





Browse the Book

In this chapter you'll learn how an SAP HANA instance is structured, how it manages resources and activities, and how to use different tools to administer the instance and the databases it contains.

-  **"Instance Administration"**
-  **Contents**
-  **Index**
-  **The Authors**

Mark Mergaerts, Bert Vanstechelman

SAP HANA 2.0 Administration

1056 Pages, 2022, \$89.95

ISBN 978-1-4932-2104-2

 www.sap-press.com/5287

Chapter 4

Instance Administration

Now that you've been introduced to the various SAP HANA administration tools, in this chapter you'll learn how an SAP HANA instance is structured, how it manages resources and activities, and how to use different tools to administer the instance and the databases it contains.

4

SAP HANA is a database management system (DBMS)—perhaps one with very peculiar characteristics, but a DBMS nonetheless. When describing any DBMS, it's customary to make a distinction between the database, which is the physical repository of the data, and the technical infrastructure that is necessary to operate the database. This technical infrastructure is to a large extent defined and managed by software objects like executable code, processes, and memory structures. We use the term *SAP HANA instance*, or simply *instance*, to refer to this infrastructure as installed on one host. In a single-node installation, there is only one instance; in a multinode installation, there are several. The SAP HANA instance or instances and the database together form the *SAP HANA system* (or just *system*).

One thing that comes up in the sales pitch of every DBMS vendor is that the system needs very little work—that it is, so to speak, self-managing. As anyone with a background in database administration or consulting in that domain can attest, self-managing database systems live in the same imaginary universe as the one in which all lunches are free. Like any complex and often business-critical application, a certain amount of management work will always be required; SAP HANA is no exception to that rule. In this chapter, we'll try to provide you with the necessary knowledge of how an SAP HANA instance is organized and how you can carry out key management tasks. This chapter focuses on instance administration, and Chapter 5 focuses on database administration, but there are areas where the dividing line between the two isn't sharp, so there's inevitably some overlap. This chapter ventures into the database realm, for example, when describing workload classes or database logging.

The first section in this chapter describes how an SAP instance is structured: you'll learn how an instance is uniquely identified, which server-level entities (like operating system users, groups, and directory trees) it uses, which processes it launches, and which network ports these processes make use of. One of the basic things an SAP HANA administrator has to know is how to start and stop an instance, which is the subject of the second section. Then we'll talk about the parameters that define how the instance is configured. The fourth section focuses on that most critical of SAP HANA resources:

memory. We'll see how SAP HANA organizes its memory and how it behaves when there isn't enough of it. We then move on to sessions and transactions, which represent the work applications actually perform in the database. The subject of Section 4.6 is load management and distribution: here we'll see how to control and optimize the allocation of precious server resources to different workloads. Section 4.7 looks at some special SAP HANA utilities that run on the database server. The last section of the chapter deals with redo logs, which hold a persistent record of all transactions. As we said earlier, this subject might just as well be part of the chapter on database management, but because we concentrate mainly on the management aspect, we cover it here.



Metadata and Monitoring Views

In this and future chapters, you'll encounter several references to *monitoring views*. We discuss these in detail in Chapter 15, but it's useful to briefly introduce them here. SAP HANA provides an extensive collection of database views that contain information about its inner workings. These views are organized into three categories:

- *Catalog views* contain the metadata—that is, information about data objects such as schemas, tables, views, services, and so on. These are covered in Chapter 11.
- *Monitoring views* provide current runtime data about the active instance, including statistics and status information. Monitoring views aren't physical objects in the database. The data they contain isn't stored anywhere; it's calculated on the spot when you query the view. All monitoring views have names beginning with "M_". We discuss monitoring views in Chapter 15.
- *Statistics service views* (often shorted to *statistics views*) contain historical information about the status, resource usage, and performance of the system. They can be used to study the behavior of the SAP HANA system over time. Like the monitoring views, we cover statistics views in Chapter 15.

4.1 Structure of an SAP HANA Instance

When designing SAP HANA, SAP chose not to invent an entirely new organizational model but instead to make use of an existing model that has proven its quality and manageability since the introduction of the first SAP R/3 release in the early 1990s; this is the model used for SAP application servers. If you have an SAP administration background, then the way an SAP HANA instance is organized will look very familiar. Even if you have no such background, you'll find this structure easy to understand and navigate. In this section, we look at the system ID (SID), the operating system users and groups, the directory organization, and the process hierarchy, each time pointing out the close resemblance between the two types of instances. In Sections 4.1.6 to 4.1.8, we look at some directories and files that are specific to SAP HANA: the database directories where you find the data and log volumes; the INI files, which

contain the configuration parameters; and the trace files, in which runtime information about the instance is collected. To navigate the different file system areas with more ease, the SAP HANA installation creates a set of aliases (command shortcuts), which we list in the last section.

4.1.1 SAP HANA System ID and Instance Number

Every SAP system, ABAP or Java, has a name: the system ID (SID). Likewise, an SAP HANA system also has a system ID, which in this chapter we will always denote with *HSID*. Where it appears as part of a longer string, such as in directory paths, we use *<HSID>* as a placeholder. Like the SID, the HSID consists of three characters. The first is an uppercase letter and the second and third either an uppercase letter or a number. Apart from some reserved names, the HSID can be chosen freely at installation time. A frequently used (but not mandatory) convention for SAP HANA is to let the system ID begin with *H*.

Again, like every SAP application instance, an SAP HANA instance is uniquely identified by a two-digit number in a range from 00 to 97 (98 and 99 are reserved). In this chapter, we'll use *<xx>* as a placeholder for the instance number.

When using the command line, the administrator will often have to use the HSID and instance number as command parameters. To make life easier and to provide better portability of, say, management scripts, two environment variables are set in the OS environment of the SAP HANA administrator user. These are shown in Table 4.1.

Environment Variable	Description
SAPSYSTEMNAME	SAP HANA system ID
TINSTANCE	Instance number

Table 4.1 Environment Variables for HSID and Instance Number

4.1.2 Users and Groups

The ID of the operating system user for the SAP HANA administrator is *<hsid>adm*, which is again the exact equivalent of the *<sid>adm* user of SAP application instances. All management activities regarding the SAP HANA instance are performed while logged in as this user.

The primary group of the *<hsid>adm* user also has the same name as in an application instance: *sapsys*. With SAP HANA, the user is also a member of the secondary group *<hsid>shm*.

With multitenant database installations, it's possible that, for security reasons, the management of different tenant databases must be strictly segregated. If this is the

case, then the SAP HANA system can be configured for *high isolation* (the alternative, and default, is *low isolation*). In a high isolation setup, separate OS users and groups are created for each tenant database. The database-specific services then run with the identity of these dedicated users, who also have the ownership of the database files.

