxDB Version 7.7. the I/O time measurement is always active. The time measurement for Wait (suspend) situations and for command runtime which should be checked via the *Database Analyzer* tool (receive/reply times) has still to be activated explicitly (using the same button as described before).

You can activate the time measurement as well using DBMCLI :
db_cons time enable

Database Analyzer: Status



When you notice bad response times of your SAP system and you suspect that the problem is caused by the database use the *Database Analyzer* to get closer to the reason for the performance problem. The *Database Analyzer* is completely integrated into transaction DB50: *Problem Analysis –> Performance –> Database Analyzer -> Bottlenecks.*

The *Database Analyzer* is a tool for long-term analysis. It is automatically started when you start your SAP System and it should always be active on productive systems.
If you want to see the current status of *Database Analyzer* in transaction DB50, choose *Determine Status* in the *Database Analyzer* menu.

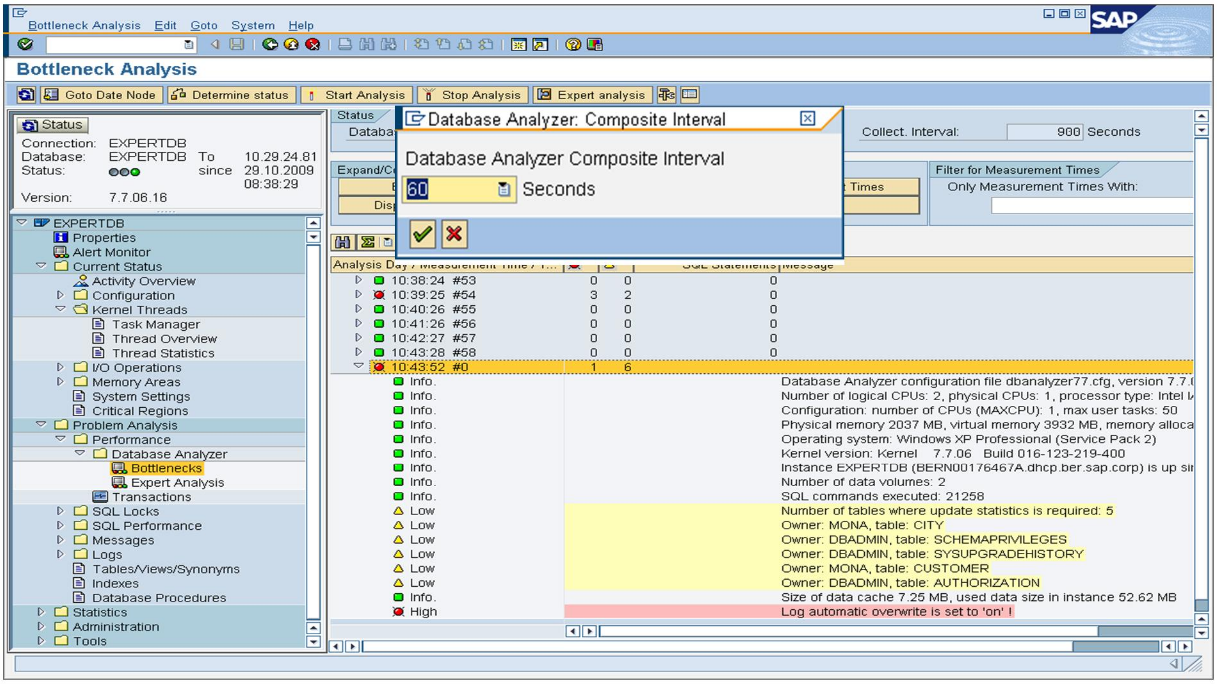
In the *dbanalyzer<version>.cfg* configuration file, SAP defines rules and commands which are used by *Database Ana*lyzer when determining performance-relevant information.

The *Database Analyzer* config file is version specific and part of the independent software. The *Database Analyzer* obtains the performance relevant information either directly from system tables in the database, or calculates it from the data in those tables.

The configuration file also describes the four classes of messages. Information marked with an I or INFO and three levels of warnings: W1 to W3 - warning levels 1 to 3 with low, medium and high priority.

All messages are logged in the DBAN.prt log file.

Starting the Database Analyzer



If problems with the *Database Analyzer* program occur, the system writes messages to the following log files:
DBAN.err: Error file
DBAN.inf: Runtime information of the *Database Analyzer* that was most recently started.

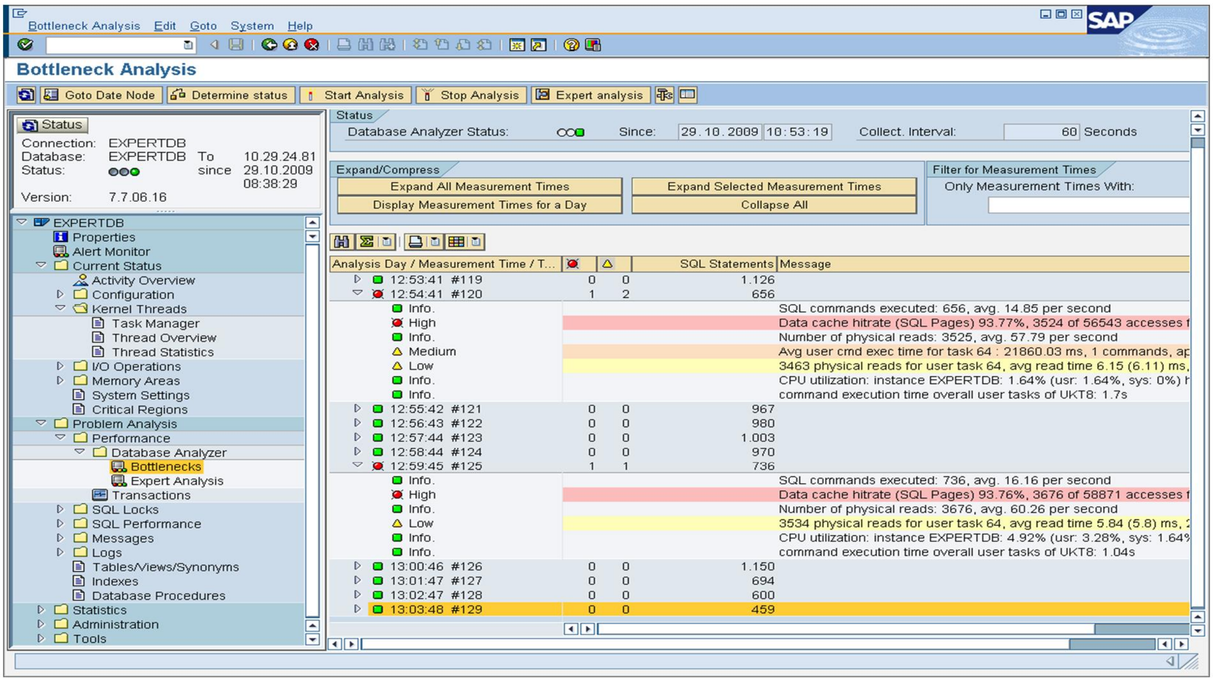
The default interval in which the *Database Analyzer* is collecting the data is 900 seconds (= 15 minutes). When you are planning to do a detailed performance analysis it is recommended to change that interval. To do so you need to stop and restart the *Database Analyzer* using the buttons *Stop Analysis* and *Start Analysis* .

When you start the *Database Analyzer* you can choose the new interval – it is recommended to use 60 seconds during a short term analysis.
In the first interval after starting the *Database Analyzer* and at the beginning of each day the *Database Analyzer* logs some general system information:

- version information about the *Database Analyzer* configuration file
- hardware information
- database configuration information
- information about required statistics update

The *Database Analyzer* logs all information and warnings in log files in directory *analyzer*, which is a subdirectory of the run directory of the database. For each day the *Database Analyzer* is running a new subdirectory with the current date is created. This makes it easy to analyze and compare the *Database Analyzer* data of different days.

Database Analyzer: Bottleneck Analysis (1)

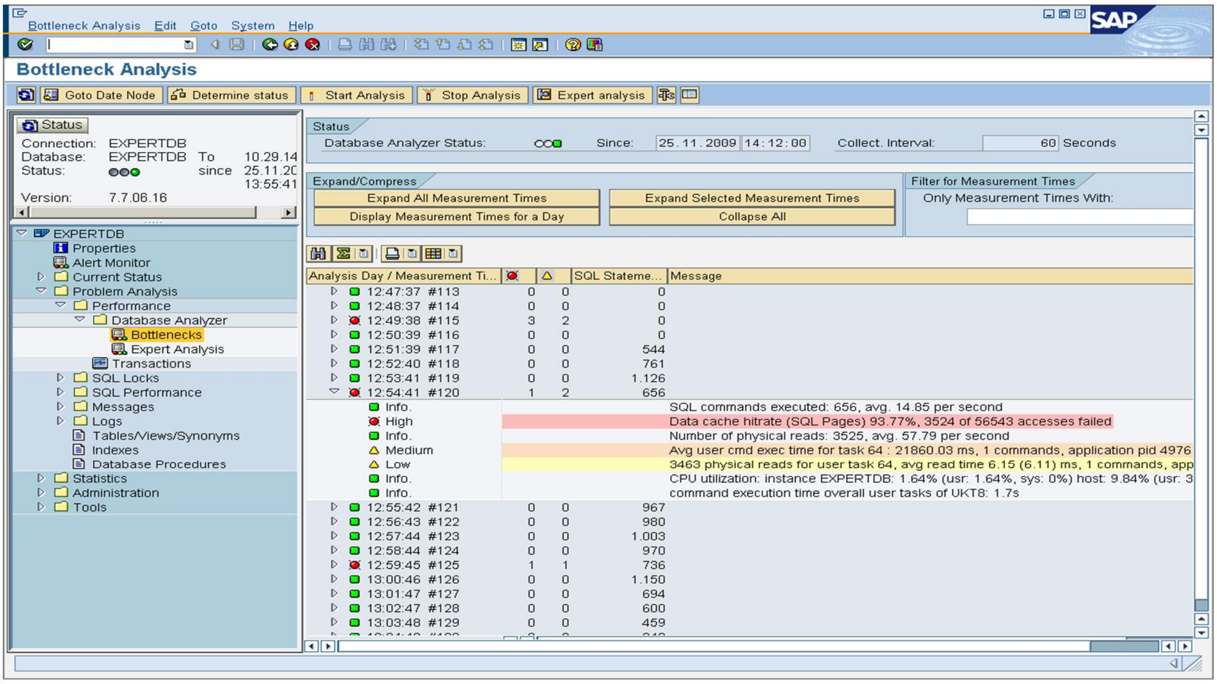


Notice: not every message which is marked as red is a critical one. It depends on the workload (number of SQL-Statements) of the database if a message should be checked in detail.

In this example we can see several messages:

Message: *High* : Data cache hitrate (SQL Pages) 93.76%, 3676 of 58871 accesses failed
Explanation: The hit rate when accessing the data cache is too low. Data cache hit rates that are lower than 99% over a fifteen-minute average should be avoided. Lower hit rates may occur for short periods of time, for example, when objects are accessed for the first time. Low hit rate for longer periods of time indicate a performance problem which has to be analyzed in detail.
Next Steps: Check the configuration of the database instance: Kernel parameter *CacheMemorySize* should be >= 1% of the total data volume size.
However, increasing the Data Cache size is often not the best solution, if individual SQL statements cause a high percentage of the total logical and physical read activity. Increasing the cache just moves the load from the hard disk to the CPU, even though, for example, an additional index could turn a read-intensive table scan into a quick direct access.

Database Analyzer: Bottleneck Analysis (2)

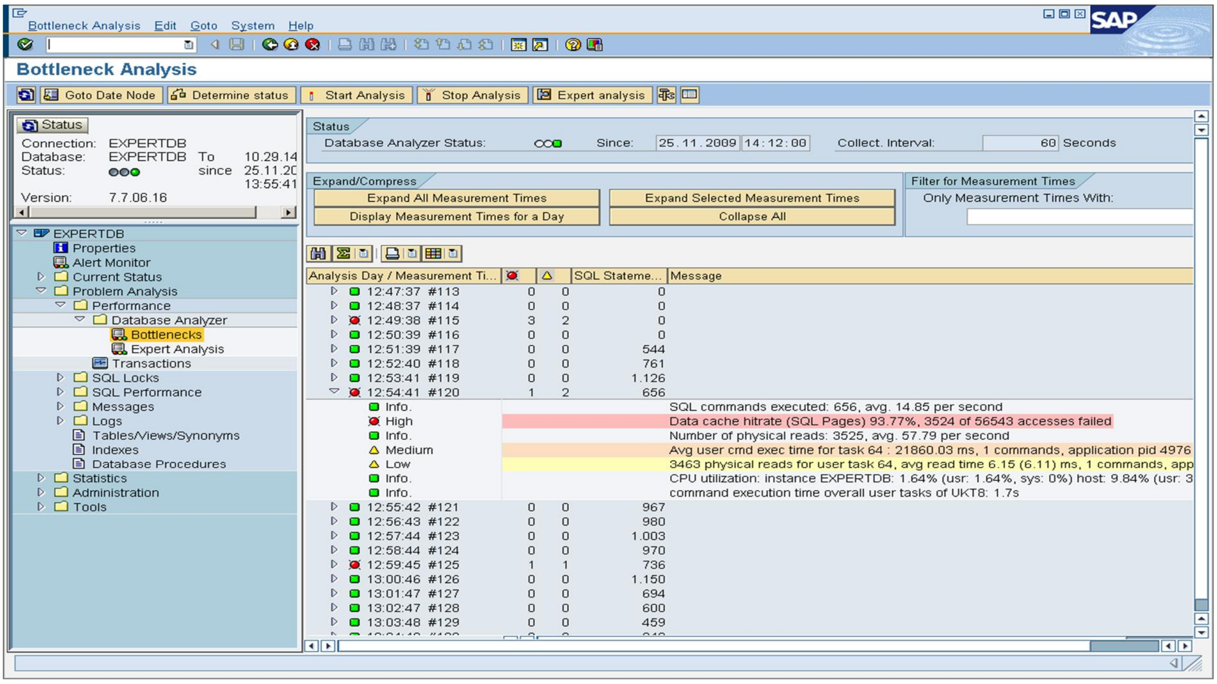


Message: *Medium* : Avg user cmd exec time for task 64: 21860,03 ms, 1 commands, application pid 4976
Explanation: The time needed by the specified user task to execute the statements is very long. You see the average execution time of a statement, the number of executed statements, and the process ID of the corresponding application process.
Next Steps: Whether this counts as a bottleneck depends on the application structure.

