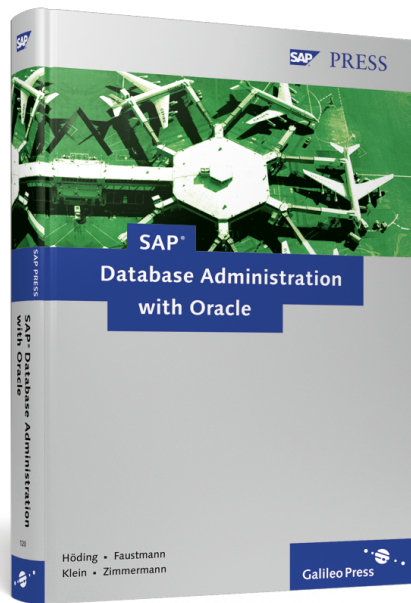


Michael Höding, André Faustmann,
Gunnar Klein, Ronny Zimmermann

SAP® Database Administration with Oracle



Contents at a Glance

1	Introduction	15
2	SAP Fundamentals	27
3	Oracle Fundamentals	65
4	SAP and Oracle	149
5	Planning the System Landscape	243
6	SAP Change and Transport Management	277
7	System Lifecycle	341
8	Performance	405
9	System Operation and Monitoring	495
10	Backup, Restore, and Recovery	545
11	Administrating the Java Stack	669
12	SAP NetWeaver BI and Oracle	723
13	Afterword and Outlook	785
A	Flight Data Model	789
B	General Options of the BR*Tools	791
C	References	797
D	Authors	799

Contents

1	Introduction	15
1.1	Reasons for Owning This Book	16
1.2	Tasks of the SAP Basis	18
1.3	Structure of This Book	19
1.4	Conventions and Other Information	23
1.5	Acknowledgements	24
2	SAP Fundamentals	27
2.1	Overview of the SAP Software	27
2.1.1	Standard Software versus Individual Software	27
2.1.2	Integration	28
2.1.3	Development of SAP R/3	28
2.1.4	SAP Terminology	32
2.2	Architecture and Scalability	35
2.3	Application Server	37
2.3.1	Overview of the SAP Application Server Processes	38
2.3.2	Memory Structures	39
2.3.3	Dispatcher	40
2.3.4	Dialog Work Process	42
2.3.5	Batch Work Process (BTC)	45
2.3.6	Update Process (UPD)	47
2.3.7	Lock Management with the Enqueue Process (ENQ) ...	48
2.3.8	Message Server Process	49
2.3.9	Gateway Process	49
2.3.10	Variants of an Instance	49
2.4	SAP Administration	50
2.4.1	Profile Files and System Startup	50
2.4.2	Software Maintenance	52
2.4.3	Database Administration	53
2.4.4	Data Backup	54
2.4.5	Performance Optimization	55
2.5	SAP and Software Development	56
2.5.1	ABAP Framework	57
2.5.2	Java in the SAP Kernel	60
2.5.3	Internet Transaction Server	61
2.5.4	Remote Function Call	62
2.6	Summary	63

3	Oracle Fundamentals	65
3.1	Basics of Database Technology	65
3.1.1	Motivation and History	65
3.1.2	Relational Data Model and SQL	71
3.1.3	Short Overview of SQL	75
3.1.4	Implementation Techniques for DBMS	79
3.2	Development of Oracle	81
3.3	Tools for the Oracle Administrator	84
3.3.1	sqlplus	84
3.3.2	isqlplus	86
3.3.3	Oracle Enterprise Manager	87
3.4	Oracle Kernel	89
3.4.1	Oracle Processes	90
3.4.2	Oracle Main Memory Structures	93
3.4.3	Oracle File System	96
3.4.4	Oracle Tablespace Concept	98
3.4.5	Other Important Files	109
3.4.6	Interaction of Processes and Storage Structures.....	116
3.4.7	Accessing Oracle with Oracle Net	119
3.4.8	Query Optimization	126
3.4.9	Data Backup and Recovery	133
3.4.10	Users and Privileges	140
3.4.11	Monitoring an Oracle Instance with the Enterprise Manager	145
3.5	Summary	147
4	SAP and Oracle	149
4.1	Processes of SAP and Oracle Systems	150
4.1.1	System Startup	152
4.1.2	Relationships Between Processes	161
4.1.3	Communication Between SAP Instances and Oracle Processes	163
4.1.4	Log and Trace Files	165
4.1.5	System Stop	171
4.2	Requirements at the Operating System Level	173
4.2.1	Users, Groups, and Environment Variables (UNIX)	177
4.2.2	Oracle Client	180
4.2.3	SAP Kernel	185
4.3	Authentication Between SAP and Oracle	190
4.3.1	Database Users	190

4.3.2	Login Processes	192
4.3.3	Privileges in the Database	197
4.3.4	Security Aspects	199
4.4	SAP Tablespaces	202
4.4.1	Tablespace Layout	202
4.4.2	Tablespace Types	207
4.4.3	Object Assignment and Object Parameters	209
4.4.4	Reorganizing Tables and Tablespaces	211
4.5	Administrating Oracle with the BR*Tools	216
4.5.1	Development and Content	217
4.5.2	Environment, Options, and Log Files	219
4.5.3	BRTOOLS and BRGUI	224
4.5.4	BRCONNECT	227
4.5.5	BRSPACE	230
4.5.6	Parameter Maintenance for Oracle	235
4.6	Summary	240

5 Planning the System Landscape243

5.1	From Product to Solution Landscape	243
5.2	Overview of Planning Criteria	245
5.2.1	Construction Infrastructure	246
5.2.2	Server Technology and Platforms	250
5.2.3	Storage and SAN Infrastructure	254
5.2.4	Backup	257
5.2.5	Frontend	259
5.3	High Availability	263
5.4	IT and System Security	267
5.5	Extending an Existing Landscape	274
5.6	Summary	276

6 SAP Change and Transport Management277

6.1	Standard Software and Changes to the Standard	278
6.2	Basic Principles of Software Logistics	280
6.2.1	Data in the SAP System	280
6.2.2	System Landscape	284
6.2.3	Change and Transport System (CTS)	288
6.2.4	Recording Changes	291
6.2.5	Change Requests	292
6.2.6	Transport Management System	296
6.2.7	Transport Routes and Transport Layers	302

6.3	Change Management for Customizing Settings	309
6.4	Change Management for Developments	313
6.5	Transport Management	318
6.5.1	Transport Organizer Tools	318
6.5.2	Transport Strategy	319
6.5.3	Import Queue and Import Buffer	326
6.5.4	Transports Between Transport Groups	328
6.5.5	Transports Between Transport Domains	329
6.5.6	Using the Transport Control Program	333
6.5.7	Logging Transports	335
6.5.8	Transports Between Unicode and Non-Unicode Systems	337
6.6	Tips and Tricks	338
6.7	Summary	339
7	System Lifecycle	341
7.1	Installation	341
7.1.1	Installation Tools	341
7.1.2	Phases	344
7.1.3	Unattended Installation	355
7.2	System Maintenance	357
7.2.1	Client Tools	357
7.2.2	SAP Support Packages, Patches, and Corrections.....	364
7.2.3	Maintaining the Oracle Database	377
7.2.4	Maintaining the Operating System	384
7.3	Upgrades	386
7.3.1	SAP Upgrade	386
7.3.2	Oracle Upgrade	402
7.4	Summary	404
8	Performance	405
8.1	Administrative and Program-Based Problems	406
8.2	Analyzing Administrative Performance Problems	408
8.2.1	Analyzing the Hardware and Operating System	409
8.2.2	Analyzing the Database	416
8.2.3	Analyzing the SAP System	458
8.3	Analyzing Program-Based Performance Problems:	
	SQL Optimization	473
8.3.1	Two Goals: Functionality and Performance	474

8.3.2	Effects	475
8.3.3	Problem Analysis Tools	477
8.3.4	Detailed Analysis of SQL Statements	480
8.3.5	Prevention: The Silver Bullet	485
8.3.6	Indexes for Faster Access	488
8.4	Summary	492

9 System Operation and Monitoring495

9.1	Monitoring in the SAP Environment: Motivation and Scope ...	495
9.1.1	Monitoring Areas	496
9.1.2	Monitoring Problems	499
9.1.3	Solutions for SAP Monitoring	500
9.2	Parameters for Monitoring an SAP System Based on Oracle ...	502
9.2.1	Monitoring the Oracle Database	503
9.2.2	Monitoring the SAP System	517
9.2.3	Monitoring the Hardware and Operating System	533
9.3	Background Jobs in the Scope of Monitoring	538
9.3.1	SAP Standard Jobs	538
9.3.2	Oracle Jobs	543
9.4	Summary	544

10 Backup, Restore, and Recovery545

10.1	What Must Be Backed Up?	547
10.1.1	Interaction of Oracle Processes and Database Objects	549
10.1.2	Operating Modes of the Oracle Database	553
10.1.3	Archiver Stuck	556
10.2	Data Backup Methods	561
10.2.1	Data Export	561
10.2.2	Offline Data Backup	562
10.2.3	Online Data Backup	564
10.3	Recovery Methods	568
10.3.1	Recovery from an Offline Data Backup	572
10.3.2	Recovery from an Online Data Backup	573
10.3.3	Error Scenario: Loss of a Normal Tablespace	574
10.3.4	Partial Restore and Complete Recovery	575
10.3.5	Database Reset	576
10.3.6	Point-in-Time Recovery	577
10.3.7	Full Restore and Complete Recovery	578

10.4	BR*Tools for Backup, Restore, and Recovery	579
10.4.1	Data Backup Using BRBACKUP	582
10.4.2	Backing Up Redo Log Files Using BRARCHIVE.....	592
10.4.3	Restoring Using BRRESTORE	602
10.4.4	Recovering the Database Using BRRECOVER	606
10.4.5	Post-Processing for an Incomplete Recovery	612
10.4.6	BR*Tools and Temporary Tablespaces	614
10.4.7	Disaster Recovery Using BR*Tools	614
10.4.8	BR*Tools in Windows Environments	616
10.4.9	Backup Media and Volume Management	617
10.5	Oracle Recovery Manager (RMAN)	622
10.5.1	Backups Without Backup Library	627
10.5.2	Backups with the SAP Backup Library	628
10.5.3	Backups with an External Backup Library	629
10.6	Other Error Scenarios	629
10.6.1	Loss of a Control File	630
10.6.2	Loss of All Control Files and the System Tablespace and Rollback Tablespace	631
10.6.3	Loss of the System Tablespace and Rollback Tablespace	636
10.6.4	Loss of a Data File of a Temporary Tablespace.....	637
10.6.5	Loss of an Online Redo Log File	639
10.6.6	Loss of an Online Redo Log Group	641
10.6.7	Loss of All Online Redo Log Files	645
10.6.8	Loss of Offline Redo Log Files	646
10.6.9	Database Crash During an Online Backup	647
10.7	Backup Strategies	649
10.7.1	Partial Backups	655
10.7.2	Two-Phase Data Backup	657
10.7.3	Standby Databases	658
10.7.4	Split Mirror Databases	660
10.8	Tips and Tricks	665
10.9	Summary	666

11 Administrating the Java Stack669

11.1	Using Java in SAP Systems	670
11.2	Architecture of the J2EE Engine	673
11.2.1	Internal Structure	678
11.2.2	Interplay of the Java Stack and the Database	686
11.2.3	Monitoring	695

11.3	Java Software Logistics	705
11.3.1	SAP NetWeaver Development Infrastructure	705
11.3.2	SAP Component Model	707
11.3.3	Patching Java Instances and Applications	709
11.4	Tips and Tricks	714
11.4.1	Profile Parameters for the J2EE Engine	714
11.4.2	Parameters of the Property File	717
11.4.3	Minimum Configuration of the instance.properties File	719
11.5	Summary	720

12 SAP NetWeaver BI and Oracle 723

12.1	Basics and Concepts of Data Warehousing	723
12.1.1	OLAP and OLTP	723
12.1.2	Data Warehouse Architecture	724
12.1.3	Extraction, Transformation, Loading	726
12.1.4	Data Structures and Design of a Data Warehouse	728
12.1.5	Operations for Data Analysis	732
12.1.6	Data Mining	732
12.1.7	Benefits	733
12.2	Data Warehousing with Oracle	733
12.2.1	Technology and Architecture	734
12.2.2	Concepts and Language Extensions in Oracle	738
12.2.3	Tools	741
12.3	SAP NetWeaver BI: An Overview	742
12.3.1	Business Content	744
12.3.2	Data Modeling	745
12.3.3	Modeling of Business Intelligence Objects	750
12.3.4	Basics of Data Extraction	752
12.3.5	Loading of Master and Transaction Data	755
12.3.6	Delta Extraction from Source Systems	758
12.3.7	Reporting	760
12.4	SAP NetWeaver BI on Oracle	765
12.4.1	SAP NetWeaver BI Tables and Indices in the Oracle Database	766
12.4.2	Configurations of the Oracle Database	770
12.4.3	PGA and Temporary Tablespace	774
12.4.4	Statistics for SAP NetWeaver BI Tables	778
12.4.5	Oracle Data Miner and the SAP BI Accelerator.....	779
12.5	Summary	782

13 Afterword and Outlook	785
Appendix	787
A Flight Data Model	789
A.1 SQL Script for Creating the Database	789
A.2 Perl Script for Generating the Database Content	790
B General Options of the BR*Tools	791
B.1 Functions of BRSPACE	791
B.2 Functions of BRARCHIVE	792
B.3 Functions of BRCONNECT	793
B.4 Functions of BRBACKUP	794
B.5 Functions of BRRESTORE	795
B.6 Functions of BRRECOVER	796
C References	797
D Authors	799
Index	801

Performance is not everything, but without performance everything is nothing, because most users find waiting almost impossible to bear.

8 Performance

In IT, performance describes a system's ability to carry out a task within a given timeframe. Therefore, performance is always measured in terms of tasks per time, such as FLOPS (*floating-point operations per Second*) or SAPS (*SAP Application Performance Standard*). One hundred SAPS are defined as 2,000 completely finished order items per hour, that is, 6,000 technical dialog steps and 2,000 update processes.

Usually, the transfer of a specific amount of data is the challenge, so that the unit can be described as quantity per time unit. Standardization of units is essential for comparing different systems and their performance. Such comparisons of defined and reproducible performance are referred to as *benchmarking*.

In the real world, it is the user who determines whether the performance of a system is good or poor. It's a matter of personal or subjective perception. A user does not necessarily recognize the scope of a task that has to be processed by a system. Consequently, on the one hand, good performance is an absolute characteristic when comparing systems, and, on the other hand, it has to be regarded in relation to the requirements. In addition, sociological factors play a role in this field: Five seconds of queue time for a data warehouse request is no problem for a user in an enterprise, whereas a customer of a web shop might be less patient.

Therefore, the overall goal of a system administrator must be to meet the different performance requirements to enable users to efficiently use a system.

Performance optimization is part of the lifecycle of every system and involves the steps of *implementation*, *operation*, and *revision*. Figure 8.1 shows a lifecycle including its different phases.

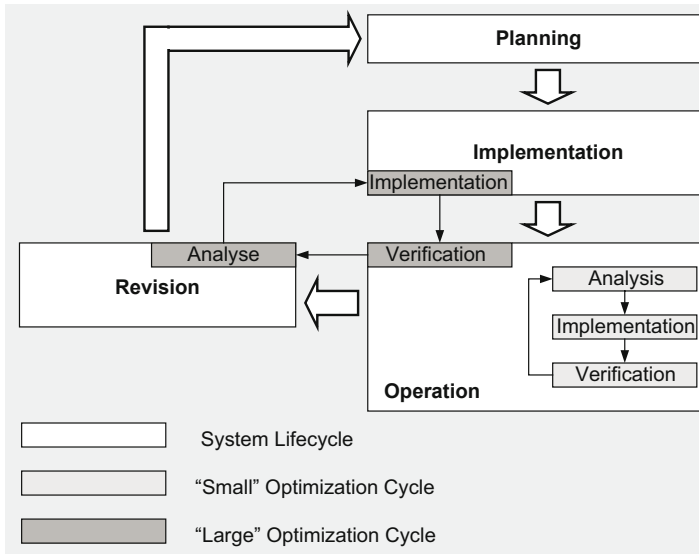


Figure 8.1 System Lifecycle and Optimization Cycles

The process of optimization is divided into the phases of *analyzing*, *implementing*, and *verifying*. There are two optimization cycles. The first, "small," cycle is carried out entirely in the lifecycle phase of *operation*. This is the phase of performance optimization, which has no or only short time effects on a system's availability. Examples of these processes include SAP kernel parameters for memory areas or work processes as well as the parameters of the Oracle database, some of which can be dynamically changed during runtime.

The "big" cycle covers the lifecycle phases of *implementation*, *operation*, and *revision*. Optimizations that require more time to be completed, as they involve tests and affect the system operation, must be performed within this wider context. Modifications on application code or extensive reorganizations of databases are also part of this cycle.

This chapter describes the *analysis*, *implementation*, and *verification* phases: Which analyses can be performed on a combination of SAP with an Oracle database and how can the results be implemented? To answer this question, we should first categorize the problems.

8.1 Administrative and Program-Based Problems

There are three sources of performance problems: program-based, administrative, and user-specific problems.

If performance problems are caused by the code of an application, the cause is *program-based*. Unfortunately, there are many examples of poorly coded software, as coding involves a wide range of bug types, such as memory leaks and inefficient algorithms, to name just a few. If, on top of that, a database is used, the probable bugs cover SQL statements that are required for the interaction. The issue of program-related performance problems is covered in Section 8.3, *Analyzing Program-Based Performance Problems*.

Administrative performance problems are caused by the configuration of hardware and software. This covers a wide range of areas, from incorrect disk layout to insufficient storage parameters of the database and the SAP system to the incorrect assignment of permissions. The methods used to analyze and solve these problems are described in Section 8.2, *Analyzing Administrative Performance Problems*.

The third source for possible problems is the behavior of users, in other words, *user-specific* causes. In this context, the problematic question is: Who caused the problem? The user who, for example, runs extensive queries and therefore causes the system performance to go down, or the programmer or administrator who does not prevent different kinds of "excessive use," by setting maximum values for input boxes or running plausibility checks? User-specific performance problems are not further discussed in this book. The solution to this type of problem is not the administration of SAP and Oracle databases but the development of applications or the administration of user permissions.

Besides the causes of the problems, the *locations of problems* represents the second part of a problem analysis, in this context, location means: Where does the performance problem occur? For a further specification of this issue, a system must be divided into its individual components. From a performance analysis point of view, an SAP system consists of the following components:

1. Hardware
2. Operating system
3. Database
4. SAP Basis (that is, SAP Kernel + SAP Basis = SAP NetWeaver Application Server)
5. SAP application

Table 8.1 shows an overview of the possible combinations of cause and location for the assignment of performance problems. Note that this chapter focuses only on the problematic points related to Oracle and SAP.

Cause\Location	Program-Based	Administrative
Hardware	Errors in firmware	Selection of inappropriate components, such as slow hard disks or storage
Operating system	<ul style="list-style-type: none"> ▶ Inefficient storage management by operating system kernel ▶ Device drivers not optimized 	<ul style="list-style-type: none"> ▶ Incorrect parameterization of the operating system kernel ▶ Inappropriate layout for massive I/O operations
Database	<ul style="list-style-type: none"> ▶ Use of "expensive" SQL statements ▶ Poor indexing 	<ul style="list-style-type: none"> ▶ Insufficient parameterization ▶ Inefficient table structures, for instance, because of too many extents
SAP Basis	Errors in the ABAP code for basic functions such as in communication components	<ul style="list-style-type: none"> ▶ Inefficient buffer sizes ▶ Wrong parameters for the SAP kernel
Application	<ul style="list-style-type: none"> ▶ Incorrect use of standard SAP functions in custom developments ▶ Bugs in the standard SAP system 	None

Table 8.1 Overview with Examples of Performance Optimization Problems

We will now continue this chapter in two parts. The first part, Section 8.2 *Analyzing Administrative Performance Problems* deals with administrative performance problems in all fields including hardware, operating system, the Oracle database, and the SAP system, with a particular focus on Oracle and the SAP system, according to the intention of this book. Then, Section 8.3, *Analyzing Program-Based Performance Problems*, deals with the program-related issues, such as expensive SQL statements, indexing, and to a smaller extent, ABAP programming.

8.2 Analyzing Administrative Performance Problems

For an analysis of performance problems, two points of entry are quite useful: the *general analysis of the system status* or of the components and the *workload analysis*. The analysis of the components, which is also referred to as the system analysis, refers to the state, that is, the configuration and utili-

zation of the system components, such as hardware, operating system, database, and the SAP Basis. The most important key figures for this type of analysis are filling levels, hit ratios, error statistics, and so on. Therefore, this section deals particularly with the examination of the components, gives information on how to solve problems, and provides configuration reference values.

The SAP workload analysis uses the times required for processing the individual dialog steps (roll-in and roll-out, database time, CPU time, and so on). These time values are collected in the system. This analysis used to monitor not only the components of the system, but also their interaction. Time is obviously the relevant key figure is in this context.

Experience has shown that starting with the workload analysis is useful when individual users complain about performance problems or when the problem occurs only at certain times. If the performance is generally poor or if a system analysis is carried out on a regular basis, starting with a general component analysis is preferable. For a complete analysis of the system performance, you should carry out both analyses.

The SAP workload analysis is not further covered here, as this would go beyond the scope of this chapter. Some excellent literature is already available on this topic, such as the following: *SAP Performance Optimization* by Thomas Schneider (SAP PRESS 2006).

A similar performance analysis method is available in the context of Oracle databases: the *wait event analysis*. As the name suggests, the flow of a transaction is analyzed on the basis of the different wait times of that transaction within a database. Section 8.2.2, *Analyzing the Database*, provides further information about the wait event analysis.

8.2.1 Analyzing the Hardware and Operating System

From the point of view of an SAP Basis administrator, no clear separation can be drawn when analyzing the hardware and the operating system, as the SAP Basis does not allow for that. However, such a separation is not necessary, as both the hardware and operating system form the basis of the SAP system and database and can therefore be viewed as one entity. If a performance problem is detected in the hardware or operating system, the system administrator or hardware partner often participates in the process of finding a solution, as the SAP partner normally does not have any access to the operating system level or lacks the required knowledge.

All data that is available for a hardware and operating system analysis in the SAP system is collected by the *SAP OS Collector* (SAPOSCOL), which is a component of the SAP Kernel that depends on the hardware and operating system. A background job (SAP_COLLECTOR_FOR_PERFORMANCE) reads the data and writes it to the performance database of the SAP system (Table MONI).

The analysis is started via the operating system monitor, which uses the data from the performance database or queries the SAPOSCOL directly. Transaction ST06 starts the OS monitor for the local instance of the system. For systems that have several instances on different servers, Transaction OS07 is used to navigate to the corresponding operating system monitor of an instance that is installed on a different server.

In general, the performance checks of hardware and operating system focus on four areas: CPU, memory requirement, I/O load, and network. All data for these components are collected by the SAP OS Collector and stored in the SAP database using a batch job. This allows access to current data as well as to a data history. Figure 8.2 shows the operating system monitor of an SAP instance.

```

Mon Jun 26 13:18:19 2006 interval 10 sec.
CPU
-----
Utilization user %      1 Count
              system %  1 Load average 1 min  0,1
              idle %    97          5 min  0,1
              io wait %  1          15 min 0,1
System calls/s      2.371 Context switches/s 30
Interrupts/s       2.443

Memory
-----
Physical mem avail Kb 33.538.048 Physical mem free Kb 396.18
Pages in/s           0 Kb paged in/s
Pages out/s          0 Kb paged out/s

Swap
-----
Configured swap Kb 54.525.952 Maximum swap-space Kb 88.064.00
Free in swap-space Kb 74.378.252 Actual swap-space Kb 88.064.00

Disk with highest response time
-----
Name          c7t6d0 Response time ms 1
Utilization   1 Queue
Avg wait time ms 0 Avg service time ms 1
Kb transferred/s 13 Operations/s

Lan (sum)
-----
Packets in/s 74 Errors in/s
Packets out/s 75 Errors out/s
Collisions 0

```

Figure 8.2 Operating System Monitor (ST06 or OS07)

Table 8.2 provides information about the meaning of the most important key figures in the operating system monitor and states critical performance limits where possible or wherever it makes sense.

Field	Description	Critical Value
Utilization user	CPU load caused by user processes including SAP system and database	$S > 80\%$ (Ø per h)
Utilization system	CPU load caused by operating system kernel	
Utilization idle	CPU idle	$<20\%$ (Ø per h)
Utilization i/o wait	CPU load caused by waiting for I/O operations	$>40\%$ (Ø per h)
Count	Number of CPUs	–
Load average	Number of processes waiting for a CPU	>3.0 (specific OS, such as Solaris, also count the active processes, then $>3 + \text{number of CPUs}$)
Phy. mem avail	Complete main memory of the server	–
Phy. mem avail	Free memory of the server	$<3\%$ Phy. mem avail (except AIX, which uses the free RAM as file cache)
Pages in/out	Number of memory pages paged in and out between RAM and swap	Windows: $\text{Kb-in} \times 3600 > 20\% \text{ RAM}$ UNIX: $\text{Kb-out} \times 3600 > 5\% \text{ RAM}$
Kb in/out	Size of memory pages paged in and out between RAM and swap	–
Swap-space (Free, Maximum, Actual)	Display depends on operating system (SAP Note 63906)	–
Disk	Hard disk with currently highest response time (menu path Detail analyses menu • Disk)	Utilization $> 50\%$ (Ø per h)
Packets in/out	Number of sent and received network packets (total of all network interfaces)	–
Errors in/out	Error when sending and receiving network packets (total of all network interfaces)	Should no longer occur with the current state of technology; therefore should be checked in the case of >0
Collision	Collisions on the network	

Table 8.2 Overview of Operating System Monitor

The critical values are not absolute values; rather, they indicate problems. If one of these values is exceeded or fallen below, you should check further.¹

8.2.1.1 Reference Values for Hardware Components

A CPU utilization of more than 80% per hour (also **idle + i/o wait** < 20%) is referred to as a CPU bottleneck. Many hardware partners, however, recommend a maximum utilization of 60% to 70% for production systems to ensure sufficient reserves for peak loads. However, you should keep in mind that the CPU values in Transaction ST06 are average values across all CPUs of the server; that is, a machine with two CPUs with a utilization of 70% has less reserve than a machine with eight CPUs with a utilization of 80% (at equal CPU performance).

Different values are available regarding the size of a swap memory. SAP recommends using three times more swap memory than physical memory, but at least 3.5 GB. This recommendation, however, is unrealistic for systems with a memory of more than 64 GB. In that case, it is difficult to reserve the appropriate amount of swap memory on the local disks. However, the operating systems often provide the corresponding solution, such as the use of a pseudo-swap for HP-UX.

Paging, that is, outsourcing memory pages from the memory and transferring them to the swap partition or the swap file on the hard disk, should generally be regarded as critical. The swap memory is merely a kind of emergency help for the operating system in order to be able to start more processes than the existing memory allows and to prevent processes from failing in situations of extreme memory load. With paging, you should always bear in mind that, theoretically, the factor that's responsible for the difference in access speed between the hard disk and RAM is approximately 500,000 (8 milliseconds for the positioning of the hard disk head – 15 nanoseconds of latency time for memory access). Although these values are only theoretical values that can be changed considerably by employing different hardware techniques, such as hard disk arrays or parallel memory access, a considerable difference still remains.

Generally, we advise that you not page out more than 5% of the memory within one hour. The best thing, however, is to entirely avoid paging and to size the memory according to your specific requirements. As a rule, when

¹ Five percent can also be a poor result when the RAM is larger than 8 GB or the I/O for swap memory is too slow, for example due to a software RAID.

planning hardware resources, the memory should be of the highest importance.

The third component is the I/O load. When you double-click on the current **Disk with highest response time** in the operating system monitor, a list with all hard disks of the system displays including their current statuses. If a hard disk is indicated with **Utilization 100%**, this does not necessarily mean that there's a bottleneck. In fact, you should merely ensure that the average **Utilization** per hour does not exceed 50%. The history of the I/O load of each hard disk is displayed under **Detail Analysis Menu • Disk**.

The network can be checked from within the SAP system using a simple ping test. This LAN check of the presentation servers (SAP GUI) only works if the servers don't access the system via an SAP router. A second and much better way of checking the network is to use the `niping` program, which can be called from Transaction SM49 as an external operating system command. SAP Note 500235 contains detailed instructions on how you can use `niping`.

The operating system monitor displays the number of received network packets per second and provides a summary on an hourly basis. Critical points are the sent errors and collisions within a period of one hour. With today's modern network structures in a LAN, every value above 0 is suspicious and should be checked together with the network administrator (be insistent).

8.2.1.2 Identifying the Causes of Bottlenecks in Hardware Components

If a CPU, memory or I/O bottleneck is detected, you must search for the cause of the bottleneck in a second step.

The processes at the operating system level are responsible for the CPU utilization. In Transaction ST06, the current processes of the server are listed in the order of their CPU utilization under **Detail Analysis Menu • Top CPU**. The displayed CPU utilization percentage always refers to a CPU of the system; that is, in a system with multiple processors (n CPUs), the maximum utilization is $n \times 100\%$. If it is possible, you can also use the tools provided by the operating system, such as "top" that's available in the different UNIX derivatives.

The further procedure depends on the processes that are identified as CPU users. SAP work processes can be recognized by the `<sid>adm` user and the process names, `dw.sap<instance>` (UNIX) and `Disp+Work` (Windows). If

these are the processes that produce the CPU load, these are further analyzed against the SAP process overview (Transaction SM50; see Section 8.2.3, *Analyzing the SAP System*). Individual processes are identified by a process ID (PID), which is displayed in the operating system monitor and in the SAP process overview.

Oracle processes are typically executed under the ora<sid> (UNIX) or SAPService<sid> (Windows) user, respectively, and have the identifier ora<sid> (shadow processes) or ora_<process>_<sid> (Oracle system process). If one of the Oracle processes utilizes an unreasonably high CPU capacity, further analysis is performed using the database process monitor (Transaction ST04N; see Section 8.2.2, *Analyzing the Database*). Different reasons exist for an extensive CPU utilization by the Oracle processes; refer to SAP Note 712624 for further information.