4.1.3 Directory Structure

Figure 4.1 shows the file system setup of an ABAP or Java dialog instance on the left and that of an SAP HANA instance on the right. It will be obvious that the two are virtually identical.

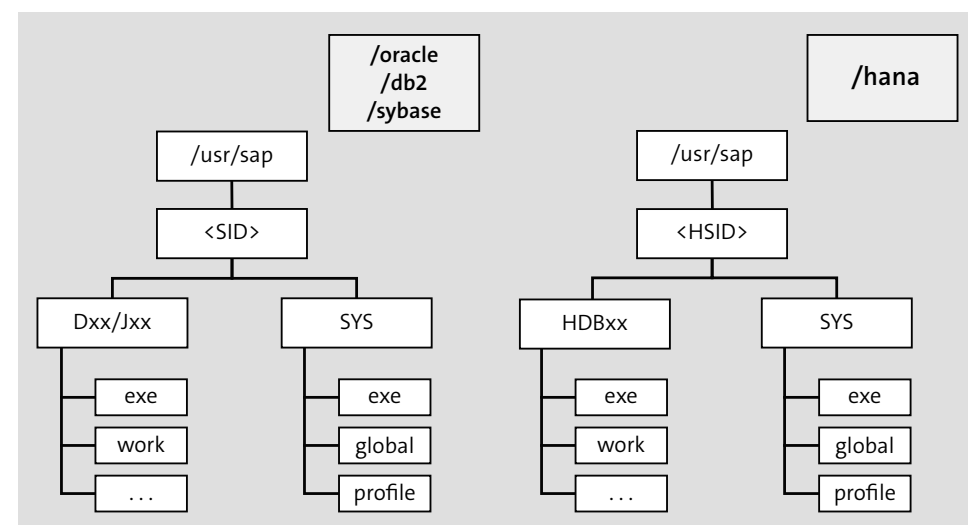


Figure 4.1 File System Structure of ABAP/Java and SAP HANA Instances

The top-level directory of an SAP dialog instance refers to the SAP HANA system ID, the HSID. Under this top node, the SAP dialog instance has a *SYS* subdirectory, which contains files shared by the entire system, and an instance-specific directory called *Dxx* (for ABAP) or *Jxx* (for Java), where the D and J denote the instance type (other types exist as well) and *xx* is the two-digit instance number. By the same token, the SAP HANA instance has a *SYS* subdirectory and an instance subdirectory *HDB<xx>*, where HDB is the instance type and *xx* is the instance number in the same range as that for dialog instances.



Central Services Instance

Apart from dialog instances, ABAP and Java systems also have a central services instance, which groups the processes that represent single points of failure. No equivalent exists for SAP HANA because high availability is not managed at this level.

In ABAP and Java systems, the database resides in a separate directory tree, whose top node typically has the name of the database system—for instance, */oracle*, */db2*, or */sybase*. The same is true for SAP HANA: the database files (including the data and log volumes) have their own directory tree under */hana*. In the case of SAP HANA, however, the SAP area (under */usr/sap/<HSID>*) and the database area (under */hana*) are connected by symbolic links. In fact, if you drilled down into the */usr/sap/<HSID>* structure, you would discover that physically very little exists there and that most subdirectories are in fact symbolic links to a subdirectory under */hana/shared*. There is little point going into the deep details of this, however: the structure is intended to create a recognizable and easily navigable environment. These two takeaways are sufficient:

- The SAP HANA instance directory tree is structured so as to be nearly identical to that of a classical SAP application instance.
- The actual instance files are located mostly in and below the */hana/shared* directory.

No /sapmnt File System

In contrast to SAP application instances on UNIX and Linux, there is no */sapmnt* file system.

4.1.4 Process Tree

Another important similarity between SAP dialog and SAP HANA instances is that they run under the control of the SAP start service framework. At the top of the process tree is the SAP service process *sapstartsrv*. This process is responsible for starting and stopping the processes in the instance and acts as their parent process. Figure 4.2 compares the process tree of an SAP HANA instance to that of an ABAP dialog instance. On the ABAP side, *sapstartsrv* launches the *sapstart* program, which starts the dispatcher process. The dispatcher then spawns a series of clones of itself, which become the instance work processes, and also starts some other specialized processes, like the RFC gateway and the Internet Communication Manager. On the SAP HANA side, we also find *sapstartsrv* and *sapstart*, which start the HDB daemon process; the daemon in turn launches the different SAP HANA service processes, which were described in Chapter 2.

sapstart

A note for the purists: strictly speaking, *sapstart* is not a child process of *sapstartsrv*. The *sapstart* process is detached from the process tree and attaches directly to the OS system daemon (PID 1). However, this technical behavior does not affect the validity of the example.

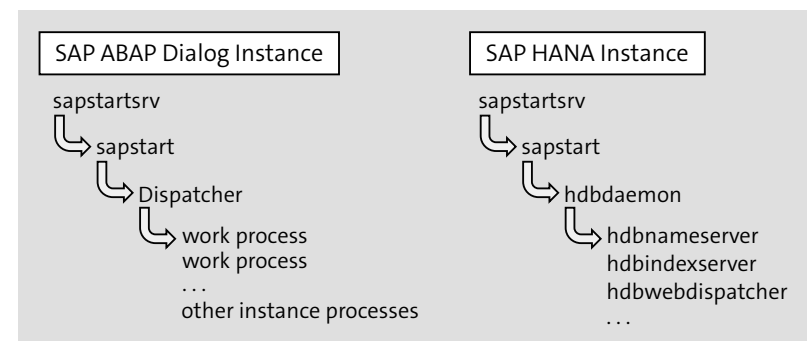


Figure 4.2 Process Tree of ABAP and SAP HANA Instances

If the instance is up and running, then the server command `HDB info` clearly displays this hierarchy. Figure 4.3 shows the process tree for an active SAP HANA, express edition instance (HANA SID HXE). The master process is `sapstartsrvt` (PID 1628, near the end of the list). The `sapstart` process (PID 4377) is the parent of process `hdb.sapHXE_HDB90`. This is nothing more than an alias (symbolic link) for the `hdbdaemon` process:

```
ls -l /usr/sap/HXE/HDB90/hxehost/trace/hdb.sapHXE_HDB90
/usr/sap/HXE/HDB90/hxehost/trace/hdb.sapHXE_HDB90 -> /usr/sap/HXE/HDB90/exe/
hdbdaemon
```