Mass statements in background processes can often cause long runtimes. Additionally, locks on SQL objects, **physical reads and writes**, or dispatching caused by the prioritization of other tasks, can cause internal kernel wait situations that increase runtimes.

We see a lot of physical reads as well in this example. So we will focus on that to find the reason for the *Database Analyzer* messages.

Database Analyzer: Bottleneck Analysis (3)



Message: *low:* 3463 physical reads for user task 64, avg read time 5.84 (5.8) ms, 23 commands, application pid 4996
Explanation: A large number of physical reads are taking place on the volumes of the database as the data requested by the applications was not found in the data cache.
If a table is accessed for the first time, or if it has not been used for a long time, and was therefore displaced from the data cache, then this situation is not a problem.

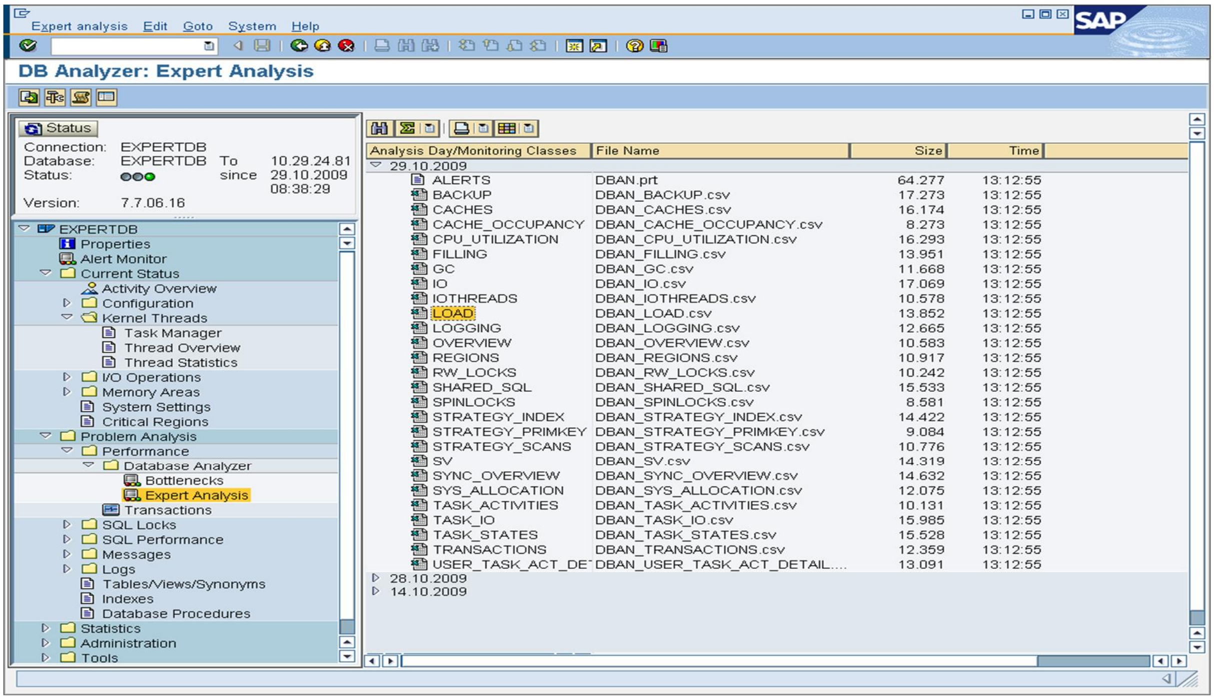
However, if this does not explain the read activity, you should check the hit rate for the data cache, and increase the size of the data cache, if necessary.

Furthermore make sure that the SQL statements specified by the application do not read significantly more data than is necessary for processing, because of poor search strategies, for example.

If the special database parameter UseDataCacheScanOptimization has the value NO, then table scans use only 10% of the data cache for caching the table. This means that the table cannot be held completely in the data cache, and the next scan has to read it from the disks again. If the parameter is set to YES, the complete cache can be used for scans.

In the database analyzer manual you will find a lot more information about the Database Analyzer messages.

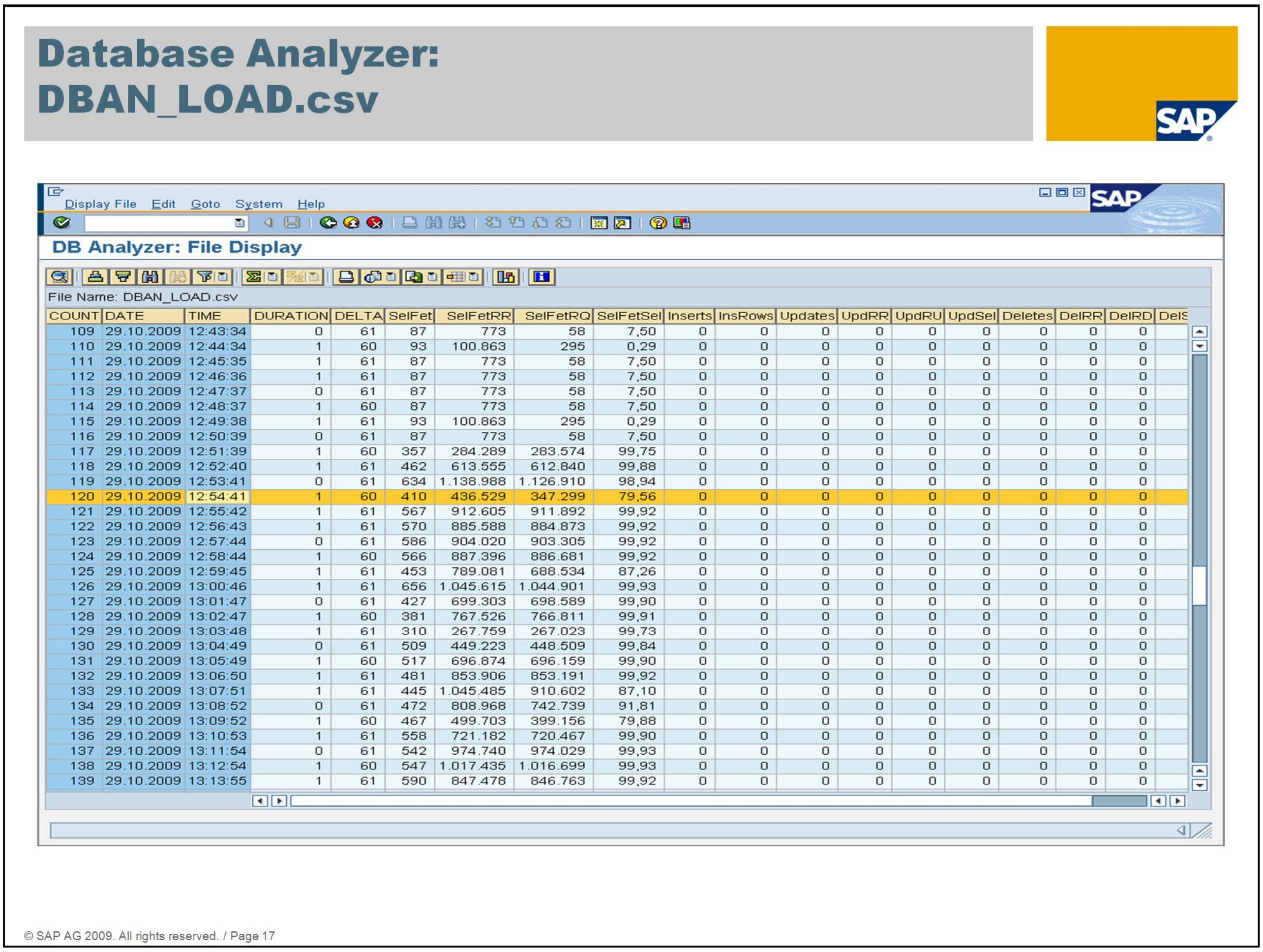
Database Analyzer: Expert Analysis



Let’s have a short look into the Database Analyzer *Expert Analysis*.
The expert analysis provides access to all available database analyzer log files. You can get detailed information about the database activities and used resources.

Expert Analysis is used by MaxDB Experts in Development Support to analyze complex database performance problems.

Let’s have a short look into two of these files: LOAD and CACHES



COUNT	DATE	TIME	DURATION	DELTA	SelFet	SelFetRR	SelFetRQ	SelFetSel	Inserts	InsRows	Updates	UpdRR	UpdRU	UpdSel	Deletes	DelRR	DelRD	DelS
109	29.10.2009	12:43:34	0	61	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
110	29.10.2009	12:44:34	1	60	93	100.863	295	0,29	0	0	0	0	0	0	0	0	0	
111	29.10.2009	12:45:35	1	61	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
112	29.10.2009	12:46:36	1	61	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
113	29.10.2009	12:47:37	0	61	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
114	29.10.2009	12:48:37	1	60	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
115	29.10.2009	12:49:38	1	61	93	100.863	295	0,29	0	0	0	0	0	0	0	0	0	
116	29.10.2009	12:50:39	0	61	87	773	58	7,50	0	0	0	0	0	0	0	0	0	
117	29.10.2009	12:51:39	1	60	357	284.289	283.574	99,75	0	0	0	0	0	0	0	0	0	
118	29.10.2009	12:52:40	1	61	462	613.555	612.840	99,88	0	0	0	0	0	0	0	0	0	
119	29.10.2009	12:53:41	0	61	634	1.138.988	1.126.910	98,94	0	0	0	0	0	0	0	0	0	
120	29.10.2009	12:54:41	1	60	410	436.529	347.299	79,56	0	0	0	0	0	0	0	0	0	
121	29.10.2009	12:55:42	1	61	567	912.605	911.892	99,92	0	0	0	0	0	0	0	0	0	
122	29.10.2009	12:56:43	1	61	570	885.588	884.873	99,92	0	0	0	0	0	0	0	0	0	
123	29.10.2009	12:57:44	0	61	586	904.020	903.305	99,92	0	0	0	0	0	0	0	0	0	
124	29.10.2009	12:58:44	1	60	566	887.396	886.681	99,92	0	0	0	0	0	0	0	0	0	
125	29.10.2009	12:59:45	1	61	453	789.081	688.534	87,26	0	0	0	0	0	0	0	0	0	
126	29.10.2009	13:00:46	1	61	656	1.045.615	1.044.901	99,93	0	0	0	0	0	0	0	0	0	
127	29.10.2009	13:01:47	0	61	427	699.303	698.589	99,90	0	0	0	0	0	0	0	0	0	
128	29.10.2009	13:02:47	1	60	381	767.526	766.811	99,91	0	0	0	0	0	0	0	0	0	
129	29.10.2009	13:03:48	1	61	310	267.759	267.023	99,73	0	0	0	0	0	0	0	0	0	
130	29.10.2009	13:04:49	0	61	509	449.223	448.509	99,84	0	0	0	0	0	0	0	0	0	
131	29.10.2009	13:05:49	1	60	517	696.874	696.159	99,90	0	0	0	0	0	0	0	0	0	
132	29.10.2009	13:06:50	1	61	481	853.906	853.191	99,92	0	0	0	0	0	0	0	0	0	
133	29.10.2009	13:07:51	1	61	445	1.045.485	910.602	87,10	0	0	0	0	0	0	0	0	0	
134	29.10.2009	13:08:52	0	61	472	808.968	742.739	91,81	0	0	0	0	0	0	0	0	0	
135	29.10.2009	13:09:52	1	60	467	499.703	399.156	79,88	0	0	0	0	0	0	0	0	0	
136	29.10.2009	13:10:53	1	61	558	721.182	720.467	99,90	0	0	0	0	0	0	0	0	0	
137	29.10.2009	13:11:54	0	61	542	974.740	974.029	99,93	0	0	0	0	0	0	0	0	0	
138	29.10.2009	13:12:54	1	60	547	1.017.435	1.016.699	99,93	0	0	0	0	0	0	0	0	0	
139	29.10.2009	13:13:55	1	61	590	847.478	846.763	99,92	0	0	0	0	0	0	0	0	0	

We will check the log file DBAN_LOAD.csv (LOAD) .

Here you can get an overview of the SQL activities on your database.

You get information about accesses and selectivity of SELECT, FETCH, INSERT, UPDATE, and DELETE statements.

In our EXPERTDB we do not have any Inserts, Updates and deletes.
The activities are caused by Select commands.
We get detailed information about:
- the number of selects and fetches (410),
- the number of rows read for those selects and fetches (436529)
- the number of rows qualified of those number of rows read (347299) -> nearly 90.000 less

Why we need to read so much more data than needed for the result?

Continue the performance analysis on SQL command level using the tool *Command Monitor*.