Important Note

This book mainly refers to the application server of SAP Releases 6.40 and 7.00. Transaction ST04N, which will be mentioned frequently in the following sections, will no longer be available in the coming Release 7.10. In the new release, it will be called ST04 again. Moreover, from Release 7.00 onward, Transaction DBACOCKPIT for Oracle is available.

If an external process causes the high CPU load, this process has to be analyzed, and the bottleneck has to be eliminated in collaboration with the operating system administrator.

Analyzing the memory utilization by the processes is much more complicated than the analysis of the CPU utilization. On the one hand, that's because the memory is used in various different ways, for instance, as a local process memory or as a shared memory; on the other hand, the different operating systems use different methods of memory management.

There are considerable differences regarding the management of swap and memory – not only between the Windows and UNIX worlds but also between the different UNIX derivatives. In general, in-depth knowledge about the operating system is essential for a precise analysis.

First, you must use the options provided by the operating system to determine whether the memory utilization is caused by the SAP system or the Oracle database or by other processes. External and high memory utilization – which is not caused by SAP or Oracle – is often caused by the file system cache, which reserves a particular percentage of the memory as a buffer for

data access. The maximum size that can be recommended depends on whether the server runs both the Oracle and SAP instances or only the SAP instance. Oracle generally recommends disabling all I/O buffers for the database access, as the access to database blocks then won't be buffered twice, namely, by the file system cache and the SGA memory. Instead, the memory should be completely allocated to the Oracle-optimized SGA buffer. If you use raw devices, you can't use the file system cache. The cache of a server with an SAP instance should not exceed 8% to 10% of memory, but no more than 1 GB. In AIX, you should also use the file system cache as little as possible. Refer to Section 8.2.2 for further information on the different types of I/O.

If the memory utilization occurs within the SAP system, that is, with the SAP work processes, a further analysis of the SAP memory areas is executed. Note that the SAP system is only capable of allocating the memory in accordance with the relevant instance parameters (see Section 8.2.3). The analysis of the memory usage of the Oracle database should be performed in the same way (see Section 8.2.2).

There are three possible causes for a high I/O load: massive paging in the swap area, a high load on the database, or an external program. If you recognize a high paging rate in the operating system monitor, you can use the disk analysis (**ST06 • Detail Analysis Menu • Disk**) to verify if the hard disk that contains the swap area has a high load. If that is the case, solving the paging problem also solves the problem with the I/O load. Because the swap area is never located on the same hard disk as the database, an I/O problem caused by paging usually never causes any I/O performance problems in the database.

If the high load occurs in the area where the database is installed, further analysis is required (see Section 8.2.2).

Problems with the communication hardware can theoretically occur for three connections:

- ▶ SAP instance: SAP GUI
- ▶ SAP instance: Oracle database server
- ▶ SAP instance: connected systems

The connection to the database server and possibly to the connected third-party system, for example, as a data source in SAP NetWeaver BI, has a particularly high bandwidth utilization. In this context, SAP requires the SAP

instance and the database server to be located together in a 100-Mbit LAN. A network overload can only be recognized in the SAP system by means of indications such as collisions and long runtimes in the LAN. The SAP system administrator can perform further checks using the `niping` program. A precise analysis and identification of causes requires the use of external network tools and the profound knowledge of a system administrator.

8.2.2 Analyzing the Database

Because the database is the core of the SAP system, its performance is essential for the performance of the entire system.

The analysis includes the following performance-relevant components:

▶ **Buffers**

The buffer areas of the Oracle database store frequently used information in the main memory of the server to provide a considerably faster access than is made possible by the hard disk storage.

▶ **Wait event**

The analysis of wait events indicates when and which event the database has to wait for during the processing of a request. This is a relatively simple way to identify bottlenecks in the database.

▶ **General parameterization**

In addition to the buffer parameters, there are many other performance-relevant Oracle parameters. These must also be included in the complete analysis.

▶ **Statistics**

The Oracle Cost-Based Optimizer (CBO) calculates the costs of the potential access paths (for example, full scan, index range scan) to determine the fastest possible access path.

▶ **I/O**

The task of a database is to read and write data blocks. Therefore, having the best possible I/O for the performance of an SAP system is essential.

▶ **SQL analysis**

The quality of SQL queries affects the speed of the database significantly. Consequently, the identification and enhancement of bad SQL queries represent an important task in the context of performance optimization.

When is an analysis of the database useful? The amount of the total response time for the database in an SAP system is the best indicator. Usually, you can

use a workload analysis to determine this amount, which should normally be less than 40%. A further indicator is the ratio between **Busy wait time** and **CPU time** (Transaction ST04N; see Figure 8.4), which should be approximately 60:40. If the **Busy wait time** is considerably higher, you can perform a wait event analysis (see Section 8.2.2.2, *Analyzing Wait Events*), whereas an increased **CPU time** indicates a CPU resource bottleneck. Another factor to be taken into account regarding the **Busy wait time** is the following: Possibly, Transaction ST04 also includes idle wait events in the **Busy wait time**. For this reason, you should check the correctness of the **Busy wait time** using the data from the V\$SYSTEM_EVENT view.

As a prerequisite for the analysis of Oracle performance data, you must set the TIMED_STATISTICS parameter to TRUE. However, this is already the case after a standard SAP installation. Otherwise, you can set this parameter dynamically as a SYS database user (logon via sysdba):

```
f05:oram05 1>sqlplus "/as sysdba"
SQL> alter system set TIMED_STATISTICS = TRUE;
```

8.2.2.1 Analyzing the Database Buffers

Two factors determine the quality of a buffer: On the one hand, there is the logical access (*logical read*), which refers to every access to a block, and the physical access to a block on the hard disk (*physical read*). Figure 8.3 illustrates this concept.

The buffer quality can be calculated based on these factors by using the following formula:

$$\text{Quality (hit ratio)} = \text{Number of hits} / \text{Number of queries} \times 100\%$$

Basically, you must consider that all buffers are initialized after a system startup and therefore have no informational value. For a buffer analysis, the system has to be in an *established* state. In general, you can assume that this state is reached after one or two days of operation. The number of logical reads on the buffer cache of the database is another reference value for the established state. This value should be greater than 50,000,000.

The database buffers of the Oracle database are located in the system global area (SGA). You can find a detailed description of the individual buffer areas and their functions in Chapter 3, *Oracle Fundamentals*.

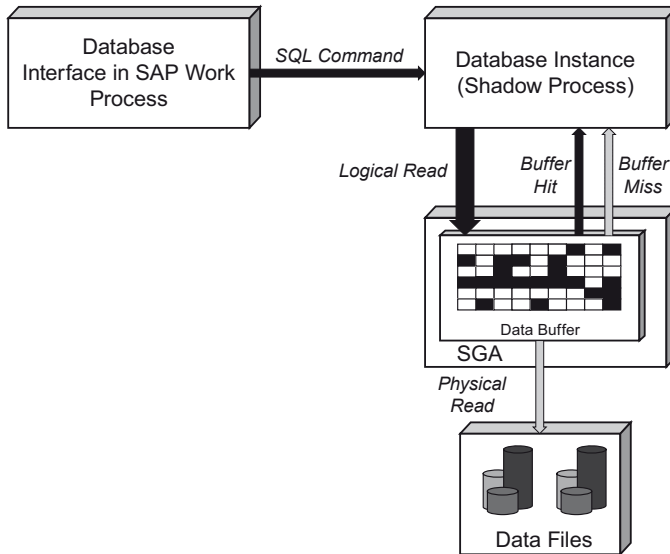


Figure 8.3 Access to Buffers and the Database

The overview in Table 8.3 includes the most important buffers in the SGA of the Oracle database.

SGA Buffer	Description
Data buffer	Contains the buffered data blocks from the data files on the hard disk. Parameter: DB_CACHE_SIZE
Shared pool	The two main subcaches: data dictionary cache and library cache. Parameter: SHARED_POOL_SIZE
Java pool	Used by the Oracle JVM, but not by the SAP system. Parameter: JAVA_POOL_SIZE
Large pool	Buffer for special data (for example, message buffer for processes running parallel queries). This buffer is very small and hardly used in SAP systems. Parameter: LARGE_POOL_SIZE
Streams pool	New buffer area in Oracle 10g for Oracle Stream, which manages data and events in distributed environments — not used in the SAP system. Parameter: STREAMS_POOL_SIZE
Redo buffer	Buffer for redo log data. Parameter: LOG_BUFFER

Table 8.3 Overview of SGA Buffers

As of Oracle Version 9i, the administrator can change the most important parameters (DB_CACHE_SIZE and SHARED_POOL_SIZE) of the SGA at runtime

of the Oracle instance. This feature is referred to as dynamic SGA and should not be confused with the *Automatic Shared Memory Management (ASMM)*. The old parameters from Oracle 8.1.x for the data buffer (`DB_BLOCK_BUFFERS`) can no longer be used. SAP has generally approved the use of the dynamic SGA, which is enabled by default during SAP installations based on SAP Basis 6.40.

As of Oracle 10g, you can fully automate the SGA management function. In that case, Oracle adjusts the individual areas, `DB_CACHE_SIZE`, `SHARED_POOL_SIZE`, `JAVA_POOL_SIZE`, `LARGE_POOL_SIZE`, and `STREAMS_POOL_SIZE`, to your current requirements. The `SGA_TARGET` parameter provides the total size of the SGA and enables the ASMM. Moreover, if the `DB_CACHE_SIZE` and `SHARED_POOL_SIZE` parameters are set, they provide the lower limits for each buffer area. Due to the lack of experience with ASMM, its use in an SAP system is not recommended. Nevertheless, it makes sense to use this parameter in a nonproduction environment to minimize the administrative effort, but only if you do not intend to use the system as an image of the production system for testing purposes.

Access to analysis data in the SAP system occurs via the Oracle database monitor (Transaction ST04N). This monitor provides you with all information about the Oracle database, which can be accessed from within the SAP system. The information about the database monitor originates from the Oracle database, specifically from the V\$ views. Figure 8.4 shows the initial screen of the database monitor.

Table 8.4 explains the meanings of the most important key figures in the Oracle database monitor and provides recommendations for its optimal states after establishing the database.

In general, recommendations for particular buffers and characteristics are only reference values. The values may deviate significantly without affecting the performance of the SAP system. The user load on a training system, for example, barely reaches the user load of a production system, so the load that's generated by administrative tasks clearly prevails. Because these activities, such as the standard SAP background jobs and the creation of the database statistics, focus on different load aspects, the individual performance characteristics vary substantially. In this case, a ratio of User calls to Recursive calls or <0.5 would not be unusual.

Data Buffer			
Size (kB)	1.048.576	Logical reads	5.210.486.341
Quality (%)	98,9	Physical reads	58.536.384
		Physical writes	3.291.932
		Buffer busy waits	103.503
		Buffer wait time (s)	173

Shared pool		Log buffer	
Size (kB)	770.048	Size (kB)	1.292
DD-cache Quality (%)	88,0	Entries	46.706.977
SQL area getratio(%)	95,9	Allocation retries	1.758
SQL area pinratio(%)	99,3	Alloc fault rate(%)	0,0
SQLA.Reloads/pins(%)	0,0432	Redo log wait (s)	46
		Log files (in use)	8 (8)

Calls			
User calls	91.548.413	Recursive calls	111.978.800
User commits	2.564.304	Parse count	8.738.674
User rollbacks	1.788	User/recursive calls	0,8
		Log_Reads/User Calls	56,9

Time statistics			
Busy wait time (s)	97.502	Sessions busy (%)	0,19
CPU time session (s)	65.881	CPU usage (%)	0,17
Time/User call (ms)	2,00	Number of CPUs	8

Redo logging			
Redo writes	3.300.996	Redo write time (s)	17
OS blocks written	24.900.682	MB written	22.095
Latching time (s)	2		

Figure 8.4 Oracle Database Monitor – Main View

Buffer/ Characteristic	Description	Recom- mendation
Data buffer	Main database buffer for the data blocks (warning: This recommendation is very general, because there are extreme cases in both directions, which means there are systems running with 80% without problems and systems having serious problems at 98%.)	>94%
DD cache	Data Dictionary buffer contains metadata of the database (structures, users, authorizations)	>80%

Table 8.4 Essential Characteristics of the Oracle Database Monitor

Buffer/ Characteristic	Description	Recommendation
SQL area get ratio	SQL cache stores the parse tree and execution plan of previously run SQL statements Get ratio = $S(\text{hit})/S(\text{request}) \times 100$	>95%
SQL area pin ratio	Indicates the quantity of all requests (in percent) that still have required objects to be executed in the memory: Pin ratio = $S(\text{executions} = \text{pin hits})/S(\text{requests for execution} = \text{pins}) \times 100$	>99%
Reloads/Pin	Ratio between reloads of a SQL statement (for example, invalidated entry due to age) and execution requests	<0.04
User/Recursive calls	Ratio between user requests to the database and requests that the database executes in addition to the user requests (for example, due to a missing dictionary cache entry)	>2
Busy wait time	Total amount of all wait times of the database in terms of seconds, without idle events (see Section 8.2.2.2, <i>Analyzing Wait Events</i>)	Ratio approx. 60:40
CPU time	Total amount of CPU time consumed by all Oracle sessions	

Table 8.4 Essential Characteristics of the Oracle Database Monitor (Cont.)

In the following text, we will look more closely at the relevant Oracle buffers in the SAP environment with regard to performance.

Without a doubt, the *data buffer* for the actual data blocks of the database has the greatest impact on performance because it reduces the total number of physical disk accesses.

The logical reads include all reading requests to the database. During a *buffer get*, the system tries to read the corresponding data block from the data buffer for all requests that are not declared as *direct path*, which means they don't have an explicit direct access to the database. A successful read access is referred to as a *buffer hit*, whereas a failure leads to a physical read in which the block is read from the data files on the hard disk (see Figure 8.4).

The hit ratio for the data buffer can be calculated as follows:

$$\text{Quality (hit ratio)} = (\text{Logical reads} - \text{Physical reads})/\text{Logical read} \times 100\%$$

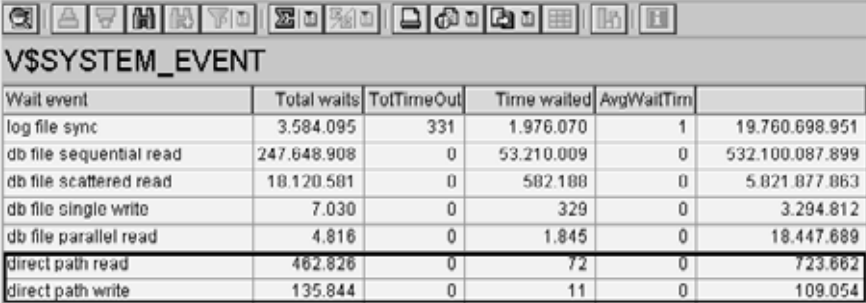
To obtain a good database performance, the buffer should process at least 94% of all block accesses (except for direct path operations) to the database. You must check the following possible causes if the actual value falls below this reference value:

- ▶ The data buffer is too small
- ▶ Many direct path operations that circumvent the data buffer when accessing data blocks (Note: Direct path operations are deducted from the hit ratio when displayed in ST04N, but not in ST04. Therefore, there may be differences.)
- ▶ Expensive SQL statements (see Section 8.3)

The direct path operations include the following wait events:

- ▶ `direct path read` and `direct path write`
- ▶ `direct path read (lob)` and `direct path write (lob)` (Oracle 9i)
- ▶ `direct path read temp` and `direct path write temp` (Oracle 10g)

You can view the number of `direct path` operations in Transaction ST04N under **Additional Function • Display V\$ • V\$SYSTEM_EVENT** (see Figure 8.5). This number should be very small (<0.5%), primarily in comparison with the number of regular accesses to data blocks via the data buffer (`db file sequential read`). SAP NetWeaver BI systems are an exception in this case, because substantially higher values are acceptable here (see Chapter 12, *SAP NetWeaver BI and Oracle*). You can use direct path operations, for example, to access the PSAPTEMP tablespace. As an example, increasing the PGA memory of individual database work processes can help you reduce the number of these accesses to the temporary tablespace for `JOIN` or `SORT` operations. Figure 8.5 shows an excerpt of the `V$SYSTEM_EVENT` view. Another note regarding this view: This view contains only the wait events that occurred after the last database startup. You shouldn't be surprised if you don't see all of the wait events described above in this excerpt.



Wait event	Total waits	TotTimeOut	Time waited	AvgWaitTim	
log file sync	3,584,095	331	1,976,070	1	19,760,698.951
db file sequential read	247,648,908	0	53,210,009	0	532,100,087.899
db file scattered read	18,120,581	0	582,188	0	5,821,877.863
db file single write	7,030	0	329	0	3,294.812
db file parallel read	4,816	0	1,845	0	18,447.689
direct path read	462,826	0	72	0	723.662
direct path write	135,844	0	11	0	109.054

Figure 8.5 Direct Path Operations in `V$SYSTEM_EVENT`

If you can exclude expensive SQL statements and `direct path` operations as a reason for a poor hit ratio, you should, if possible, try to improve the per-

formance by increasing the data buffer. If you can only implement this increase by extending the hardware, you should first exclude all possible causes for a performance degradation before making a corresponding investment.

The initial configuration of the data buffer depends completely on the intended use and the load on your system. In real life, many SAP production systems work with data buffers of more than 100 GB. Therefore, we cannot provide a general recommendation at this point. We would rather recommend that you have SAP experts perform sizing sessions.

Since the introduction of the dynamic SGA with Oracle 9i, the Oracle administrator can test changes to the data buffer in a simple and convenient way. The `V$DB_CACHE_ADVICE` view enables you to check how a change to the buffer size affects the number of physical reads. The factor representing the changes between the physical database accesses and the current status represents the possible reduction of the buffer in MB without a significant performance degradation or the efficient expansion of the buffer in MB to further minimize the physical reads. For this purpose, you must enable the dynamic SGA and set the Oracle parameter, `DB_CACHE_ADVICE`, to `ON`.

In addition to the actual data buffer, the *keep pool* and the *recycling pool* also buffer data blocks. If you use the dynamic SGA (Oracle 9i), you can see that these two pools are no longer part of the data buffer but are included separately in the SGA. The keep pool can be used for tables and blocks that should not be displaced from the data buffer. The recycling pool, on the other hand, can be used for tables that should not displace other blocks from the data buffer but whose own blocks can be displaced immediately. The standard settings in SAP do not use these pools; however, their usage is recommended under specific circumstances (see SAP Note 762808).

Apart from the data buffer, other important buffers of the Oracle database are located in the *shared pool*, namely, *SQL cache* and *dictionary cache*.

The SQL cache (formerly known as *shared cursor cache*) is located in the library cache and stores all Oracle-internal information for later reuse, if required. This information is related to an SQL statement call, such as the parse tree and the execution plan.

Another key figure for the SQL cache in the shared pool is the *pin ratio*. You can calculate the pin ratio as follows:

$$\text{Pin ratio} = (\text{Pin hits}/\text{Pin}) \times 100\%$$

Oracle processes an SQL statement by splitting it up into different components. A pin is the request for the reuse of one of the components resulting from the decomposition of the SQL statement, and, accordingly, a *pin hit* is the successful reuse. However, this is not always the case because cache entries may be invalidated by timeouts or displaced by other entries. If the described reuse fails, the system must reload the corresponding SQL command with its new components. The "Reloads per pin" key figure results from the relationship between requests (pins) and reloads (see Figure 8.4 and Table 8.4 above).

A buffer hit in the SQL cache simply means that the parsing of the queried SQL statement was already performed. However, the pin ratio indicates the number of successful reuses for a found cache entry. If the reuse fails, the system reloads the corresponding component.

In the Oracle library cache, you can find further subcaches for the PL/SQL (Procedural Language/Structured Query Language) packages as well as for the control structures, such as locks and library cache handles. These play only a minor role regarding the system performance.

The dictionary cache buffers rows from the dictionary of the Oracle database, that is, information about structures of tables, authorizations, and so on. This metadata of the database is needed regularly to process user requests.

According to SAP, the minimum size of the shared pool should be approximately 400 MB. If the hit ratio values are permanently under the values listed in Table 8.3, it may be useful to increase the value of the `SHARED_POOL_SIZE` parameter. However, you should take into account that, for instance, the structure of the database statistics may temporarily decrease the hit rate in the shared pool significantly.

Possibly, you can also minimize the shared pool again if the performance values (see Table 8.4) are acceptable and a larger subarea of the shared pools remains free (>50 MB). You can find the free area in the shared pool in Table **V\$SGASTAT • free memory** or by using the following SQL command:

```
Select bytes from V$SGASTAT
where pool = 'shared pool' AND name = 'free memory';
```

In the `V$SHARED_POOL_ADVICE` view, you can also see how changes to the shared pool size affect the cache hits and then resize the pool accordingly.

As of Oracle 10g, you have the option to display the history of the load of the shared pool. Furthermore, the `DBA_HIST_SGASTAT` view displays the progress of the free space development.

The *Program Global Area* (PGA) component of the Oracle memory is locally assigned to a server process (shadow process or background process). The entire PGA memory of an Oracle instance can be calculated based on the amount of PGAs of all database processes. You can find the total amount of allocated PGA memory in Transaction `ST04N` or under **Additional Function • Display VS/GV\$ Views and Values • V\$PGASTAT • total PGA allocated** or by using the following SQL call:

```
Select VALUE from V$PGASTAT
where name = 'total PGA allocated';
```

The PGA of a process contains only the data and information that is needed or to be processed. The size of the PGA plays a particularly important role for memory-intensive sort and hash operations. Consequently, the administrator should place special emphasis on the optimum configuration of this memory, in particular in the SAP NetWeaver BI environment (see Chapter 12).

However, the configuration of the PGA has been considerably simplified since Oracle Release 9i. Whereas the administrator of older Oracle versions had to specify the PGAs for individual operations (for example, `SORT_AREA_SIZE` and `HASH_AREA_SIZE`), you can now use the automatic PGA management function. Similar to the ASMM for the SGA, Oracle adjusts the PGAs for all server processes automatically. It is worth mentioning that, in contrast to earlier versions, PGA memory that is no longer needed is released using the automatic PGA management. You can limit the entire size of the PGA using the `PGA_AGGREGATE_TARGET` parameter. As of Oracle 9i, the automatic PGA management can be used (according to SAP Note 619876, it is even recommended) and is enabled by default in every SAP installation with SAP Basis 6.40 or higher.

To better understand the PGA tuning settings, we will now introduce some terms. To execute an operation, the Oracle process needs local memory, the *work area*. If the available PGA memory for the process is sufficient for the entire work area, we refer to this as an *optimal work area size*, and the corresponding operation is called *optimal execution*. If the PGA is not sufficient, the operation uses the temporary permanent storage (`PSAPTEMP`). The resulting I/O activities (direct path operations without buffering) have a sig-

nificantly negative impact on the system's performance. If the first pass (first recursion level) of the PSAPTEMP is successful, we refer to it as a one-pass operation. If the PSAPTEMP is used for several passes, it is called multi-pass operation.

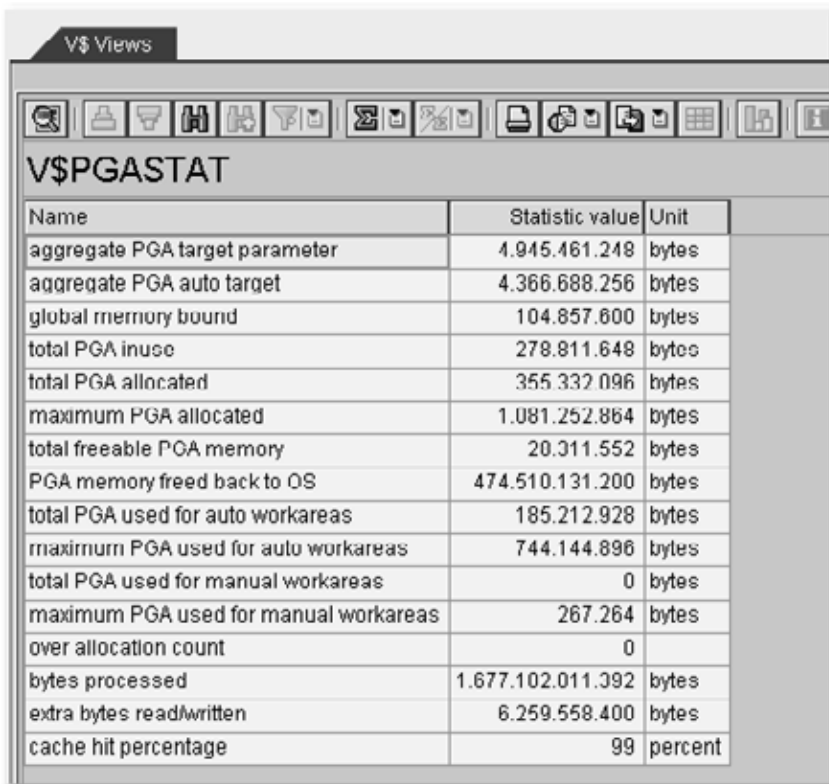
There are different indicators to determine if the configured PGA memory is too small. First, you should verify the following values in the V\$PGASTAT view (this applies to the automatic PGA management; see Figure 8.6):

► **Over allocation count**

Specifies how often the PGA memory was insufficient. This value should be around 0; otherwise, you must extend the PGA.

► **Cache hit percentage**

Specifies the number of hits for the optimal work area size. Ideally, if the value is 100%, no one-pass or multipass operations exist. This value should be >70% for a normal ERP system and >90% for a BI system.



Name	Statistic value	Unit
aggregate PGA target parameter	4.945.461.248	bytes
aggregate PGA auto target	4.366.688.256	bytes
global memory bound	104.857.600	bytes
total PGA inuse	278.811.648	bytes
total PGA allocated	355.332.096	bytes
maximum PGA allocated	1.081.252.864	bytes
total freeable PGA memory	20.311.552	bytes
PGA memory freed back to OS	474.510.131.200	bytes
total PGA used for auto workareas	185.212.928	bytes
maximum PGA used for auto workareas	744.144.896	bytes
total PGA used for manual workareas	0	bytes
maximum PGA used for manual workareas	267.264	bytes
over allocation count	0	
bytes processed	1.677.102.011.392	bytes
extra bytes read/written	6.259.558.400	bytes
cache hit percentage	99	percent

Figure 8.6 Characteristics of the PGA

The `V$PGA_TARGET_ADVICE` view displays how a change to the `PGA_AGGREGATE_TARGET` parameter affects the PGA quality (over allocation count and cache hit percentage).

The `V$SQL_WORKAREA_HISTOGRAM` view displays the frequency and quantity of the PGA memory used by a process and, accordingly, when an optimum, one-pass, or multipass operation was performed. The goal of sizing the PGA is to avoid multipass operations entirely. Depending on the type of system (ERP or BI), the percentage of optimum operations should be >70% or >90%, respectively.

8.2.2.2 Analyzing Wait Events

The quality of the individual buffer memories (data buffer, shared SQL, and so on) is not sufficient to make a reliable statement about the Oracle performance. If, for example, a query resulted in a hit, this does not tell you anything about the processing speed for the query and the output of results. That's where *Oracle wait events* come into play. The database response time consists of two elements:

- ▶ **CPU time:** the time during which the Oracle session uses the CPU
- ▶ **Wait event:** the times during which the Oracle session waits for an event, such as reading a data block from a hard disk

A wait event is a situation in which an Oracle session waits for an event. This event can come from different database areas. For example, the wait event, `log buffer space`, indicates that the session had to wait for free space in the redo log buffer. After starting the database, all wait events are collected in `X$` tables and can be queried using different `V$` views. The most important of these views are as follows:

- ▶ **V\$SYSTEM_EVENT**
Contains all wait events since the database was started including their frequency and average length.
- ▶ **V\$SESSION_EVENT**
Contains all waits since the database was started including their frequency and the average and maximum lengths for every Oracle session.
- ▶ **V\$SESSION_WAIT**
Contains the current waits for every Oracle session or the information that the CPU is currently being utilized.

With Oracle database Release 9i or lower, all monitoring data for the wait events are deleted after restarting the database. Oracle 10g, however, includes some history tables or views that store historical data. You can find the history of wait events in the `DBA_HIST_SYSTEM_EVENT` view.

The consideration of waits allows you to precisely determine which actions an Oracle session is currently performing or for which actions it is currently. This makes it easier for the administrator to identify potential problems. Moreover, the analysis of the system-wide collected waits provides details on the optimization potential within the database.

Wait events are always composed of an event name and up to three optional parameters to include more specific information on the event, as described in the following example:

- ▶ **Event:** `direct path read`: Waiting for a read operation on a data block from the hard disk while circumventing the data buffer
- ▶ **Parameter 1:** `file number`: File number of the file to be read
- ▶ **Parameter 2:** `first dba`: First block to be read in the file
- ▶ **Parameter 3:** `block count`: Quantity of blocks to be read

You can use the following SQL command to determine the file name for a file number and the corresponding tablespace:

```
Select tablespace_name, file_name
From dba_data_files
Where file_id = 'ID';
```

Oracle 10g contains more than 850 wait events (Oracle 9i has about 400), which are grouped in the classes shown in Table 8.5 to provide a better overview (as of Oracle 10g).

Wait Event Class	Number of Wait Events in Class
Administrative	46
Idle	62
Application	12
Network	26
Cluster	47
Scheduler	1

Table 8.5 Wait Event Classes (Number of Wait Events)

Wait Event Class	Number of Wait Events in Class
System I/O	24
Commit	1
User I/O	17
Concurrency	24
Other	591
Configuration	23

Table 8.5 Wait Event Classes (Number of Wait Events) (Cont.)