This is again something also found with ABAP instances, where the dispatcher and work processes (whose real program name is `disp+work`) appear under the name of a symbolic link `dw.sapSID_Dxx`.

```
$ HDB info
USER      PID      PPID     RSS  COMMAND
hxeadm    1950     1887    3756  sshd: hxeadm@pts/0
hxeadm    1951     1950    5284  \_ -bash
hxeadm    572      568     3860  sshd: hxeadm@pts/1
hxeadm    573      572     5364  \_ -bash
hxeadm    8584     573     3456  \_ /bin/sh /usr/sap/HXE/HDB90/HDB info
hxeadm    8615     8584    2892  \_ ps fx -U hxeadm -o user:8,pid:8,ppid:8,pcpu:5,vsz:10,rss:10,args
hxeadm    4377     1      3036  sapstart pf=/usr/sap/HXE/SYS/profile/HXE_HDB90_hxehost
hxeadm    4385     4377   69844  \_ /usr/sap/HXE/HDB90/hxehost/trace/hdb.sapHXE_HDB90 -d -nw -f /usr/sap/HXE/
hxeadm    4403     4385   555936  \_ hdbnameserver
hxeadm    4745     4385  123004  \_ hdbcompileserver
hxeadm    4768     4385   920092  \_ hdbindexserver -port 39003
hxeadm    4970     4385  308608  \_ hdbdiserver
hxeadm    4973     4385   542288  \_ hdbwebdispatcher
hxeadm    1628     1      37872  /usr/sap/HXE/HDB90/exe/sapstartsrvt pf=/usr/sap/HXE/SYS/profile/HXE_HDB90_hxe
hxeadm    1546     1      4752  /usr/lib/systemd/systemd --user
hxeadm    1549     1546    1604  \_ (sd-pam)
```

Figure 4.3 SAP HANA Instance Processes

4.1.5 Services and Ports

The services that make up an SAP HANA instance either work on behalf of the entire system or belong to a specific tenant only. In the system metadata, a service is almost always identified by the host where it's running and the network port it's using for all communication. This combination of hostname and port number can therefore be

seen as a unique identification for a given service, and later in this chapter and in other parts of the book we'll encounter many situations in which you must use or interpret this identification to obtain data about the system.

The SAP HANA installation defines the services in a range starting at `3xx00`, where `xx` denotes the instance number; for example, port numbers for an instance `01` would start at `30100`. Table 4.2 shows standard port number assignments as set by the SAP HANA installation process or SAP HANA lifecycle manager (`hdb1cm` executable).

Port	Database	Service Name
3xx00	-	daemon
3xx01	System DB	nameserver
3xx02	System DB	preprocessor
3xx03	Initial tenant	indexserver
3xx04	Initial tenant	scriptserver
3xx05	Initial tenant	diserver
3xx06	Initial tenant	webdispatcher
3xx07, 3xx08	Initial tenant	xsengine
3xx10	Initial tenant	compileserver
3xx11	Initial tenant	dpserver
3xx40–3xx99	Other tenants	Services assigned by installation or <code>hdb1cm</code> when creating additional tenants

Table 4.2 Port Numbers in SAP HANA System

When a tenant database is created via SQL with the `CREATE DATABASE` statement, it's possible to assign the port numbers explicitly. In that case, tenant port numbers may differ from this standard.

You can find the services and their associated port numbers by querying the `M_SERVICES` monitoring view.

4.1.6 SAP HANA Database Directories

The top node for the database directory tree is `/hana`. This area contains the physical directories and files of the SAP HANA application (which the symbolic links in `/usr/sap/<HSID>` point to), as well as the data and log volumes of the system and tenant databases.

The data and log volumes reside under a directory node known as the *basepath*. Its physical path may differ between installations, but it can be determined from two configuration parameters in the *global.ini* profile in the [persistence] section:

- `basepath_datavolume`
- `basepath_logvolumes`

If SAP HANA is up and running, then you can find the values of these parameters by querying the `M_CONFIGURATION_PARAMETER_VALUES` monitoring view, as shown in Listing 4.1.

```
select host, port, key, value from M_CONFIGURATION_PARAMETER_VALUES
where file_name = 'global.ini' and key in
    ('basepath_datavolumes', 'basepath_logvolumes');
HOST,PORT,KEY,VALUE
"prdhana1",33003,"basepath_datavolumes","/hana/data/HP1"
"prdhana1",33007,"basepath_datavolumes","/hana/data/HP1"
"prdhana1",33003,"basepath_logvolumes","/hana/log/HP1"
"prdhana1",33007,"basepath_logvolumes","/hana/log/HP1"
```

Listing 4.1 Retrieving Data and Log Volume Basepaths

As you can see, the parameters are set twice: for service 33003 and 33007 (which, based on Table 4.2, correspond to the standard ports for the index server and SAP HANA XS engine). Each SAP HANA service that needs persistent storage has its own data and log volumes and therefore its own storage path (although in practice they will rarely, if ever, be different between services).

Data Volumes

Under the basepath, the data volumes are organized as shown in Figure 4.4.

The directories `mnt00001`, `mnt00002`, and so on correspond to different hosts: `mnt00001` always exists; scale-out systems will also have `mnt00002` and higher depending on the number of nodes. The `hdb00001` subdirectory contains the data volume of the name server of the system database. The tenants have their own subdirectories, like `hdb0000n.ddd` containing the data volumes for the index server and the other services that use persistent storage. The `ddd` suffix is a sequence number indicating the tenant.

By default, a data volume consists of a single data file with the name `datavolume_0000.dat`. The process of writing data from memory to persistent storage continuously adds, changes, and deletes pages to this file. When additional space is required, the file will automatically expand; however, it does *not* shrink when space requirements decrease. Shrinking a data volume is possible, but it's a manual operation.

The data volume of the index server can be partitioned in such a way that its data is distributed over different disk stripes, which can improve performance.