Database Analyzer: DBAN_CACHES.csv

Display File Edit Goto System Help

DB Analyzer: File Display

File Name: DBAN_CACHES.csv

COUNT	DATE	TIME	DURATION	DELTA	DC_Acc	DC_Succ	DC_Fails	DC_Hit	Cat_Acc	Cat_Succ	Cat_Fails	Cat_Hit	Parse_Activity	Hist_Acc	Hist_Succ	Hist_Hit
96	29.10.2009	12:30:25	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,83	0	0	0
97	29.10.2009	12:31:26	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
98	29.10.2009	12:32:26	1	60	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
99	29.10.2009	12:33:27	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
100	29.10.2009	12:34:28	0	61	53.613	50.107	3.506	93,46	404	295	109	73,02	10,69	0	0	0
101	29.10.2009	12:35:28	1	60	35.402	35.402	0	100,00	395	288	107	72,91	10,83	0	0	0
102	29.10.2009	12:36:29	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
103	29.10.2009	12:37:30	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
104	29.10.2009	12:38:30	1	60	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
105	29.10.2009	12:39:31	1	61	53.585	50.086	3.499	93,47	404	295	109	73,02	10,69	0	0	0
106	29.10.2009	12:40:32	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,83	0	0	0
107	29.10.2009	12:41:32	1	60	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
108	29.10.2009	12:42:33	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
109	29.10.2009	12:43:34	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
110	29.10.2009	12:44:34	1	60	53.582	50.084	3.498	93,47	404	295	109	73,02	10,69	0	0	0
111	29.10.2009	12:45:35	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,83	0	0	0
112	29.10.2009	12:46:36	1	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
113	29.10.2009	12:47:37	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
114	29.10.2009	12:48:37	1	60	35.402	35.402	0	100,00	395	288	107	72,91	10,69	0	0	0
115	29.10.2009	12:49:38	1	61	53.601	50.098	3.503	93,46	404	295	109	73,02	10,69	0	0	0
116	29.10.2009	12:50:39	0	61	35.402	35.402	0	100,00	395	288	107	72,91	10,83	0	0	0
117	29.10.2009	12:51:39	1	60	38.323	38.310	13	99,97	1.210	1.097	113	90,66	10,69	0	0	0
118	29.10.2009	12:52:40	1	61	40.345	40.328	17	99,96	1.530	1.422	108	92,94	3,65	0	0	0
119	29.10.2009	12:53:41	0	61	44.956	44.901	55	99,88	2.013	1.893	120	94,04	2,87	0	0	0
120	29.10.2009	12:54:41	1	60	56.543	53.019	3.524	93,77	1.361	1.254	107	92,14	2,91	0	0	0
121	29.10.2009	12:55:42	1	61	42.235	42.226	9	99,98	1.827	1.720	107	94,14	3,09	0	0	0
122	29.10.2009	12:56:43	1	61	42.175	42.175	0	100,00	1.846	1.739	107	94,20	2,38	0	0	0
123	29.10.2009	12:57:44	0	61	42.329	42.329	0	100,00	1.884	1.777	107	94,32	2,28	0	0	0
124	29.10.2009	12:58:44	1	60	42.149	42.149	0	100,00	1.833	1.726	107	94,16	2,23	0	0	0
125	29.10.2009	12:59:45	1	61	58.871	55.195	3.676	93,76	1.479	1.372	107	92,77	2,30	0	0	0
126	29.10.2009	13:00:46	1	61	43.379	43.359	20	99,95	2.103	1.996	107	94,91	2,84	0	0	0
127	29.10.2009	13:01:47	0	61	40.470	40.469	1	100,00	1.417	1.310	107	92,45	2,00	0	0	0

Remember: We also got a message about very bad Data Cache hit rate.

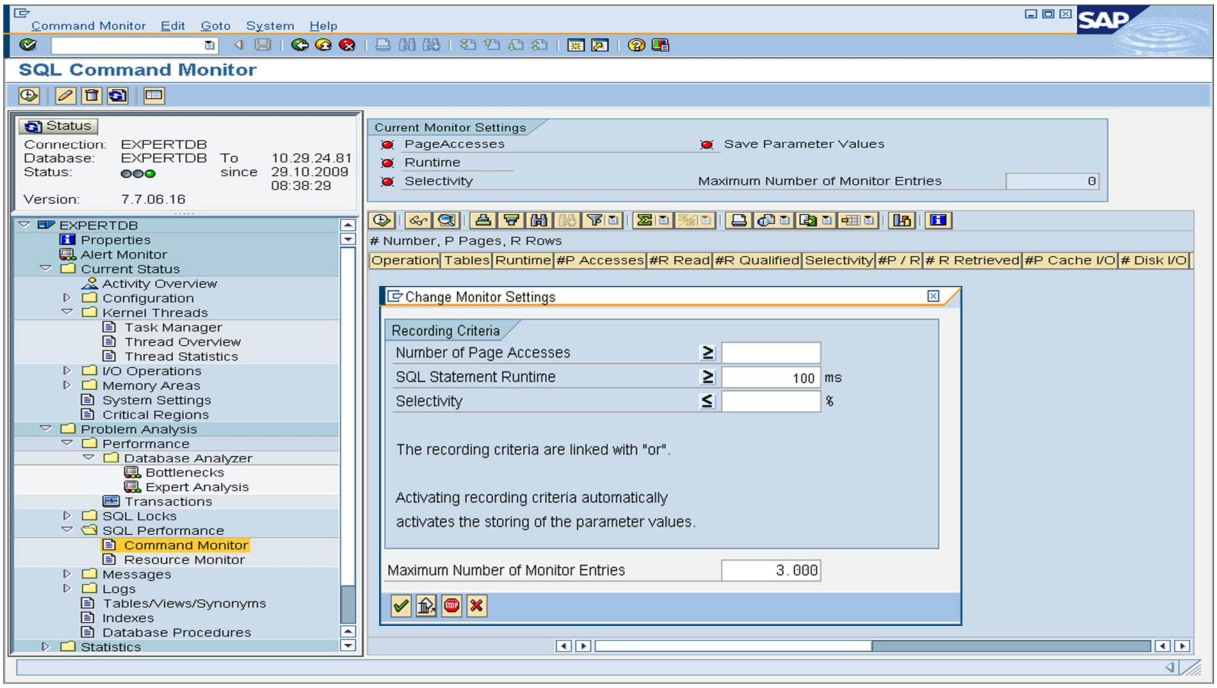
You get more information about the details of the caches with a closer look into file DBAN_CACHES.csv (CACHES). Successful and unsuccessful accesses to the MaxDB caches, and also hit rates are listed here.
What you should know about the several caches is that the data cache (DC) is the most important cache listed here. All application data (tables/indexes) which is processed must be located in the DC. If the pages are not available in the DC the data is read from the disk which causes physical I/O.

You don't need to focus on the catalog cache. The catalog cache is used for meta data of tables, indexes and SQL commands, e.g. access plans.

When you access the MaxDB catalog information e.g. the system tables this information is stored in the catalog cache as well.
When the command monitor is switched on all executed SQL commands will be parsed again. The Catalog Cache hitrate decreases.
Or if you are working with transaction DB50 – doing a lot of catalog scans caused by system table accesses - the catalog cache rate decreases.
This should not have a negative impact to your system performance.
We focus on the data Cache hitrate (DC_HIT). A database which is running fine should have a data cache hitrate of more than 98%.

We see here in the EXPERTDB that the data cache hitrate is bad (93%) from time to time.

Activating the Command Monitor



Use the *Command Monitor* if the analysis of the database bottlenecks (Database Analyzer) reports inefficient database accesses. With this tool, you can identify long running SQL statements in a systematic way.

The tool should be used for short-term analysis since the number of SQL statements logged is limited. You can restrict the number of SQL statements logged by specifying logging criteria. This enables you to concentrate on long-running SQL statements.

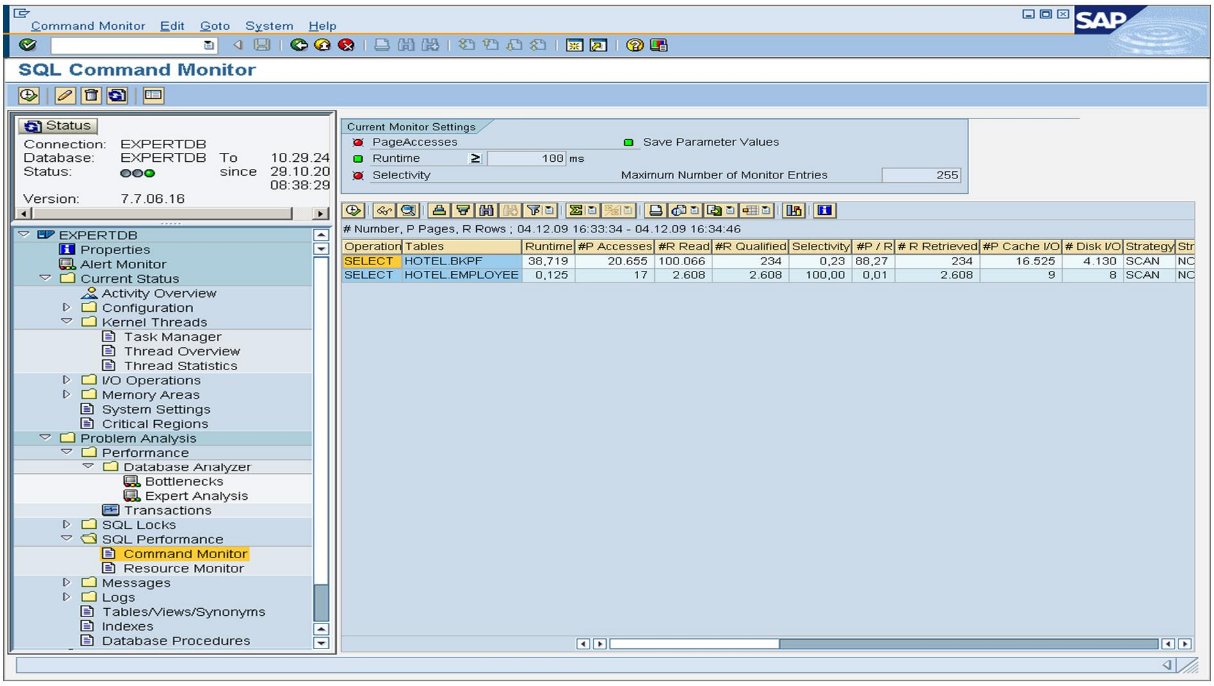
Enter the desired recording criteria in the *Change Monitor Settings* display. The recording criteria determine which SQL statements are to be logged in the command monitor tables. If one of the set criteria is full filled the SQL statement is logged in the command monitor.

Number of Page Accesses: A SQL statement is logged if the number of specified page accesses is exceeded.
SQL Statement Runtime: A SQL statement is logged if the specified runtime is exceeded.
Selectivity: A SQL statement is logged in the command monitor tables if the ratio of qualified records to read records falls below the specified percentage.

SAP provides default values for these thresholds. Use *Apply SAP default settings* and then adapt or confirm these values.

In our Expert Session we will use only the runtime with >= 100 ms.

Command Monitor: Long Running SQL Statements



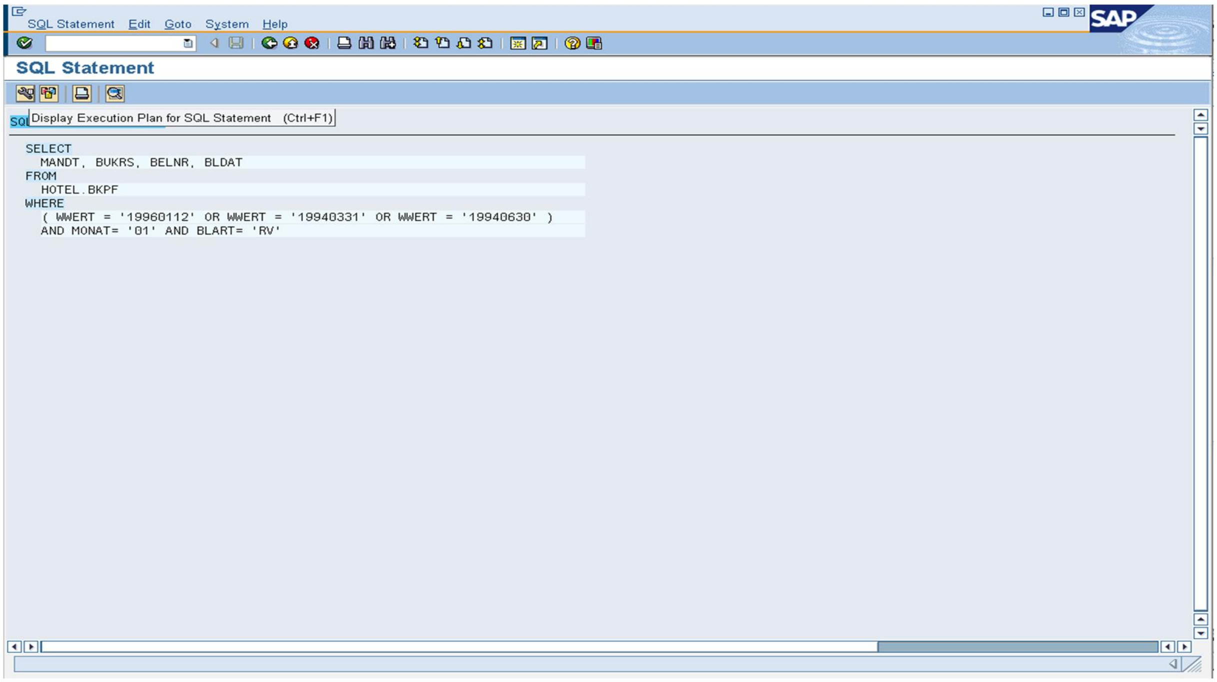
After the command monitor has been switched on all SQL statements which are newly executed and which violate one of the thresholds are logged in the monitoring tables (up to max. 3000 statements).

The list of SQL statements is sorted by the runtime (default). Statements with the longest runtime are on top.

We start the analysis with those commands which take the longest runtime (in msec). In our example this is a SELECT statement on table HOTEL.BKPF.
This select took more than 38 seconds, accessed more than 20.000 pages and more than 100.000 records were read but only 234 were qualified for the result. This explains the Database Analyzer information about bad selectivity.
The next thing we need to check is, why the database needs to read so many records to get the result - maybe there is an index missing.

With a double click on the SQL statement we get detail information about this statement.

Command Monitor: Detailed SQL Statement



The complete SQL Statement is displayed.

Display Execution Plan for an SQL Statement
You use this functionality to see which strategy the MaxDB Optimizer uses to select the data. The *Execution Plan* (EXPLAIN) is listed.