The following SQL statement can be used to determine to which class a wait event belongs:

```
Select WAIT_CLASS from dba_hist_event_name where EVENT_NAME='Name';
```

It is important to know that some wait events don't influence the database response time at all and can therefore be neglected in performance analyses. On the one hand, these are all events that belong to class `Idle`. These events are reported if an Oracle process is in idle state (that is, not performing any action). The most commonly known and used event of this class is `SQL*Net message from client`, which occurs if an Oracle shadow process is waiting for a new query. On the other hand, there are wait events that are irrelevant to the database, especially in the context of SAP. One reason for such a situation can be that the time of an event is already included in another event; for example, `log file parallel write` is already covered by `log file sync`. Furthermore, many (but not all) events that occur in Oracle shadow processes (`DBWR`, `PMON`, `SMON`, etc.) are only of secondary importance, because the corresponding operations are performed asynchronously to the Oracle work processes.

The following list shows the most frequent wait events that are usually irrelevant from the SAP perspective.

Oracle Wait Events not Relevant to SAP

- ▶ `db file parallel write`
- ▶ `log file sequential read`
- ▶ `smon timer`
- ▶ `SQL*Net message from client`
- ▶ `Log archive I/O`
- ▶ `ARCH wait on SENDREG`

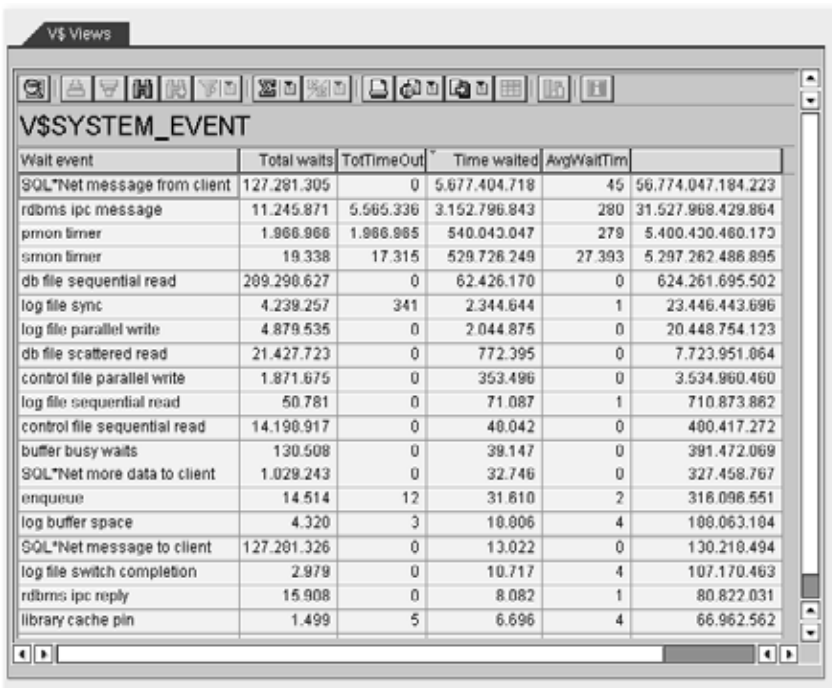
Oracle Wait Events not Relevant to SAP

- ▶ rdbms ipc message
- ▶ jobq slave wait
- ▶ log file parallel write
- ▶ pmon timer
- ▶ Streams AQ: <action>

As already described in Table 8.4, the ratio between **Busy wait time** and **CPU time** (ideally 60:40) is generally a first indicator.

When starting a general wait event analysis, it is useful to create a list containing the top wait events, that is, a list with the totaled wait times in descending order. You can create this list in Transaction ST04N (see Figure 8.6) using the V\$SYSTEM_EVENT view or by executing the following SQL command:

```
select EVENT, TIME_WAITED, AVERAGE_WAIT
from V$SYSTEM_EVENT
order by TIME_WAITED desc;
```



Wait event	Total waits	TotTimeOut	Time waited	AvgWaitTim	
SQL*Net message from client	127.281.305	0	5.677.404.718	45	56.774.047.184.223
rdbms ipc message	11.245.871	5.565.336	3.152.796.843	280	31.527.968.429.864
pmon timer	1.966.966	1.966.965	540.043.047	279	5.400.430.460.173
smon timer	19.338	17.315	529.726.249	27.393	5.297.262.486.895
db file sequential read	289.290.627	0	62.426.170	0	624.261.695.502
log file sync	4.239.257	341	2.344.644	1	23.446.443.696
log file parallel write	4.879.535	0	2.044.875	0	20.448.754.123
db file scattered read	21.427.723	0	772.395	0	7.723.951.064
control file parallel write	1.871.675	0	353.496	0	3.534.960.460
log file sequential read	50.781	0	71.087	1	710.873.862
control file sequential read	14.190.917	0	40.042	0	400.417.272
buffer busy waits	130.508	0	39.147	0	391.472.069
SQL*Net more data to client	1.029.243	0	32.746	0	327.458.767
enqueue	14.514	12	31.610	2	318.096.551
log buffer space	4.320	3	10.906	4	100.063.104
SQL*Net message to client	127.281.326	0	13.022	0	130.210.494
log file switch completion	2.979	0	10.717	4	107.170.463
rdbms ipc reply	15.908	0	8.082	1	80.822.031
library cache pin	1.499	5	6.696	4	66.962.562

Figure 8.7 Lists of Wait Events in the V\$SYSTEM_EVENT View

The AVERAGE_WAIT column is formatted differently in Oracle 9i than in 10g. In Oracle 9i, the contained values are displayed as 1/100 seconds without decimal places, so they are not precise enough for a serious performance analysis. As an alternative, you can use the TIME_WAITED_MICRO column, which contains the total wait time in microseconds. The exact average wait time can then be calculated by dividing TIME_WAITED_MICRO by TOTAL_WAITS.

Table 8.6 provides an overview of the individual columns of the V\$SYSTEM_EVENT view including their meaning.

Column	Description
TOTAL_WAITS	Number of occurrences of the wait event since the last start of the Oracle database
TOTAL_TIMEOUTS	Number of waits for which the corresponding event has not occurred
TIME_WAITED	Total wait time for the wait event in hundredths of a second
AVERAGE_WAIT	Average wait time for the event in hundredths of a second (AVERAGE_WAIT = TIME_WAITED / TOTAL_WAITS)
TIME_WAITED_MICRO (last column in Figure 8.7)	Total wait time for the wait event in microseconds

Table 8.6 Columns of the V\$SYSTEM_EVENT View

Once the list has been created, it is searched from top to bottom to find critical wait events; during this step, idle wait events are ignored.

SID	SPMID	WAIT_EVENT	PTEXT	PARAM_1	PARAM_2	PTEXT_2	PTEXT_3	PTEXT_4	PTEXT_5	WAIT_TIME	SEC_WAIT	SESSION_STATE		
1	50	pmon timer	duration	300	00000000012C	0	00000	0	00000	0	2.785	WAITING		
2	63	rdbms ipc message	timeout	300	00000000012C	0	00000	0	00000	0	1.732	WAITING		
3	11	rdbms ipc message	timeout	300	00000000012C	0	00000	0	00000	0	0	WAITING		
4	2	rdbms ipc message	timeout	100	000000000120	0	00000	0	00000	0	619	WAITING		
5	41	rdbms ipc message	timeout	30	000000000170	0	00000	0	00000	0	5.675	WAITING		
6	71	rdbms ipc message	timeout	6	0000000001770	0	00000	0	00000	0	12	WAITING		
7	34	rdbms ipc message	timeout	300	00000000012C	0	00000	0	00000	0	0	WAITING		
8	31	pmon timer	sleep time	300	00000000012C	0	00000	0	00000	0	0	WAITING		
9	42	SQL*Net message to client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	1	0	WAITING KNOWN TIME
10	37	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	9	WAITING
11	49	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	114	WAITING
12	51	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	54	WAITING
13	6	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	5	WAITING
14	29	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	291	WAITING
15	34	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	27	WAITING
16	37	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	281	WAITING
17	37	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	293	WAITING
18	61	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	295	WAITING
19	346	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	26	WAITING
20	31	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	29	WAITING
21	120	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	2.705	WAITING
22	49	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	54	WAITING
23	9	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	384	WAITING
24	64	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	27	WAITING
25	54	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	173	WAITING
26	34	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	3	414	WAITED KNOWN TIME
27	320	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	1	292	WAITED KNOWN TIME
28	8	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	54	WAITING
29	827	SQL*Net message from client	driver id	1	413.087.536	000054435000	#bytes	1	00001	0	00000	0	64	WAITED KNOWN TIME

Figure 8.8 V\$SESSION_WAIT View

You can obtain further important information in the `V$SESSION_WAIT` view (see Figure 8.8). This view displays the current or most recently active wait events of all Oracle processes.

Table 8.7 describes the meaning of the individual columns in the `V$SESSION_WAIT` view.

Column	Description
SID	Session ID of the Oracle process.
P1TEXT, P2TEXT, P3TEXT	Description and unit of the corresponding parameter.
P1, P2, P3	Parameter values of the wait event.
WAIT TIME	Time waited for the wait event (in hundredths of a second) once the wait event is no longer active. The value of an active wait event is 0. Moreover, there are two special values: Value = -1 if the duration of the event was below the measurement accuracy and value = -2 if <code>TIMED_STATISTICS</code> is not active.
STATE	Wait events can have the following statuses: <ul style="list-style-type: none"> ▶ WAITING: waiting/active (WAIT TIME = 0) ▶ WAITED KNOWN TIME: has expired and had a duration of more than 1/100 sec (WAIT TIME > 0) ▶ WAITED UNKNOWN TIME: has expired and had a duration of less than 1/100 sec (WAIT TIME = -1) ▶ TIME STATISTICS OFF: has expired, but statistics are not recorded (WAIT TIME = -2)

Table 8.7 Columns of the `V$SESSION_WAIT` View

Furthermore, it is often necessary or it simply makes sense to map the SAP work processes to an Oracle work process or vice versa. The easiest way to do this is to use the Oracle Session Monitor. In the **Clnt proc** column, this component includes the process ID of the linked SAP work process for every entry of an Oracle work process. This client PID corresponds to the **Pid** column of the SAP process monitor (Transaction SM50).

Warning

A CPU bottleneck can also cause a large number of different wait events. If the CPU load is very high, it is possible that Oracle processes that currently hold a lock are displaced. If other processes are waiting for this lock, several wait times can increase drastically. You should therefore first ensure that sufficient CPU resources are available.

Remark

As mentioned earlier, compared to the previous Release Oracle 9i, the number of wait events was increased considerably in Oracle 10g. This is reflected, for instance, in the splitting up of wait events for a more detailed root cause specification. We will mainly use the wait events from Oracle 10g and only point out the differences in comparison to Oracle 9i in a few situations.

The following sections describe the most important wait events and provide some background information on these. You'll find several tables with the most important details followed by a text section containing a description of the wait event.

Name	db file sequential read/db file parallel read
Parameter 1	File number(s)
Parameter 2	Block number(s)
Parameter 3	One or a number of parallel reads
Meaning	These events represent the process of waiting for one or more parallel read operations to be performed on blocks on the hard disk. In this case, parallel does not refer to reading several blocks successively, but to simultaneous reads of different, nonsuccessive blocks.
Rating	Average wait time should be less than 2, i.e., $2/100 \text{ s} = 20 \text{ ms}$.

Table 8.8 db file sequential read/db file parallel read

If there are problem values for the average wait time, this primarily indicates an I/O performance bottleneck. For information on the analysis of I/O problems, refer to Section 8.2.2.3, *Analyzing the Database I/O*. Another important factor apart from wait time is the occurrence frequency of `db file sequential read`. If this value is very high, the wait event usually occurs in conjunction with a bad hit ratio of the data buffer. In this case, there are two solution scenarios: You either tune potentially existing bad SQL statements (see Section 8.3) or you increase the data buffer size.

Name	db file scattered read
Parameter 1	File number
Parameter 2	Block number
Parameter 3	Number of blocks

Table 8.9 db file scattered read

Name	db file scattered read
Meaning	If this event occurs, an Oracle session is waiting for a successive read operation on several blocks from the hard disk.
Rating	A maximum of 10% of the WAIT_TIME of db file sequential read (exception: in case of SAP NetWeaver BI, a higher value can be accepted).

Table 8.9 db file scattered read (Cont.)

A successive read operation on several blocks usually occurs only with Full table scan or Index fast full scan. These access types reduce performance significantly and should thus be avoided if possible. Once again, SAP NetWeaver BI represents an exception to this rule (see Chapter 12). If the occurrence of db file scattered read exceeds the values listed in Table 8.9, you should determine the SQL commands that cause this situation. Information on these commands can be found using the function **SQL Request** in Transaction ST04N or following menu path **Resource Consumption • SQL Request**. Pay attention to SQL statements with high disk reads. Better yet, focus only on commands that include a full scan in their process.

SAP Note 619188 describes an SQL command that can be used as of Oracle 9i. Using this command, you can determine the 20 SQL statements that generate the largest number of disk reads because of full scans. You should then determine if these commands can be tuned. **Attention:** If you make extensive use of the Oracle transactions in the SAP system (for example, Transaction ST04N) during performance analysis, this may be reflected in the results. In this case, some of the top 20 SQL statements contain queries on Oracle specifications or Oracle monitoring data that are not associated with the normal business-related SQL queries. These SQL statements should be ignored in your analysis.

Name	direct path read/direct path read temp direct path write/direct path write temp
Parameter 1	File number
Parameter 2	Block number
Parameter 3	Number of blocks

Table 8.10 direct path [read|write|temp]

Name	direct path read/direct path read temp direct path write/direct path write temp
Meaning	This wait event is registered if the data buffer is circumvented when data blocks are accessed. As of Oracle 10g, waits are categorized by either access to "normal" blocks or access to temporary blocks from the PSAPTEMP tablespace (temp).
Rating	None of these events should be among the first 10 in the wait event list (in descending order according to the totaled wait time; see Figure 8.7). Furthermore, similar to <code>db file sequential read</code> , a maximum value of 2 applies to the average wait time, i.e., $2/100\text{ s} = 20\text{ ms}$.

Table 8.10 direct path [read|write|temp] (Cont.)

If the problem is caused by a too long average wait time, this is probably also caused by an I/O bottleneck. In this case, you should perform the steps described in Section 8.2.1.2, *Identifying the Causes of Bottlenecks in Hardware Components*.

If the `direct path` operations are performed too often and are therefore displayed among the first entries in the list, you must distinguish between the reasons for these operations in subsequent actions.

For the Oracle database, there are three reasons why `direct path` operations are performed:

1. PSAPTEMP accesses
2. Parallel queries
3. Access to LOB data (large object)

For many operations on the PSAPTEMP tablespace (these can be recognized by the wait event, `direct path read/write temp`), increasing the PGA storage (see Section 8.2.2.1, *Analyzing the Database Buffers*) is a possible solution to the problem. Operations that are performed for the second or third reason cannot be distinguished in Oracle 10g. In contrast, when accessing unbuffered LOB data in Oracle 9i, a separate wait event is generated: `direct path read/write (lob)`. LOB data, that is, large unstructured data in table columns, is primarily used by the SAP system in tables with ABAP code. Because of their size (up to four gigabytes), these LOBs are no longer buffered in the data buffer. This again results in `direct path` operations. If problems occur, buffering special LOB data might be advisable (see SAP Note 563359).

Parallel queries, that is, performing special actions such as a full table scan in parallel, are generally not used by SAP. They are only used for SAP NetWeaver BI systems. The reason for this is that these queries have several disadvantages regarding the CBO and the resulting resource allocation (see SAP Note 651060).

Name	log file sync/log buffer space/log file parallel write
Parameter 1	Number of buffers/ - / file number
Parameter 2	- / - / Number of blocks
Parameter 3	- / - / Number of I/O requests
Meaning	<ul style="list-style-type: none"> ▶ log file sync Represents the process of waiting for the full synchronization of the log files with the redo buffer by the LGWR (e.g., after a COMMIT). ▶ log buffer space Represents the process of waiting for a free block in the redo buffer. ▶ log file parallel write Occurs if there is a wait time for the writing of blocks in the redo log files.
Rating	For all three wait events, a maximum average wait time of 4 applies, i.e., $4/100 \text{ s} = 40 \text{ ms}$. However, for current hardware, significantly lower values should be obtained that allow for about 15 ms.

Table 8.11 log file [sync|parallel] and log buffer space

All three wait events described above usually depend directly on I/O performance during write operations for redo log files. You should therefore first analyze and examine if there are I/O problems and whether these areas can be optimized (see Section 8.2.1.2, *Identifying the Causes of Bottlenecks in Hardware Components*, and Section 8.2.2.3, *Analyzing the Database I/O*). The redo log files are the most I/O-intensive area of an Oracle database and therefore have special requirements regarding their storage location and parameterization. Section 5.2.3, *Storage and SAN Infrastructure*, provides further information on this topic.

Under certain circumstances, it makes sense to completely deactivate logging when importing or modifying large data volumes. This takes place, for instance, during the initial loading of the SAP system in the database. However, after deactivating logging, you can only restore and recover data up to the time when this action was performed. This means you need to create a new full backup of the system once you reactivate the logging function.

Another aspect is the size of the redo buffer. If this buffer is configured with less than one megabyte – contrary to the SAP recommendation – this may also result in log buffer space wait events. If this situation occurs, you need to change the `LOG_BUFFER` parameter to the size of one megabyte (offline). There are a few other cause of the `log file sync` wait event, such as enqueue wait situations (see SAP Note 745639, Section 12).

Name	<code>log file switch completion/(archiving needed)/(checkpoint incomplete)/(private strand flush incomplete)</code>
Parameters	–
Meaning	These wait events are reported if the system needs to wait for a log file switch for different reasons (see below).
Rating	<ul style="list-style-type: none"> ▶ <code>archiving needed</code> Should never occur ▶ <code>checkpoint incomplete</code> Should never occur (special SAP Note 79341) ▶ <code>completion</code> Should not be among the top 10 of the wait event list, at most one log switch per minute ▶ <code>private strand flush incomplete</code> Should never occur

Table 8.12 log file switch completion

The term *log file switch* is a generic term for several wait events that occur when switching to the next redo log file:

- ▶ **log file switch (archiving needed)**
Occurs if the log switch cannot be performed, because the next redo log file has not been archived yet
- ▶ **log file switch (checkpoint incomplete)**
Represents the process of waiting for completion of the checkpoint of the subsequent redo logs before the log switch
- ▶ **log file switch completion**
Represents the process of waiting for completion of the log switch
- ▶ **log file switch (private strand flush incomplete)**
Occurs if the LGWR waits for the DBWR to completely write the in-memory UNDO buffer (IMU) into the log buffer

An "archiving needed" event always has an average wait time of 98–100 (time in one-hundredth of one per second), because the writing Oracle processes (LGWR, DBWR, etc.) always wait for exactly one second in case of an

archiver stuck before a new write attempt is carried out. An archiver stuck must not occur in an SAP production system, as this would cause a system standstill. Only in a nonproduction system is a short archiver stuck acceptable under certain circumstances, if the system reaches a very high load, for instance, during client copies or data loading at night-time. To avoid this standstill of the Oracle database, your backup strategy must ensure that the disk volume on which the offline redo logs are saved (usually the directory *oraarch*) is always backed up and purged so that there is sufficient space for new offline redo logs after a redo switch. If you define other archiver destinations using the `LOG_ARCHIVE_DEST` parameter, you need to ensure that these are backed up as well.

In the case of a log file switch (`checkpoint incomplete`) wait event, the "checkpoint not complete" error has occurred and was recorded in the Oracle alert log. Checkpoints are performed during every log switch. Several checkpoints can be active at the same time. The "checkpoint not complete" error is recorded if a log switch is to be performed to a redo log with a checkpoint that has not yet been completed.

The following four situations can cause a repeated occurrence of the wait event or the "checkpoint not complete" situation:

1. Numerous redo logs are being written.
2. DBWR performance bottleneck.
3. Not enough redo logs.
4. Redo logs are too small.

If the Oracle database writes many redo logs, you should first examine whether you are dealing with an operational load, that is, whether the number of redo logs is caused by the normal system usage. If there is no indication that your applications are responsible for the high redo log frequency, there are several other possible reasons, such as misconfigurations and Oracle bugs. Read SAP Note 584548 for a description of the possible causes.

Usually, the reason for a high amount of redo logs can, of course, be found in the system operation. As a first step, you should ensure that no more than one redo log switch is performed per minute. If this is not the case, you should increase the size of your redo log files. To do that, proceed as follows:

1. Log on to the database as sysdba:

```
sqlplus "/ as sysdba"
```

2. Delete a log file group:

```
ALTER DATABASE DROP LOGFILE GROUP 11;
```

If the error ORA-01623 is reported, use the following command to switch to a new redo log file and wait for a few seconds before you repeat the DROP command:

```
ALTER SYSTEM SWITCH LOGFILE;
```

If the error ORA-01624 is reported, the current checkpoint has not yet been completed. Wait for a few seconds and repeat the DROP command.

3. Delete the corresponding operating system files in the redo log directory.
4. Set up the log file group 11 with a new larger size (<new_size> in MB):

```
ALTER DATABASE ADD LOGFILE GROUP 11
('/oracle/<sid>/origlogA/log_g11_m1.dbf',
'/oracle/<sid>/mirrlogA/log_g11_m2.dbf')
SIZE <new_size>M;
```

5. Repeat steps 2 through 4 for all existing redo log groups.

In the standard SAP installation, the redo log files of the four groups have a size of 50 MB each. Increase the files incrementally and verify whether this solves the problem. The scope of the increase depends on the number of redo logs that are written per minute. If five log switches are performed per minute with a size of 50 MB, there is no point in increasing the log size to 100 MB, but, change the size to, for instance, 300 MB right away.

If the log file switch (private strand flush incomplete) wait event occurs or there are other indications of a bottleneck in the DBWR process, for example, from the free buffer waits wait events (see below), you can increase the number of DBWR processes to enhance write performance. To do this, set the DB_WRITER_PROCESSES parameter using the following command (prerequisite: parameter management with SPFILE):

```
alter system set db_writer_processes=X scope=spfile;
```

Attention: The number of DBWR processes should not exceed the number of available CPUs.

Another way to enhance write performance is, of course, to tune the Oracle environment, that is, all I/O relevant components. Refer to Section 8.2.2.3 for further information on this topic.

The `log file switch completion wait` event occurs if an Oracle shadow process must wait for the completion of a log switch. As described above, if too many redo log switches exist during operation (more than once per minute), this results in a critical condition regarding the database performance. However, if this happens, proceed as described earlier.

Name	read by other session/buffer busy wait
Parameter 1	File number
Parameter 2	Block number
Parameter 3	ID
Meaning	These wait events describe the process of waiting for a block in the data buffer, because this block is currently being read (<code>read by other session</code>) or modified (<code>buffer busy wait</code>).
Rating	The average wait time value should be below 2, i.e., $2/100 \text{ s} = 20 \text{ ms}$.

Table 8.13 read by other session/buffer busy wait

In Oracle 9i, both events were named `buffer busy wait`, and parameter value 3 indicated the reason: The IDs started with 1 or 2 and contained further places depending on the exact reason. If the ID starts with 1 (ID = 1xx), the event deals with the reading of a block. If it starts with 2 (ID = 2xx), the wait event is caused by a write or change operation for a block. As of Oracle 10g, the name of the wait event already distinguishes whether the event was caused by a read or write operation. Detailed information on the event cause can be obtained from the parameters.

As all data that are read from or saved in the database "pass through" the data buffer (an exception is the already mentioned direct path access), high I/O loads always result in `buffer busy waits`. You always have the option to reduce I/O load to decrease the amount of waits on the data buffer. This can either be done by redistributing data loads or by tuning SQL statements so that fewer data blocks must be read (see Section 8.3).

The second criteria besides I/O load is the management of the data blocks themselves. In this area, in particular, Oracle 9i provided significant enhancements with the introduction of *Automatic Segment Space Management* (ASSM). Previously, the individual blocks of a tablespace or a segment

were managed using the `PCTUSED`, `PCTFREE`, `FREELISTS`, and `FREELISTGROUPS` parameters. These parameters were used, for example, to define the fill level of a block (`PCTUSED`): As long as this value was not reached, the block would accept new data. From this group of parameters, only one was also implemented in ASSM so that you can still use its function: `PCTFREE`.

Without ASSM, the database administrator had to or could decide for each table how the individual blocks of the segments were used. This made it possible to choose between performance and efficient space usage depending on the change frequency. This task is now performed by ASSM. SAP made the use of ASSM possible as of Version 9.2.0.5, and on installations with SAP Basis 6.40 and higher all data tablespaces are set to ASSM by default. If problems occur that are related to `buffer busy waits`, you can now switch to ASSM to resolve issues relating to segment management. In Oracle 9i, this switching procedure involves downtime, whereas Oracle 10g allows you to make the transition online. SAP Note 620803 provides step-by-step instructions for this transition.

Name	write complete waits/free buffer waits
Parameter 1	File number
Parameter 2	Block number
Parameter 3	ID
Meaning	These wait events occur if an Oracle process must wait for the DBWR process to write a block into the relevant data file.
Rating	Both wait events must not be among the first 10 entries in the wait event list.

Table 8.14 write complete waits/free buffer waits

If these wait events occur too often, the data buffer may be too small or the performance of the DBWR process is poor. If possible, resolve this problem by increasing the value for the `DB_CACHE_SIZE` parameter or optimizing the I/O performance (Section 8.2.2.3). Furthermore, you can raise the number of DBWR processes as described in the previous section.

Name	rdbms ipc reply
Parameter 1	PID of the background process
Parameter 2	Timeout in seconds
Parameter 3	–

Table 8.15 rdbms ipc reply

Name	rdbms ipc reply
Meaning	This wait event occurs if an Oracle shadow process must wait for a background process.
Rating	This event must not be among the top 10 entries in the wait event list.

Table 8.15 rdbms ipc reply (Cont.)

In general, the occurrence of the `rdbms ipc reply` wait event is not a problem, as there are various reasons why a process must wait for a background process. The crucial factor is the duration of the wait time for the background process. The main reason for this event is wait situations in the `BEGIN BACKUP`, `TRUNCATE`, and `DROP` operations, because the `CKPT` process must perform a checkpoint in these operations. In Oracle 9i and lower, there is the additional drawback of a design weakness that results in the entire data buffer being searched for affected blocks in a `DROP` or `TRUNCATE` operation; this process can take quite some time with larger buffer sizes. This problem does longer exists in Oracle 10g.

In general, the duration of `rdbms ipc reply` wait events is very short. For this reason, the average wait time should not exceed 10 ms, as this would indicate several wait periods that are much too long and increase the average value. In this case, it is advisable to examine the enqueue wait events, because some of them are closely related to the `rdbms ipc reply` wait event. Refer to SAP Note 745639 for further information on this topic.

If you suspect a problem with this wait event, you can use the Oracle Session Monitor (Transaction `ST04N`; **Resource Consumption • Oracle Session**) and `V$SESSION_WAIT` to determine which Oracle work process is waiting for which background process. In `V$SESSION_WAIT` you'll find the `SID` of the process that is waiting for the `rdbms ipc reply` wait event, while parameter 1 (column `P1`) displays the `PID` of the background process for which the work process is waiting. Subsequently, you can use the Session Monitor to find out which actions the background process is currently performing. For a more detailed analysis of the actions performed by Oracle processes, you should use the functions of the `ORADEBUG` trace. The procedure is described in SAP Note 613872.

Name	latch free/latch: <latch_name>/wait list latch free
Parameter 1	Latch address
Parameter 2	Latch number

Table 8.16 Latch Wait Events

Name	latch free/latch: <latch_name>/wait list latch free
Parameter 3	Number of sleeps
Meaning	This wait event occurs if a process must wait for the release of a latch.
Rating	The rating heavily depends on the specific latch wait event. In general, latch wait events should not appear among the top 10 entries in the wait event list.

Table 8.16 Latch Wait Events (Cont.)

A *latch* is a very low-level lock mechanism for the SGA memory structures. In contrast to a lock, a latch is applied only for a very short time. For this reason, latch requests are not placed in a queue, but the requesting processes permanently try to apply the latch. This so-called spinning process is performed as many times as set in the `_SPIN_COUNT` parameter. If a process applies a latch and another process tries to access the respective memory area, but does not succeed in doing so during the spin phase, a latch <latch_name> wait event is activated. Oracle 10g contains 27 latch wait events, and their names specify the location or the memory structure in which the latch is applied. All "irrelevant" latches are referred to as "latch free."

In older Oracle releases (before 10g), all waits are summarized under the term *latch free*. As an analysis of the different latch wait events would go far beyond the scope of this chapter, we refer you to SAP Note 767414 for more detailed information on this topic.

Name	enqueue (9i)/enq: <type> - <description> (10g)
Parameter 1	Type
Parameter 2	ID1 (9i)/Detailed information in plain text (10g)
Parameter 3	ID2 (9i)/Detailed information in plain text (10g)
Meaning	An event of this type occurs if a process is waiting for the release of an Oracle lock.
Rating	The average wait time value (of all enqueue events) should be below 10, i.e., $10/100\text{ s} = 100\text{ ms}$. As all possible lock situations are collected under one event in 9i, it is generally not a problem if these events are listed among the lower top 10 entries in the wait event list. With Oracle 10g, however, the different enqueue waits should not appear among the top 10 (an exception is TX "row lock contention").

Table 8.17 Enqueue Wait Events

Finally, we want to introduce another important Oracle wait event related to Oracle database locks (enqueues and locks). The meaning of this event can be seen from the development between 9i and 10g. In Oracle 9i, all enqueue waits were still accumulated as one event so that it was rather difficult to perform an analysis. With 10g, however, 184 enqueue wait events were introduced, which, are grouped in different classes (see Section 8.2.2.4, *Other Performance-Relevant Aspects of the Oracle Database*; this section describes database locks in greater detail).

8.2.2.3 Analyzing the Database I/O

If you have found a hint on problems in the I/O area in one of the preceding analyses, for example, due to an anomaly of a corresponding wait event, you inevitably get to the I/O analysis. Unfortunately, from the SAP administrator's point of view, the analysis of an I/O system is restricted. In addition, an I/O bottleneck can generally not be solved during the operation if it is caused by hardware or the distribution of data.