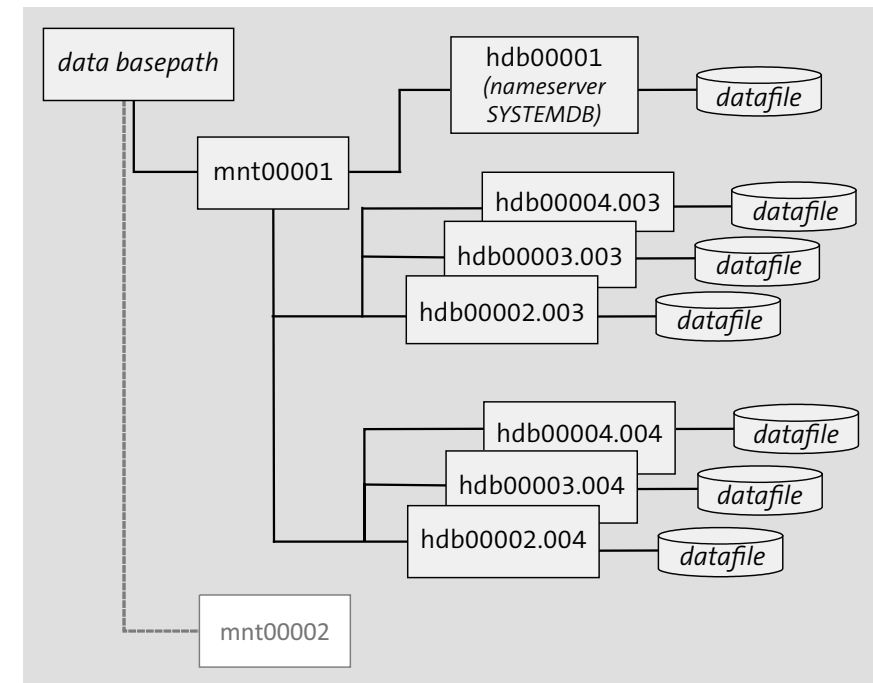


Figure 4.4 Directory Structure for Data Volumes

The data files of the database are listed in the `M_DATA_VOLUMES` monitoring view. Join this view with `M_SERVICES` to see the association between the volume and the service, as shown in Figure 4.5.

```
1 select d.host, d.port, s.service_name, file_name, to_integer(size/(1024*1024)) as SIZE_MB
2 from m_data_volumes d
3 inner join m_services s on s.host = d.host and s.port = d.port
4 order by 1,2;
5
```

HOST	PORT	SERVICE_NAME	FILE_NAME	SIZE_MB
exphanap1	30158	indexserver	/hana/data/HQ1/mnt00001/hdb00003.00005/datavolume_0000.dat	147072
exphanap1	30161	xsengine	/hana/data/HQ1/mnt00001/hdb00002.00005/datavolume_0000.dat	192
exphanap1	30164	dpserver	/hana/data/HQ1/mnt00001/hdb00004.00005/datavolume_0000.dat	192
exphanap1	30167	docstore	/hana/data/HQ1/mnt00001/hdb00005.00005/datavolume_0000.dat	192
exphanap1	30170	scriptserver	/hana/data/HQ1/mnt00001/hdb01024.00005/datavolume_0000.dat	192

Figure 4.5 Listing Data Volumes of Database

Log Volumes

Figure 4.6 shows the directory structure for the log volumes. For readability, Figure 4.6 only shows the directory structure for a single-tenant database.

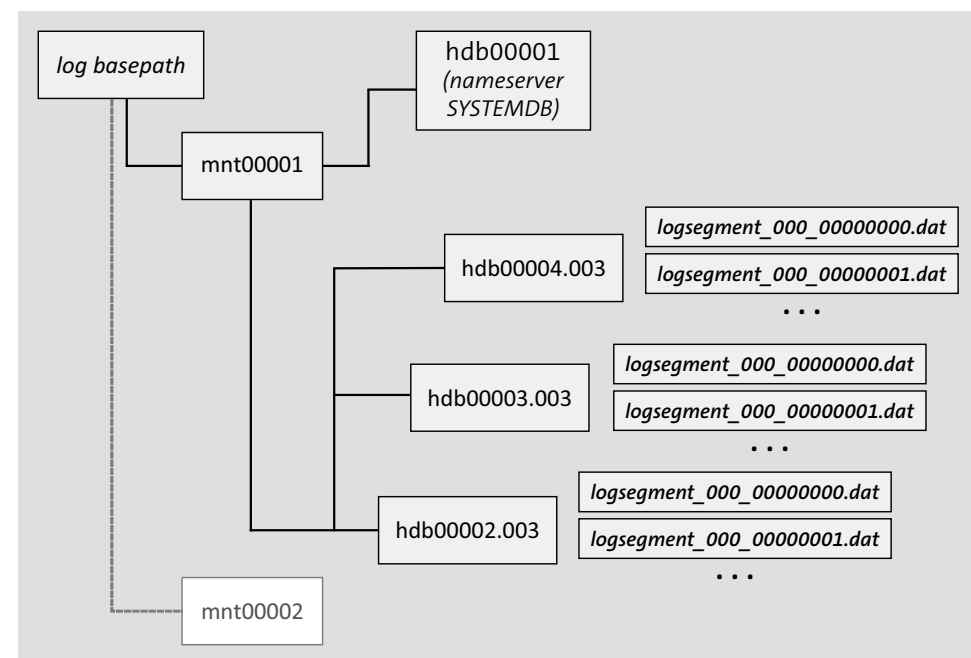


Figure 4.6 Directory Structure for Log Volumes

The directories are organized in precisely the same way as for the data volumes. The only difference is in the contents of the subdirectories. Instead of a single data file, there is a constantly evolving collection of redo log files. These files are called *log segments* and are numbered sequentially starting with 0. To prevent the disks containing the logs from overflowing, it's necessary to schedule regular log backups. Redo log management and backups are subjects we'll cover later in this chapter.

In addition to the redo logs, each log volume also contains a log directory file with the name *logsegment_000_directory.dat*.

Now that you know where to find the data and log volumes of the SAP HANA database, it's time for a warning—a *dire* warning.



Here Be Dragons!

Never—we repeat, *n-e-v-e-r*—manipulate the SAP HANA data and log volumes with operating system commands! Operations on the files should be handled only by the DBMS processes or through SQL commands. Do not rename, move, or—worst of all—delete files manually. You'll end up with an unusable database that can no longer be started. When the SAP HANA installation creates the data and log directories, it places an empty file called `__DO_NOT_TOUCH_FILES_IN_THIS_DIRECTORY__` there. That file is there for a reason.

Threatening as it sounds, there is one exception to this rule: in a situation where the disk containing the redo log is full, you will probably have to move log segments by hand—but you'll have to move them back to where they came from later.

4.1.7 INI Files: Database Parameters

All SAP HANA parameters are stored in a collection of server files, commonly known as INI files (from the file extension *.ini*). If you're old enough to remember preregistry versions of Microsoft Windows, then both the term INI file and the structure of these files will bring back memories because this was the typical format Windows-based applications used for their parameter settings. In fact, the SAP HANA parameter files are formatted in precisely the same way as their Windows ancestors: every INI file is subdivided into sections titled with names in square brackets; the parameters themselves are entered as key/value pairs, as shown schematically in Listing 4.2.

```
[sectionA]
parameter1 = value1
parameter2 = value2
[sectionB]
parameter3 = value3
```

Listing 4.2 General Format of INI File

In the SAP HANA documentation, a parameter therefore always has a three-part identification: the INI file, the section, and the parameter file. We already saw an example of this when we mentioned the parameters for the data and log basepaths in Section 4.1.6.

Parameters are defined at four different levels or, to use their precise name, *layers*. These layers are listed in Table 4.3.