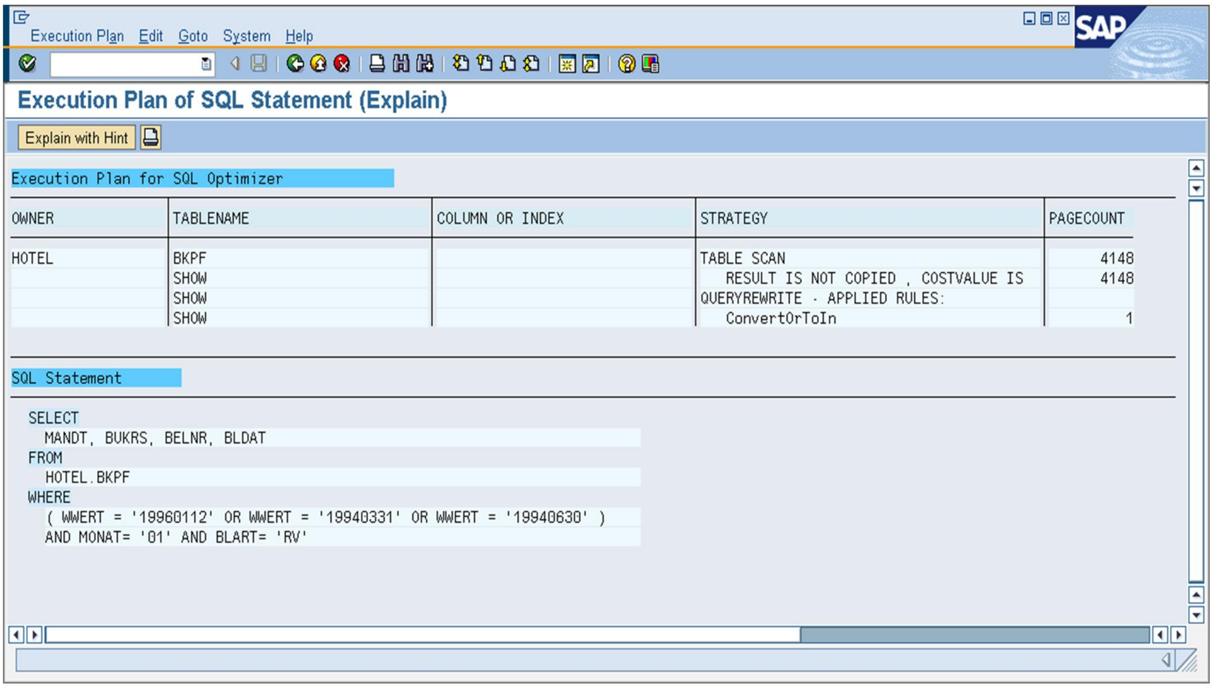
Display/Trace Execution Plan for an SQL Statement
The SAP Support might ask you to create this SQL Optimizer trace. This trace is used by the support to analyze why the MaxDB Optimizer chooses the strategy displayed in the *Execution Plan*.

Display Call in ABAP Program
If a logged SQL statement was called from an ABAP program, you can trace the SQL statement back to that program. To do this choose *Display Call in ABAP Program* (only available for the database instance that is the basis of the current SAP Web AS system).

In this example there are 4 columns in the SELECT list - the order of those columns is NOT important for the MaxDB Optimizer strategy search.
Remember: it is better to use only those columns in the SELECT list which are really needed than to use the * for all columns of a table.

During a SQL analysis focus on the WHERE condition of the SQL Statement. In this example only those rows are requested for which the conditions MONAT = ’01’ and BLART = ‘RV’ and one of the 3 different OR terms for WWERT apply.

Command Monitor: Display the Execution Plan

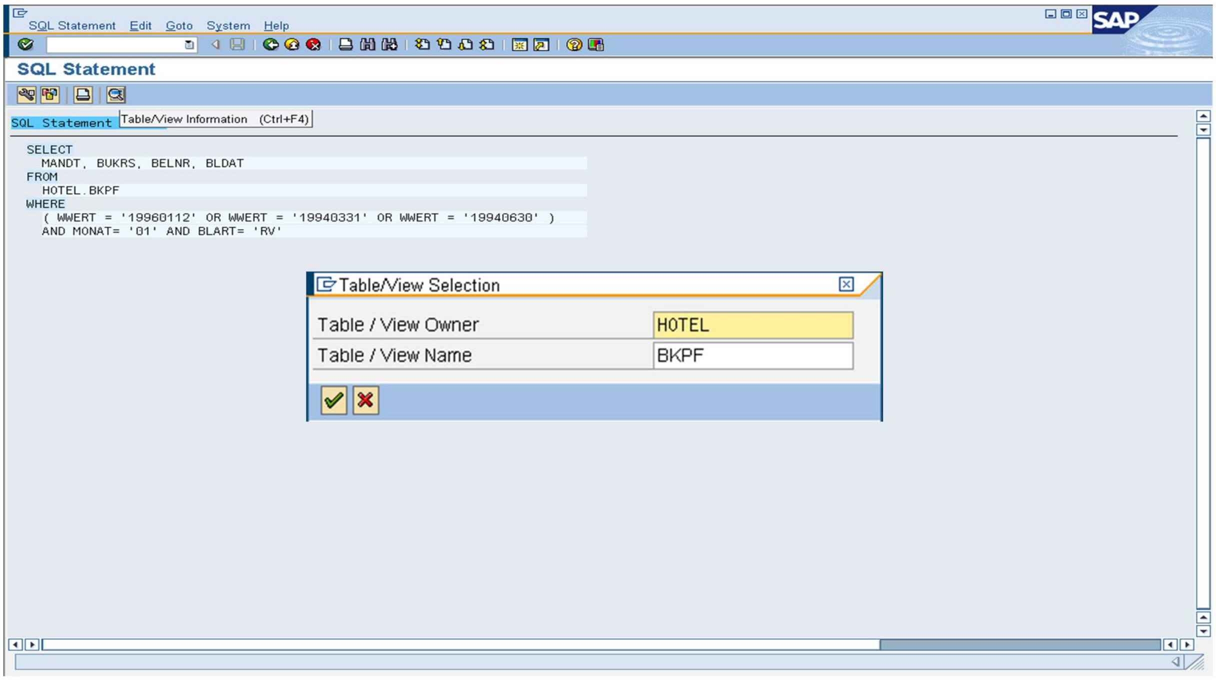


The execution plan of this SQL statement shows that the optimizer is using a *Table Scan* to select the relevant data.

If table BKPF is a very large one this may be an expensive access. Check in the next step why no index is used.

To get more information about existing indexes for this table choose *back*.

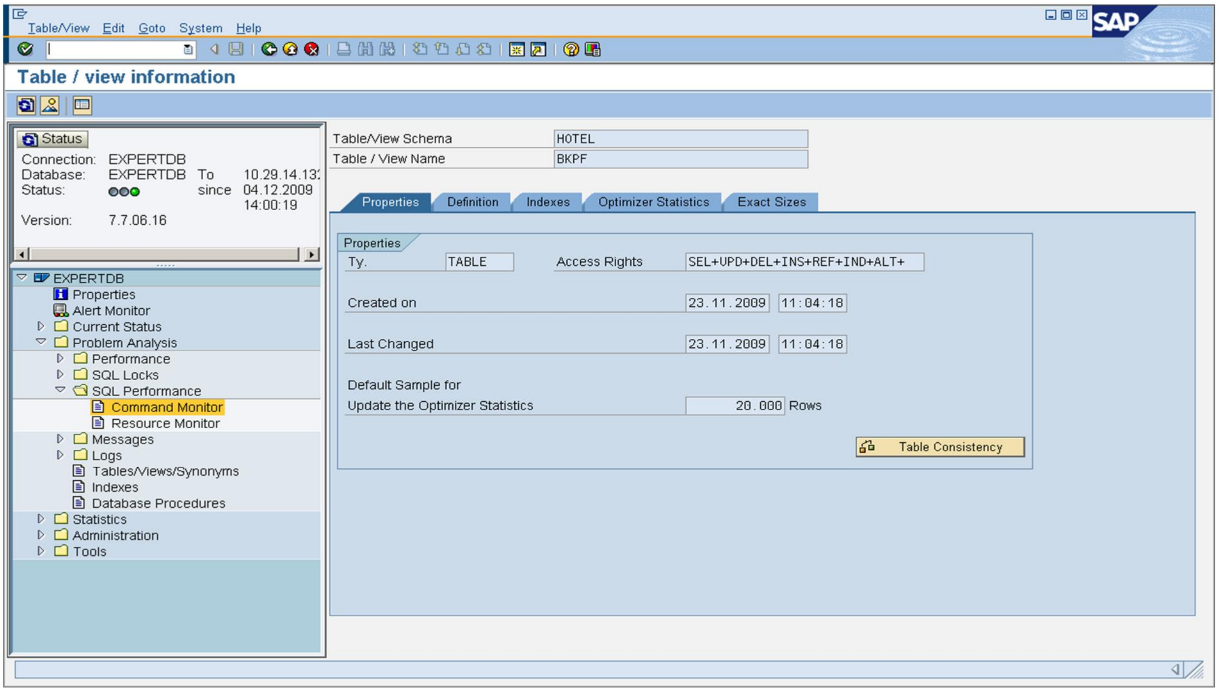
Command Monitor: Display Table Information



In this screen choose *Table/View information*.
Automatically the relevant table is inserted into the selection screen.
After confirming the table name you get detailed information about table BKPF.

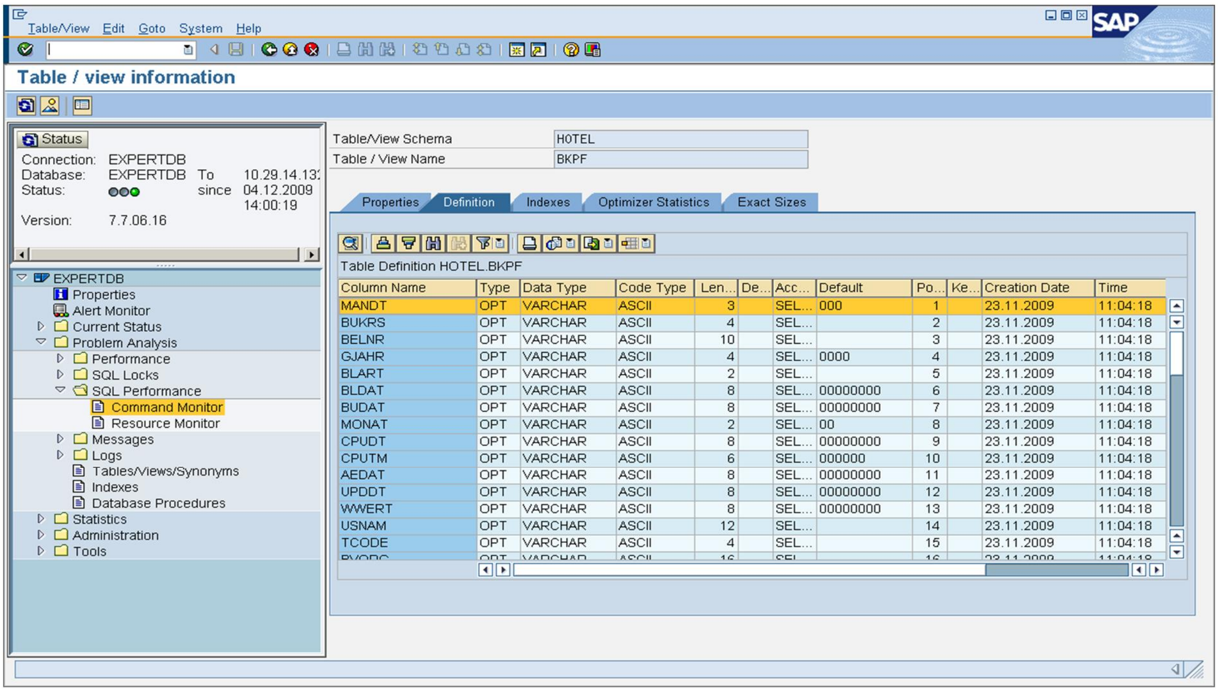
Alternatively you could display the same table information by choosing *Problem Analysis –> Tables/Views/Synonyms* in transaction DB50.

Table Details: Properties



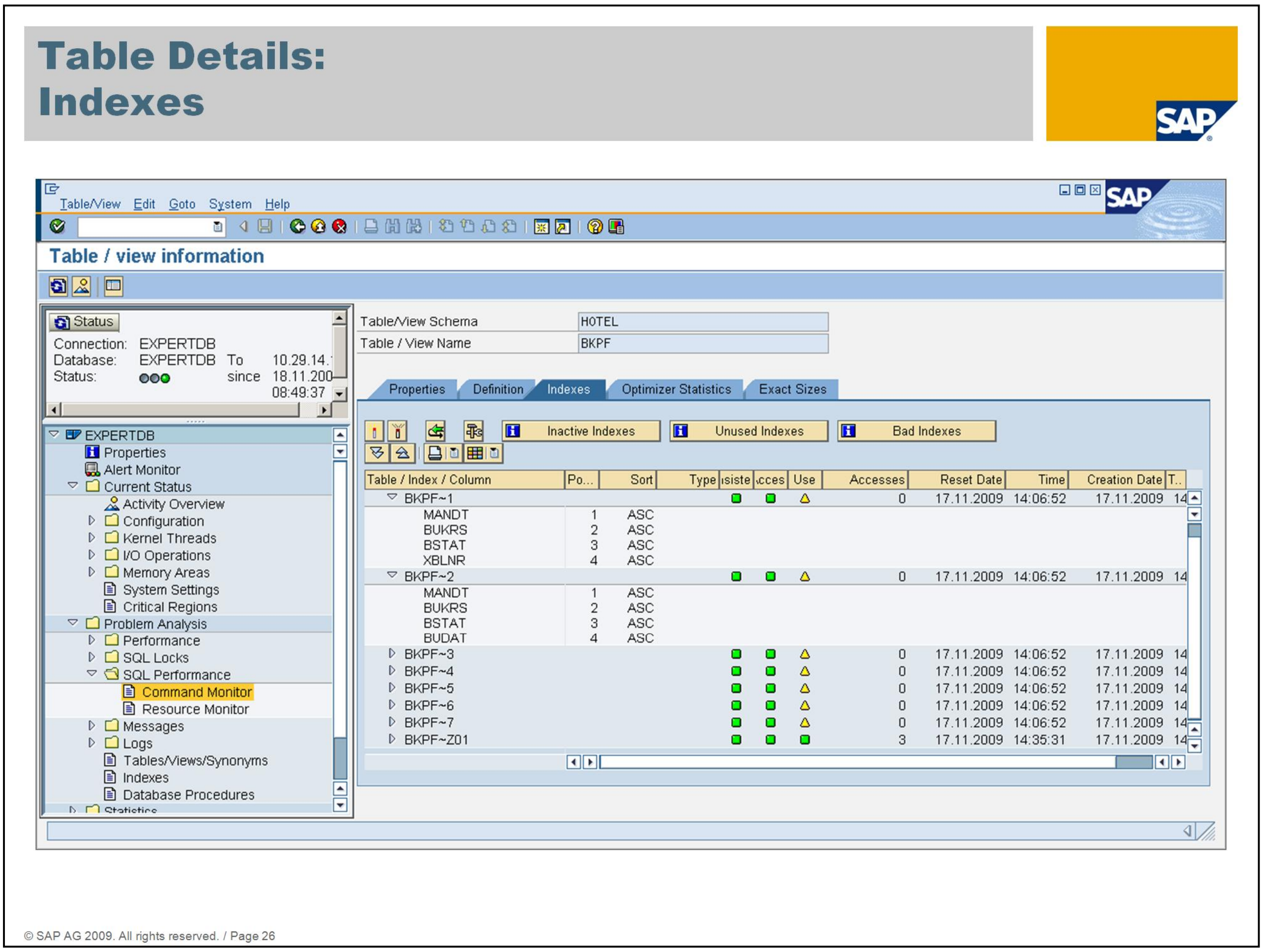
You can choose from the following displays:
Properties: Information about table type, access rights, creation and alter date, default sample size for the update of optimizer statistics (only for tables), and cost savings caused by clustering of tables (only for tables) is displayed. Additionally you can check the database structure of the selected table.