Term Clarification

- ▶ Before getting started with the analysis, we should clarify the term *hard disk*. (Also refer to Figure 8.11.) Regarding server systems for business-critical applications, namely, the world of SAP and Oracle, the term *hard disk* has two meanings:
- ▶ Physical: magnetic memory; the hardware part
- ▶ Virtual: operating system resource (device); the software part

In the following text, the term *hard disk* is always used with the second meaning, because the SAP system or the database considers only the operating system with the resources available as the underlying level. Regardless of whether the hard disk is visible as a hard disk with file system or as a raw device, the virtual hard disk in a modern IT infrastructure is far more than a physical hard disk. In fact, there is a complex storage architecture with many different components behind the operating system as the abstraction layer. All of these parts of a storage system, for instance, interface cards to a SAN, storage switches, or array controllers, can be relevant for I/O problems. Unfortunately, you as an SAP or Oracle administrator have no chance to identify and solve problems but need assistance from specialists of the respective hardware partner. You should keep this definition in mind and remember the "veiled" complexity of the term in the relevant parts of the following sections.

Critical points in the structure of an Oracle database are as follows: The most I/O intensive areas are without any doubt the redo log files followed by the data files. Out of those, the undo (or rollback) and PSAPTEMP tablespaces can be pointed out, which always (undo) or, especially in the OLAP environment (PSAPTEMP), show increased access rates. Offline redo logs and Oracle

executables are less important. Read Section 5.2.3, *Storage and SAN Infrastructure*, to find out more about the optimal distribution of Oracle files.

We already briefly touched upon analyses in Section 8.2. In the operating system monitor (ST06) you can view the current utilization of all hard disks of the operating system (see Figure 8.9) under **Detail Analysis Menu** using the **Disk** button.

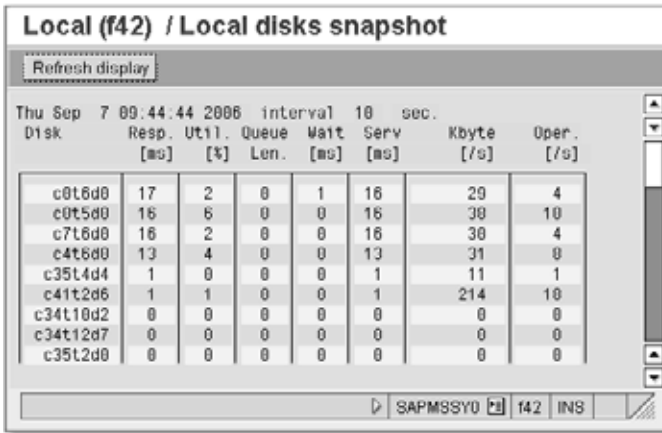


Figure 8.9 Overview of the Hard Disks in the Operating System Monitor

By double-clicking on one of the displayed disks, you can view the utilization of the selected disk during the last 24 hours. The most important key figure, **Utilization**, shows a mean value over a period of one hour (see Figure 8.10).

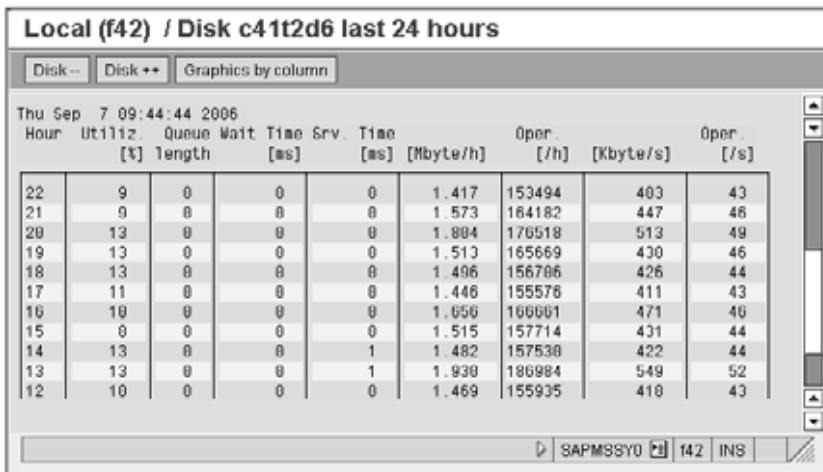


Figure 8.10 History of a Single Hard Disk

If you discover a problem with a hard disk via the I/O utilization, you must identify which parts of the Oracle and SAP installation are on the disk. Unfortunately, the SAP system does not enable you to retrace the direct assignment of files and raw devices to hard disks. The reason is the already mentioned resource of the virtual hard disk that is displayed in the operating system monitor. Between the virtual hard disk and the actual files, server systems have another virtualization layer, such as the typical Logical Volume Manager (LVM) for UNIX operating systems. Figure 8.11 illustrates the relationship between the individual components.

For example, if you want to create a connection between the hard disk and the Oracle data files or raw devices, you must use tools of the operating system.

You have two options to increase the I/O performance: You can increase the performance of the hard disk(s) or try to reduce the I/O load.

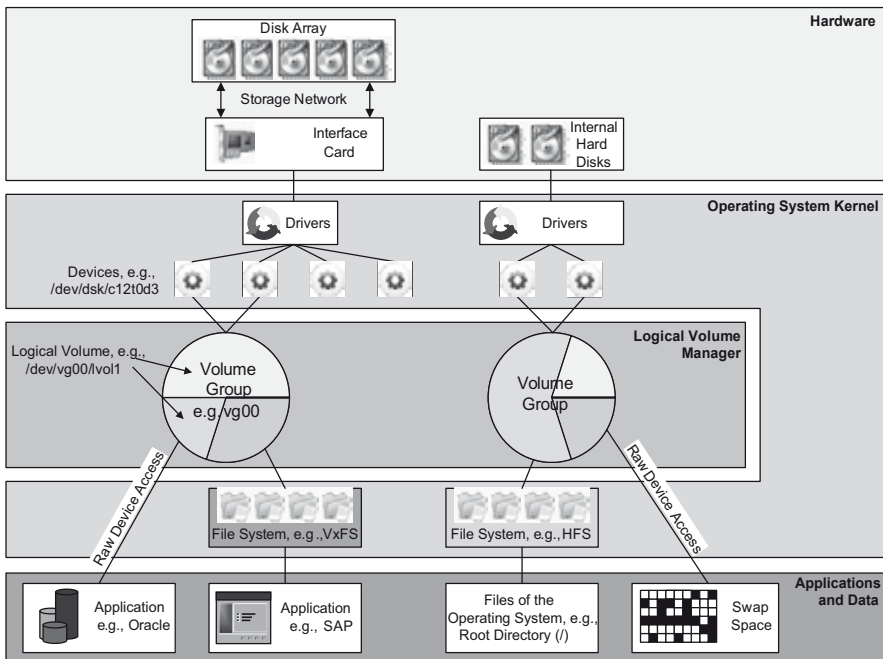


Figure 8.11 Overview of a UNIX System with Logical Volume Manager

Let us first take a look at the options to increase the hard disk performance. First, you should check if you have used all options of the operating system for optimal performance:

1. Does the layout of the structure of your Oracle files correspond to the recommendations, especially regarding the separation of load-intensive files (see Chapter 5, *Planning the System Landscape*)?
2. Have you used all possible options of I/O processing?
3. Are all current drivers for I/O subsystems installed?

You should pay particular attention to the options of I/O processing. In general, all operating systems provide two functions for I/O operations: the *file system caching* and *file lock mechanisms*. File system caching works in a similar way to the Oracle data buffer (but is easier) as a temporary storage for the access of applications to I/O systems. File locks serve as write locks so that data or files cannot be changed simultaneously by two different processes. For the options of I/O processing, the handling of the mentioned operating system functions is decisive. Table 8.18 lists all options in descending order of their performance.

Name	Description
Raw I/O	When raw I/O (or raw devices) is used, the operating system functions are bypassed completely, and logical volumes and hard disks are directly accessed. Furthermore, there is no file system and correspondingly no data in the traditional sense.
Concurrent I/O	As is the case for raw I/O, file system caching and file locks are bypassed completely, but there is a file system and thus normal files. Of course, the used file system must support concurrent I/O (e.g., Veritas VxFS). This I/O type can be used for Oracle databases, as the Oracle-internal locking functions already provide a collision-free access and hence ensure the integrity of data. Caution: Only volumes that contain only Oracle data or redo log files may be operated in this mode (Oracle executables are also excluded).
Direct I/O	Direct I/O disables only the file system caching of the operating system but uses the locking mechanism for the data access.
Cached I/O	Uses all operating system functions for I/O and is generally the default setting for I/O processing.

Table 8.18 Options of I/O Processing

In addition to the I/O modes already mentioned, there are two more independent options: *synchronous I/O* and *asynchronous I/O*. When a process performs a synchronous input or output, the process must wait until the operation is completed and only then can continue to work or perform the next input or output. This is not the case for asynchronous I/O. Thus, the process can continue working simultaneously with the I/O operation. Generally, you should use asynchronous I/O.

SAP Note 834343 provides a table with the currently supported combinations of operating system, file system, and I/O options.

Another remark on raw I/O: Basically, with raw I/O, you can assume that you can reach an I/O performance that is about 20% higher than the performance with other I/O options. Why is raw I/O not always used? The main disadvantage of raw I/O is that it always involves significantly increased administration efforts for setting up and managing the Oracle database. However, a psychological aspect also assumes an important role regarding the usage of raw I/O: The administrator "misses" his files. The Oracle recommendation for the usage of raw devices is as follows:

"Oracle recommends that raw devices should only be considered when the Oracle database is I/O bound."²

Raw I/O is fully supported and integrated by SAP, Oracle, and all relevant monitoring and backup solutions so that it can be especially used for performance-critical installations, such as SAP Business Information Warehouse. In this case it is also possible to operate only parts of the Oracle database, for instance, redo log files or the temporary tablespace, on raw devices.

For the usage of the described I/O options, the administrator also has to keep two relevant Oracle parameters in mind:

- ▶ **DISK_ASYNC_IO** (Default: TRUE)
This option ensures that asynchronous I/O is always used when it is offered by the operating system.
- ▶ **FILESYSTEMIO_OPTIONS** (Default: none on the part of Oracle, however, depending on the version SAP uses `ASYNC (9i)` or `SETALL (10g)`)
This parameter overrides `DISK_ASYNC_IO` when file systems are used and includes the following options:
 - ▶ **NONE**
No direct I/O and no asynchronous I/O.
 - ▶ **DIRECTIO**
Enables direct I/O.
 - ▶ **DIRECTIO**
Enables asynchronous I/O.
 - ▶ **SETALL**
Enables concurrent I/O (if available), direct I/O, and asynchronous I/O.

² Oracle recommends that raw devices should only be considered when the Oracle database is I/O bound. See www.oracle-training.cc/oracle_tips_raw_devices.htm.

For example, this can be used for installations with the combination of raw devices and file system to use asynchronous I/O on the raw device (`DISK_ASYNC_IO=TRUE`) on the one hand, but also to use cached I/O for file system data (`FILESYSTEMIO_OPTIONS=NONE`).

In addition to the options for I/O processing, SAP supports further parameters for the different systems that can affect the I/O performance. SAP Note 793113 is a good starting point that refers to the individual operating system-specific notes. To increase the hard disk performance, you can also exchange hardware. Especially regarding the already described complex storage systems that are now used in important areas, many components can be significantly enhanced when they are replaced by a new generation. Together with your hardware partner, you should decide whether such an exchange makes sense or not. Let us now take a look at the second option for increasing the I/O performance: the attempt to reduce the I/O load. Here, you must bear in mind which I/O type occurs where. Table 8.19 provides notes on the I/O reduction.

Data type/ I/O type	Read I/O	Write I/O
Redo log files	/	<ul style="list-style-type: none"> ▶ Keep data files no longer than necessary in backup mode (parameter <code>backup_dev_type</code>) ▶ If possible, use <code>NOLOGGING</code> ▶ Avoid long transactions with timeouts and rollbacks (because changes are unnecessarily logged) ▶ Avoid unnecessary <code>INSERT</code>-, <code>UPDATE</code>-, and <code>DELETE</code> operations ▶ Avoid unnecessary indexes
Data files	<ul style="list-style-type: none"> ▶ Tuning of expensive SQL statements (Section 8.3) ▶ Caching of LOB accesses (SAP Note 563359) ▶ Extension of the Oracle buffer pool ▶ Extension of the PGA 	<ul style="list-style-type: none"> ▶ Time between log switches more than one minute ▶ Distribution of data files across different volumes or hard disks ▶ Extension of the PGA ▶ Extension of the buffer pool (at free buffer waits)
Temporary data files (PSAPTEMP)	<ul style="list-style-type: none"> ▶ Tuning of expensive sorting with sort, hash, or bitmap functions ▶ Extension of the PGA 	<ul style="list-style-type: none"> ▶ Tuning of expensive sorting with sort, hash, or bitmap functions ▶ Extension of the PGA

Table 8.19 Notes on the I/O Load Reduction

Regarding the reduction of I/O load, we will concentrate on one aspect in more detail: the distribution of data files to reduce the accesses to individual volumes and hard disks. "File system requests" in the Oracle monitor (Transaction ST04N; **Overall Activity**) are suited to get an impression of the number of accesses to the individual data files. Figure 8.12 shows the corresponding view, sorted by the number of read accesses.

File#	Name	Full Path	Reads	Blk Rds	Blk/Rd	Rd Avg (ms)	Rds/File(%)	S
13	a70.data12	/oracleA70/sapdata4/a70_12/a70.data12	3578146	3580572	1	0	100.0000	3
16	a70.data15	/oracleA70/sapdata/a70_15/a70.data15	3321152	3323214	1	1	100.0000	3
17	a70.data11	/oracleA70/sapdata4/a70_11/a70.data11	2581500	2583867	1	1	100.0000	2
17	a70.data18	/oracleA70/sapdata6/a70_18/a70.data18	2125181	2127155	1	1	100.0000	2
7	a70.data6	/oracleA70/sapdata2/a70_6/a70.data6	1925069	1928196	1	0	100.0000	1
6	a70.data5	/oracleA70/sapdata2/a70_5/a70.data5	1824735	1828188	1	0	100.0000	1
1	system.data1	/oracleA70/sapdata1/system_1/system.data1	1550707	8233283	3	0	100.0000	6
15	a70.data14	/oracleA70/sapdata5/a70_14/a70.data14	1515932	1518898	1	1	100.0000	1
2	a70.data1	/oracleA70/sapdata1/a70_1/a70.data1	1508888	1508282	1	1	100.0000	1
14	a70.data13	/oracleA70/sapdata5/a70_13/a70.data13	1431818	1435124	1	1	100.0000	1
5	a70.data4	/oracleA70/sapdata2/a70_4/a70.data4	1429792	1440945	1	1	100.0000	1
3	a70.data2	/oracleA70/sapdata1/a70_2/a70.data2	1308368	1313774	1	1	100.0000	1
4	a70.data3	/oracleA70/sapdata1/a70_3/a70.data3	1245138	1248777	1	1	100.0000	1
8	a70.data7	/oracleA70/sapdata3/a70_7/a70.data7	654136	657008	1	1	100.0000	6

Figure 8.12 Access Statistics for Oracle Data Files

With the access statistics, the administrator can, for example, distribute the top 10 data files to different media or data subsystems. You find an example for moving data files below:

1. Log in as sysdba user. The corresponding tablespace must be set to offline for moving the data files:

```
alter tablespace PSAP<SID> offline;
```

2. Move the file at the operating system level (to simplify matters from the Oracle shell with a preceding "!"):

```
! mv /path with old volume/<sid>.dataX /path with new volume/  
<sid>.dataX
```

3. Change the path of the file in the Oracle database:

```
alter tablespace PSAP<SID> rename file
```



```
'/path with old volume/<sid>.dataX' to '/path with new volume/<sid>.dataX';
```

4. Set the tablespace to online again:

```
alter tablespace PSAP<SID> online;
```

Because the data files are generally rather big and the corresponding tablespace needs to be offline, it is not possible to move the files while the SAP system is running. When files of the SYSTEM or of the UNDO tablespace need to be moved, the procedure mentioned above does not work. Instead, the database needs to be offline to move the files. The new path to the file is published in the MOUNT status with the following command:

```
alter database rename file ...
```

8.2.2.4 Other Performance-Relevant Aspects of the Oracle Database

At the end of this section we will discuss two additional very important and performance-relevant aspects:

- ▶ Oracle enqueues and Oracle optimizer statistics are locks at database level that ensure permanent access to Oracle resources, such as objects or data records in tables. For example, if an Oracle process wants to change a data record, an enqueue is created to protect this modification as an atomic operation. If a second process wants to change this data record, it also creates an enqueue that is placed in a queue. The processing of the queue follows the FIFO principle (*first in, first out*). Oracle enqueues are often referred to as *exclusive lockwaits* or *Oracle locks*. If an enqueue or a lock situation occurs, the corresponding enqueue wait event is generated. Generally, Oracle distinguishes between two types of enqueues that are again differentiated by type: User enqueues are the enqueue types that occur with "normal" data changes, for instance, inserting or deleting data records or restructuring an index. There are three types:
 - ▶ TX (transaction enqueue): Evolves from every change made to a data record when the system needs to wait for this change. This is the most frequent enqueue type.
 - ▶ TM (DML enqueue):
 - ▶ Occurs when a complete object, for example, an index, is locked.
 - ▶ UL (user enqueue): Is generated when a user sets a lock with DBMS_LOCK.REQUEST. This function is not used in the SAP standard.

- ▶ System enqueues are the enqueue types that occur in many Oracle-internal management mechanisms. There are more than 40 different types. However, most of them are only partially or not at all relevant. The most important type in the SAP environment should be mentioned, though: ST (space transaction enqueue). This enqueue is generated during extent management in DMTS (dictionary-managed tablespace).

A data lock mostly affects a row of a table if it is changed via `update`, `delete`, `insert`, or `select for update`. Such locks, as described above, are called TX enqueues. Other enqueues, for example, also lock entire segments (e.g., TM enqueues) or critical paths (ST enqueues). The set lock is always kept up to the data base command `COMMIT` and is then released again. Because a `commit` or `rollback` is carried out after every SAP transaction step (caution: not transaction), the database locks should generally be kept no longer than a few seconds.

Of course, *database locks* are generally important for data consistency in the database. Therefore, their occurrence is normal and presents no problem for the performance. However, this only applies when the locks are only held as long as necessary and no serialization effects occur. This would mean that increasingly more processes wait for the release of a database lock. You should therefore observe the Oracle lock monitor in Transaction ST04N or by following menu path **Exceptional Conditions • Lock Monitor** (or Transaction DB01). **Caution:** You will *not* see the current locks that are kept in the Oracle database but only the lockwaits and thus the requested locks that have not been immediately assigned (that are being waited on). Information about the current database locks can be found in the views `V$LOCK` and `V$LOCKED_OBJECT`. Figure 8.13 shows these two views — also called via Transaction ST04N or by going to **Additional Function • Display V\$/GV\$ Views and Values**.

The figure demonstrates how the Oracle Process **31 (Session ID)** locks the object with ID **18.992**. `V$LOCKED_OBJECT` illustrates that it is a **TM** enqueue in Mode **3**. Furthermore, `V$LOCK` demonstrates that Process **31** also keeps another **TX** enqueue in Mode **6**.

The mode shows how restrictively the object is locked. Table 8.20 illustrates the possible modes of locks.

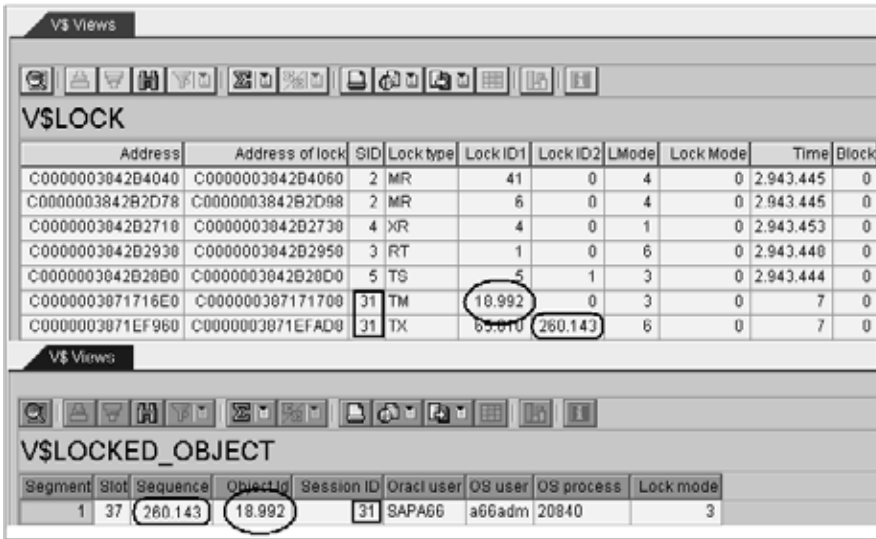


Figure 8.13 Locks of the Oracle Database

Mode	Name
0	Not hold or not requested
1	Null mode
2	Row shared table locks (RS)
3	Row exclusive table locks (RX)
4	Shared table locks (S)
5	Shared row exclusive table locks (SRX)
6	Exclusive table locks (X)

Table 8.20 Modes of Oracle Locks

The description of the individual modes and their corresponding effects would fill several pages. You should therefore refer to the Oracle documentation at www.oracle.com/pls/db102/homepage under **Concepts • Data Concurrency and Consistency**. We will only give a brief description of the basic difference between *shared* and *exclusive* locks:

- ▶ An exclusive lock protects a resource against a shared or another exclusive lock. It is set when the resource is supposed to be changed. The first process that requests and receives an exclusive lock for a resource is the only process that can change the resource until it releases the lock.

- ▶ A shared lock on a resource also allows other shared locks on that resource but prevents exclusive locks. That means write access to the resource is not possible with a shared lock. Therefore, the resource can be read consistently.

In the individual modes, these principles are linked at table and row level.

Let us take another look at the example of Figure 8.13. What is happening? As described, Process 31 holds a lock with Mode 3 on Table 18992. That means it has a shared lock on the entire table (TM enqueue) and wants to make changes to the row. This also corresponds to the meaning of mode — row exclusive table locks. Of course, changing the rows also determines an exclusive lock on the corresponding row, which can be seen in V\$LOCK. The last entry demonstrates that Process 31 holds an exclusive lock (Mode 6) on a row (TX enqueue). That is, Process 31 only makes entries or changes in Table 18992.

The Oracle session monitor (Transaction ST04N; **Resource Consumption • Oracle Session**; see also Chapter 4, *SAP and Oracle*) shows which SAP work process is linked with Oracle Process 31. Using the following SQL command, you can determine which object is specified by the object ID:

```
SQL> select object_name, object_type from dba_objects where object_id=41733;
```

Finally, we will take another look at the V\$LOCK view and, in particular, at the lock mode column in Figure 8.13. (Caution: Don't confuse this with the column with the same name in V\$LOCKED_OBJECT.) This column contains the mode of a requested lock. That is, in case of a value >0, the process waits for a lock from the lock mode that it requested. At this point, you can recognize a lockwait. As mentioned before, such a requested but not assigned lock would also be displayed by the Oracle lock monitor (Transaction DB01) in the SAP system. Furthermore, you can also determine the Oracle process that holds the lock on the "demanded" object (lock mode = 0) via the columns Lock ID1 and Lock ID2 (objects).

If you discover Oracle locks that are held longer or that are being waited on, you should identify the SAP work process that causes the lock via the client host and the client ID. Afterwards, you can determine which program sets the lock for which user and does not release it. A further analysis can then be made with the help of the user or the developer.

If you cannot identify an SAP process with the respective lock, there are two possibilities: First, the SAP work process was cancelled and the "attached"

database shadow process was not closed properly. In this case, you can delete the lock manually by cancelling the corresponding database shadow process using tools of the operating system. (Caution: Ensure that you don't cancel the wrong database shadow process or even an Oracle system process.) To prevent such a situation, you can set the parameter `SQLNET.EXPIRE_TIME` in the file `sqlnet.ora`, which enables an automatic cleanup of cancelled sessions. See SAP Note 20071 for further information. Second, the lock could also be kept by an external process or its attached database shadow process.

See the corresponding Oracle documentation and SAP Note 745639 for further information on Oracle enqueues.

The *table statistics* of the Oracle database provide another essential performance aspect. The Oracle Database Optimizer is supposed to determine the optimum access path for accessing the data. Generally, there are two types of optimizers: the *Rule-Based Optimizer* (RBO) and the *Cost-Based Optimizer* (CBO). The RBO calculates the access paths according to rules that derive from the "where" clause of the SQL statement to be optimized. However, because all databases running on SAP systems use the CBO, the exact process of the RBO is not relevant for us. The R/3 systems version 3.x or older on Oracle older than 7.3.3 were the only exceptions, because the SAP applications for the RBO were developed on these systems due to technical problems with the newly introduced CBO.

The cost-based optimizer calculates the access path to the data on the basis of the costs required for the access. In Oracle systems, you recognize the usage of the CBO with the parameter `OPTIMIZER_MODE`. In SAP systems on Oracle 9i, the parameter has the `CHOOSE` value, which determines that the CBO is always used when statistics are available for a table and that otherwise the RBO is used. From Oracle 10g onward, the RBO is no longer supported, so different levels of the CBO are provided for selection. As of Version 10g, the SAP default value for the parameter is therefore `ALL_ROWS`.

The exact definition of the access costs depends on the database. For Oracle releases older than 9i, the costs are exclusively determined via the blocks to be read, whereas versions higher than Oracle 10g allow several key values (single-block reads, multiblock reads, and CPU) for cost determination. The exact working method of the CBO is kept secret by Oracle. A description and summary of the known facts regarding the CBO can be found in SAP Note 750631, *Rules of Thumb for Cost Calculation of the CBO*. The most important thing to keep in mind is that the costs are mainly calculated on the basis of

table statistics. Therefore, the generation of these statistics is an important task of administrators. Optimizer statistics are generated in two steps:

1. Analyzing the table
2. Creating the statistics

In the first step, all tables of the database are analyzed to determine for which table statistics have to be renewed. Because the actual creation of statistics is very resource-intensive, this two-phase process prevents unnecessary new statistics from being created when the content of a table is insignificantly changed. SAP urgently recommends that you use only the SAP tools to create optimizer statistics, as these tools are especially customized to meet the requirements of the SAP software for the Oracle database. The BR*Tools introduced in Chapter 4 can be called for the creation of statistics by means of the command line (as <sid>adm user)

```
brconnect -u / -c -f stats -t [<TABLESPACE_NAME>|ALL]
```

or can be scheduled via the DBA planning calendar (Transaction DB13) (see Figure 8.14).

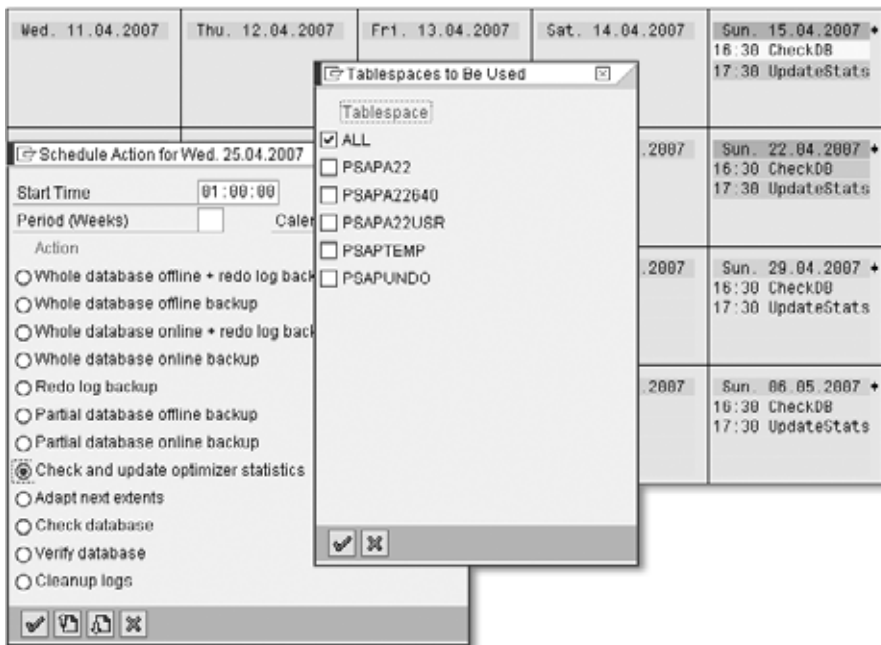


Figure 8.14 DBA Planning Calendar

Chapter 9, *System Operation and Monitoring*, provides further information on the DBA planning calendar.

For every execution of the BR*Tools, and thus for every table analysis run and statistics creation, you find the corresponding log file in the log display for DBA operations (Transaction DB14). There, by clicking on the **BRCONNECT** button, you can view all brconnect logs. You can identify the logs for the update of the optimizer statistics by means of the description or abbreviation "sta" in the FID column. By double-clicking, you can view the log file and find an entry for every table, whose statistics are recalculated (method = C) or estimated (method = E), such as:

```
BR0881I Collecting statistics for table SAPA22.CCMSBIDATA with
method/sample C ...
BR0881I Collecting statistics for table SAPA22.CPRTYPET with
method/sample E/P30 ...
```

Which table of the Oracle database is analyzed and how it is analyzed is controlled by two aspects:

- ▶ Most important are the rules that are programmed in the BRCONNECT program. Those are:
 - ▶ First, it is determined whether new statistics are required or not (on the basis of the number of changed rows), followed by the actual creation of the statistics (two-phase concept).
 - ▶ No statistics on pool and cluster tables for Oracle 9i or earlier.
 - ▶ Accuracy of the statistics based on the number of entries in the table and so on.
- ▶ The second aspect is the content of the DBSTATC table. Using this table, the system administrator can influence the statistics creation for individual tables. Therefore, the table is also referred to as exception table. In every row of DBSTATC, the parameters for running the statistics creation are set for a specific table. The maintenance of table DBSTATC is performed via Transaction DB21 (see Figure 8.15).

Table 8.21 lists the most important columns of the DBSTATC table.

You can also implement new tables in the DBSTATC table that were, for example, created through developments. See SAP Note 106047 for further information on the maintenance of the DBSTATC table.