Layer	Scope
DEFAULT	SAP HANA default settings
SYSTEM	Entire SAP HANA system
HOST	Single host
DATABASE	Individual database (multitenant only)

Table 4.3 Parameter Layers

We return to this when we talk about parameter maintenance, but we're mentioning it here because the layer affects the location of the INI files.

Although parameters exist at four layers, they can only be set at three. The settings at the DEFAULT layer are read-only and can't be changed.

Every layer has its own directory; we list these in Table 4.4.

Layer	Directory
DEFAULT (read-only)	<code>/usr/sap/<HSID>/HDB<xx>/exe/config</code>
SYSTEM	<code>/usr/sap/<HSID>/SYS/global/hdb/custom/config</code>
HOST	<code>/usr/sap/<HSID>/HDB<xx>/<hostname></code>
DATABASE	<code>/usr/sap/<HSID>/SYS/global/hdb/custom/config/DB_<tenant></code>

Table 4.4 Directories for INI Files

Not every INI file is defined for every layer (except the default layer, which has a version of all INI files). For example, the `daemon.ini` file, which contains the parameters for the HDB daemon process, only exists at the system and host layer, not at the database layer. Conversely, the majority of INI files exist at the system and database layers but not at the host layer. The directory of the default layer contains a file called `inifiles.ini`, which lists all existing INI files, indicates for each layer whether they are defined there, and also shows the affected services and a brief description. Listing 4.3 shows a small extract from this file.

```
[file:global.ini]
layered=yes
default=yes
system=yes
host=no
database=yes
services=*
description=global settings for all services
```

```
[file:indexserver.ini]
layered=yes
default=yes
system=yes
host=no
database=yes
services=indexserver,docstore,xsengine,dpserver,etsserver
description=hdbindexserver
```

```
[file:inifiles.ini]
hidden=yes
layered=yes
default=yes
system=no
database=no
```

```
host=no
description=inifile meta information
```

```
[file:nameserver.ini]
layered=yes
default=yes
system=yes
host=no
database=no
services=nameserver,webdispatcher
description=nameserver
```

Listing 4.3 Extract from inifiles.ini

To see which INI files are in use and at which layers, query the `M_INIFILES` monitoring view. Figure 4.7 shows the contents of the view in an SAP HANA Express system. At the time of the query, only the `global.ini` and `indexserver.ini` files exist at the database layer for the HXE tenant. Listing the contents of the corresponding server directory shows that indeed only those two INI files exist there.

	FILE_NAME	DEFAULT_LAYER	SYSTEM_LAYER	DATABASE_LAYER	HOST_LAYER
1	attributes.ini	TRUE	FALSE	FALSE	FALSE
2	computeserver.ini	TRUE	FALSE	FALSE	FALSE
3	diserver.ini	TRUE	TRUE	FALSE	FALSE
4	docstore.ini	TRUE	TRUE	FALSE	FALSE
5	dpserver.ini	TRUE	FALSE	FALSE	FALSE
6	esserver.ini	TRUE	FALSE	FALSE	FALSE
7	executor.ini	TRUE	FALSE	FALSE	FALSE
8	global.ini	TRUE	TRUE	TRUE	FALSE
9	indexserver.ini	TRUE	TRUE	TRUE	FALSE
10	multidb.ini	TRUE	FALSE	FALSE	FALSE
11	scriptserver.ini	TRUE	FALSE	FALSE	FALSE
12	statisticsserver.ini	TRUE	TRUE	FALSE	FALSE
13	streamingserver...	TRUE	FALSE	FALSE	FALSE
14	xscontroller.ini	TRUE	TRUE	FALSE	FALSE
15	xsengine.ini	TRUE	TRUE	FALSE	FALSE

```
hxeadm@hxehost: /usr/sap/HXE/SYS/global/hdb/custom/config/DB_HXE> ls -l
total 8
-rw-r--r-- 1 hxeadm sapsys 133 May 17 16:25 global.ini
-rw-r--r-- 1 hxeadm sapsys 374 May 14 16:33 indexserver.ini
```

Figure 4.7 M_INIFILES Monitoring View

4.1.8 Trace Files

Trace files are text files with diagnostic messages, which are continuously being written to by the SAP HANA processes. These files are an important source of information

about the functioning of the system, and they are the first place to turn to if you notice or suspect a malfunction. Not all information in the trace files is intended for common mortals, but it may still be very useful to SAP support specialists if you have to call upon them to solve a problem.

We'll defer a full discussion of trace files until Chapter 15, which is about monitoring SAP HANA, but because they play an important role in the management of the instance, it's necessary to at least mention their existence here.

4.1.9 Command Aliases for Directory Navigation

Even after you've become familiar with the directory organization, constantly having to remember what's where and typing the full directory paths gets a little tedious. To make this easier, the SAP HANA installation creates a set of aliases for the `cd` (change directory) command in the shell environment of the `<hsid>adm` user. These aliases can be used to quickly switch to a directory of interest. There are about 20 of these aliases; you won't use all of them frequently. Those listed in Table 4.5 are the ones that you might find the most useful, in the authors' experience.

Alias	Target	Description
<code>cdco</code>	<code>/usr/sap/<HSID>/HDB<xx>/exe/config</code>	Default parameter files (INI files)
<code>cdcoc</code>	<code>/usr/sap/<HSID>/HDB<xx>/global/hdb/custom/config</code>	System, host, and database INI files
<code>cdexe</code>	<code>/usr/sap/<HSID>/SYS/exe/hdb</code>	SAP executables and libraries directory
<code>cdglo</code>	<code>/usr/sap/<HSID>/SYS/exe/global</code>	Global directory
<code>cdhdb</code>	<code>/usr/sap/<HSID>/HDB<xx></code>	Instance directory
<code>cdhdblcm</code>	<code>/hana/shared/<HSID>/hdblcm</code>	SAP HANA lifecycle manager directory
<code>cdpro</code>	<code>/usr/sap/<HSID>/SYS/profile</code>	Instance profile directory
<code>cdpy</code>	<code>/usr/sap/<HSID>/HDB<xx>/exe/python_support</code>	Directory containing Python support scripts
<code>cdtrace</code>	<code>/usr/sap/<HSID>/HDB<xx>/<hostname>/trace</code>	Instance and tenant database trace files

Table 4.5 Useful Aliases for `cd` Command

To list all available aliases for directory switching, use the `alias | fgrep cd` command.

4.2 Starting and Stopping the Instance

You can start and stop the SAP HANA instance via the command line and with SAP HANA Studio; we describe both methods in the first two sections. The SAP documentation vaguely mentions that it's also possible to do with the SAP HANA cockpit, but this doesn't appear to be true at this time. By default, starting the SAP HANA instance also starts the tenant databases, but it's of course also possible to start and stop them individually. You can also disable the automatic startup for one or more tenants. We cover these subjects in Section 4.2.4. In Section 4.2.5, we show how you can check the current status of the instance and the tenant databases, and in Section 4.2.6 we take a closer look at the SAP start service process.