Table Details: Definition



Definition: The definition of the object in the database instance is displayed (this is not the object definition from the ABAP Dictionary but the object definition according to the system tables of the database system).

For our example analysis we check in the definition if our table has a primary key and which columns belong to the primary key: table BKPF does not have a primary key.



Next the Indexes are checked.
Indexes: Display of all indexes (including inactive, unused, bad indexes) which are defined for this object. Among other things, you can activate or deactivate indexes, restore a corrupted index or reset the usage counter.

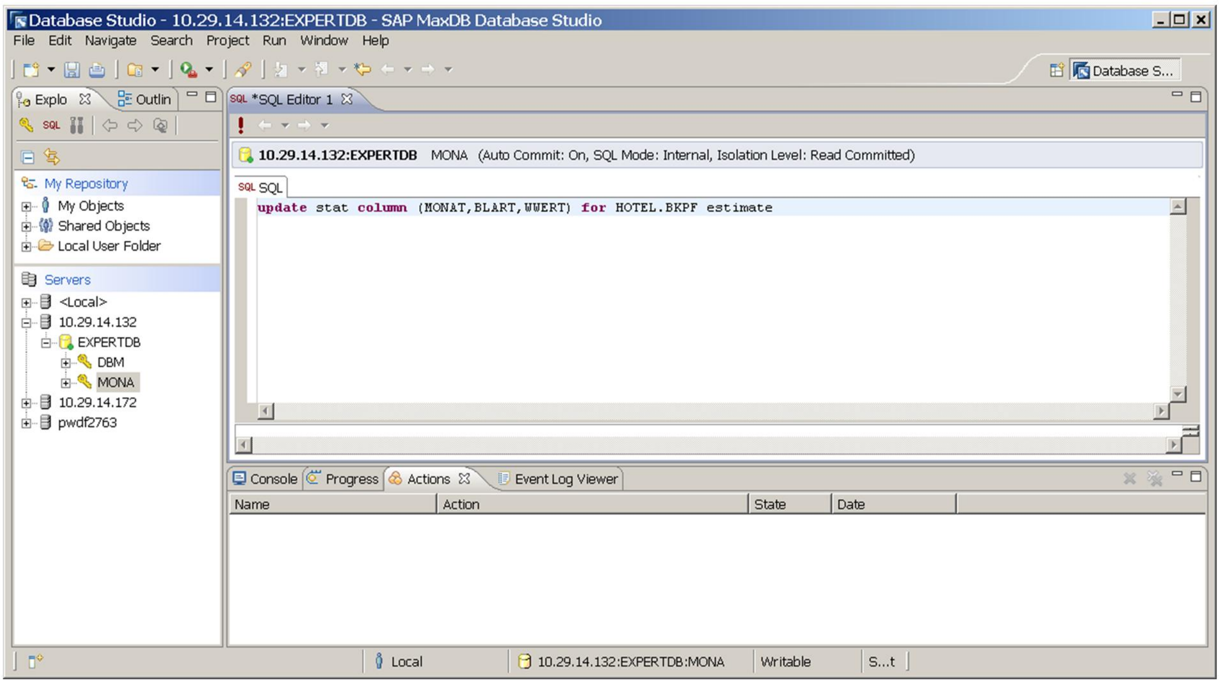
The WHERE condition of our SQL command is as follows:
where (WWERT = <value> or <value> or <value>) AND MONAT = <value> AND BLART = <value>

Check all listed indexes if they could be used to execute the analyzed SQL statement. None of the available indexes contains the columns of the WHERE condition. That means, the optimizer can only access the data via a table scan.

A new index should be created to optimize the access.
The order of the columns in the WHERE condition is not relevant for the column order in the index.
Important is the number of different values in a column to reduce the number of rows which has to be read to get the result. And important is the kind of qualification for each column.

The next step is to check the Optimizer Statistics of table BKPF to get the information about the distinct values of the qualified columns.

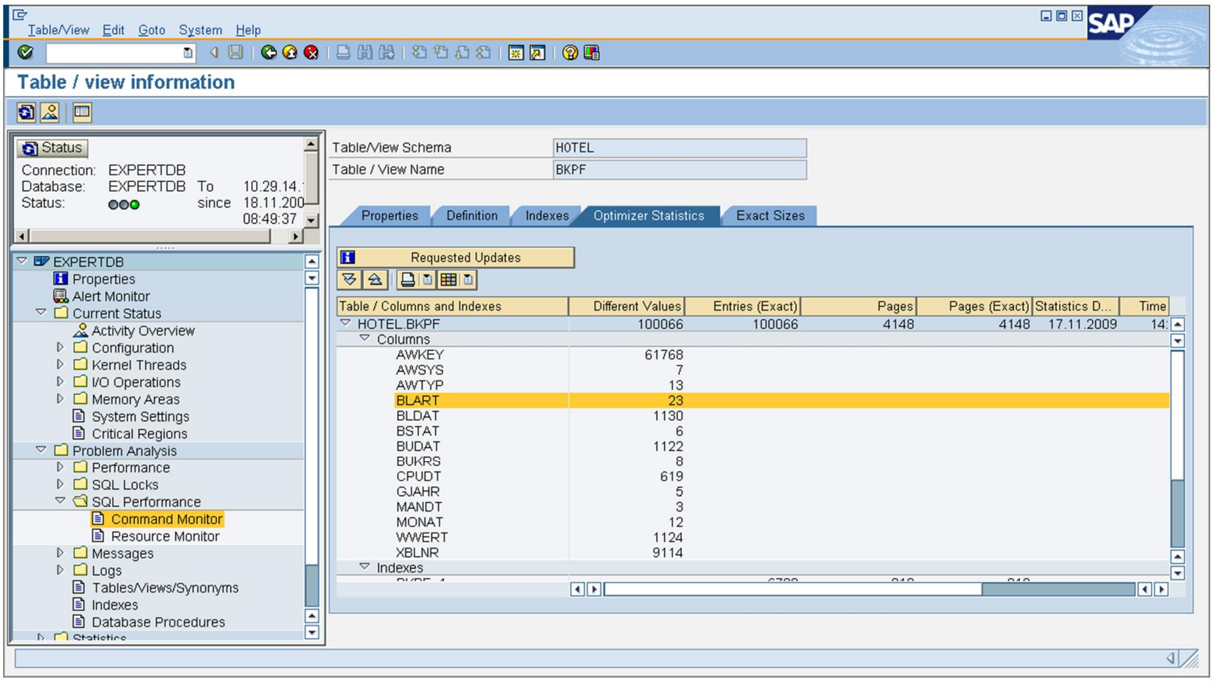
Update Statistics Column



If there are no statistics available for the columns WWERT, MONAT and BLART, you can start an *update statistics column* using tool Database Studio.

The statistics could also be updated in the table application on tab *Optimizer Statistics*.

Table Details: Optimizer Statistics



The distinct values of the columns and the qualification type in the WHERE condition are both relevant to find the best index structure.

Remember that the values of an index are sorted by the indexed columns.

The fasted access can be reached when the number of rows which have to be read is minimized. The distinct values give a hint which column should be the first one of the index: you should choose the column which has the most different values (distinct values).

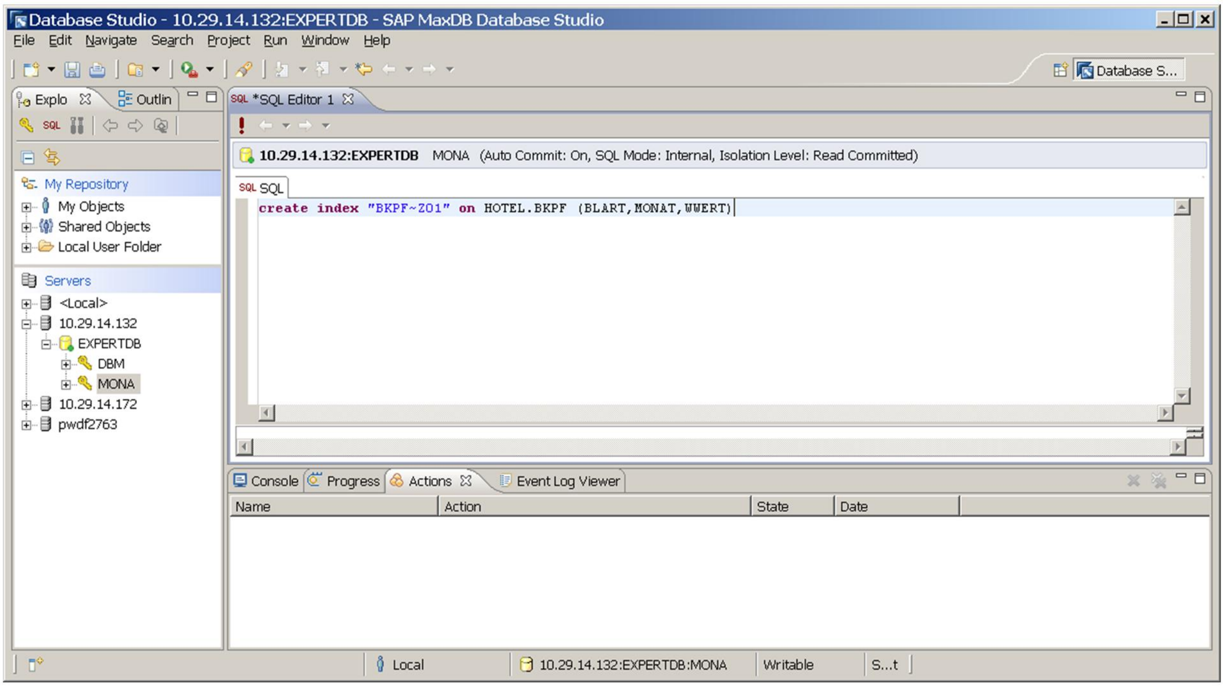
This is column WWERT, but in the WHERE condition 3 different values of WWERT are selected.

Using column WWERT as the first index column would result in reading more data to find the result as if another column is used as the first index column, which is selected with an equal condition on one value.

There are two other where conditions in the statement with an equal condition (one value each): on columns BLART and MONAT.

BLART has more distinct values so the first index field should be BLART, the second MONAT and the last WWERT.

Creating an Index



To optimize the statement create an additional index on table BKPF containing the columns (BLART,MONAT,WWERT). Make sure to use exactly this column order!

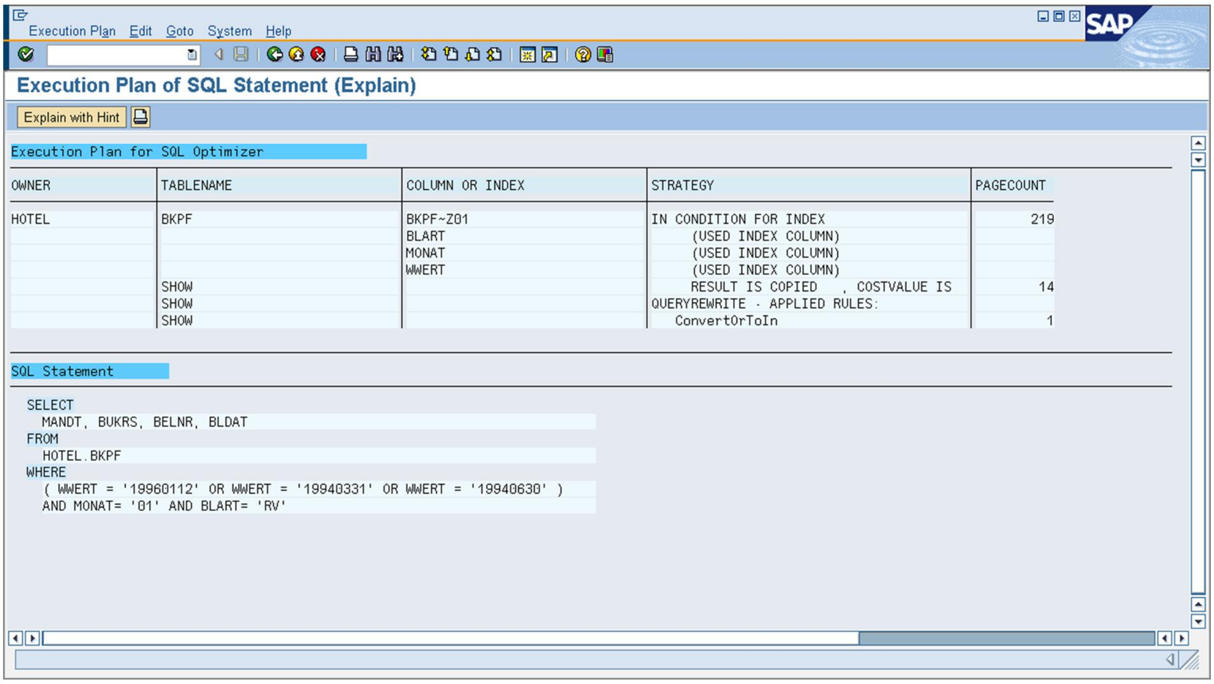
In an SAP Environment you should use transaction SE11/SE14 and the transport system to create an index in a test environment first and transport the new index to your productive system. Use the customer name space for non SAP standard indexes.