Database Object	Database	Use	Active	ToDo	Method	Sample	Hist.	Cust.
AFPU	ORACLE	A	A		C		X	
AFRII	ORACLE	A	A		C	R6	X	
AFVC	ORACLE	A	I		E	P30	X	
AFVU	ORACLE	A	A		E	P30	X	
AFVV	ORACLE	A	A		E	P30	X	
AGR_1010	ORACLE	O	A		C			
AGR_1250	ORACLE	O	A		E	P10		
AGR_1251	ORACLE	O	A		E	P3		

Figure 8.15 Excerpt from the DBSTATC Table

Column	Description
Database object	Name of the table
Usage	A = Application Monitor (ST09) and Optimizer O = Only for the optimizer
Active	Controls if the statistics for the table are renewed. Possible values are, for example: <ul style="list-style-type: none"> ▶ A: Active (is checked and updated, if required) ▶ I: Ignore ▶ U: Unconditional (statistics are always updated) ▶ N: No statistics ▶ R: Only temporary statistics
ToDo	Forces the statistic to be generated once.
Method	How the statistic is generated — either by the exact analysis of the entire table (C) or by the estimation according to procedure <sample> (E).
Sample	You have two options: <ul style="list-style-type: none"> ▶ P <n> – n = Percent of the table rows are analyzed ▶ R <n> – n × 1,000 table rows are analyzed

Table 8.21 Columns in the DBSTATC Table

8.2.3 Analyzing the SAP System

Having discussed the analysis of the hardware and operating system as well as of the Oracle database, we now turn to the analysis of the SAP system, or more precisely, the individual instances of the SAP system.

The criteria that are relevant for the performance of an SAP instance can be divided into two main categories:

1. Configuration of the characteristics of an instance, for example, the number of work processes and their types or, even more importantly, the memory configuration of the instance.
2. Configuration of the SAP buffers, that is, the analysis and administration of table buffering, of number range buffering, and of program buffering in other internal SAP buffers.

The following sections briefly describe these categories and give an overview of the most important options and settings.

8.2.3.1 Configuring the Memory of an SAP Instance

In general, the importance of the SAP memory configuration (not of the SAP buffer) has decreased due to the technological progress in recent years. Above all, the enhancement of the main memory capacities, which also involves changing to the 64-bit technology, means the main memory is usually no longer the limiting factor of an SAP instance. However, the configuration must still be performed so that the instance can work with high performance. In the following text, we will briefly introduce the SAP memories and their configuration and provide information that is relevant to performance. However, we won't go into too much detail, as the SAP memory configuration depends on the operating system and would thus fill an entire book if it was described thoroughly.

The memory configuration of an SAP instance must be defined in the instance profile. The parameters of the individual memory areas are also in these sections.

Note

The following text refers only to the SAP memory management for typical UNIX operating systems (HP-UX, Solaris, and AIX). The concepts for configuring the memory of other platforms supported by SAP, such as Linux, Windows, or IBM iSeries, sometimes deviate considerably. For example, Windows and Linux provide an option for Zero Administration Memory Management, where only one parameter defines the total memory that is available for the instance and where the individual memory areas are automatically configured.

The SAP memory areas are:

► Roll memory

Every work process contains a roll memory area that is located in the local process memory. It stores the initial user context that is swapped to the

roll buffer when the process is changed (SAP process multiplexing). The roll buffer itself (don't confuse it with the roll memory) is also referred to as the roll area and, like the extended memory, is a shared memory area. From SAP R/3 3.0 onward, the roll memory plays only a minor role, because the main part of the user context is directly stored in the extended memory and the access change is performed through the pointer. This method is significantly faster than copying the data in the memory.

► **Extended memory**

This shared memory is used by all work processes and is the most important memory area of an SAP instance. It contains all user contexts of the users that are logged in to the instance, with the exception of the small initial context that is being copied between roll memory and roll buffer.

► **Heap memory**

This memory area is a local memory that belongs to one work process. The work process type determines when the heap memory is used. It is used for dialog processes when the extended memory or at least the part of the extended memory that may be used by a single work process is entirely utilized. For nondialog processes, the local heap memory is used immediately after the roll memory, because here no process multiplexing is performed, and the extended memory is therefore reserved for the dialog processes.

► **Paging memory**

Previously, this memory area served to reduce the load of the roll memory for operations with large amounts of data via a paging procedure similar to the paging process of an operating system. Today, the memory is only used when the ABAP commands, `EXTRACT` and `EXPORT ... TO MEMORY ...`, are used.

Table 8.22 provides an overview of the most important parameters of the SAP memory areas. The SAP Help provides a complete overview of all parameters.

SAP Memory Area	Parameter	Description
Roll memory and roll buffer (area)	<code>ztta/roll_first</code>	Defines the size of the initial local roll memory of a work process. The default value is only 1 byte.
Roll memory and roll buffer (area)	<code>ztta/roll_area</code>	Defines the size of the entire local roll memory of a work process.

Table 8.22 Parameters for the SAP Memory Areas

SAP Memory Area	Parameter	Description
Roll memory and roll buffer (area)	rdisp/ROLL_SHM	Defines the size of the roll buffer in the shared memory.
	rdisp/ROLL_MAXFS	If the roll buffer is not sufficient, an overflow file exists whose size is specified with this parameter.
Extended memory	em/initial_size_MB	Defines the total size of the extended memory of an SAP instance.
	ztta/roll_extension	Defines the maximum amount of extended memory that can be used by a single work process. Two other parameters can also be used to specify the maximum memory for dialog and nondialog processes.
	abap/heap_area_total	Defines the total size of the local heap memory as the total of all work processes in one SAP instance.
	abap/heaplimit	If one work process uses more heap memory than this parameter specifies, the process is started again at the end of the transaction to release the memory.
Paging memory	rdisp/PG_SHM	Specifies the size of the paging memory of an SAP instance.
	rdisp/PG_MAXFS	If the paging memory is not sufficient, an overflow file exists whose size is defined with this parameter.

Table 8.22 Parameters for the SAP Memory Areas (Cont.)

Transaction ST02 (memory monitor) gives you an overview of the configuration and the current status of the SAP memory areas. Figure 8.16 shows the section that is relevant for the memory areas.

SAP memory	Current use [%]	Current use [kB]	Max. use [kB]	In memory [kB]	On disk [kB]
Roll area	0,66	1.725	15.744	131.072	131.072
Paging area	0,01	10	35.888	05.536	190.608
Extended Memory	4,70	196.608	692.224	4.186.112	
Heap Memory		0	110.770		

Figure 8.16 SAP Memory Areas in Transaction ST02

The **Current use** column provides information on the current use of the instance. **Max. use** lists the maximum use since the last start of the instance. The **In memory** column contains the maximum values of the individual areas in kilobytes. The areas are defined according to the parameters from Table 8.22. Note that the parameters of the extended and heap memories are defined in megabytes or bytes, whereas the roll buffer (**roll area**) and paging memory (**paging area**) sizes are specified by the number of blocks with a size of 8k. The last column, On disk, shows the maximum sizes of the overflow files of the roll buffer and of the paging memory.

The **Detail analysis menu** button in Transaction ST02 enables you to obtain a selection with further analysis options. Using the **SAP memory** function, you can display a detailed analysis of the SAP memory areas. Here, the administrator can view which user is using how much of which SAP memory at a given time. You can also obtain a historical overview.

Before the extended memory was implemented, the roll-in and roll-out processes were critical for performance during process multiplexing, that is, copying the user context. Thanks to the usage of pointers, this problem has become obsolete. What needs to be done now is to configure the SAP memory areas with enough memory of each area and without wasting memory space that could be of more use somewhere else (for instance, for SAP buffers or the Oracle database). However, if not enough memory is available for one of the areas, the performance of the SAP instance may automatically decrease considerably, or program aborts may occur. The following list summarizes the most important rules for the configuration of SAP memory areas:

- ▶ Choose the roll buffer and the paging memory so that the overflow files are never used, that is, the value in Max. use is always smaller than the value in In memory, as shown in Figure 8.16. The SAP default values for these areas are usually sufficient. An exception is SAP NetWeaver BI, which sometimes places considerably more load on the paging memory.
- ▶ The extended memory is the most important SAP memory area. Therefore, you should calculate it more generously. Earlier recommendations of 6 to 10 MB per user are too small for today's requirements. Consequently, you should consider 20 to 30 MB per user.
- ▶ As a contingency reserve, 20% of the extended memory should remain free, that is, the Current use should never exceed 80% of the In memory value. If the Max. use value indicates that this rule has not been adhered to, you should check if the history if this contingency reserve was often used and increase the extended memory, if required.

If Max. use or the history clearly shows that the extended memory is never used up to 80%, you can reduce it accordingly and use the memory somewhere else.

- ▶ Regarding the maximum amount of the extended memory of a single work process, SAP recommends 10% to 20% of the entire extended memory.
- ▶ You can use heap memory (Current use) without a problem as long as you use it only for nondialog processes. However, if dialog processes use local heap memory, they change to the so-called PRIV mode, that is, they can no longer perform a user change and are bound to the running transaction. You can identify this status in the SAP process overview (Transaction SM50). You should avoid this status by all means. If required and possible, you should specify more extended memory.

Of course, you should keep an eye on the hardware resources when configuring the memory for an SAP instance. It is important that there is enough space in your operating system for all SAP memory areas plus the SAP buffers plus a contingency reserve. If there isn't enough space, the SAP instance won't start. The best option for the performance is that everything (including the maximum heap memory) can be mapped onto the physical memory of the server.

Note on SAP Memory Management

The extended memory is not created completely at the start of the instance but, depending on the usage, grows to its maximum size. During this process, the extended memory, which has been allocated once, is not released for the operating system even if it is no longer used by the SAP instance and is thus actually available. This is a problem if a memory bottleneck occurs on the server, because "available" memory is not recognized as being really available by the operating system and therefore can be removed during swapping, for example. Some new instance parameters that are provided as of Kernel 4.6D (with the relevant patch) can solve this problem. For more details, see SAP Note 724140.

8.2.3.2 SAP Buffers

The SAP system also uses the buffers concept to access data quickly that is often reused. This concept is equivalent to that of buffers in the database. Figure 8.17 illustrates the two-level buffer architecture and the relationships between the buffers. Note that, of course, each instance of an SAP system has its own buffers.

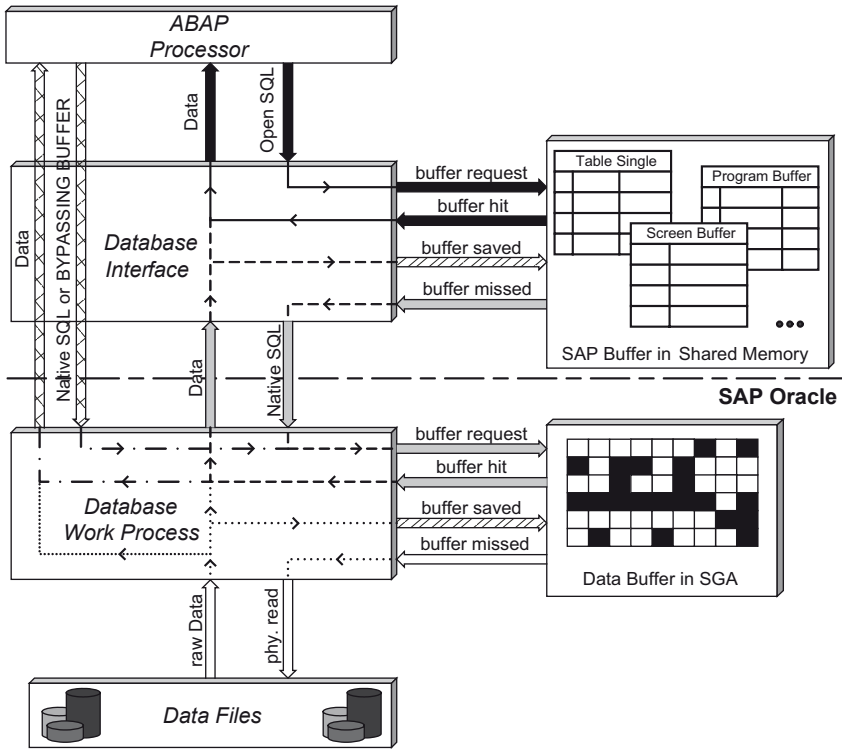


Figure 8.17 Two-Level Buffer Architecture of SAP-Oracle

The combination of database and SAP buffers involves a performance-relevant feature. Let's take a data record for currency conversion as an example. This data record is loaded from the Oracle database if a relevant transaction is performed by the user. First, the Oracle database loads the data record from the data files (or from the raw device) and stores it or the data blocks in the data buffer. The SAP system or the work process receives the data record for the current transaction and stores it also in the SAP buffer for table entries (table single buffer or table generic buffer) if the table that originally contained the data record is provided for buffering.

If the currency conversion is performed again, the data record is provided by the SAP table buffer if it has not become invalid, due to changes, for example. As a result, the data record or the respective blocks are swapped from the Oracle data buffer after a short period of time (depending on the buffer size). If they are also swapped from the SAP table buffer, for example, because other objects have been loaded in the meantime and the buffer size is not large enough, another query of the currency conversion leads inevita-

bly to a physical read of the database. That is, the objects and data that are buffered in the SAP application server are generally swapped from the database buffer, which causes a physical data access when they are queried again from the database.

Therefore, SAP buffers primarily contain other data than the database buffer. Table 8.23 shows which individual buffers are available.

SAP Buffer	Description
Table definition (TTAB)	Buffers the entries of the DDNTT table. Explanation: The name table (NTAB) contains all information on the table and field definitions in the ABAP repository. It consists of the DDNTT (table definitions) and DDNTF (field descriptions) tables.
Field descriptions (FTAB)	Buffers the entries of the DDNTF table.
Short NTAB	Stores a combination of TTAB and FTAB buffers.
Initial record	Stores the layout of the data records.
Program	Here, the compiled executable versions of the ABAP codes are buffered. The swaps are performed on the basis of the LRU concept (least recently used).
CUA	Buffers objects from the SAP GUI, such as menus or button definitions based on the LRU concept.
Screen	Stores the dynpro screens that have been generated previously.
Calendar	Here, all defined factory and holiday calendars from the TFACS and THOCS tables are buffered (also on the basis of the LRU concept).
OTR	This is the Online Text Repository buffer that stores texts that are, for example, used in BSPs.
Table generic/ Table single	These are the entries for table buffering in an SAP instance (as described below).
Export/Import Exp/Imp SHM	This buffer is used for all work processes and stores data clusters using specific ABAP commands (see SAP Note 702728).

Table 8.23 SAP Buffers

To carry out a qualitative evaluation of the buffers, it is required that they be in an established state, as is the case with database buffers.

Transaction ST02 enables you to access the memory monitor of the instance to which you are currently logged in. Figure 8.18 shows the part of the monitor that lists and evaluates the individual SAP buffers. The individual fields and their meanings are described in Table 8.24.

Buffer	Hitratio [%]	Allocated [kB]	Free space [kB]	Dir. size [%]	Dir. size Entries	Free directory Entries	Swaps [%]	Database accesses
Nametab (NTAB)								
Table definition	99,80	11 017	4 560	49,60	50 000	24 037	49,67	36 508
Field description	99,79	103 086	31 680	31,60	50 000	26 307	52,61	25 879
Short NTAB	99,85	5 563	2 918	72,95	12 500	7 763	62,10	4 738
Initial records	99,85	11 563	7 394	73,94	12 500	2 919	23,35	27 895
Program								
SOL	99,67	500 000	1 217	0,25	125 000	111 847	89,48	471 732
Screen	99,90	0 000	95	0,51	4 000	2 309	59,73	2 450
Calendar	99,75	24 415	14 088	59,83	10 000	9 189	91,89	5 852
DIR	100,00	4 006	3 531	100,00	2 000	2 000	100,00	0
Tables								
Generic key	99,85	48 820	3 052	6,55	10 000	1 267	12,67	217 260
Single record	99,87	30 000	15 730	52,09	500	281	56,20	101 131
Export/import								
Exp./imp. SHM	89,88	40 000	1 581	5,01	30 000	15 760	52,53	0
Exp./imp. SHM	95,99	4 099	2 529	99,94	2 000	1 999	99,95	0

Figure 8.18 SAP Memory Monitor (Excerpt Containing the SAP Buffers)

Field	Description
Hit ratio	The hit ratios of a buffer are indicated as a percentage. They are calculated in the same way as the hit ratio of the database buffers: $Buffer\ quality\ (hit\ ratio) = (Buffer\ requests - Database\ requests) / Buffer\ requests \times 100\%$
Allocated	Memory space (in RAM) occupied by the SAP buffer.
Free space	Memory space that is available for the buffer.
Dir. size	Maximum number of possible buffer entries.
Free directory	Number of buffer entries that are available.
Swaps	Number of swaps from the buffer since the last start of the instance.

Table 8.24 Fields in the Memory Monitor

Hit ratio and swaps are the essential criteria for SAP buffers. SAP recommends that the hit ratio of the buffers should be >98%. The two export and import buffers are an exception with an optimum hit ratio that is supposed to be >80%. If possible, swaps from buffers should be avoided. However, depending on the utilization and usage of the SAP instance, it is not always possible to avoid them. For example, if an instance runs in another operating mode to execute more batch jobs at night, other ABAP programs and tables are required that are loaded into the program or nametable buffer. This usually leads to swaps. You can therefore tolerate a small number of swaps (a few hundred swaps per day). SAP recommends that up to 10,000 swaps per day can be accepted for the program buffer.

If the number of swaps increases considerably, you must check if the *free space* or the number of *free directory entries* of the buffer is insufficient. The respective instance parameter must then be increased step by step to elimi-

nate the swaps. You can find the parameters for the SAP buffers using the **Current parameter** button in the memory monitor.

In addition to the hit ratio and swaps, you should also keep an eye on the free space for SAP buffers. You can possibly find unused memory resources if large parts of a buffer are not used in the established state. For example, in Figure 8.18, this is the case for the field description buffer because 30 MB are unused. It makes more sense to use this memory somewhere else.

There are three types of table buffering in an SAP system:

1. Single buffering

In this process, each record (row of a table) is stored individually in the TABLP buffer if it has been read once on the database. To use single buffering, it is important that all key fields are qualified in the `where` condition for a query

2. Full buffering

If a record is read from the database, the entire table is stored in the TABL buffer.

3. Generic buffering

This buffering type is specified by the number of key fields used for selection. If a record is read from a table buffered as generic 1 from the database, all other data records are buffered that are identical to the record initially read in key field 1. Corresponding buffering with n key fields is also possible. The generic buffered data records are in the TABL buffer.

Note

When full buffering is activated, a client-dependent table, that is, a table that always has the `MAN` key first, is automatically buffered as generic 1.

The entire buffer management is mapped onto the database interface of the individual SAP work processes, that is, here it is decided when which buffer is accessed. For the access of the buffer, it is decided whether all required keys of a table are specified in the `where` condition. Let us assume that the `TAB` table with the key fields `KEY1`, `KEY2`, and `KEY3` is buffered as generic 2, that is, via `KEY1` and `KEY2`. The following call would benefit from a buffering process:

```
SELECT * FROM TAB WHERE KEY1=X and KEY2=Y;
```

In contrast, the following calls could not be buffered and would thus cause an access to the database:

```
SELECT * FROM TAB WHERE KEY1=X;
SELECT * FROM TAB WHERE KEY1=X and KEY3=Z;
```

Additionally, there are numerous exceptions where the table buffer is not used either:

- ▶ When the SQL commands `SELECT FOR UPDATE` or `SELECT DISTINCT` are used
- ▶ When the aggregate functions `SUM`, `MIN`, `MAX`, and `AVG` are used
- ▶ When the Native SQL statements or the Open SQL condition `BYPASSING BUFFER` is used

Because all SAP buffers are kept separately for each instance, the table buffer is forced to synchronize the instances. If the content of a table buffer is changed, the running work process writes a corresponding entry to the DDLOG database table. The instances read this table regularly to invalidate data records in their own table buffers that are affected by the changes. That is, if a table is individually buffered, only the affected data record is declared as invalid; however, if tables are generic or fully buffered, the complete generic part of the table or the entire table in the buffer is invalidated.

This synchronization process is controlled via the profile parameters `rdisp/bufrefmode` (controls reading and writing of the DDLOG table) and `rdisp/bufreftime` (specifies the frequency at which the DDLOG table is read – default: 60s). Furthermore, you can monitor the buffer synchronization in the SAP memory monitor: **SAP_memory monitor (ST02) • Detail analysis menu • Buffer synchron.**

The synchronization of the table buffers between the instances may have a negative effect on the system performance. Therefore, some criteria must be met by tables and views (they can also be buffered) for useful buffering:

- ▶ The table must be small and read very often.
- ▶ The change rate must be very low, for example, less than 1% changes per day for tables with a size of one megabyte.
- ▶ A short-term inconsistency must be acceptable because delays during synchronization between the instances (`rdisp/bufreftime`) may occur.

On the basis of the characteristics mentioned, the individual data classes in an SAP system can be assigned relatively rigidly to one buffering procedure. SAP distinguishes between three data classes that are contained in a database in the system (apart from the actual ABAP code and “technical” data):

1. Transaction data

All data that is generated and changed in large quantities during operation, such as invoices, delivery notes, sales orders, material movements, and so on. The tables grow rapidly during operation and can thus reach a size of several gigabytes. Therefore, they are generally not suited for buffering in the SAP system.

2. Master data

Master data is changed rarely or never during live operation and contains information on material, customers, vendors, and so on. The respective tables change less than the transaction data but still reach a size of several hundred megabytes. Therefore, master data tables are also not included in the SAP table buffer.

3. Customizing data

Data that is generated when mapping the enterprise processes onto the SAP system (customizing). The most common examples are company codes, factories, sales organizations, conditioning, and so on. The respective table records are changed or supplemented rarely during operation and are therefore usually buffered in the SAP system.

On the basis of SAP's own experience, table buffering functions are already configured in the supplied versions of the different SAP software solutions.

To decide when and how a table is buffered reasonably or not, you have to monitor the SAP table buffering processes. This can be done in the SAP memory monitor (ST02) via **Detail analysis menu • Call statistic** or using Transaction ST10. A selection screen is displayed where you must select the Table type, Period, and SAP instance factors. Because every instance has its own buffers, you must theoretically analyze each instance. However, this is only virtually relevant if different tasks are performed on these instances, such as batch against dialog instances, or if organizational enterprise parts (for example, international branch offices) are distributed across different instances and thus other data must be buffered.

Figure 8.19 shows an excerpt from the table access statistics, and Table 8.25 lists the most important columns and their meanings. (Note: You can expand the individual detail columns via buttons.)

Table	Buffer state	Buf key opt	Invali- dations	Buffer size (bytes)	Size maximum (bytes)	Total	ABAP/IV Processor requests	Changes	DB activity
							Direct reads	Seq. reads	Calls
Total			20 271	48 482 095	162 951 205	482 251 527	143 708 840	247 375 811	51 086 068
ATAB_BPCDGC	pending	sng	1 548	0	273 920	39 937	2 915	35 155	482 844
BEPOGAG			0	0	0	30	0	30	31 966
WDRDGGG1			0	0	0	32	0	32	182 513
BUPUNCA			0	0	26 500 388	0	24 800 132	234	181 848
ABR_HBERT	pending	gen	0	0	17 890	21 147 134	0	21 147 134	46 838
PTDW_FWS_DB			0	0	12 734 570	0	12 734 570	0	18 995
PPD11			0	0	11 855 573	0	11 855 573	0	34 836
EXDOKL			0	0	1 990 514	1 940	1 942 201	162	260 996
KROSS			0	0	99	0	35	61	29 347
ITRNL_EXIT	absent	gen	0	0	4 882	0	4 882	0	16 238
GL_DD3			0	0	5 842 544	0	5 842 279	255	12 828
MC1SWANWSETUP			0	0	5 851 182	0	5 813 328	237 872	555 827
MC1SWD1TRIE1UP			0	0	5 850 828	0	5 813 854	237 872	555 565
SWTCS			0	0	3 723 172	2 182 411	254 881	1 376 878	6 845 886
BOZPS			0	0	5 595 811	0	5 595 545	471	5 577
STCF			0	0	1 287 682	0	811 928	455 763	1 275 488
TST01			0	0	3 744 841	2 054 251	485 881	1 688 118	6 418 476
TRML_PAB			0	0	4 888 257	0	485 881	4 418 416	4 882 155
DD101NC			0	0	75 479	0	75 479	0	75 700
ALSTL08			0	0	89 256	0	34 828	34 828	89 596
T520N	absent	gen	0	0	3 530 270	0	3 530 270	0	4 155
SOEOLMP			0	0	4 296 918	4 787	4 292 160	0	4 282 288
TAD1R	valid	sng	2	23 000	7 127 816	11 344 589	601 267	10 743 087	155
USR13	absent	gen	0	0	2 927 742	0	2 927 742	0	5 400
T1LCA			0	0	2 884 221	0	2 884 220	1	22 929

Figure 8.19 Excerpt from the Table Access Statistics

Column	Description
Table	Name of the table
Buffer state	The most important possible states are: <ul style="list-style-type: none"> Valid: Table is in the buffer and valid. Invalid: Table buffer has become invalid and cannot be loaded yet because the change has not been completed. Pending: Table buffer has become invalid and cannot be loaded yet because the grace period is still running. Loadable: Table buffer was invalid and can be loaded again. Absent: Table has never been loaded. Displaced: Table was swapped from the buffer. Error: Very important state, particularly regarding the performance, because it indicates that table buffering was cancelled (see SAP Note 618868, Section 9, Table Buffering)
Buffer key opt	Buffering type: ful = full, gen = generic, sng = single.
Buffer size	Space in the buffer currently occupied by the table.
Size maximum	Maximum space in the buffer occupied by the table.
Invalidations	Indicates how often the table buffer was invalid.
ABAP/IV proces- sor requests	Number of ABAP requests for the table, which can be broken down as follows: direct reads, sequential reads, and changes (update, inserts, deletes)
DB calls	Combination of direct and sequential fetches (transferring the results of an SQL request to the calling SAP work process).
DB rows affected	Number of data records that are transferred from the database to the SAP system. Exception: initial load of the buffer.

Table 8.25 Columns of the Table Access Statistics

The following list describes how to check the buffered tables and how to decide if it makes sense to buffer them:

► **Estimation of the database accesses (DB rows affected)**

The database accesses of buffered tables should be considerably smaller than those of unbuffered tables at a similar number of requests. If there are buffered tables at the beginning of the table access statistics after the table was sorted by "DB rows affected," you should further examine them.

► **Change rates of the tables**

They are calculated on the basis of the following formula:

(Values from the table access statistics from the ABAP request area): $\text{Change rate} = \text{Changes} / (\text{direct reads} + \text{seq. reads}) \times 100\%$

Reference values for acceptable change rates are:

- Table size < 1 MB ↔ Change rate < 1%
- Table size > 1 MB and < 5 MB Change rate < 0.1%
- Table size > 5 MB are rarely buffered; if buffered, change rate < 0.01%

► **Size of the table in the buffer**

Sort the table access statistics by the buffer size column and ensure that all larger buffered tables (>100,000 bytes) are set to valid, if possible. If this is not the case, the table buffer should be extended, if possible. Note: The **Analyze table** button enables you to perform a complete table analysis in the background. This analysis indicates, for instance, the size of the table in the database or the distribution of the generic areas.

► **Select quality**

When you double-click on a row of the table access statistics, a detail screen for the respective table opens. There, you will find the select quality as the ratio between ABAP requests and database calls (**Fetches/Exec:** fetching/changing data records). The quality should be approximately >95%.

Vice versa, the administrator searches for tables that are not buffered but should or could be buffered. For this purpose, here is a brief overview of the most important criteria:

► **Number of ABAP requests**

Sort the table access statistics by total ABAP requests (for a better overview, you can view only the unbuffered tables). The ABAP Dictionary tables, DDNTF and DDNTT, are usually listed at the top. However, these tables are already stored in the nametab buffer. Look for Customizing or customer-specific tables in the top request entries (see the Remark box on the next page).

► Change rates of the tables

Once you have found the relevant tables, determine the change rates as described above and compare them with the recommendations.

Remark

How do you recognize customizing tables? There are numerous customizing tables, such as the condition tables Axxx (xxx = 000 – 999). If you search for a table in the standard SAP system, you can find further information as follows:

1. Look at the short text of the table in the Data Dictionary (Transaction SE11) and check the specifications under **Goto • Technical Settings** in the Logical storage parameters field (see Figure 8.20).
2. Search for the table for your application (for example, ERP or SCM) in the SAP documentation under <http://help.sap.com>. Because Customizing is documented very well, this documentation mentions or describes nearly all Customizing tables.
3. Look for SAP Notes on the table in the SAP Support Portal. There are some explicit notes on buffering for some tables.

The buffering settings for a table can be made in the Data Dictionary (Transaction SE11). There, you must enter the respective table and view and change it via **Goto • Technical Settings** (see Figure 8.20).

Name	TCURR	Transparent Table
Short text	Exchange Rates	
Last Change	SAP	06.11.2003
Status	Active	Saved

Logical storage parameters	
Data class	APPL2 Organization and customizing
Size category	0 Data records expected: 0 to 15.000

Buffering	
<input type="radio"/>	Buffering not allowed
<input type="radio"/>	Buffering allowed but switched off
<input checked="" type="radio"/>	Buffering switched on

Buffering type	
<input type="checkbox"/>	Single records buff.
<input checked="" type="checkbox"/>	Generic Area Buffered
<input type="checkbox"/>	Fully Buffered
No. of key fields	1

Log data changes
 Maintain as transparent table

Figure 8.20 Table Buffering Settings

Some final remarks on table buffering: Always check whether it is a table from the standard SAP system or a customer-specific table. As already mentioned, SAP provides your standard tables with the relevant buffering settings. You should only change them if SAP explicitly recommends or authorizes this (for example, in an SAP Note). Regarding customer-specific tables, you should discuss with the developer of the table or application that uses the table if buffering is possible or makes sense.

In general, activating the buffering function is more dangerous than deactivating it. If you deactivate a buffering function, performance problems of applications that use the table are the worst that can happen. In contrast to that, if the table buffering is switched on – caused by a delay in the synchronization between instances of the SAP system – logical inconsistencies may occur.