4.2.1 With the Command Line

We already saw the commands to start and stop the SAP HANA instance in Chapter 3. These operations are implemented as options of the `HDB` command, as shown in Listing 4.4.

```
HDB start
HDB stop
HDB restart(stop + start)
```

Listing 4.4 Start/Stop Options for `HDB` Command

These commands run in the foreground; that is, the server session remains blocked until the process finishes. If you want to do other things, such as monitoring the trace files, during startup or shutdown, then you must open an extra session on the database server.

There's nothing particularly interesting about the output of the commands, so we won't show it here—except for this little intriguing line in the `HDB stop` output:

```
hdbdaemon will wait maximal 300 seconds for NewDB services finishing.
```

Notice the name `NewDB`. We think the developers left this in on purpose to commemorate the very early days of SAP HANA by referring to it—just this once—by its old working name.

4.2.2 With SAP HANA Studio

In SAP HANA Studio, you find the functions to start, stop, and restart (`stop + start`) the system when you right-click a connection to the system database (not a tenant database). You then see the options **Start System**, **Stop System**, and **Restart System** in the context menu, as shown in Figure 4.8. Let's now look at each of those options in more detail, and also take a brief look at the trace files for these processes.

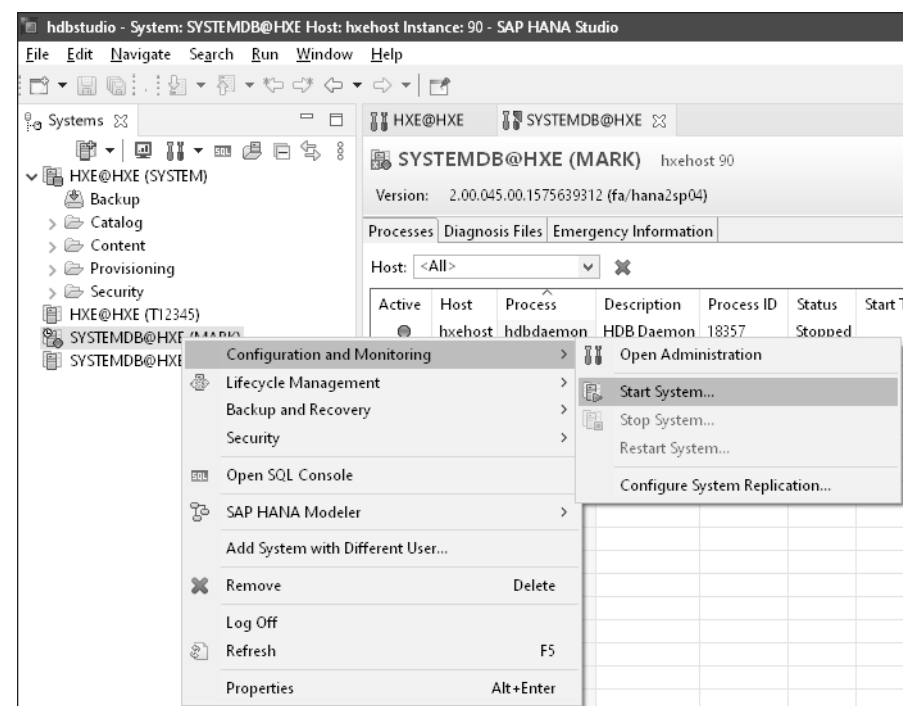


Figure 4.8 Menu to Start, Stop, or Restart System in SAP HANA Studio

Starting

Choose **Start System** from the context menu shown in Figure 4.8. Confirm in the pop-up window that you want to start the system now. The status of the services on the **Processes** screen will gradually change from yellow to green (see Figure 4.9); this screen refreshes every few seconds. The instance is fully started when all services are green.

Active	Host	Process	Description	Process ID	Status
	hxehost	hdbcompileserver	HDB Compileserver	26251	Running
	hxehost	hdbdaemon	HDB Daemon	25716	Initializing
	hxehost	hdbdiserver	HDB Deployment Infrastructure Server		Scheduled
	hxehost	hdbindexserver	HDB Indexserver-HXE	26277	Initializing
	hxehost	hdbnameserver	HDB Nameserver	25735	Running
	hxehost	hdbwebdispatcher	HDB Web Dispatcher		Scheduled

Figure 4.9 Service Status during Startup

Stopping

Choose **Stop System** from the context menu shown in Figure 4.8. The pop-up window in Figure 4.10 appears; here you choose between a soft shutdown, which waits for all active sessions to end, and a hard shutdown, which kills and rolls back the currently open transactions. For a soft shutdown, you can also specify a timeout; if there are still open sessions at the end of the timeout period, SAP HANA Studio will do a hard shutdown. The default timeout period is 10 minutes.

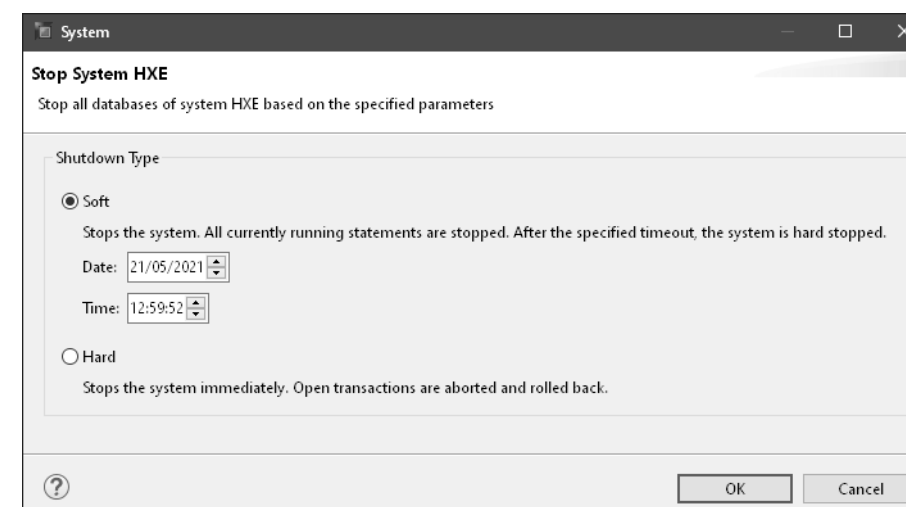


Figure 4.10 Shutdown Options

As soon as the shutdown sequence begins, the statuses of the services on the **Processes** screen will start to change from green to yellow. When the shutdown is complete, only the HDB daemon service will be listed with a red status icon (see Figure 4.11).