For test issues and in our demo Database Studio can be used to create indexes.

Important to know: Up to MaxDB version 7.6 the table is locked for change operations while the index is created. Be careful when you create new indexes on large tables, as this might take a long time. This should only be done during times of low workload.

With MaxDB Version 7.7 a non blocking create index has been implemented.
If the transaction which is creating the index does not hold any locks on the table the table won’t be locked for changes during index creation. All changes on that table during index creation are logged separately. At the end of the index creation only a short write lock on the table is set to redo these changes in the index.
Attention: if a unique index is created, the table is still exclusively locked while the index is created!

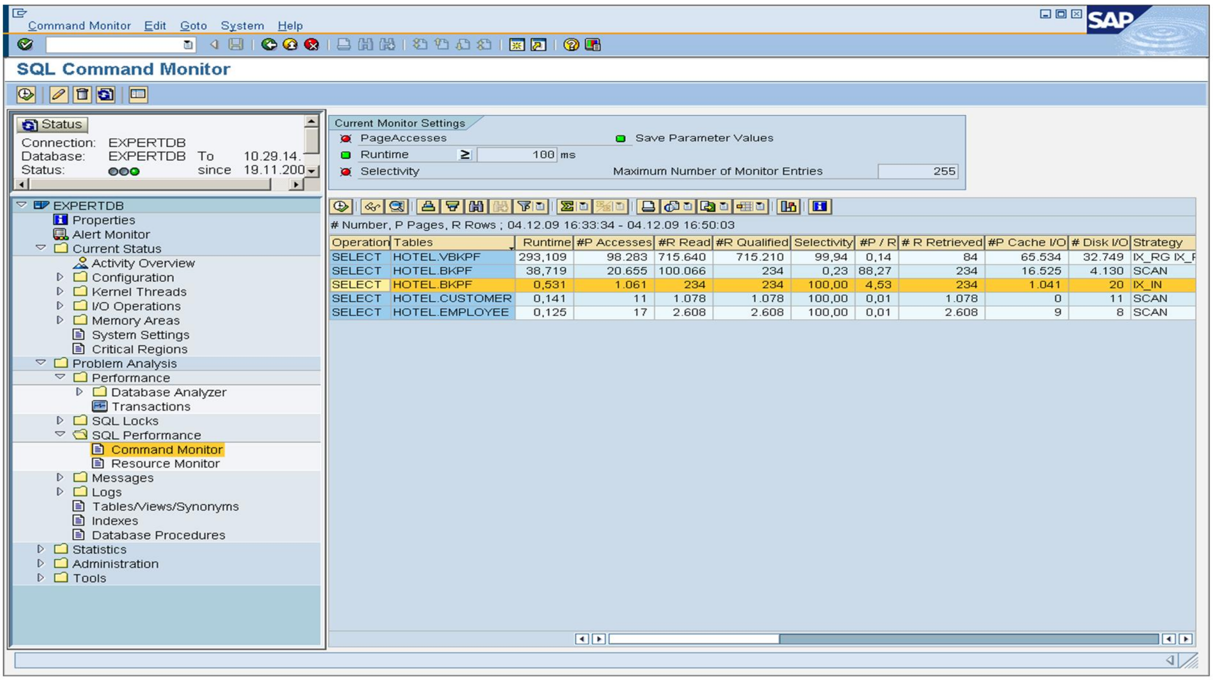
Command Monitor: Display the Execution Plan



After the index is created, open the command monitor and display the execution plan again. The execution plan is not stored with the monitoring data, but re-calculated. So the new index is taken into account by the optimizer for creating this execution plan.

As you can see, the new index is used. The calculated costs are lower than before: only 219 data pages have to be read during statement execution – compare this to the old value 4148. The costs have been reduced from 4148 to 14.

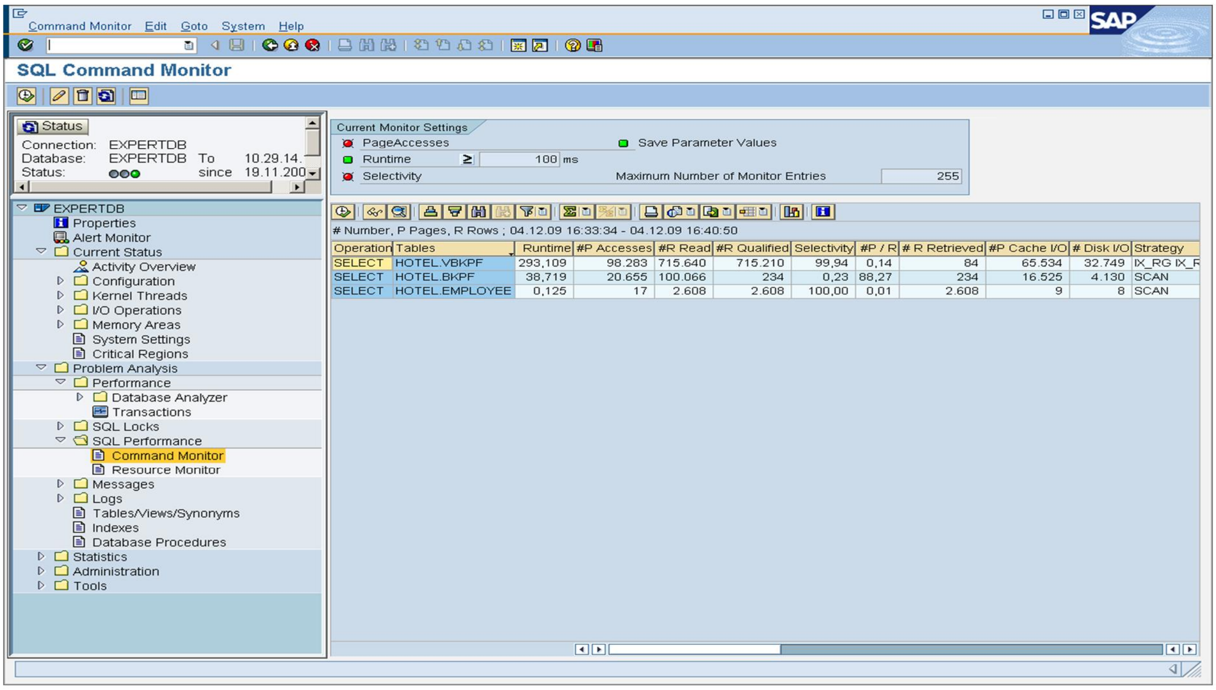
Command Monitor: Runtime Comparison



When the report is executed again, you will see that the runtime is shorter than before without the index.

The statement execution now takes only 531 msec.

Command Monitor: Long Running Join Select

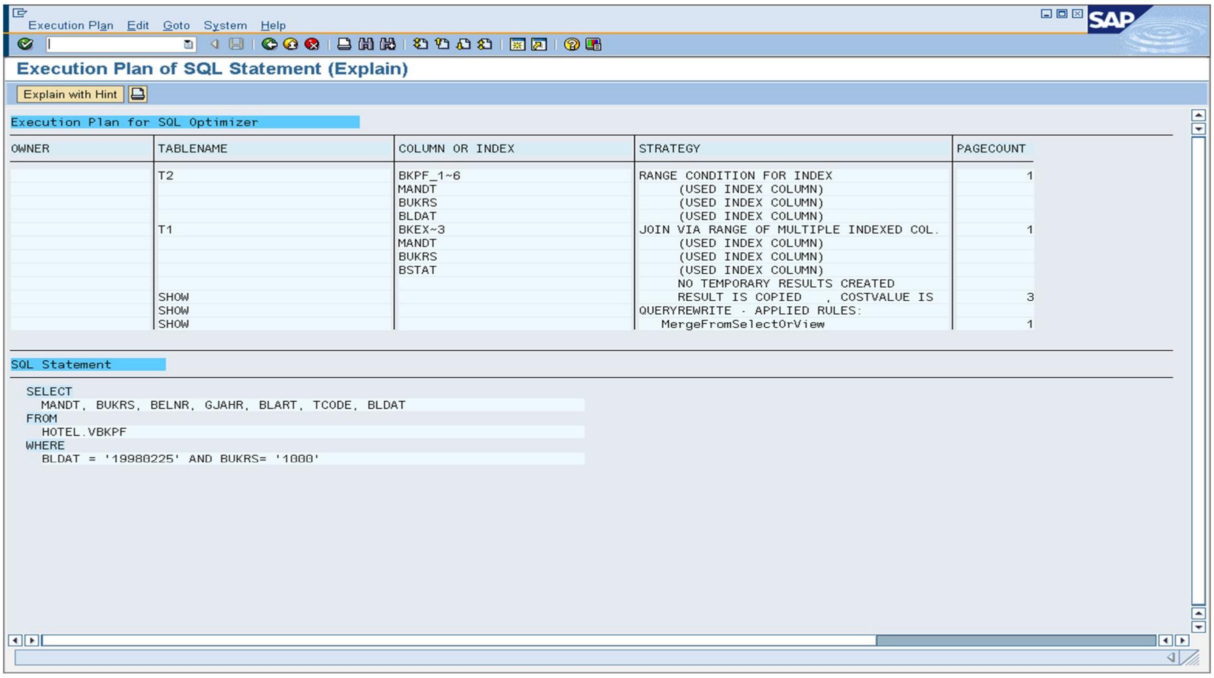


There is another SQL statement logged in the Command Monitor which should be analyzed: the SELECT statement on HOTEL.VBKPF with a total runtime of 293 seconds. You can see a lot of page accesses and although the ratio between rows read and rows qualified and therefore the selectivity is not that bad, this statement should be analyzed in more detail.

Background information: when a join statement is executed, all rows which are qualified in the first table are counted as ‘qualified’ although some of them might be excluded from the final result during one of the next join steps.

Start the analysis with detail information about the SQL statement by double-clicking the statement.

Command Monitor: Display the Join Execution Plan



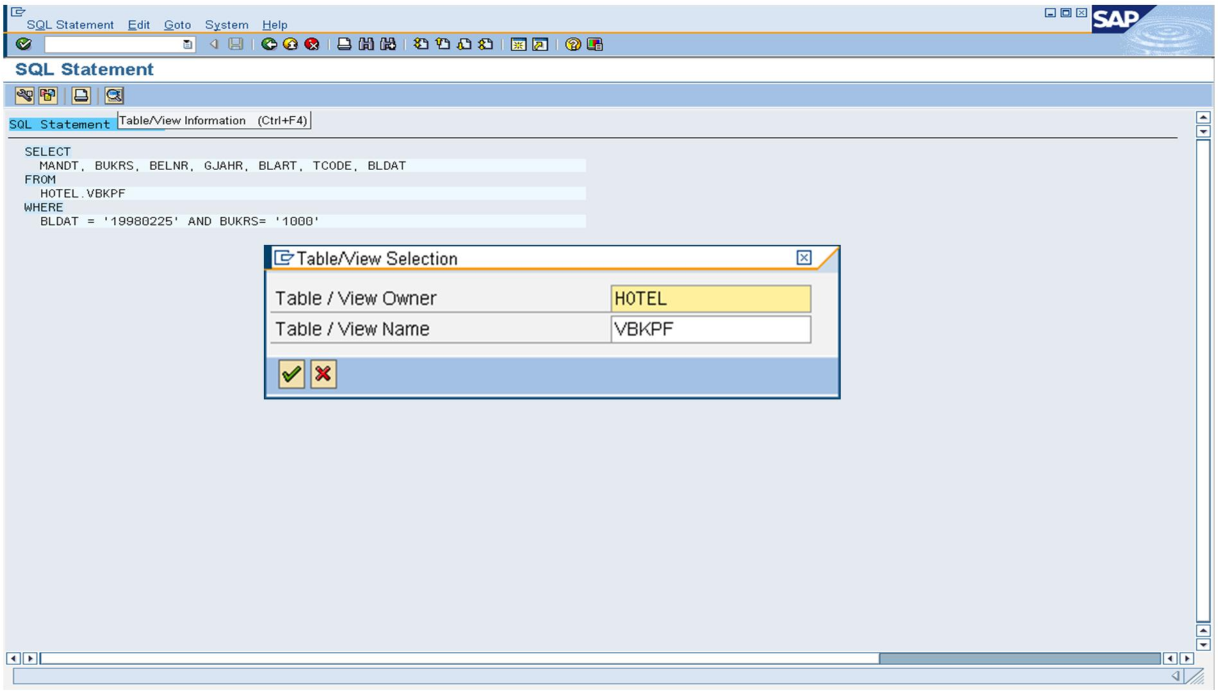
The execution plan shows that two tables are involved. View VBKPF contains two tables called T1 and T2. The Optimizer starts with selecting data on table T2 using index BKPF_1~6 and joins the results to table T1 using the index BKEX~3.

The cost value does not look bad with value 3. However, the displayed page count does not fit to the number of page accesses logged in the monitor.
It is also suspicious that this statement run for more than 290 seconds although only one page is supposed to be read for each table.