Having viewed the individual components and areas of an SAP system with an Oracle database regarding the performance under administrative aspects, the last section deals with aspects related to the program.

8.3 Analyzing Program-Based Performance Problems: SQL Optimization

In Chapter 3, *Oracle Fundamentals*, the possibility of optimizing the query language SQL via DBMS is described as one of the strengths of relational databases. The SQL query describes the desired result and leaves it up to the database system to calculate the result set. In general, the DBMS provides much better ways to create the best execution plan. Theoretically, expert software developers should design a better plan only in exceptional cases. However, in practice, the Optimizer is not able to locate the best plan in all cases. A developer with knowledge of the interdependencies between the attributes and of the value distribution often must support the Optimizer by providing hints.

As an SAP-Oracle administrator, you are responsible for detecting and avoiding program-based bottlenecks. Thomas Schneider reports that quite frequently, situations occur in which a few expensive SQL statements cause more than 50% of the database load (*SAP Performance Optimization*, SAP PRESS, 2005). These issues can often only be handled with the support of the development team, even if you as an administrator can immediately alleviate the problems by creating an additional index or using other technical measures.

Our examples show you how to develop high-performance programs, demonstrate a few typical errors, and describe their impact on the interaction between SAP and Oracle. In this context, we will introduce essential SAP

tools for analyzing SQL statements in the following sections. In addition, we'll also demonstrate how you can use and create indexes.

On the one hand, this will provide you with material for advising the developer team regarding a high-performance use of SQL in a qualified manner. On the other hand, you will learn how to use these tools to identify problems in that area, and you will get to know basic solution approaches.

8.3.1 Two Goals: Functionality and Performance

ABAP programs are often developed by application experts whose IT knowledge is rather limited. You have to ensure not only that the programs provide functionally correct results, but also that the tools be implemented in such a way that they provide high performance.

<pre>REPORT zbuch_bsp1. DATA wa LIKE sbook. SELECT * FROM sbook INTO wa. IF wa-passname = 'Cindy Lindworm'. WRITE: wa-fidate,/. ENDIF. ENDSELECT.</pre>	<pre>REPORT zbuch_bsp2. DATA wa LIKE sbook. SELECT * FROM sbook INTO wa WHERE passname = 'Cindy Lindworm'. WRITE: wa-fidate,/. ENDSELECT.</pre>
"Bad" Statement	"Good" Statement

Figure 8.21 Inefficient and Efficient Use of SQL

The example shown in Figure 8.21 shows two functionally identical variants of a report that – based on the flight data table, SBOOK – lists all days on which Cindy Lindworm booked flights. For this purpose, in the variant on the left, all data records of the SBOOK table are read from the database. After that, the data records determined for further processing (here: output) are selected. In the other variant, the selection is performed by the `where` clause, which is triggered by the Oracle DBMS. The second variant has two advantages:

1. The Oracle server process transfers only the data that is really needed for the work process.
2. Oracle can select the correct data records very quickly by using existing help data structures.

An old IT saying goes that you make mistakes whenever they are possible. Functional errors can either be avoided by using proper specifications or identified by means of testing. A performance-critical programming style may remain undetected for a long time. That's because hardware performs at

different speeds or the SAP systems in use have different loads. Consequently, the statement that the execution of the SAPBC_DATA_GENERATOR report takes about 138 seconds for creating the flight data model is of no real relevance.³

Note

Not every "bad" SQL statement causes performance problems. SAP and Oracle proactively provide mechanisms to process such statements quickly (see Chapters 2, 3, and 4). On the other hand, not every optimized SQL statement is fast. Some processes simply need a lot of time.

8.3.2 Effects

A poorly formulated SQL statement has both indirect and direct effects on performance. Figure 8.22 illustrates the direct effects. In the example shown in this figure, the transferred data packages between Oracle and the SAP application server consists of exactly one SQL query that is submitted and requests the entire SBOOK table. Therefore, the Oracle database needs to deliver the entire table, which can involve many physical reads (❶). In addition, the transfer of the entire table with delivery of an unnecessary number of data packages is involved (❷). The Oracle system itself cannot perform any optimization action because it does not know that numerous data records in the application are discarded. In the application server itself, all data records are (sequentially) verified with regard to whether they correspond to the IF condition (❸). This is quite time-consuming for the generated flight database that contains 90,000 data record in the SBOOK table.

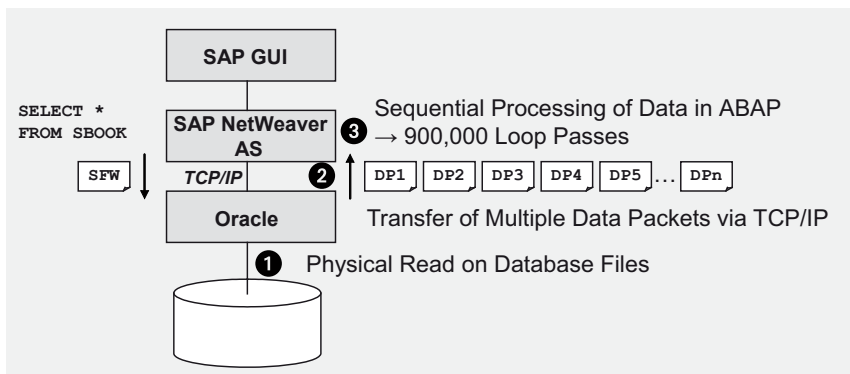


Figure 8.22 Effect of a "Bad" SQL Statement

³ We have chosen the monster data record variant and created the database in a background job (Transaction SM36).

Figure 8.23 shows the processing of a good SQL statement. The statement is improved because not all attributes of the desired data records that are marked with * are requested — only the `FLDATE` attribute is. Based on an exact description of the desired result quantity, the Oracle system can deliver the minimum result set. Consequently, only a small number of loop passes is necessary for the work process (③), and only some data is transferred, which fits into a single package in our example (②). Moreover, Oracle can create an optimized execution plan. For example, by using indexes, the physical read operations can be limited to some index data blocks as well as to those data blocks containing the requested data records.

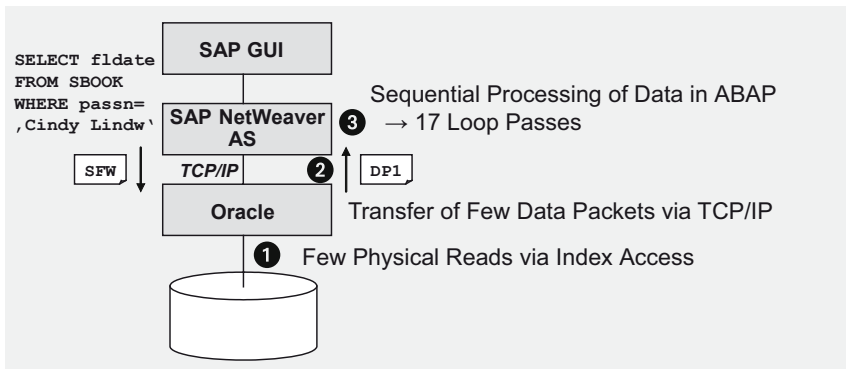


Figure 8.23 Effect of a "Good" SQL Statement

However, not every SQL statement that requests only the necessary data, like the one in Figure 8.23, is optimized. You also need to consider the processing by the Oracle system, particularly the use of indexes. From the point of view of performance, it is sometimes useful to extend the SQL queries with seemingly redundant `where` conditions to use an existing index. However, sometimes you may have to create a new index. We will describe these two aspects in greater detail in Section 8.3.6, *Indexes for Faster Access*.

In addition to the direct effects, there are also some indirect effects we should take a look at. For example, data that is requested unnecessarily occupies space in the buffers of the application server and database system. In the case of write requests, buffered data must be processed by transaction management.

For experiments and optimizing operational processes, SAP tables allow you to define whether and to what extent they are buffered in the table buffer of the application server. To do that, you must select a table in Transaction

SE11 (ABAP Dictionary), go to **Display** or **Change** and then select **Technical Settings**. Figure 8.24 shows the buffering settings. You can define whether you want a table to be buffered, whether buffering is allowed but switched off, or whether buffering is not allowed at all. If you allow buffering, you can also specify if individual data records, table sections, or entire tables should be buffered. Because the SCARR table used in the example is part of the standard SAP system, the settings can only be changed using an object key, which must be requested from SAP. If a table is buffered, it is very likely that the Oracle database is not accessed in experimental runs, which you should take into account when discussing the results.

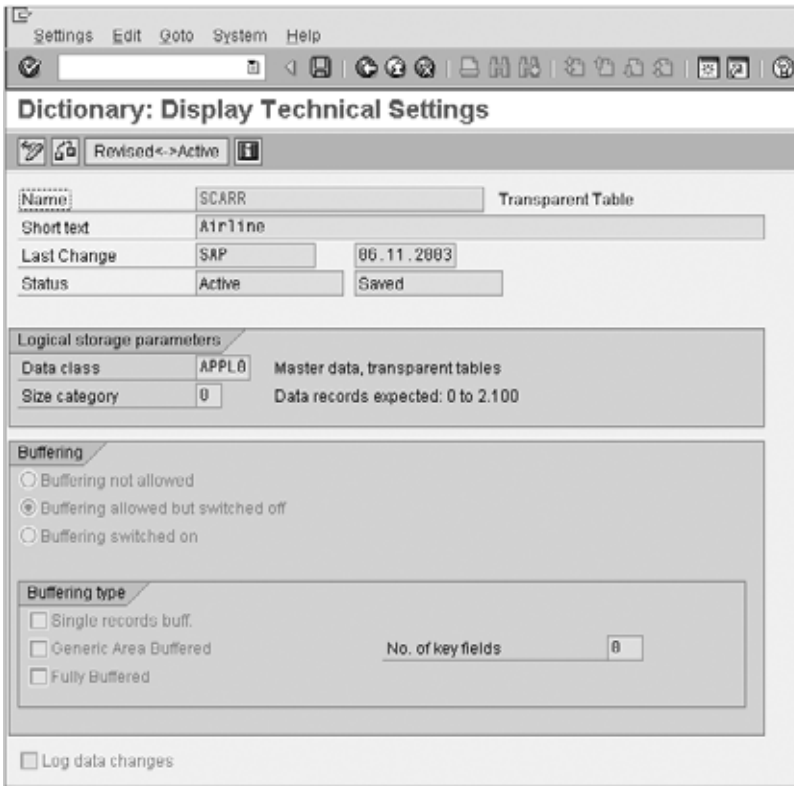


Figure 8.24 SCARR Is Not Buffered

8.3.3 Problem Analysis Tools

SAP provides a range of tools to detect and analyze performance problems that are related to SQL queries. There are tools that support the examination

of the entire system and others that enable a detailed analysis of an SQL statement or program.

In this context, the analysis of the transaction profiles (Transaction ST03) and Oracle performance (Transaction ST04 or ST04N) again plays a major role (see Figure 8.25). You can use these tools to identify and isolate problems from the point of view of the entire system first.

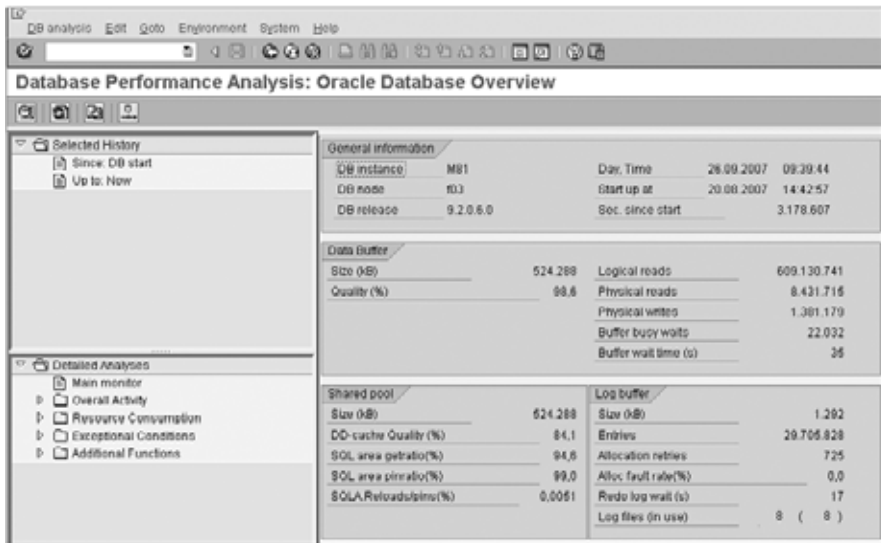


Figure 8.25 Oracle Performance Analysis with ST04N

The following parameters are important for optimizing SQL statements:

- ▶ **SQL area getratio** describes the ratio between matches and requests for an object in the library cache and should be close to 100% for a production system. This shows that the shared pool size is well selected for the actual query load, which can be caused by good queries or a generous measurement.
- ▶ **SQL area pinratio** describes the ratio between matches and requests for reading and executing objects and should also be close to 100%. Here again, expensive queries can have a negative impact due to displacements.
- ▶ Vice versa, **SQLA.Reloads/Pin** describes the ratio between the necessary reloads (SQL query parsing) and the accesses. Consequently, a value close to 0 should be reached here.

The quality of the dictionary cache should also be close to 100 because it is required for the plan creation and query processing. In addition, the size of

the data dictionary is known because of the fixed structure⁴. Therefore, the system can size the memory area in an optimal manner. This is not the case with the system used in Figure 8.25 because it has been in use only for a short period.

You should know the typical values for your own system. Deviations may occur when maintenance work is carried out on the system. Usually, you as an administrator will carry out this work or at least will be involved. In this respect, you will develop a feeling for the behavior of your system.

Deviations in the normal operation, however, must be checked thoroughly. The goal is to locate the cause of the problem, which might be related to the use of an inefficient SQL statement. The analysis of the SQL requests may be an option to identify this type of statement. In Transaction ST04, you can access a sorted list of all SQL statements that are stored in the Oracle system by selecting **Detail Analysis • SQL Request**. If you want to select the requests of a specific user in the selection screen, note that only the UNIX users of the SAP system are referred to here, but not the SAP users that are unknown to the Oracle system. As of SAP Basis 8.1, the CLIENT_ID will be available, which corresponds to the SAP end user.

Executions	Cur. BH	Disk reads	Reads/Exec	Buffer gets	Bgets/Exec	Proc. rows	Rows/Exec	Bgets/row	SQL text	CPU Time	CPU Time/Exec	Elapsed time	Elapsed Time/Exec	Wait	Wait	SQL statement
62	0	52,213	842.1	532,952	8,596.0	0	0.0	0.0	0	7,190,000	115,967.7	28,371,893	450,035	22.18	519	SELECT * FROM...
215,089	0	42,883	0.2	3,399,298	15.3	259,180	1.2	8.2	10,287	42,070,000	195.6	91,223,663	424.2	48.15	228.8	SELECT * FROM...
6	0	37,702	6,283.7	7,249,261	1,208,216.2	637,184	106,194.0	11.4	0	36,670,000	6,111,666.7	43,721,589	7,266,931.5	7,061	1,17	SELECT * FROM...
3	0	37,678	9,226.3	28,204	9,428.6	0	0.0	0.0	0	2,620,000	873,333.3	32,591,231	10,833,743.7	29,988	9.96	SELECT * FROM...
6	0	36,441	4,416.2	29,119	4,853.2	2,196	366.0	13.3	0	960,000	160,000.0	4,770,565	795,094.3	2,810	435	UPDATE * FROM...
3	0	35,361	8,453.7	35,590	11,863.3	3,757,182	1,251,364.0	0.0	0	3,960,000	1,320,000.0	6,847,314	2,282,438.0	2,887	943	SELECT * FROM...
109,650	0	33,388	0.3	3,163,032	18.7	48,635	0.4	48.3	0	13,260,000	120.9	336,038,017	3,143.4	251.7	3.03	SELECT * FROM...
10	0	31,403	2,148.3	21,826	2,182.6	0	0.0	0.0	0	1,160,000	116,000.0	1,921,844	192,184.4	761.6	76.1	SELECT * FROM...
10	0	31,400	2,148.0	21,440	2,144.0	0	0.0	0.0	0	800,000	80,000.0	813,196	81,319.6	13,184	1.31	SELECT * FROM...
3	0	30,477	8,835.7	33,727	7,875.7	51	17.0	445.6	0	2,660,000	888,666.7	6,084,606	2,028,202.0	3,424	1.14	SELECT * FROM...

Figure 8.26 Analysis of the Shared Cursor Cache

The top SQL statements should be primarily analyzed in terms of disk reads, buffer gets, and elapsed time. The read access per execution (**bgets/exec**) and the read access per record (**bgets/row**) provide important information. They are important when performing a thorough and individual analysis of an SQL statement. **bgets/exec** can be critical if a request carries out many buffer accesses that may be redundant, as shown earlier in the example in Figure

4 Apart from custom developments, patches, and upgrades, the database structures generally remain stable.

8.21. If the value for **gbets/row** is high, many blocks must be read from the database to deliver a small number of data sets to the application server. In this context, the database access is possibly performed without using appropriate indexes. However, complex joins and several inlists can entail high values for **bgets/exec** without having the technical potential for optimization.

8.3.4 Detailed Analysis of SQL Statements

Furthermore, you can view the execution plan for individual SQL queries (Ctrl-Shift-F6). The query plan for the query

```
SELECT * FROM SBOOK
INTO WA
WHERE PASSNAME = 'Cindy Lindworm'
```

provides the execution plan shown in Figure 8.27. Because no appropriate index is available, the entire table is accessed using the primary index.

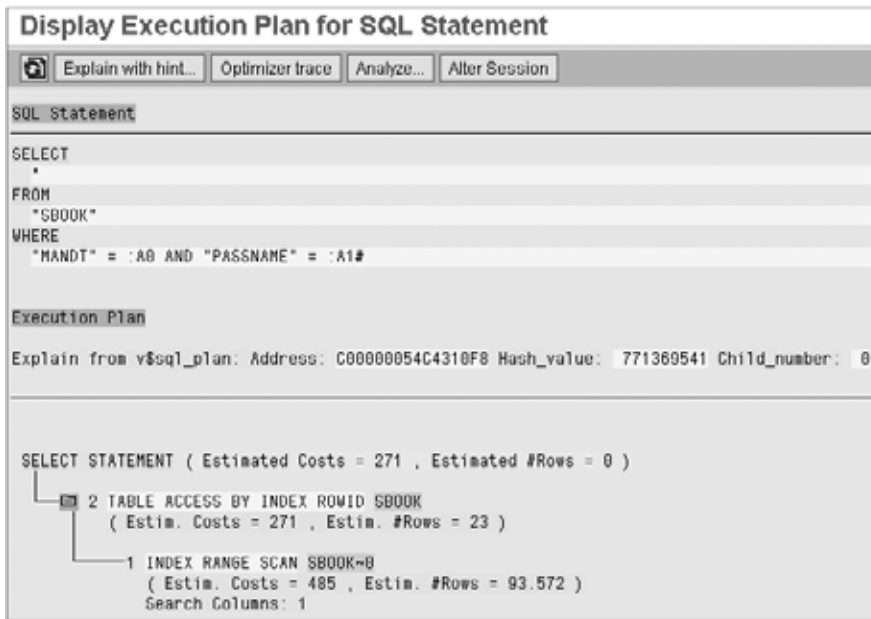


Figure 8.27 Execution Plan for an SQL Query

The ABAP Dictionary (Transaction SE11; see Figure 8.28) provides more information about the available indexes. No index is defined for the SBOOK table via the PASSNAME attribute queried by the where clause.

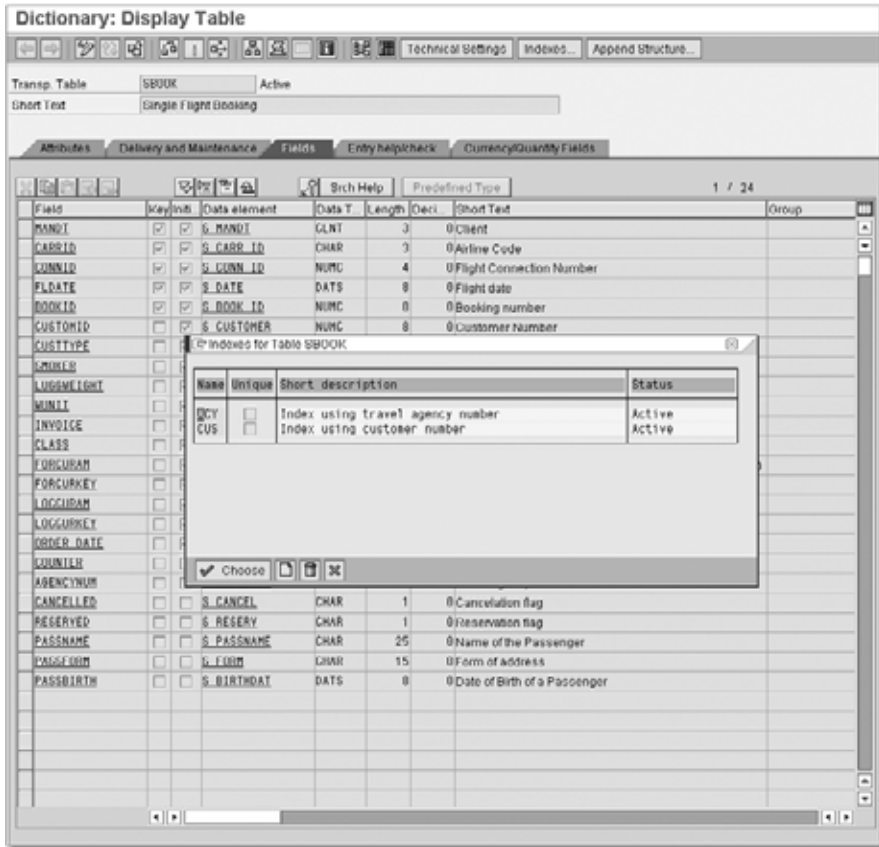


Figure 8.28 Transaction SE11 for Displaying Available Secondary Indexes

8.3.4.1 Analyzing Programs

An alternative approach is to examine the programs that produce high loads or send SQL statements causing these loads. The ABAP Dictionary provides the **Where used** function to enable the identification of programs that use the table. The list of found programs can be very long. In that case, the use of the shared SQL analysis table is more appropriate (see Figure 8.26), which also contains the name of the program that submitted the listed SQL statement first.

Starting from here, a runtime analysis of the program can be carried out (use Transaction SE30 or follow menu path **Tools • ABAP Workbench • Test • Runtime Analysis**). In the first step, the program to be examined must be executed. In the second step, you can view the evaluation (see Figure 8.29).

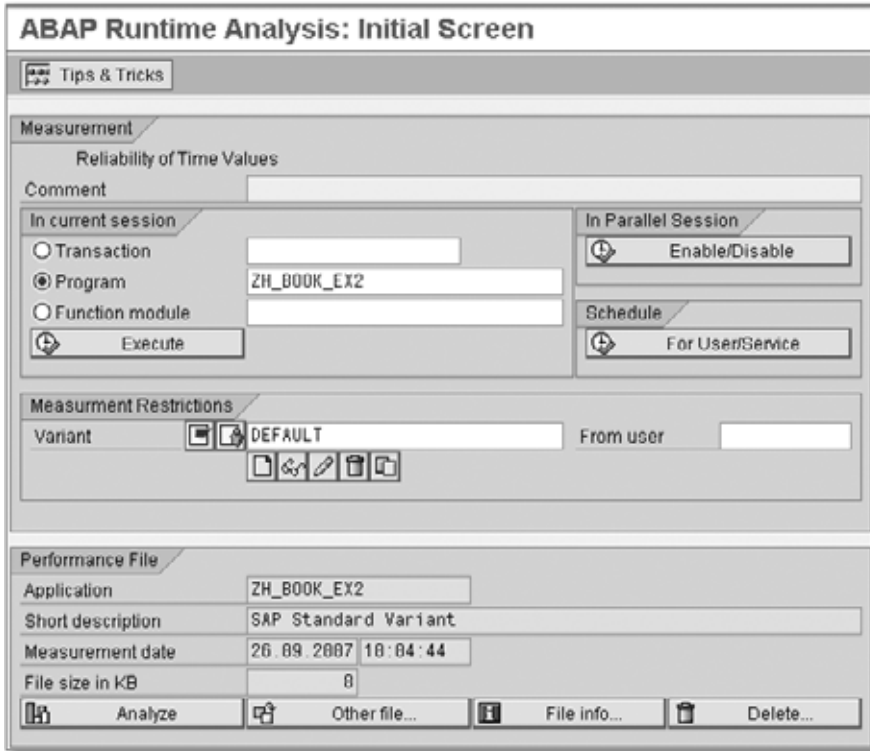


Figure 8.29 Initial Screen of the Runtime Analysis

Figure 8.30 shows the results of the "bad" SQL statement used at the beginning including the use of the `IF` condition. With a total execution time of three seconds, the program has already slowed down noticeably. Most of the time is assigned to the database system that has to transfer about 900,000 records of the `SBOOK` table to the application server. From the point of view of the architecture, this is absolutely necessary because the buffering of the `SBOOK` table is not permitted.

In contrast, the program, which has been optimized by shifting the name verification into the `where` condition, shows a runtime behavior that has been improved by a factor of >10 (see Figure 8.31). Whereas the processing with `IF` requires approximately 3 seconds, the variant using `WHERE` only takes 0.2 seconds. On the one hand, the load in the database system is lower, because only a few records needed to be delivered. On the other hand, the application server is also less stressed, because it has to perform the `SELECT` loop only a few times.

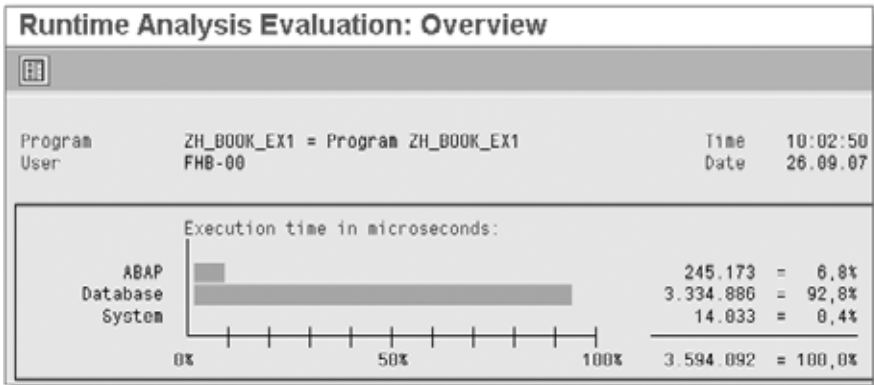


Figure 8.30 Runtime of a Bad Statement

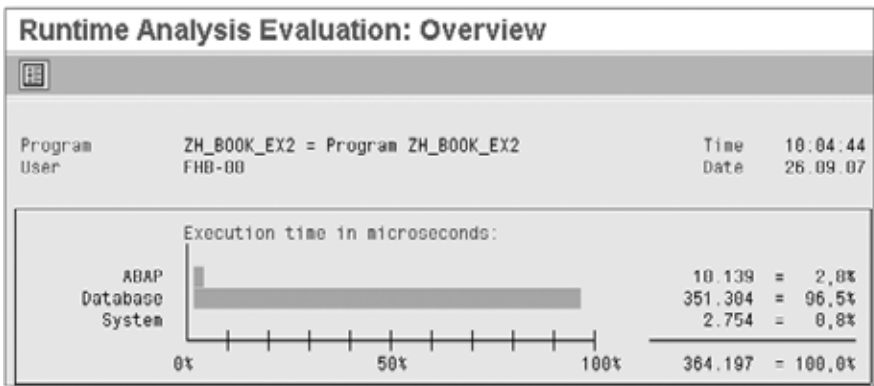


Figure 8.31 Runtime of a Better Statement

Note that the runtime measurements do not always provide the same results. For buffered tables, the initial execution can involve a high database load, whereas the second execution can leverage the data from the buffer of the application server, and the query does not access the Oracle database. The Oracle buffers behave in a similar manner. Consequently, the Oracle data buffer also buffers tables for which no buffering (in the application server) is allowed from the point of view of the SAP system. In this respect, it is not always easy to construct clear examples. The actual system behavior also has an impact on the measurements:

- Other transactions generate loads and require CPU time and data transfer volume.

- ▶ Other SQL queries displace data from the buffers to the application server and Oracle levels.
- ▶ Other SQL queries already stored the data viewed during the measurement in the buffers at both levels (which is actually positive in terms of the overall performance).

Thus, the runtime analysis is helpful, especially for addressing a specific problem.

You can use the SQL Trace (Transaction ST05 or menu path **Tools • ABAP Workbench • Test • SQL Trace**) to perform a very detailed examination of a single program. Here, the database interface on the side of the SAP system logs the processing steps for SQL queries in detail. This includes the operations provided by the record-oriented interface of the five-layer architecture (see Chapter 3). Figure 8.32 shows the large number of fetch operations that occur with an unspecific SQL query against the SBOOK table.

Duration	Obj. name	Op.	Recs.	RC	Statement
13	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'ZH_BOOK_EX1' AND
3.326	PROGDIR	FETCH	1	0	
6	DWINACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_
216	DWINACTIV	FETCH	0	1403	
6	DWINACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_
151	DWINACTIV	FETCH	0	1403	
6	TRDIR	REOPEN		0	SELECT WHERE "NAME" = 'ZH_BOOK_EX1'
255	TRDIR	FETCH	1	0	
513	SBOOK	PREPARE		0	SELECT WHERE "MANDT" = :A0
5	SBOOK	OPEN		0	SELECT WHERE "MANDT" = '901'
65.006	SBOOK	FETCH	320	0	
3.988	SBOOK	FETCH	320	0	
3.866	SBOOK	FETCH	320	0	
4.148	SBOOK	FETCH	320	0	
3.740	SBOOK	FETCH	320	0	
3.965	SBOOK	FETCH	320	0	
4.036	SBOOK	FETCH	320	0	
3.809	SBOOK	FETCH	320	0	
4.097	SBOOK	FETCH	320	0	
4.047	SBOOK	FETCH	320	0	
3.789	SBOOK	FETCH	320	0	
4.186	SBOOK	FETCH	320	0	
3.807	SBOOK	FETCH	320	0	
2.075	SBOOK	FETCH	320	0	

Figure 8.32 SQL Trace: High Number of Fetch Operations

An optimized statement that performs the selection using the `where` clause needs only one fetch for the few Cindy Lindworm records. However, this fetch runs relatively long (see Figure 8.33).