Active	Host	Process	Description	Process ID	Status	Start Time	Elapsed Time
	hxehost	hdbdaemon	HDB Daemon	18357	Stopped		

Figure 4.11 All Services Stopped

Restarting

This option shuts down and immediately restarts the instance. Choose **Restart System** from the context menu shown in Figure 4.8. As when stopping the system, you can choose between a soft shutdown (with a timeout) and a hard shutdown.

Trace Files

While the start or stop process is ongoing, it's interesting (and necessary, if there's trouble) to look at the instance trace files. In SAP HANA Studio, you can do so by opening the **Diagnosis Files** tab (to the right of the **Processes** tab in Figure 4.11). The trace file you may want to look at first is the one for the `hdbdaemon` process (`daemon_<hostname>.<service>.000.trc`). Double-click the name to display the contents of the file (see Figure 4.12). The screen will show the most recent messages; you can enable an autorefresh function (right-hand side of the toolbar) so that you always get up-to-date status information.

```

[27541]{-1}[-1/-1] 2021-05-23 15:30:59.914887 i Daemon DaemonHandle.cpp(00324) : Instance count 2. Searching for terminated processes
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.914972 i Daemon DaemonHandle.cpp(00336) : Process 'hdbindexserver', pid 27831 exited normally with
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.914982 i Daemon Events.cpp(00178) : "KillInstanceEvent" cancelled for 'hdbindexserver', pid 27831
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915036 i Daemon DaemonHandle.cpp(00427) : Found stopped instance 3 of program 'indexserver.HXE',
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915050 i Daemon Daemon.cpp(00929) : Line down current runlevel 3 to 0; next event "kill instance"
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915057 i Daemon Daemon.cpp(00958) : All instances in runlevel 3 stopped
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915060 i Daemon Daemon.cpp(00958) : All instances in runlevel 2 stopped
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915063 i Daemon Daemon.cpp(00968) : Runlevel 1 is not lined down
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915093 i Daemon DaemonHandle.cpp(00470) : Process 'hdbnameserver', pid 27559 is alive
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915106 i Daemon Daemon.cpp(01152) : Daemon state "stopping", runlevels [current:reached:target]=[
[27541]{-1}[-1/-1] 2021-05-23 15:30:59.915164 i Daemon Daemon.cpp(00172) : Shared memory updated, daemon runlevels changed [current:reac
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528064 i Daemon DaemonHandle.cpp(00324) : Instance count 1. Searching for terminated processes
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528293 i Daemon DaemonHandle.cpp(00336) : Process 'hdbnameserver', pid 27559 exited normally with
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528309 i Daemon Events.cpp(00178) : "KillInstanceEvent" cancelled for 'hdbnameserver', pid 27559,
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528393 i Daemon DaemonHandle.cpp(00427) : Found stopped instance 0 of program 'nameserver', delet
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528404 i Daemon Daemon.cpp(00929) : Line down current runlevel 1 to 0, no events scheduled
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528412 i Daemon Daemon.cpp(00958) : All instances in runlevel 1 stopped
[27541]{-1}[-1/-1] 2021-05-23 15:31:00.528418 i Daemon Daemon.cpp(00958) : All instances in runlevel 0 stopped
  
```

Figure 4.12 Monitoring Daemon Trace File

4.2.3 With SAP HANA Cockpit

According to the *SAP HANA Administration Guide*, you can start and stop the system by using the **Start System** and **Stop System** buttons at the top of the **Database Directory** screen. We cannot confirm this because in all the cockpit versions we checked, and despite satisfying the connection requirements, these buttons proved elusive.

Not that we really see this as a major shortcoming. Given that it's extremely easy to stop and start SAP HANA from the command line, this is really what we advise you to do. And if you insist on using a GUI, the function works impeccably in SAP HANA Studio, which will probably still be around by the time the issue in the cockpit gets fixed.

4.2.4 Starting and Stopping Tenant Databases

By default, the tenant databases start up automatically when the SAP HANA instance starts. It's possible to change this behavior with following SQL statement:

```
alter database <DB> no restart;
```

To reinstate the automatic start, use this statement:

```
alter database <DB> default restart;
```

These operations require a connection to the system database. You can use three methods for this:

1. With the command line

Tenant databases can be started and stopped manually with these statements (also when connected to the system database):

```
alter system stop database <DB>;
alter system start database <DB>;
```

2. With SAP HANA cockpit

In the **Database Directory** window, open a connection to the system database. In **Database Overview**, choose **Database Management** (if you don't see this button, then you're in a tenant database). On the next screen, there's a **Stop** or **Start** button to the right of each tenant (Figure 4.13).

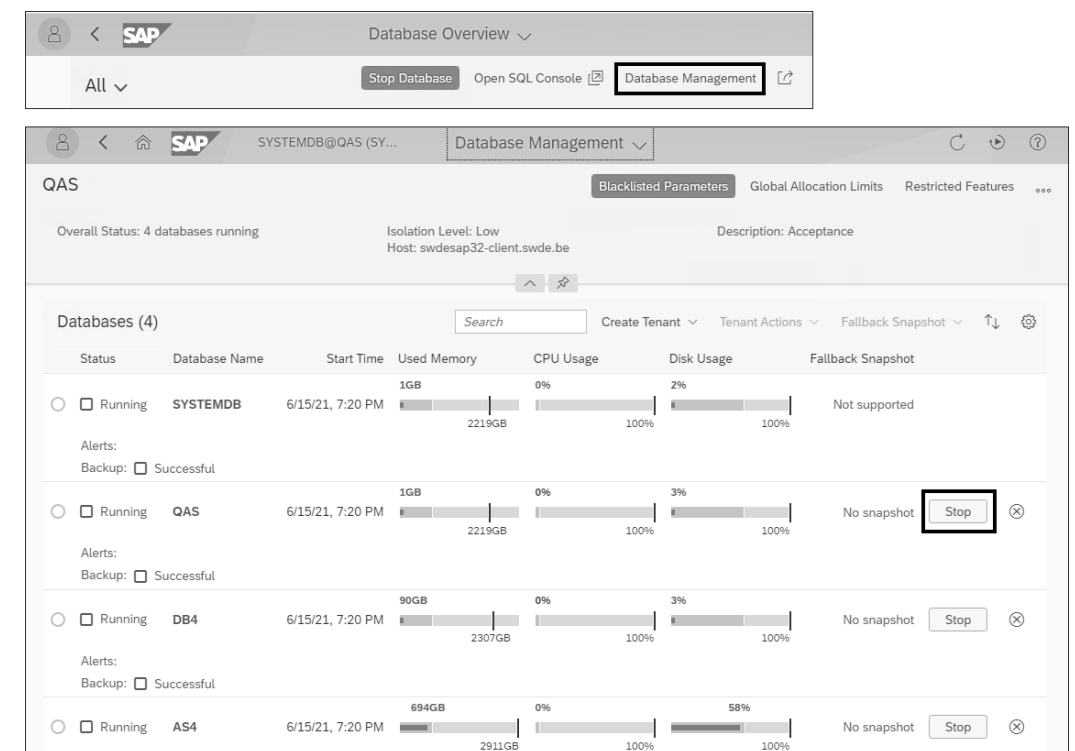


Figure 4.13 Start/Stop Tenant Database in SAP HANA Cockpit

3. With SAP HANA Studio

There is no specific function in SAP HANA Studio to stop and start tenant databases; to do this, open a SQL console for a system database connection and use the `alter system start database/stop database` statements shown earlier.