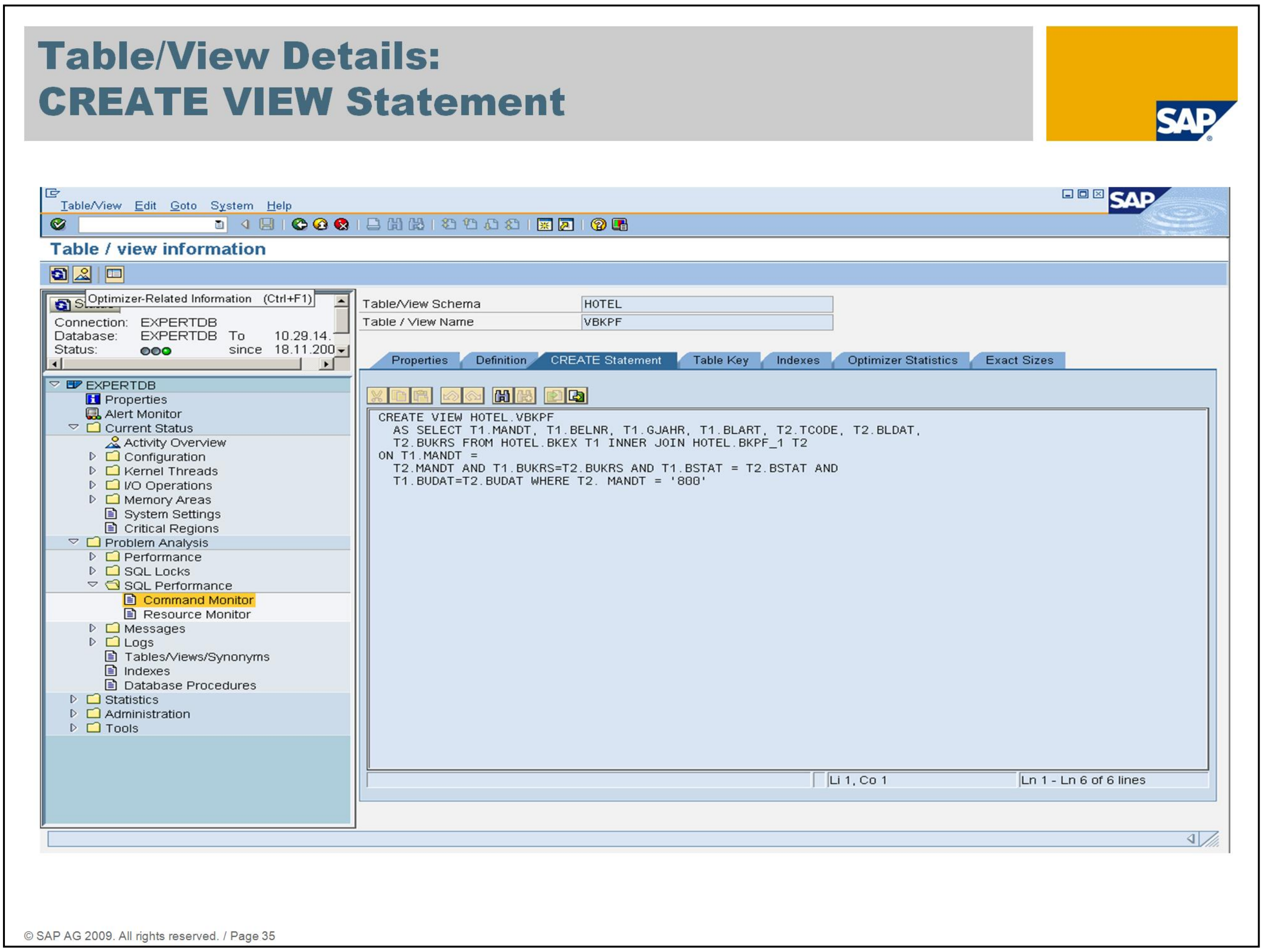
Something seems to be wrong here!

In the next step the view details have to be checked – either accessed from the SQL statement screen or in application *Tables/Views/Synonyms*.

Command Monitor: Display Table/View Information



The procedure to analyze a view access is nearly the same as the procedure to analyze an SQL statement on a single table.



Having a closer look at the join definition is very important when a join select has to be analyzed. You can see that table HOTEL.BKEX (T1) and HOTEL.BKPF_1 (T2) are used in the view.

The following columns of table HOTEL.BKEX are selected:
MANDT, BELNR ,GJAHR and BLART
The following columns of table HOTEL.BKPF_1 are selected:
TCODE, BLDAT and BUKRS

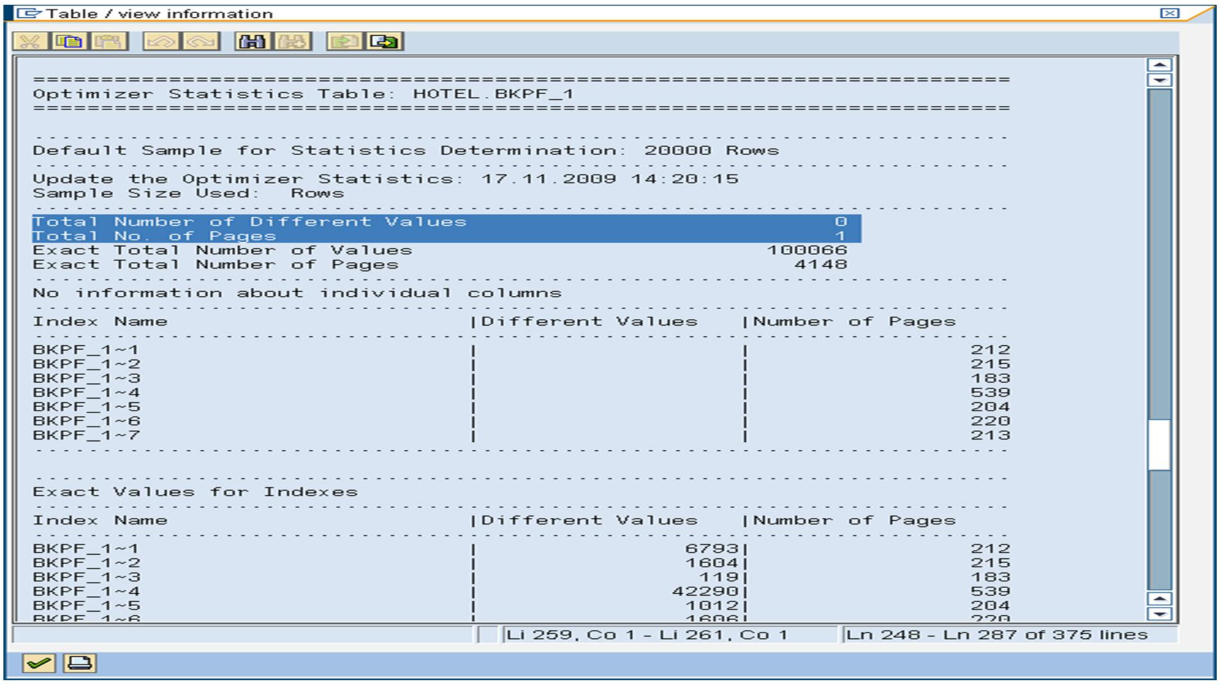
Both tables are joined via the following columns (join condition):
T1.MANDT = T2.MANDT
T1.BUKRS = T2.BUKRS
T1.BSTAT = T2.BSTAT
T1.BUDAT = T2.BUDAT

The view selects only rows which belong to MANDT = ‘800’ (local predicate).

Check the explain plan: The optimizer starts with table BKPF_1 and joins the result with table BKEX.

Remember: so far it cannot be explained why the execution plan shows that only one page of each table has to be read to get the result. The Command Monitor shows that 98.000 pages were accessed!
The optimizer uses column statistics to find the best strategy for join selects – and the execution plan is also based on statistic values. Therefore the optimizer statistics should be checked. This can be done using either button *Optimizer Related Information* or using the *Optimizer Statistics* tab.

Table/View Details: Optimizer Related Information (1)



```
Table / view information
=====================================================================
Optimizer Statistics Table: HOTEL.BKPF_1
=====================================================================
Default Sample for Statistics Determination: 20000 Rows
Update the Optimizer Statistics: 17.11.2009 14:20:15
Sample Size Used:    Rows
Total Number of Different Values                            0
Total No. of Pages                                          1
Exact Total Number of Values                           100066
Exact Total Number of Pages                              4148
No information about individual columns
Index Name                     |Different Values   |Number of Pages
BKPF_1~1                       |                   |            212
BKPF_1~2                       |                   |            215
BKPF_1~3                       |                   |            183
BKPF_1~4                       |                   |            539
BKPF_1~5                       |                   |            204
BKPF_1~6                       |                   |            220
BKPF_1~7                       |                   |            213
Exact Values for Indexes
Index Name                     |Different Values   |Number of Pages
BKPF_1~1                       |              6793|             212
BKPF_1~2                       |              1604|             215
BKPF_1~3                       |               119|             183
BKPF_1~4                       |             42290|             539
BKPF_1~5                       |              1012|             204
Li 259, Co 1 - Li 261, Co 1    Ln 248 - Ln 287 of 375 lines
```


The optimizer needs statistics about the size of the tables and indexes and the cardinality of the different columns to determine the ideal access strategy for join statements.

The *Total Number of different Values* is displayed as 0.
The *Total Number of Pages* is 1. This represents the currently available statistic information.

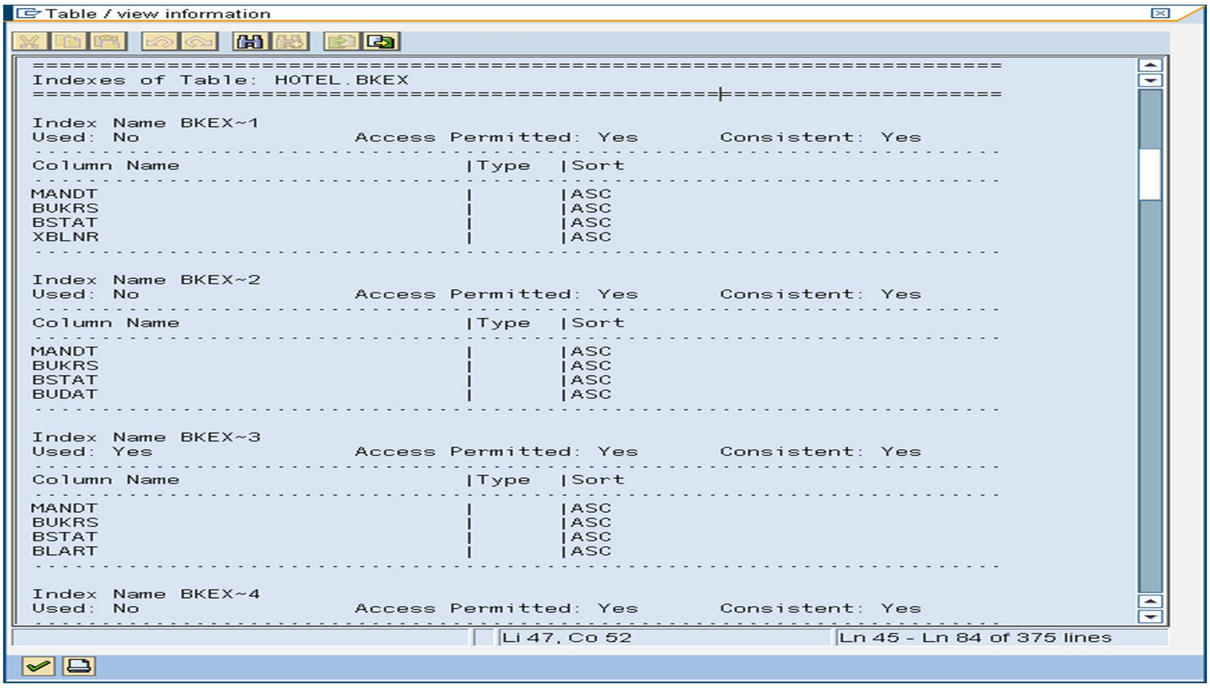
As of MaxDB version 7.6 the exact number of table records is available additionally to the table statistics. The number of records is updated after each delete and insert.

The *Exact Total Number of Values* and *Exact Total Number of Pages* differ from the displayed statistic information. This indicates that the statistic data is outdated. But the optimizer needs accurate statistic values to determine the best access strategy for join selects. Therefore the optimizer statistics should be updated!

Nice to know: For single table accesses those column statistics are irrelevant because the optimizer uses the evaluation feature during strategy search to get information about the cardinality of columns.

To avoid problems which are caused by missing or outdated optimizer statistics, update the SQL optimizer statistics regularly. Do a performance analysis of individual SQL statements (particularly join statements) only when the statistics are up-to-date. Schedule regular updates of the SQL optimizer statistics in the DBA Planning Calendar. Ensure that the scheduling has been successfully executed and thus that the SQL Optimizer statistics are current.

Table/View Details: Optimizer Related Information (2)



The columns which are used in the join condition are important when analyzing if the best strategy is used for the join transition.

T1.MANDT = T2.MANDT
T1.BUKRS = T2.BUKRS
T1.BSTAT = T2.BSTAT
T1.BUDAT = T2.BUDAT

Check the existing primary key and indexes if they can be used for the join transition.

The execution plan displays that index BKEX~3 is used for the join transition. It has to be checked, why the optimizer does not use index BKEX~2, which reflects exactly the join condition. This might also be caused by the outdated statistics.

Update Statistics

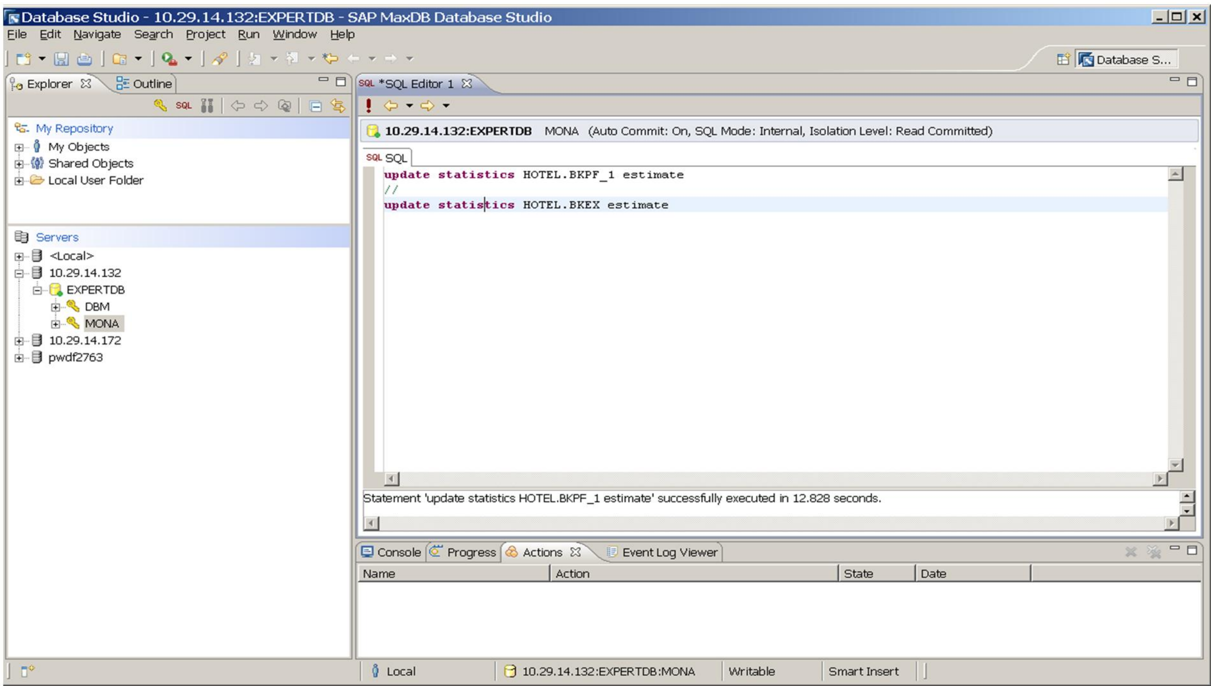


Table BKPF_1 and table BKEX are used in view VBKPF. Therefore new optimizer statistics have to be created for both tables.