Note that when you use an SQL trace, all SQL queries are logged if no filters are used for specific tables, users, or work process numbers. The result can thus be falsified by the side effects of other transactions. Therefore, you should use the filtering options. Tracing processes are time-consuming and may falsify the result.

TRANSACTION	CPU_INT	WORK_PROCESS_NO	PROC. TYPE	STA	CLIENT	DBI	USER	FHB_NO	TRANSID	DATE
									48C000F18C801248F180000000C0000	24.09.2007
FUNCTION	OBJ. NAME	OP.	RECS.	PG.	STATEMENT					
14	PRODIR	REOPEN		0	SELECT WHERE "NAME" = 'ZM_BOOK_EX2' AND "STATE" = 'A'					
837	PRODIR	FETCH	1	0						
0	DWINGACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_NAME" = 'ZM_BOOK_EX2'					
213	DWINGACTIV	FETCH	0	1403						
0	DWINGACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_NAME" = 'ZM_BOOK_EX2'					
145	DWINGACTIV	FETCH	0	1403						
5	TRDIR	REOPEN		0	SELECT WHERE "NAME" = 'ZM_BOOK_EX2'					
251	TRDIR	FETCH	1	0						
560	SBOOK	PREPARE		0	SELECT WHERE "MAND" = 'AB' AND "PASSNAME" = 'A1'					
0	SBOOK	OPEN		0	SELECT WHERE "MAND" = 'S01' AND "PASSNAME" = 'Chantal Ryan'					
307.010	SBOOK	FETCH	19	1403						
0	TRDIR	REOPEN		0	SELECT WHERE "NAME" = 'ZM_BOOK_EX2'					
353	TRDIR	FETCH	1	0						

Figure 8.33 SQL Access with the Where Clause

The large number of logged operations can be reduced to a more manageable amount by defining suitable restrictions during the preselection. It is useful to specify the SAP user (FHB-00 in the example), because the SAP application server logs the operations of the database layer.

You can also view the execution plan for individual SQL statements from within the SQL trace (Explain – F9). In addition, you can change and test SQL statements on a trial basis (Ctrl-F6).

We have now briefly discussed some of the main approaches for detecting and analyzing problematic SQL queries. Let's take a closer look at the solution to these problems.

8.3.5 Prevention: The Silver Bullet

The best way to avoid problematic SQL requests is prevention. The SAP system itself offers numerous good and bad examples via the **Tips and Tricks** function of the runtime analysis (Transaction SE30) or through the ABAP Workbench (Transaction SE80 or **Environment • Examples • Performance Examples**). Based on small code examples, you can become familiar with the alternative programming methods and can start measuring the runtime right

away. The examples illustrate typical programming errors including their effects and troubleshooting options (see Figure 8.34). As a prerequisite, you need the flight data model in each case.

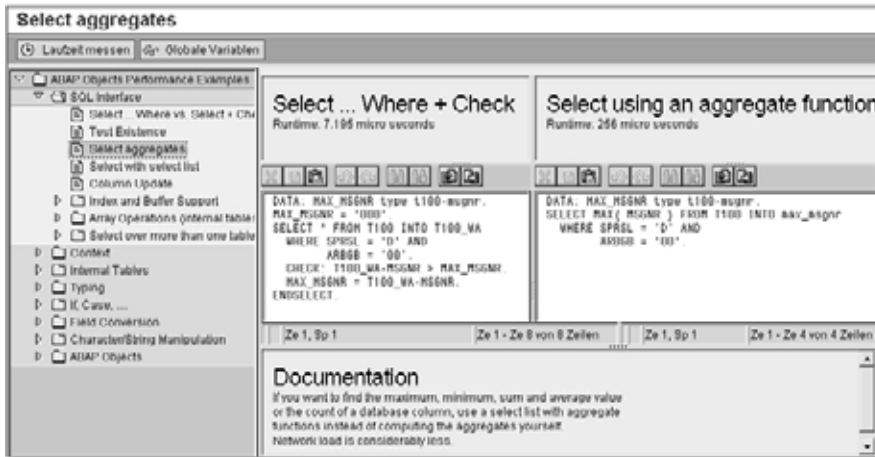


Figure 8.34 Performance Examples with Runtime Measurement

The examples also illustrate the *five golden rules of high-performing SQL programming* (see Schneider, Thomas: *SAP Performance Optimization*, SAP PRESS, 2005), which we will describe briefly here:

1. The number of data records to be transferred between the DBMS and SAP application server must be kept as small as possible. The impact of non-specific SQL requests that transport large amounts of data has been discussed several times in this section. In addition, multiple reads of identical data by a program can cause large data volumes. You can detect such behavior in the SQL Trace using the **Display Identical Selects** function (Ctrl-Shift-F8).
2. The transported data volume must be kept as small as possible. In addition to Rule 1, you must ensure that no complete data records are transferred, that is, you should avoid using `select *`. Furthermore, for typical calculations, such as calculating averages or totals, you should use the aggregate functions of the Oracle system (see the example in Figure 8.34).
3. The number of transfers between the Oracle database and the application server should also be kept small. Consequently, it is better to use a few SQL requests with large⁵ return quantities than many SQL requests with

⁵ Of course, taking into account Rules 1 and 2.

very small return quantities. On one hand, this is because processing SQL requests involves a certain overhead. On the other hand, data packages that contain small amounts of data and have a constant size on the network unnecessarily consume resources. With many small selects, the transferred gross quantity of data can be a multiple of the net quantity of data. If you succeed in determining all necessary data with a single SQL request, waste occurs only in the last package, which is probably not completely filled with data.

For this reason, you should avoid using SQL requests with nested loops. In current SAP versions, join operators are available for this purpose. In older SAP versions (older than 4.0), joins could be emulated by defining views. The alternative and intuitive way of formulating joins as nested loops can still be found in older developments and offers some potential for optimization.

4. You can keep the overhead for processing the request small by using a `where` clause that corresponds to the existing indexes. For this purpose, the `where` condition should be simple, that is, it should consist of `AND` links as far as possible. `AND` constrains the search area, whereas `OR` extends it. When formulating a request, it is often possible to transfer `OR` conditions into `AND` conditions.⁶ We'll describe the use of the correct index in this context in greater detail later in this chapter.
5. It can be useful to shift the load more toward the application server. The architecture of the SAP system can best be scaled at the level of the application server. It is possible to use a larger number of instances to increase the computing performance at that level. However, most SAP installations use only a single integrating database server (see Chapter 2, *SAP Fundamentals*). For operations such as sorting or grouping, which can be performed equally at the database server and application server levels, a shift toward the application server is advisable. Here, you can also use particularly efficient algorithms internally, such as a sorting algorithm. Of course, you need to take Rules 1 to 3 into account in this context, that is, this procedure only makes sense if the entire data quantity considered is used in the application.

Whereas Rules 1 through 4 are universally valid to a large extent, Rule 5 sometimes doesn't make sense or may even be counter-productive in your

⁶ Procedures for the conversion into disjunctive normal form and conjunctive normal form (by negation) are described in the algorithm literature, for example the application of De Morgan or the Quine-McCluskey procedure.

specific installation. If the application level and database level are located on the same server computer, this would cause a load on the same CPUs, irrespective of the execution level. If the database server computer is comparably overdimensioned, an execution in the database layer would certainly have some advantages. In this context, it is essential to understand the underlying mechanisms to obtain a working solution.

8.3.6 Indexes for Faster Access

Indexes support the quick access to specific data in tables. Take this book as an example: Without a doubt, you can find the key word *Runtime analysis* by reading all pages of the book in their entirety. However, this kind of search is pretty time-consuming. Instead, you may want to use the index at the end of the book where – thanks to the alphabetic sorting – you will only need to search one or two pages more closely to find the reference to the correct page. Ultimately, you will only have to read 3 to 5 pages instead of approximately 800 pages (in the worst case).

For tables, you can create several indexes, which connect single attributes or combinations of attributes. First, there is always the primary index that contains the key attributes. It can be mapped by means of a sorted storage of the data records. Other indexes are referred to as secondary indexes. These secondary indexes store the combinations of attribute values with references to the associated data records in additional memory pages. The organization can occur, for example, as a B-tree, via hash procedures, or as a bitmap index.

An important aspect in this context is that an index has a highly efficient selection function, which means it can help you significantly reduce the number of accesses to the data record blocks. Furthermore, the index should be necessary, that is, the attribute combination should be frequently used by application programs in the `where` clause. Moreover, it is useful if an index consumes significantly fewer memory pages than the data, that is, it is composed of only a few attributes.

In addition to the memory consumption, the maintenance of indexes costs time as well. Insert, change, and delete operations require the adjustment of associated indexes. If many indexes exist, this can considerably slow down the system operation. For this reason, you should act with special care when creating indexes. The creation of unnecessary indexes should be avoided.

8.3.6.1 Using Indexes

In the following text, we will demonstrate the effectiveness of indexes and optimization options on the basis of some examples. Figure 8.27 showed the execution plan for a selection by passenger names. In this context, the primary index is used, it doesn't provide any advantage. Actually, 277 seems to be a pretty high value for the costs.⁷ However, the passenger name can be replaced with the customer ID, because the sample data contains a customer ID for every passenger name and vice versa. If you modify the request by performing an additional selection via the CUSTOMID attribute, the Oracle Optimizer leverages the secondary index via the CUSTOMID attribute (see Figure 8.28). This is shown in the plan in Figure 8.35, which now displays an estimated cost of 3.

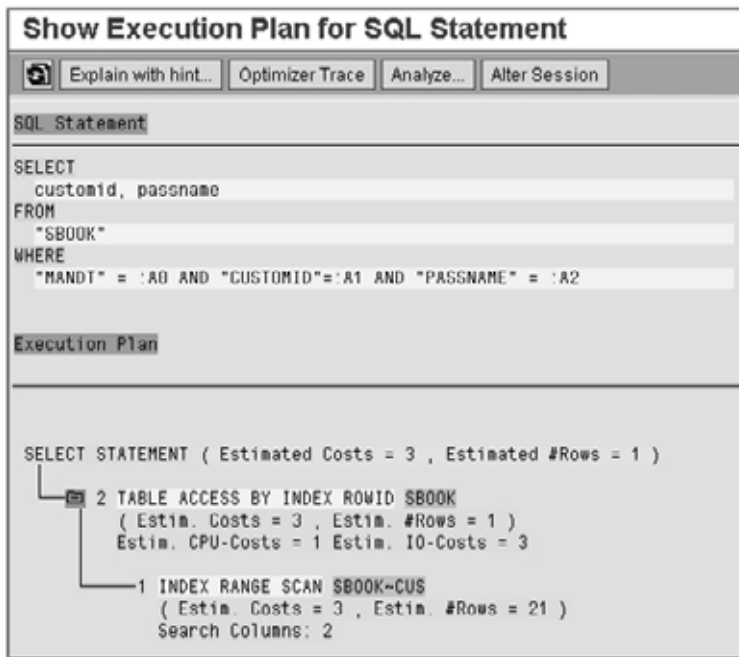


Figure 8.35 Plan with Additional CUSTOMID

Note that by taking into account existing indexes, you can often accelerate the program execution considerably. In this context, you should particularly consider attributes that do not change from the perspective of a specific

⁷ The costs (Estim. Costs, Estim. Rows) are relative values, which support the comparison of alternative plans. See also SAP Note 766349.

application and are therefore often ignored when a request is formulated. Important examples in this context include the `MANDT` and `BUKRS` attributes. Sometimes these attributes must be added to the `where` clause to make the optimizer use a specific index.⁸

8.3.6.2 Creating Indexes

It may become necessary to create an additional index. However, note that this is a substantial intervention in the system. You should only create indexes for tables of the standard SAP system after consulting the SAP Notes and by taking into account the above considerations. Also, note that creating an index takes some time, and the affected table may be locked for that time. For this reason, you should create the index at times of low usage. Because indexes consume additional resources, you should check the effectiveness of the index. In general, however, the creation or deletion of secondary indexes does not affect the functionality of the system, if they are not unique indexes.

The following example shows how an index is created and becomes effective. As changes to the structure of Table `SBOOK` are not allowed without an access key, we'll first create a copy of that table. To do that, you must start Transaction `SE11` and click on the **Copy** icon. This creates an empty table, `ZBOOK`, which has the same structure as `SBOOK` including the secondary indexes.

The following small ABAP program allows you to copy all data records from `SBOOK` to `ZBOOK`:

```
REPORT zh_sbook_copy.
DATA booking LIKE sbook.
SELECT * FROM sbook INTO booking.
    insert into zsbook values booking.
ENDSELECT.
```

For Table `ZBOOK`, the runtime analysis shows the values measured in Figure 8.36 for a selection by "Cindy Lindworm." No index is used here, because no suitable index exists for that passenger name. Compared to a selection using Table `SBOOK`, the values shown here are slightly different. One possible reason might be the different physical characteristics of the two tables.

⁸ See Schneider, T.: SAP Performance Optimization. SAP PRESS 2005.

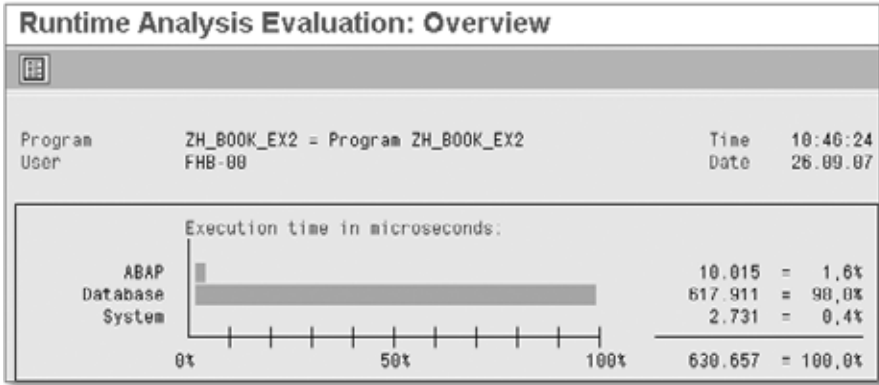


Figure 8.36 Runtime Analysis Without Index

As shown in Figure 8.37, you can use Transaction SE11 to create an index, PNI, for the PASSNAME attribute.

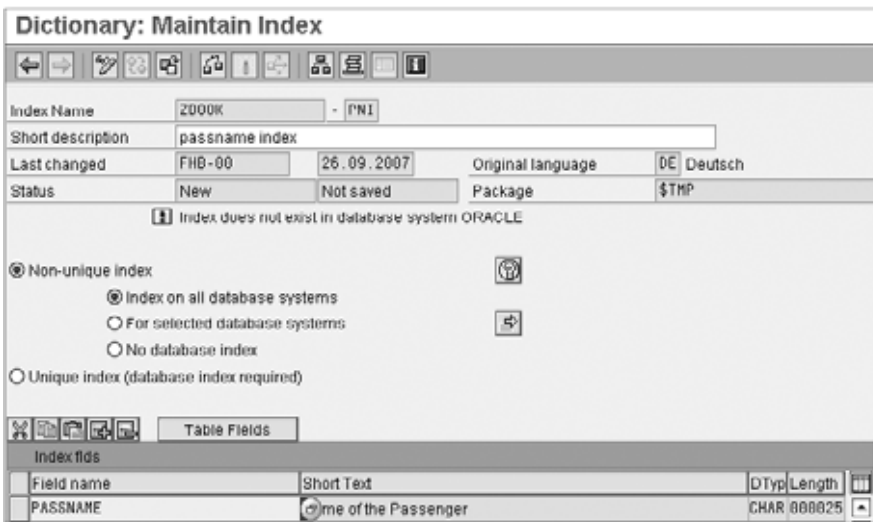


Figure 8.37 Creating the PNI Index for PASSNAME

When measuring the unchanged program, the new results are significantly better, as shown in Figure 8.38. The processing in the application server requires a similar amount of time. The number of packages transported between the database and application server is the same. However, in the example, processing in Oracle is 250 times faster due to the use of the index.

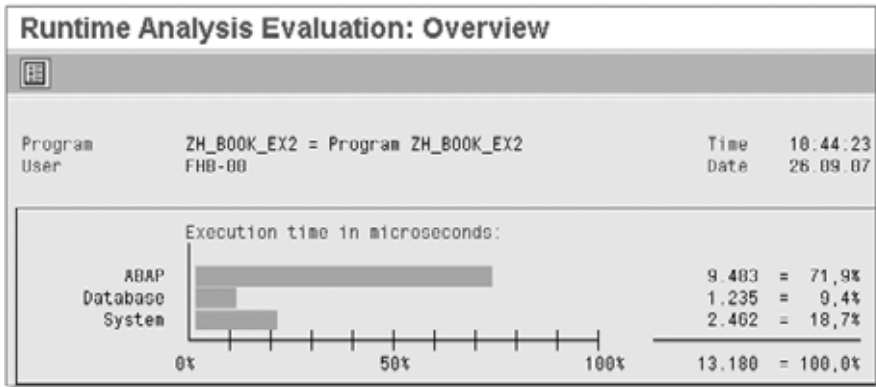


Figure 8.38 Runtime Analysis with Index

The display of the selected plan (see Figure 8.39) shows the use of index PZBOOK-PNI via the PASSNAME attribute, as expected.

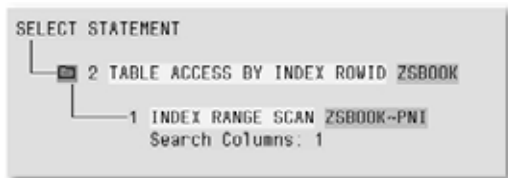


Figure 8.39 Execution Plan with PASSNAME Index

Note that this is an ideal-world example. It makes more sense to perform a selection not via the passenger name, but via the equivalent customer ID. It is very likely that other mechanisms of SAP or Oracle become effective in similar cases to achieve very short response times, even if the formulation of the request is not made in the best possible way. The powerful statistical optimizer from Oracle implements numerous methods and heuristics, which frequently (but not always) enable an ideal plan creation. In addition, the optimizer is enhanced continuously.

8.4 Summary

Performance is a far-ranging and complex topic. Hardware – operating system – Oracle – SAP system: Nobody knows everything. The intention of this chapter was to provide an overall picture to enable you to handle per-

formance problems. Whenever possible, you should try to consult an expert on the topic you need help with.

Avoid making snap decisions. Performance is important, but isn't everything. Data security is always the top priority. If the system halts, immediate action is required, but this situation should generally happen very rarely. Try to adhere to the small or big optimization cycle: Problem analysis – troubleshooting – verification.

Performance-critical SQL statements may significantly affect the overall system and are often caused by the primarily function-oriented programming style of the application developers. Prevention helps here as well as training the developers with regard to a high-performance programming style. You can often improve problematic SQL statements in collaboration with the development team. In this chapter, we demonstrated ways to recognize and precisely analyze this type of statement. Another option, which, however, should be the responsibility of an experienced Oracle administrator, is to create additional secondary indexes. If you identify a problem with an original SAP program, you're probably not the first to encounter that problem. In this case, the SAP Service Marketplace or the SAP Notes often contain essential information for problem handling, either by means of a correction or a workaround.

Index

28-day backup cycle 651
64 bit 252
64 bits
 database 253
 hardware 253
 operating system 253
 software 252

A

ABAP 33
 buffer 40, 43
 Dictionary 59, 477
 dump 530
 framework 33, 57
 processor 43
 report 34
 Support Package 365
 Workbench 278, 313
ABC analysis 66
abort 133
Access
 control 68
 costs 455
 path 80
access
 algorithm 128
ACID properties 47, 69, 135
action log 336
Add-On Installation Tool 365, 374
Admin option 143
Administrator Workbench 743
After-image 550, 566
Aggregate function 78
Aggregation 728
Ajax 119
ALERT log 136, 570
Algebraic optimization 128
all_ 101
ALL_ROWS 133
alter table 73, 75
Analytic workspace 738, 741
Analyze command 132

Application data 282
Application layer 35
Application server 35, 37
 memory structure 39
 process 38
Application Service Provider 245
ARCH 90
Archive directory 134, 135
ArchiveLog mode 553
 activate 555
 online data backup 564
Archiver directory 93, 555, 601, 665
 change 559
Archiver process 553
Archiver stuck 508, 556, 592, 600
 avoidance 558
Archiving 90, 93, 391
ARIS 745
AS SYSDBA 85
ASCII terminal 29
Assignment of privileges 134
Asynchronous I/O 447
Asynchronous update 48
Attribute 71, 747
Authentication 121
Authorization model 141
autoallocate 106
autoextend 103
Auto-join 78
Automatic Segment Space Management (ASSM) 106, 440
Automatic Shared Memory Management (ASMM) 419
Automatic Storage Management (ASM) 82, 97, 736
Automatic Tuning Optimizer (ATO) 126
Availability 37, 264

B

Background job 355, 538
BACKINT 587, 600
 disaster recovery 616

- Oracle Recovery Manager 624
 - split mirror database* 664
 - volume management* 619
- Backup 112, 133, 257, 354
 - cumulative incremental* 139
 - cycle* 618, 650
 - development and test system* 655
 - device* 617
 - differential incremental* 139
 - domain controller* 299
 - incremental* 138, 585, 653
 - level* 138
 - library* 624, 627
 - log* 510
 - mechanism* 70
 - offline* 138, 391, 562
 - online* 138, 564, 647
 - partial tablespace* 655
 - scheduler* 140
 - size* 510
 - speed* 652
 - split-mirror* 258
 - strategy* 257, 649
 - technology* 138
 - time* 70, 510
 - two-phase* 657
- Backup library
 - external* 629
- Backup, incremental 625
- BAPI (Business Application Programming Interface) 60
- Basic operations 67
- Basis extent 103
- Batch input sessions 46
- Batch mode 32
- Batch work process 39, 45
- bdump 136
- Before-image 566
- BEx Analyzer 760
- BEx Map 763
- BI 723
- BIGFILE 106
- Bitmap 105
 - Index* 488
 - index* 737
 - join index* 737
 - tablespace* 105
- Blade server 37
- Blade technology 37
- Block boundary 118
- Block split 567
- Bootstrap 682, 690
- BR*Tools 54, 89, 140, 216, 368, 543, 579
 - attended versus unattended mode* 221
 - BRCONNECT* 227
 - BRGUI* 226
 - BRSPACE* 230, 237
 - BRTOOLS (frontend)* 224
 - configuration file* 220
 - general options* 220
 - information* 219
 - log files* 222
 - operating system user* 219
 - Oracle Recovery Manager* 623, 627
 - SAPDBA* 217
 - standby database* 659
 - temporary tablespace* 614
 - Windows environment* 616
- BRARCHIVE 580, 592
 - backup device* 599
 - backup mode* 593
 - compression* 598
 - continual backup* 595
 - database object* 592
 - detailed log file* 598
 - Oracle Recovery Manager* 623
 - split mirror database* 663
 - standby database* 600, 659
 - summary log file* 595
 - tape drive* 594
 - unattended* 599
 - verification* 600
 - volume* 619
- BRBACKUP 580, 582
 - backup device* 587
 - backup mode* 585
 - backup type* 586
 - compression* 588
 - database object* 584
 - Oracle Recovery Manager* 623
 - parallelization* 588
 - split mirror database* 663
 - split scenario* 664
 - standby database* 659
 - unattended* 590

- user* 590
- verification* 591
- volume* 619
- volume management* 589
- BRCONNECT 544
- BRRECOVER 580, 606
 - backup device* 608
 - detailed log file* 611
 - disaster recovery* 614
 - parallelization* 608
 - post-processing* 612
 - recovery scenario* 609
 - structural change* 610
 - summary log file* 611
 - unattended* 609
 - user* 610
- BRRESTORE 580, 602
 - backup device* 602
 - compression* 604
 - data file* 603
 - detailed log file* 605
 - parallelization* 604
 - redo log file* 603
 - restore mode* 602
 - summary log file* 606
 - unattended* 604
 - user* 604
 - verification* 605
 - volume* 604
- BRTOOLS 555, 580
- BSP (Business Server Page) 60
- B-tree index 768
- Buffer 536
 - administration* 472
 - management* 81
 - quality* 417
 - synchronization* 468
- Buffer busy wait 440
- Bug fix 378
- Business Add-In 279
- Business Application Programming Inter-
face (BAPI) 60
- Business Content 744
- Business Explorer 743, 760
- Business Intelligence 30, 723
- Business process 28
- Business Server Pages (BSP) 60, 261
- Business transaction 32
- Busy wait time 417, 421, 430

C

- Cached I/O 447
- Calendar 465
- Cardinality, high 752
- Catalog 67
- Central instance 49
- Change and Transport System (CTS) 288
- Change Management Service (CMS) 706
- Change request 293
 - local* 308
 - release* 320
- Characteristic 750
- Check table 68
- Checkpoint 93, 113, 550
- Checkpoint not complete 438
- Checkpointner 90, 93, 551
- Classical star configuration 746
- Classification 733
- Client 35, 280, 357
 - administration* 291
 - background job* 363
 - export* 360
 - import* 360
 - initial storage space* 360
 - maintenance* 285
 - new creation* 362
 - number* 280
 - role* 285
 - transport* 358
- Client copy 35, 355, 358
 - copy log* 362
 - copy profile* 358
 - local* 359
 - remote* 359
 - test run* 362
 - very large production clients* 363
- Client library 83
- Client process 121
- Client/server system 29
- Climate control 245, 246, 247
- Clustering 733
- Codd's rules 67
- Collector job 538
- COMMIT 95
 - fast* 93
- Communication hardware 415
- Communication SAP – Oracle 163
- Communication structure 754

Company code 35
 Compliance 29
 Component
 Build Service 706
 development 708
 partitioning 737
 software 708
 Support Package 365
 Component model 707
 Compression 618
 BRARCHIVE 598
 BRBACKUP 588
 compute_statistics 130
 Concurrent I/O 447
 Config Tool 673
 Configuration file 111
 Configuration with one machine 36
 CONNECT 143
 Connecting SAP and Oracle 158
 Connection pooling 124
 Consistency
 archive log 514
 spooler 541
 TemSe 541
 Consolidation route 303
 Consolidation system 303
 Constraint 82
 Control file 109, 111, 136, 321, 551,
 630, 631
 Copy profile 358
 Correction 316
 Cost-Based Optimizer (CBO) 455, 516
 Cost-based selection 128
 count 77
 CPU 497, 536
 time 417, 421, 430
 utilization 412, 413
 create any table 142
 create index 129
 create session 142
 create table 73, 75
 Critical patch update 382
 CUA 465
 Cube 728, 740, 743
 Cursor 59
 Customer namespace 57, 314
 Customizing 34, 279
 data 281, 469
 request 293
 tables 472

D

Data
 backup 133
 backup method 561
 buffer 117, 418, 421, 423
 cleaning up 727
 cleansing 727, 741
 cube 728
 definition language 74
 Dictionary 80, 117, 127
 element 59
 export 561
 file 134, 321, 547
 flow 758
 integration 733
 manipulation language 74
 mart 730, 734, 738
 mining 724, 732
 model 71, 745
 modeling 71
 package 475
 protection 140
 pump 725, 735
 quality 741
 security 54, 66, 70, 133, 135
 storage 36
 target 742
 type 59, 74
 warehouse 724
 Warehousing Workbench 743
 Data basis, initial 34
 Data buffer 420
 Data class (database object) 209, 469
 Data Definition Language (DDL) 74
 Data Manipulation Language (DML) 74
 Database
 buffer 549
 buffer cache 94
 check 227
 consistency 511
 Creation Assistant (DBCA) 89
 export 513
 growth 618
 interface 43, 467
 layer 36
 lock 452
 management software 17
 object 140, 547, 584, 592

- point-in-time recovery* 577
- reorganization* 81
- reset* 576
- standby* 514
- statistics* 517
- structures* 99
- user* 160, 190, 200
- Writer* 90, 93
- writer* 549
- DataSource 693, 753
- db file parallel read 433
- db file scattered read 433
- db file sequential read 433
- DB LUW 47
- DB Reconnect 163
- DB reconnect 197
- DB release 83
- DB_BLOCK_BUFFERS 419
- DB_CACHE_SIZE 418
- DBA 143
 - planning calendar* 456, 512, 514
 - views* 101
- dba_data_files 100
- dba_lmt_free_space 107
- DBMS_STATS 132, 778
- dbms_xplan 129
- DBSNMP 145
- DBSTATC 457
- DBVerify 512
- DBWR 90, 93, 117
- DD cache 420
- DDIC user 46
- DDL 74
- Dedicated server 119
 - process* 92, 121
- Default
 - password (Oracle)* 191
- Default profile 51, 153
- Default tablespace 195
- delete_from 76
- Delivery client 282
- Delivery route 303
- Delta
 - procedure* 759
 - queue* 759
 - update* 739
 - upload* 758
- Demilitarized zone (DMZ) 272
- Dependency analysis 732
- Deployment 57, 675, 709
- Description 732
- Design process 73
- Design Time Repository (DTR) 706
- Developer key 57, 313
- Development 279
 - class* 58
 - environment* 34
 - landscape* 134
 - object* 708
 - partnership* 29
 - system* 284, 655
 - task* 706
- Deviation analysis 732
- Dialog mode 29
- Dialog work process 38, 42, 549
- Dicing 732
- Dictionary cache 95, 424, 478
- Dictionary statistics 516
- Dictionary-Managed Tablespace (DTMS) 100, 102
- Dimension 728, 744, 750
- Dimension ID 747
- Direct hand-off process 121
- Direct I/O 447
- direct path operation 421, 422, 435
- direct path read 422, 434
- direct path read temp 434
- direct path write 422, 434
- direct path write temp 434
- Dirty 93, 118
- Disaster recovery 607, 614
 - logging* 616
 - prerequisite* 615
- Disk layout 256
- DISK_ASYNC_IO 448
- Disp+Work program 38
- Dispatcher 38, 40, 91, 523
 - queue* 40, 526
- Distribution strategy 41
- DML enqueue (TM) 451
- DMTS 100, 102, 771
- domain link 329
- Downtime 388, 400
- Downtime minimized 390
- dpmon 526
- Drill-across 732
- Drill-down 732, 740
- Driver class 687

drop any table 142
 drop table 73, 75
 Dummy file 558, 666
 Duplicates 728
 Dynamization 82
 Dynpro 33
 Dynpro processor 42