Important for finding the best access strategy for a join statement are the column statistics. You don’t need to create those statistics with a special *update statistics column* command. During the statistics update for the table the column statistics (of indexed columns) are updated implicitly. The statistics for the indexes are automatically created as well.

Command Monitor: Display the Join Execution Plan

Execution Plan Edit Goto System Help

Execution Plan of SQL Statement (Explain)

Explain with Hint

Execution Plan for SQL Optimizer

OWNER	TABLENAME	COLUMN OR INDEX	STRATEGY	PAGECOUNT
	T2	BKPF_1~6	RANGE CONDITION FOR INDEX	4148
		MANDT	(USED INDEX COLUMN)	
		BUKRS	(USED INDEX COLUMN)	
		BLDAT	(USED INDEX COLUMN)	
	T1	BKEX~2	JOIN VIA MULTIPLE INDEXED COLUMNS	4148
		MANDT	(USED INDEX COLUMN)	
		BUKRS	(USED INDEX COLUMN)	
		BSTAT	(USED INDEX COLUMN)	
		BUDAT	(USED INDEX COLUMN)	
			NO TEMPORARY RESULTS CREATED	
	SHOW		RESULT IS COPIED , COSTVALUE IS	42
	SHOW		QUERYREWRITE - APPLIED RULES:	
	SHOW		MergeFromSelectOrView	1

SQL Statement

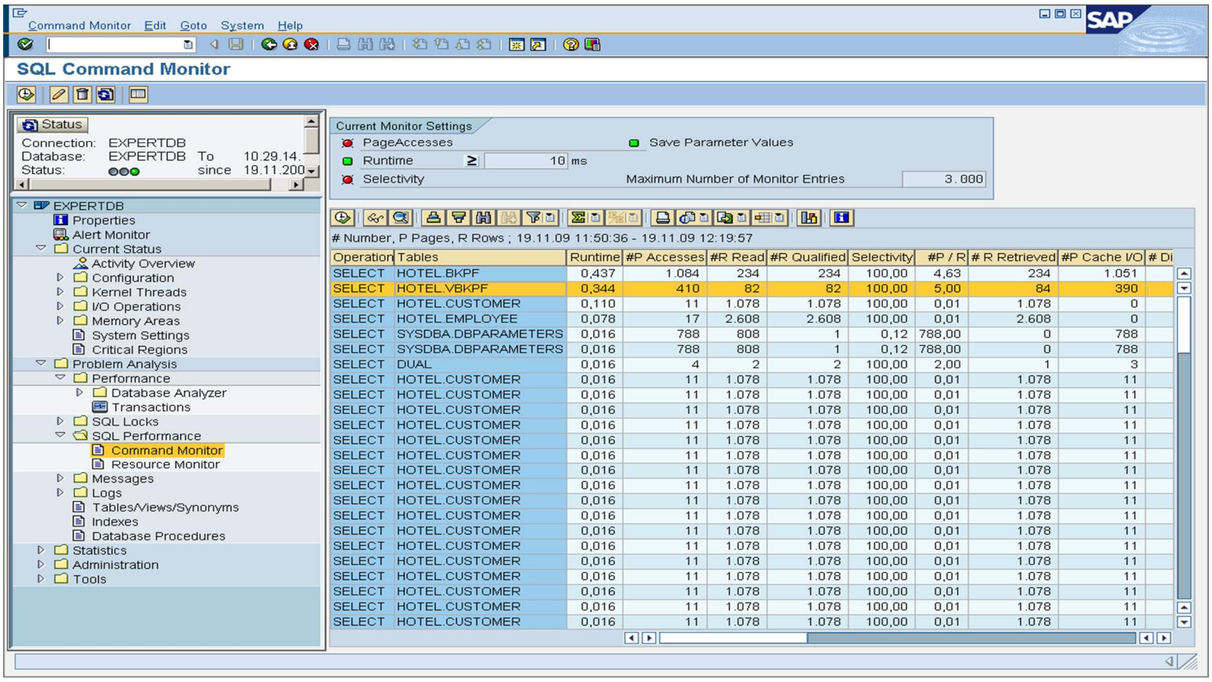
```
SELECT
   MANDT, BUKRS, BELNR, GJAHR, BLART, TCODE, BLDAT
FROM
   HOTEL.VBKPF
WHERE
   BLDAT = '19980225' AND BUKRS= '1000'
```


When the execution plan is displayed again, you can see that the strategy has changed. The optimizer still starts the access with index BKPF_1~6 but the join to the second table BKEX is now done via index BKEX~2.

Now the displayed costs are higher than before but the lower costs of the old execution plan were based on wrong optimizer statistics.

To check the current runtime of this command the report can be executed again.

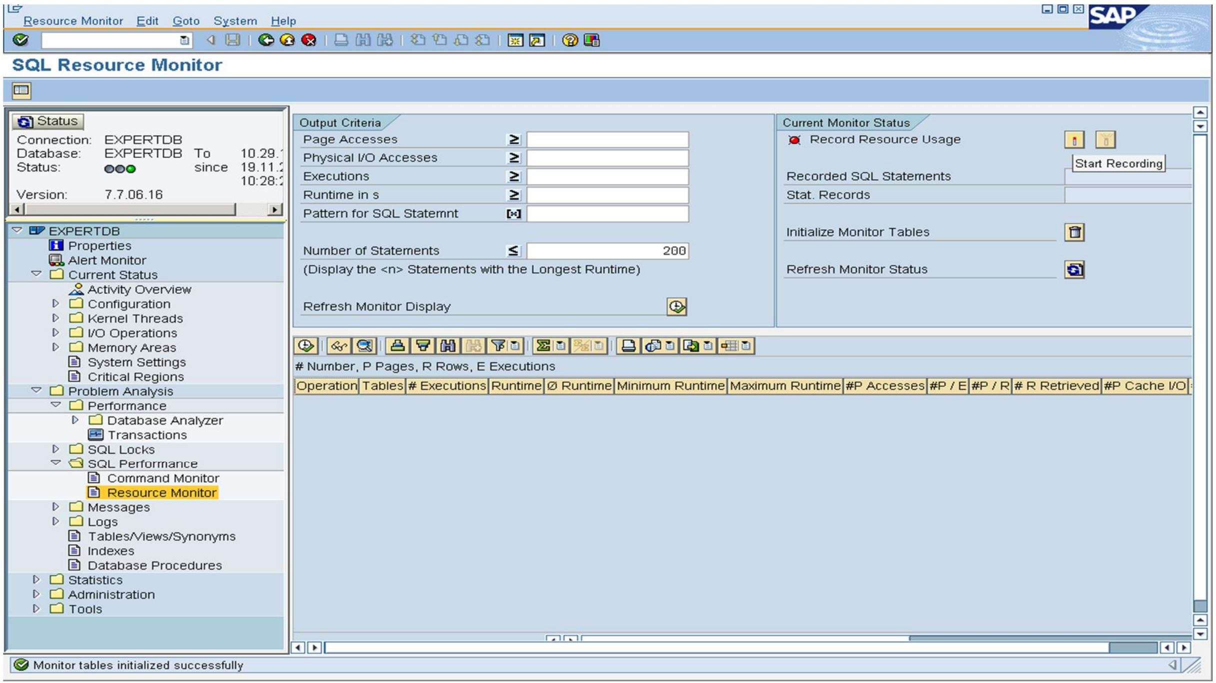
Command Monitor: Runtime Comparison



Operation	Tables	Runtime	#P Accesses	#R Read	#R Qualified	Selectivity	#P / R	# R Retrieved	#P Cache I/O	# Di
SELECT	HOTEL.BKPF	0,437	1.084	234	234	100,00	4,63	234	1.051	
SELECT	HOTEL.VBKPF	0,344	410	82	82	100,00	5,00	84	390	
SELECT	HOTEL.CUSTOMER	0,110	11	1.078	1.078	100,00	0,01	1.078	0	
SELECT	HOTEL.EMPLOYEE	0,078	17	2.608	2.608	100,00	0,01	2.608	0	
SELECT	SYSDBA.DBPARAMETERS	0,016	788	808	1	0,12	788,00	0	788	
SELECT	SYSDBA.DBPARAMETERS	0,016	788	808	1	0,12	788,00	0	788	
SELECT	DUAL	0,016	4	2	2	100,00	2,00	1	3	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	
SELECT	HOTEL.CUSTOMER	0,016	11	1.078	1.078	100,00	0,01	1.078	11	

The runtime of the SELECT statement on join view VBKPF decreased to 344 msec.

Resource Monitor (1)



You can use the *Resource Monitor* to get an overview of the SQL statements which used the most resources in a specified period. To access the *Resource Monitor*, choose menu *Problem Analysis -> SQL Performance -> Resource Monitor* in transaction DB50.

The status of the *Resource Monitor* is displayed in the *Current Monitor Status* area and can be refreshed using *Refresh Monitor Status* .

As of MaxDB Version 7.8. the *Resource Monitor* is always active.
In older database versions the *Resource Monitor* has to be activated manually if needed for an analysis.

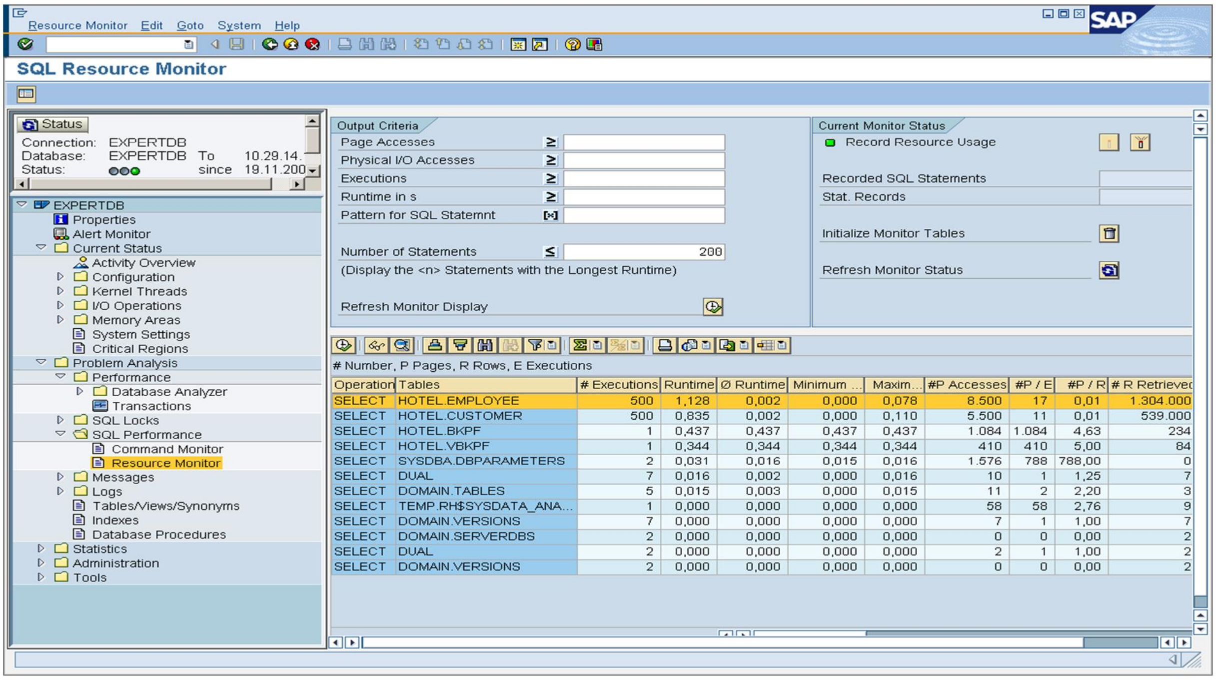
To start or stop the *Resource Monitor*, choose *Start Recording* respectively *Stop Recording*.

Choose *Initialize Monitor Tables* to delete all logged SQL statements from the monitoring tables.

In contrast to the *Command Monitor* the *Resource Monitor* logs all executed SQL statements – not only statements which violate specific thresholds.

In the *Output Criteria* area you can specify filters to reduce the number of displayed statements.

Resource Monitor (2)



The runtime data is aggregated for all executions of the same SQL statement. This way you get an overview, which statement is most expensive – not necessarily a single execution of the statement but the sum of all executions.
In contrast to the *Command Monitor* the parameter values of the single statement executions are not recorded in the *Resource Monitor*.

As a default the output list is sorted by the *Total Runtime*.

Statements which should be analyzed are statements:

- with a high total runtime

- with a very high number of executions

- with a bad selectivity

Statements with a high total runtime are the most expensive ones. This can have different reasons: either each single execution was fast but the statement was executed very often or even a single execution took a long time.

For statements which are executed very often, the application needs to be analyzed. It has to be checked if it is really necessary to execute the same statement over and over again. Furthermore it should be checked, if only the required columns are selected and if the WHERE condition is sufficient. It doesn't make sense to select more data than necessary and filter it afterwards by the application.

Statements with a bad selectivity need to be analyzed in the same way as shown in the *Command Monitor* example.

Questions and Answers

Thank You!
Bye, Bye – And Remember Next Session

Feedback and further information:
http://www.sdn.sap.com/irj/sdn/maxdb

End of January:
Session about Analysis of Database Corruptions

Disclaimer

This presentation reflects current planning.

Contents may be changed without prior notice, and are in no way binding upon SAP.