E

EarlyWatch 284
 E-Business 30
 E-Commerce 30, 138
 Effects forecast 733
 Embedding of SQL 34
 Enqueue 443, 522
 lock 523
 process 39, 48
 replication server 523
 service 675
 ensmon 522
 Enterprise Core Component (ECC) 31
 Enterprise Extension Set (EES) 31
 Enterprise Java Beans 60
 Entity 71
 Entity Relationship (ER)
 diagram 72
 model 71
 schema 72
 Entity relationship (ER)
 model 745
 Environment variable 179
 db_s_ora_schema 179
 DIR_LIBRARY 186
 ORACLE_BASE 179
 ORACLE_HOME 179
 ORACLE_SID 179
 SAPDATA_HOME 179, 206
 SAPSYSTEMNAME 179
 Ergonomic presentation 29
 Error class 546
 Error scenario 574, 629
 control file 630, 631
 offline redo log file 646
 online backup 647
 online redo log file 639, 645
 redo log group 641
 rollback tablespace 636

system tablespace 636
 tablespace 574
 temporary tablespace 637
 ETL process 726, 734, 742, 752
 Exception reporting 763
 Exclusive lock 453
 Exclusive lockwaits 451
 Executables 110, 111
 Execution plan 116, 129, 473, 480, 485
 Exp/Imp SHM 465
 Expert mode 349
 Expert system 126
 explain_plan 129
 Export 735
 Export/Import 465
 Extended memory 460, 462
 Extended SAP star configuration 747
 dimension ID 747
 surrogate ID 748
 Extension semantics 34
 Extent 102
 Extraction 726

F

Factors
 external 546
 Failover 266
 Fast refresh mechanism 739
 Fat client 35
 Federated database system 727
 Fiber channel (FC) 257
 Field descriptions (FTAB) 465
 File
 flat 759
 File cache 97
 File system
 caching mechanism 97
 local 96
 File system cache 414
 File, flat 753
 FILESYSTEMIO_OPTIONS 448
 Fire protection 269
 Firmware 384
 FIRST_ROWS 133
 Five-layer architecture 79
 Flash backup technology 140
 Flashback interval 140

Flashback recovery log 140
 Foreign key 73
 Foreign system 753
 Forwarding
 direct 755
 flexible 756
 rule 754
 Fragmentation 107
 external 107
 internal 108
 free buffer waits 441
 Free list 104
 Frequent itemset 741
 Frontend 401
 Full buffering 467
 Full restore and complete recovery 578
 Full table scan 128
 Full upload 758

G

Galaxy configuration 731
 Gateway 528
 monitor 529
 process 39, 49
 Generic Request and Message Genera-
 tor 696
 Geofeature 764
 Granularity 752
 Grid computing 82
 group by 78
 grouping 78, 487, 740, 751
 gwmon 529

H

Hardware error 546
 Hash
 grouping 129
 join 128, 737
 partitioning 737
 HASH_AREA_SIZE 425
 having 78
 Heap memory 460, 463
 Heterogeneity 726
 Heuristic 128
 Hierarchy 747, 750

High availability 49, 245, 249, 266
 license 353
 High-availability system 345
 Hint 127, 133
 Hit ratio 417
 HP-UX 385
 HTML 60
 Humidity 248

I

I/O
 analysis 444
 configuration 777
 load 413, 415, 446, 449
 modes 447
 performance 97, 117, 446
 Idle wait event 429
 IDoc (Intermediate Document) 63
 Implementation Guide 278, 309
 Import 735
 Monitor 322
 overview 321
 Puffer 326
 queue 321, 326
 Index 80, 127, 488
 range scan 131
 rebuild 234
 Individual software 27, 278
 InfoCube 751
 InfoObject 750
 InfoPackage 755
 InfoProvider 742
 InfoSet 761
 InfoSource 754
 init.ora 96, 111
 init.sap 582
 Initial record 465
 insert_into 76
 Installation
 phases 341, 344
 planning 110
 tools 341
 Instance 36, 49
 name 50
 number 345
 profile 51, 153
 Instance recovery 551

Instant Client 119, 124, 688
 Integration 28, 67, 725
 Integration Engine 725
 Integration platform 36
 Integration system 303
 Integrity check 68
 Integrity rules 74
 Interactive Query Language (IQL) 74
 Interface 681
 Interim patch 378, 386
 critical patch update 382
 installation 382
 merge patch 384
 MOPatch 384
 OPatch 381
 Intermediate Document (IDoc) 63
 Internet Communication Manager (ICM)
 31, 670
 Internet Graphics Service 367
 Internet Transaction Server (ITS) 61, 259
 integrated 260
 IPC 120, 123
 IQL 74
 ISO/OSI-seven-layer model 120
 isqlplus 86, 87

J

J2EE
 cluster 678
 engine 673
 server 672
 specification 672
 Java 60
 cluster 670
 Connector 63, 670
 pool 418
 schema 686
 stack 670
 Startup and Control Framework 682,
 716
 Support Package 366
 Support Package Manager 365, 709
 Upgrade program 709
 upgrade program 714
 JAVA_POOL_SIZE 418
 JCMon 684, 700
 jcontrol 682
 JDBC driver 686

jlaunch 682
 JNDI 693
 Jobs
 log 539
 Join 74, 77, 480
 JPREPARE 393

K

Keep pool 423
 Kernel 534
 Kernel parameter 96
 Key 73
 Key figure 728, 750, 762

L

Large database
 backup 652
 Large objects 106
 Large pool 418
 LARGE_POOL_SIZE 418
 Latch 443
 latch free 442
 LCK 90
 LDAP 140
 Least Recently Used (LRU) 93
 Least recently used block 550
 Legal regulations 29, 133
 Level 0 backup 624
 Level 1 backup 625
 lgtst 521
 LGWR 90, 93, 117, 118, 140
 Library 681
 Library cache 423
 Lifecycle 364
 like 77
 Line item 752
 Linux 385
 List partitioniung 737
 Listener 121
 listener.ora 122
 LISTSCHEMA 760
 Load balancing 49
 Load distribution 124
 Loading 728
 LOB 106
 Local client copy 359

- Locally Managed Tablespace (LMTS)
 - 100, 105, 771, 776
 - Location of problems 407
 - Lock conflict 515
 - Lock management 48, 81
 - Lock table 48
 - Log 165
 - Listener log* 165
 - listener log* 166
 - Listener trace* 165
 - Oracle alert log* 165
 - SAP work process trace* 165
 - session traces* 165
 - start and stop logs – Oracle* 165
 - start and stop logs – SAP* 165
 - log buffer space 427, 436
 - Log file 69, 70
 - log file parallel write 436
 - log file switch completion 437
 - log file sync 436
 - Log Sequence Number 552, 560
 - Log sequence number 114
 - Log switch 113, 114
 - Log writer 93, 135, 550
 - LOG_DIRECTORY_LISTENER 122
 - LOG_FILE_LISTENER 122
 - Logical optimization 128
 - Logical read 417
 - Logical Unit of Work (LUW) 41, 47, 69
 - Logical Volume Manager (LVM) 97, 446
 - Login process 192
 - Login shell 179
 - Logon group 521
 - Log-Writer 90
 - Long-term storage 651
 - LRU algorithm 93
 - LRU procedure 95
 - lsnrctl 125
- M**
-
- Main memory 134
 - Mainframe 29
 - Maintenance Optimizer 366
 - Maintenance release 84
 - Maintenance window 264
 - Mass deletion 108
 - Master controller process 735
 - Master data 469, 747, 750, 755
 - Match 741
 - Matching, fuzzy 77
 - Materialized View 738
 - max 78
 - MaxDB 70
 - MCP 735
 - Media error 70, 133, 135
 - Memory 414, 497
 - areas* 459, 461
 - configuration* 459
 - gap* 94
 - monitor* 461
 - physical* 535
 - swap* 535
 - Merge 739, 741
 - Merge patch 378
 - Message server 39, 518, 675, 677
 - process* 49
 - Metadata 735
 - Metadata Repository 745
 - Metric 127
 - Microsoft Management Console 153
 - Mirroring 97, 736
 - Mirror-Log 110
 - MMAN 91
 - MMON 91
 - Model concept 740
 - Model View Controller concept (MVC)
 - 61, 672
 - Model-View-Controller 261
 - Modification 279
 - Modification adjustment 400
 - MOLAP 731, 738
 - Monitoring 495, 695
 - category* 498
 - external* 501
 - hardware* 496
 - load caused by* 500
 - network* 498
 - performance* 533
 - Service* 696
 - software* 498
 - table buffering* 469
 - MOPatch 384
 - msmon 519
 - msprot 518
 - MultiCube 761
 - Multidimensional data model 728
 - Multidimensional Entity Relationship
 - model 745

Multidimensional OLAP 731
 Multidimensionality 743
 Multi-pass operation 426
 Multiple Components in One Database
 (MCOB) 202, 347, 577, 603, 610
 Multiple storage 67
 Multitable insert 740
 Multiuser mode 65

N

Named pipes 120
 Native SQL 43
 Net Manager 124
 netca 124
 netmgr 124
 NetWeaver
 Developer Studio 669, 705
 Development Infrastructure 669, 673,
 705
 Network 413
 Network performance 164
 Next extent 103
 NoArchiveLog mode 554
 offline data backup 563
 Non-Unicode system 301, 337
 Normalization 73
 Note Assistant 375
 Note correction 364
 NTBACKUP 616
 NULL value 76

O

Object assignment (database object) 209
 Object directory entry 316
 copy 316
 original 316
 Object Navigator 59
 Object parameter (database object) 209
 Object privilege 141
 Object-oriented concept 33, 82
 Object-relational database 82
 OCI 120
 OCSS process 97
 ODM connector 779
 ODS object 751

Offline
 backup 138
 data backup 562
 redo logs 444
 OLAP (Online Analytical Processing) 723
 cube 728
 engine 734
 processor 725
 OLTP (Online Transaction Processing)
 89, 723
 Online
 backup 138
 data backup 564, 647
 transaction processing 723
 OPatch 381
 open resetlogs 612
 Open SQL 43, 57, 71
 Operating modes 553
 Operating system 173, 549
 kernel 160, 174
 maintenance 384
 monitor 410, 411, 445
 patch 174
 user 177, 194, 199
 Operating system group 178, 193
 dba 178
 oper 178
 sapsys 178
 Operating system-specific environment
 variable 183, 187
 Operation 67
 Operation error 133
 Operation mode switching 45
 Operation type 355
 OPI 120
 OPSS\$ process 194, 195, 196
 Optimal execution 425
 Optimal work area size 425
 Optimization cycle 406
 Optimization, internal 128
 Optimizer 92, 116, 126, 473
 Optimizer hint 132
 Optimizer statistics 456
 OPTIMIZER_MODE 455
 Oracle
 availability 503
 block 567
 Business Intelligence Discoverer 741
 Call Interface (OCI) 120

- CBO statistics* 229
- character set* 182
- client* 180
- Client 9i* 181
- Cluster Synchronization Service (OCSS)* 97
- control file* 255
- data buffer* 464
- data file* 255
- Data Migration Assistant* 402
- Data Miner* 742, 779
- Databank Optimizer* 455
- database monitor* 419
- Database Upgrade Assistant* 402
- E-Business Suite* 17
- Enqueues* 451
- Enterprise Manager* 87
- installation* 548
- Instant Client (10g)* 183
- job* 543
- latch* 443
- Listener* 156, 160, 164, 200, 503, 689
- lock mode* 452
- lock monitor* 452
- Managed Files (OMF)* 97
- mirror log* 255
- Optimizer* 489, 739
- parameter* 160, 236, 416
- Program Interface (OPI)* 120
- Real Application Cluster (RAC)* 37, 125
- redo log* 255
- schema ID* 205
- server process* 92
- Session Monitor* 161, 432
- shadow process* 159
- statistics* 778
- storage areas* 771
- stream* 734
- system ID* 177
- version number* 84
- wait events* 427
- Warehouse Builder* 741
- Oracle Net 89
 - architecture* 120
 - Configuration Assistant* 89, 124
 - Manager* 89
 - Services* 184
- Oracle parameter
 - CREATE_BITMAP_AREA_SIZE* 775
 - DB_CACHE_ADVICE* 423
 - DB_FILE_MULTIBLOCK_READ_COUNT* 772
 - FILESYSTEMIO_OPTIONS* 772
 - HASH_AREA_SIZE* 775
 - PARALLEL_MAX_SERVER* 772
 - PGA_AGGREGATE_TARGET* 774
 - PGA_MAX_SIZE* 774
 - SORT_AREA_SIZE* 775
 - WORKAREA_SIZE_POLICY* 774
- Oracle parameters
 - administration* 237
 - dynamic versus static* 236
- Oracle process 90, 151, 157, 549
 - ARC process* 151
 - CKPT* 151
 - DB work process* 151
 - DB Writer* 151
 - LGWR* 151
 - listener* 151
 - PMON* 151
 - RECO* 151
 - SMON* 151
- Oracle Recovery Manager 585, 622, 653
 - backup library* 627
 - BR*Tools* 627
 - external backup library* 629
 - features* 622
 - level 0 backup* 624
 - level 1 backup* 625
 - SAP backup library* 628
 - use* 624
- Oracle upgrade 402
 - tools* 402
- ORACLE_HOME 84
- ORACLE_SID 84
- OracleBI
 - Beans* 742
 - Discoverer* 741
 - Spreadsheet* 741
- Oracle-Parameter
 - PARALLEL_EXECUTION_MESSAGE_SIZE* 772
- Oracle-Upgrade
 - paths* 402
- ORADEBUG 167
- order by 78
- Orig-Log 110
- OSS Note 34
- OTR (Online Text Repository) 465

P

Package 58, 304
 \$TMP 308
 Paging 412, 415
 Paging memory 460
 PAI area (Process After Input) 42
 Parallel processes 361
 Parallel query 436
 Parameter file 96
 Parameter maintenance (Oracle) 235
 Parameterization file 547
 Partial backup 655
 Partial Restore and Complete Recovery 575
 Partitioning 737
 Password 140
 Password change (Oracle) 191
 Patchset 378, 386
 installation 379
 prerequisite 379
 PBO area (Process Before Output) 43
 pctfree 104, 441
 pctincrease 104
 pctused 104
 Performance 55, 70, 126, 475
 Performance problems 407
 administrative 407
 analysis 408
 program-based 407
 user-specific 407
 Persistent Staging Area (PSA) 755
 Personalization 279
 PGA_AGGREGATE_TARGET 425
 Physical read 417
 Pivoting 732
 Planning phase 244
 PMON 90, 121
 Point-in-time recovery 577, 610
 Post installation 352
 Power supply 248
 uninterruptable 247
 Preliminary correction 317
 PREPARE 393, 395, 396
 log file 398
 modules 397
 Prepare 116
 Prerequisites Check 346
 Presentation layer 35

Prevention 485
 Primary index 480
 Primary key 73, 75
 Privilege (Oracle) 197, 201
 Process 90, 150
 Process After Input (PAI) 42
 Process Before Output (PBO) 43
 Process overview 523
 Product Availability Matrix (PAM) 386
 Production key 748
 Profile file 50
 Program Global Area (PGA) 94, 425, 771
 Administration 777
 administration 776
 management 425
 Project IMG 310
 CTS function 310
 view 311
 Projection 128
 Projection list 77
 Protocol Support 120
 protocol.ora 122
 PSAPTEMP 422, 425, 444, 777
 PSPO 91
 Punchcard stacks 29

Q

QA approval procedure 309
 Quality assurance system 284
 Query 763
 definition 761
 Designer 761
 language 76
 processing 92
 rewrite 739
 Quick Sizer 250

R

R/2 system 49
 R3trans 153, 290, 323, 333
 RAC 125
 RAID 81, 97, 110, 133, 135, 254
 Range partitioning 737
 Raw device 97, 207, 448
 Raw I/O 447

- rdbms ipc reply 441
 - RDDIMDPD 323
 - read by other session 440
 - Read-only tablespaces 102
 - Real Application Cluster (RAC) 36, 125
 - Recipient system 303
 - RECO 90
 - Reconnect mechanism 562
 - Recording changes 291
 - client-dependent* 291
 - client-independent* 292
 - Recovery 112, 133, 135, 140, 569
 - catalog* 623
 - Manager (RMAN)* 89, 138, 140
 - training* 133
 - writer* 140
 - Recovery method 568
 - offline data backup* 572
 - online data backup* 573
 - Recovery scenario 571, 606
 - Recycling pool 423
 - redirect process 121
 - Redo buffer 418, 437
 - Redo information 114, 140
 - Redo log buffer 93, 114, 118, 550
 - Redo log file 93, 110, 112, 118, 134, 364, 438, 551, 565, 592, 650
 - backup* 557, 593
 - loss* 639, 645
 - loss, offline* 646
 - offline* 554
 - online* 551
 - split mirror database* 663
 - status* 552
 - Redo log group 113, 552, 641
 - Redo log switch 552
 - Redundancy 67, 73
 - Redundancy-free storage 36
 - Referential integrity 68
 - Relation 73
 - Relational database management system 36
 - Relational OLAP 730
 - Relationship 71
 - Release upgrade 243
 - Remote copy 359
 - RemoteCube 753, 761
 - Renaming 77
 - Reorganization 211, 233
 - online versus offline* 212
 - performance* 215
 - process* 214
 - tools* 213
 - Repair 316
 - Repair flag 316
 - Reporting 760
 - Repository data 282
 - Request ID 751
 - Request optimization 126
 - Request processing 126
 - RESOURCE 143
 - Resource minimized 390
 - Response time 537
 - Restore 112, 569
 - RFC 62
 - interface* 62
 - server group* 361
 - RKWR 91, 140
 - RMAN (Oracle Recovery Manager) 55, 89, 140, 735
 - ROLAP 730, 738
 - Role 141
 - Roll buffer 40
 - Roll memory 459
 - Roll-up 732, 740
 - Rule-Based Optimizer (RBO) 455
 - runInstaller 344
 - Runtime analysis 481, 488
 - Runtime measurement 483
- ## S
-
- SAP
 - application server* 67
 - backup library* 624, 628
 - backup object* 548
 - buffer* 459, 463, 465
 - data file names* 206
 - database library* 186, 188
 - directory structure* 174, 206
 - LUW (logical unit of work)* 47
 - memory monitor* 465, 469
 - NetWeaver* 30
 - NetWeaver Developer Studio* 60
 - NetWeaver Portal* 262

- Notes 34, 375
- original client* 283
- OS Collector 153, 173, 410
- port* 274
- process overview* 414
- Reference Implementation Guide (SAP Reference IMG)* 309
- SAP up 393, 395, 399
- SCM 29, 765
- SCM namespace 767
- Service Marketplace 34, 57
- Software Change Registration 314
- software development 56
- Solution Manager 501
- Solution Manager Key* 349
- standard 278
- Start Service 153
- system ID 177
- system log 529
- system startup 152
- tablespace 202, 232
- Transport Tool 53
- user 46
- work process 92
- work process overview 168
- SAP authorization role (Oracle) 197
- SAPCONN 198
- SAPDBA role 198
- SAP Basis 32
- SAP Easy Access Menu 33
- SAP GUI 35
 - HTML 259
 - Java 259
 - Windows 259
- SAP instance
 - memory area 459
- SAP kernel 32, 185, 188, 367, 386
 - double-stack system 369
 - Internet Graphics Service 367
 - parameters 189
 - SAPCPE 368
 - update 368
- SAP NetWeaver 669
- SAP NetWeaver Application Server 60
- SAP NetWeaver BI 30, 49
 - namespace 766
 - performance 766
- SAP NetWeaver Business Intelligence
 - Accelerator 780
 - SAP NetWeaver Exchange Infrastructure (XI) 669
- SAP patch 364
 - kernel files 367
- SAP process 150, 154
 - gateway process 150
 - Internet Communication Manager 150
 - message server 150
 - process ID 154
 - sapstart 150
 - syslog collector 150
 - syslog sender 150
 - work process 150
- SAP R/3 table buffer 39
- SAPCPE 368
- SAPDBA 544, 579
- SAPinst 341
 - GUI 342
 - log file 351
- SAPJup 393, 395
- SAPOS Collector 531
- SAProuter 273, 343
- SAPS 250, 405
- Scalability 36
- Scalable architecture 29
- Scheduler 757
- Screen 465
- SDP (Session Description Protocol) 120
- SDU (session data unit) 123
- Secondary index 116, 488
- Secondary memory 134
- Secure Store 688
- Security 199
 - database 271
 - gap 384
 - network 272
 - technical 270
- Segment 102
- Segment shrinking 216
- select from 77
- Selection 77, 128
- Selectivity 129
- Self-tuning 736
- Sequence number 135
- Server overview1 523
- Server process 90, 91, 121
- Service 681
- Service definition 123
- Service level agreement (SLA) 264

- Service Manager 680
- Service-oriented architecture 31
- SGA_TARGET 419
- Shadow instance 399
- Shadow process 44, 92
- Shadow system 390, 392
- Shape file 764
- Shared
 - cursor cache* 423
 - lock* 454
 - memory* 96
 - memory segment* 96
 - pool* 418, 424
 - server* 124
 - server operation* 91
 - server process* 92
 - SQL pool* 95
- SHARED_POOL_SIZE 418, 424
- Shopping cart analysis 741
- Short NTAB 465
- Size category (database object) 209
- Sizing 246, 250
- Slicing 732
- SMON 90
- Snapshot 90, 653
- Snowflake configuration 731
- SNPn 90
- Socket 122
- Software
 - Component Archive* 366
 - Deployment Manager* 365, 675, 690, 709
 - error* 546
 - maintenance* 52
- Solaris 385
- SORT_AREA_SIZE 425
- Sorting 78, 117, 487
- Source system 753
- Space management 98
- SPFile 96, 111
- Split mirror database 660
- Split scenario 664
- SPOF (Single point of failure) 517
- Spool process 39
- SQL
 - *NetV2* 119
 - area get ratio* 478
 - area pin ratio* 478
 - cache* 423
 - IDL* 76
 - Native* 692
 - Open* 692
 - optimization* 473
 - query* 116
 - trace* 485
 - Vendor* 692
- SQLA.Reloads/Pin 478
- SQLNET.EXPIRE_TIME 455
- sqlnet.ora 122
- sqlplus 84, 137
- SSL 120
- Stability 70
- Stack processing 29
- Staging 753
- Staging area 728
- Standard
 - job* 46
 - software* 27, 278
 - transport layer* 304
- Standard business software 27
- Standby
 - cold* 265
 - database* 600, 658
 - hot* 265
 - load balancing* 267
- Star configuration
 - classical* 746
 - extended* 747
- Star schema 731
- Start profile 51, 153
- startdb 153, 155, 166
- startsap 153, 166
- Startup problem 159
- Statistic 127
- statistic_level 132
- statistics 130
- Stop mark 327
- stopdb 166, 172
- stopsap 166
- Storage
 - area network (SAN)* 254
 - array* 254
 - gap* 79
 - structure* 90
 - system* 97
- Streams pool 418
- STREAMS_POOL_SIZE 418
- Striping 97, 736

- Structural change 610
 - Subcube 730
 - sum 78
 - Support Package
 - ABAP 365
 - Component 365
 - Java 366
 - Maintenance Optimizer 366
 - memory requirements 366
 - SAINT 365
 - Software Component Archive 366
 - SPAM 365
 - Stack 369
 - system landscape 376
 - types 365
 - Support package 29, 53, 364, 386
 - Support Package Manager
 - conventional mode 373
 - define queue 371
 - generation 373
 - import scenario 372
 - install queue 372
 - system landscape 376
 - update 370
 - Support Package Manager (SPAM) 365, 370
 - Support Package Stack (SPS) 369, 713
 - Surrogate ID 748
 - Swap memory 160, 412
 - Swaps 466
 - Synchronization 69
 - Synchronous I/O 447
 - SYS 143
 - SYS tablespace 100
 - SYSAUX tablespace 100
 - SYSDBA 100
 - SYSDBA/SYSOPER - Connect 193
 - SYSTEM 143
 - System
 - analysis 408
 - catalog 67
 - Change Number 551
 - change option 291
 - downtime 140, 545
 - Enqueue 452
 - error 70, 133, 135
 - file 111
 - Landscape 243
 - Monitor 90
 - privilege 141
 - process 537
 - R 79
 - statistic 516
 - stop 171
 - switch upgrade 399
 - System Global Area (SGA) 94, 417
 - buffer 415
 - dynamic 419
 - memory 415
 - System landscape 37
 - multisystem landscape 287
 - three-system landscape 284
 - two-system landscape 287
 - System, external 331
- ## T
-
- Table
 - generic 465
 - internal 59
 - single 465
 - statistics 455
 - Table buffering
 - generic buffering 467
 - monitoring 469
 - single buffering 467
 - Table definition (TTAB) 465
 - Tablespace 564, 655
 - backup mode 564, 567, 647
 - concept 98
 - dictionary managed 507
 - locally managed 506
 - loss 574, 631, 636, 637
 - point-in-time recovery 577
 - reorganization 206
 - rollback 631
 - system 631
 - temporary 614, 637
 - Tablespace layout 202, 568
 - new layout 204, 212
 - old layout 203
 - Tablespace type 98, 207
 - changeover 208
 - TAF (Transport Application Failover) 125
 - Takeover 658

- Tape
 - backup* 134
 - pool* 618
 - virtual libraries* 258
 - Task 295
 - Task Handler 44
 - Task type 317
 - TCP/IP 120
 - Technical terminology 32
 - Tertiary memory 134
 - Test system 655
 - Text 747, 750
 - Thin JDBC 119
 - Three-layer client/server architecture 35
 - Time stamp 135
 - TIMED_STATISTICS 417
 - TNS 120, 121
 - TNS_ADMIN 124
 - tnsnames.ora 111, 122
 - Top SQL statements 479
 - Total response time 416
 - tp 290, 301, 323, 333
 - Trace
 - application* 698
 - developer* 684, 699
 - file* 112, 570
 - JCo* 704
 - level* 684
 - performance* 698
 - single activity* 699
 - SQL* 699
 - Training system 134
 - Transaction 32, 69
 - data* 469, 755
 - enqueue (TX)* 451
 - error* 133
 - log* 81, 110, 118
 - management* 92
 - number* 33
 - profile* 478
 - Transfer structure 753
 - Transformation 726
 - Transformation rules 754
 - Transmission rule 754
 - Transparent Application Failover (TAF) 125
 - Transparent Network Substrate (TNS) 120
 - Transparent table 59
 - Transport
 - directory* 297, 345
 - domain* 299, 329
 - domain controller* 299
 - file* 321
 - group* 299, 328
 - layer* 303, 315
 - log* 336
 - management system* 290
 - request* 58
 - route* 302
 - route editor* 304
 - step* 336
 - strategy* 319
 - system* 134
 - Transport Organizer 288, 318
 - Tools* 318
 - Transportable tablespace 734
 - TREX 745, 781
 - Trigger 74, 82
 - Two Task Common (TTC) 120
 - Two-phase data backup 657
 - Two-stage login 196
- ## U
-
- Ultra Large Database 734
 - UML 71
 - Undo information 140
 - Unicode 31, 301, 337, 347
 - Unique index 490
 - UPD (update process) 47, 48
 - UPD2 48
 - Update 47, 76
 - error* 527
 - Update process 39, 527
 - type 1* 48
 - type 2* 48
 - Upgrade 341, 386, 402
 - assistant* 394
 - downtime* 400
 - frontend* 401
 - Oracle* 402
 - Prepare* 396
 - shadow instance* 399
 - strategy* 388, 390
 - system switch* 399
 - tools* 393
 - UPS 248
 - URL prefix table 702

User 140
 User context 44
 User enqueue (UL) 451
 User process 537
 user_ 101
 user_role_privs 144
 user_sys_privs 144
 user_tab_privs 144
 user_tablespace 99

V

V\$ view 419
 V\$DB_CACHE_ADVICE 423
 V\$PGA_POOL_ADVICE 423
 V\$SHARED_POOL_ADVICE 424
 v\$controlfile 112
 V\$LOCK 452
 V\$LOCKED_OBJECT 452
 v\$log 115
 v\$log_history 115
 V\$PGASTAT 425, 426
 V\$SESSION_EVENT 427
 V\$SESSION_WAIT 427, 432, 442
 V\$SGASTAT 424
 V\$SQL_WORKAREA_HISTOGRAM 427
 v\$sysaux_occupants 101
 V\$SYSTEM_EVENT 427, 430
 v\$tablespace 107
 V\$-View
 V\$PGA_TARGET_ADVICE 776
 V\$PGASTAT 776
 V\$SQL_WORKAREA_HISTOGRAM
 776
 Value help 33
 Variable 762
 VB* tables 48
 vi 85
 View 68
 View expansion 116, 127
 Virtual host name 342
 Virtual system 306
 Virtual view 101
 Visual Administrator 693

Volume 589, 600, 618
 check 620
 initialization 619
 management 589, 619

W

Wait event 422
 analysis 409
 classes 428
 structure 428
 Waste 98
 Web Dynpro 670
 ABAP 262
 Java 261
 Web reporting 761
 Webservice 31, 61
 where 77
 Windows 385
 with admin option 142
 Work area 425
 Work process
 components 169
 multiplexing 40, 41
 PRIV mode 524
 Workbench request 293
 Workload analysis 408, 417
 Workload repository 91
 Wrapper 62
 write complete waits 441

X

XML 82, 726

Z

Zero Administration Memory Manage-
 ment 459
 ZZZ_myindex_123 439
 ZZZ_myindex_20 413
 ZZZ_myindex_404 127
 ZZZAdd-on 366, 374
 ZZZASM instance 97