4.2.5 Checking the Instance and Database Status

To check the instance status from the server command line, use `HDB info`.

We already showed the output of this command in Figure 4.3. If the instance is running, you'll see the HDB daemon process with the service processes underneath it. The presence of one or more `hdbindexserver` processes indicates that the tenant databases are also active.

To see the status of all databases, use the following query when connected to the system database:

```
select * from M_DATABASES;
```

The `ACTIVE_STATUS` column shows the current state of the tenant database; if `ACTIVE_STATUS` is 'NO', then `ACTIVE_STATUS_DETAILS` gives the reason the tenant is down—for example, because it was stopped manually or because `autorestart` is disabled.



M_DATABASES View

The `M_DATABASES` view only provides information about all databases when executed in the system database. The view also exists for the tenant databases, but there it only shows the data for the tenant itself.

4.2.6 Managing the SAP Start Service

Stopping the SAP HANA instance using any of the methods we saw in the previous sections does not end the SAP start service (`sapstartsrv`) process itself. This is fine; there's normally no need to stop it, and its resource use is close to nothing. There may be exceptional situations, however, when you do want to stop the service process; furthermore, `sapstartsrv` has numerous management and information functions that you might want to make use of.

All communication with the `sapstartsrv` process is done using another program, `sapcontrol`. This program interfaces with the service process using web service calls. A `sapcontrol` command generally takes this form:

```
sapcontrol -nr <xx> -function <servicename> [<parameters>]
```

Here, `<xx>` is the two-digit SAP HANA instance number, `<servicename>` is the name of the service function you want to call, and `<parameters>` represents the parameters to be passed to the service function (not every function takes parameters). There are more command line arguments than this, but only `-nr` and `-function` are mandatory. Use `sapcontrol -h` for a list of all options and all service functions with their parameters.

To facilitate the use of commands like `sapcontrol` that need the instance number, an environment variable `TINSTANCE` is normally set in the environment of the `<hsid>adm` instance administrator. In the first section we first describe how you can stop and start

the SAP start service process; then we look at some of the interesting services you can invoke with `sapcontrol`.

Stopping and Starting `sapstartsrv`

To stop the service process, use this command:

```
sapcontrol -nr <xx> StopService
```

The service will normally stop almost immediately. It's safe to do this even while the SAP HANA instance is running; this is because, as we mentioned earlier, `sapstartsrv` is not attached to the SAP HANA process tree (the process at the top of the tree is `sapstart`). However, as long as the service process is stopped, you can't use HDB to stop or start SAP HANA because it needs to interact with `sapstartsrv`.

To restart the service process, use this command:

```
sapcontrol -nr <xx> StartService <HSID>
```

Notice that the `StartService` function has the `HSID` as a mandatory parameter.

There is also a function to restart (stop + start) the service, as follows:

```
sapcontrol -nr <xx> RestartService
```

Other Service Functions

We won't go through the entire list of available functions (and some don't apply to SAP HANA anyway), but we'll show some of the more useful ones.

To check the status of the SAP HANA instance processes, use this command:

```
sapcontrol -nr $TINSTANCE GetProcessList
```

If the instance is up, the output is a list of processes with `GREEN` status. If the output contains a `YELLOW` status, then this means that the process is not fully operational (e.g., still starting), and `GRAY` means that the process is stopped. For a stopped instance, you'll see something like Listing 4.5.

```
sapcontrol -nr $TINSTANCE -function GetProcessList
GetProcessList
OK
name, description, dispstatus, textstatus, starttime, elapsedtime, pid
hdbdaemon, HDB Daemon, GRAY, Stopped, , , 15427
```

Listing 4.5 `sapcontrol` Showing Stopped Process

Use following command to display the log of the `sapstartsrv` process:

```
sapcontrol -nr $TINSTANCE GetTraceFile
```

The next command lists the environment in which the service process was started; this can sometimes be useful when troubleshooting if you suspect that environment settings might have been incorrect when the instance was started:

```
sapcontrol -nr $TINSTANCE GetEnvironment
```

For version information (of the SAP start service, not of SAP HANA), use this command:

```
sapcontrol -nr $TINSTANCE GetVersionInfo
/usr/sap/HXE/HDB90/exe/sapstartsrv, 753, patch 512, changelist 1949210, ...
```

4.3 Maintaining Parameters

We saw in Section 4.1.7 that parameters are stored in INI files on the server and that these files exist at four layers. The default layer controls the default values of all parameters and cannot be changed. At the other three layers, system, database, and host, parameters can be set in separate layer specific INI files. The hierarchy of layers is as follows, from highest to lowest:

- HOST
- DATABASE
- SYSTEM
- DEFAULT

Settings of a higher layer override the values set at a lower layer. Which INI files are operational at which layer can be found in the `M_INIFILES` monitoring view, which we also encountered in Section 4.1.7.

Although the meaning and the hierarchy of the layers seems straightforward, there is one catch: when you specify the `SYSTEM` layer while connected to a tenant database, you are actually setting that parameter at the `DATABASE` layer. We come back to this in the SQL statement examples.

The INI files are text files, which means that you can not only read but also change them using a text editor. However, as of version 2.0 SPS 05, the practice of directly editing the INI files has been deprecated and should no longer be used. In the past, manually changing the INI files was the only way to modify parameters when SAP HANA was offline. As of SAP HANA 2.0 SPS 05, SAP delivers a Python script to maintain parameters; this script can be used whether SAP HANA is online or offline. In the following paragraphs, we describe five different methods to maintain the parameters: using SQL statements, with the SAP HANA cockpit, with Transaction `DBACOCKPIT` in ABAP systems, with SAP HANA Studio, and with the new Python script.

Before we start with this, one word about the documentation: there are hundreds of parameters, and unless you are vying for the title of SAP HANA uber nerd, there's no

reason to know them all. However, it's still useful to have complete documentation for them available. When you download the SAP HANA documentation (see Appendix B), the package includes a zipped archive with the *SAP HANA Configuration Parameter Reference Guide*. Extract this archive along with the other guides; then, to display the guide in a browser, open the `index.html` document (see Figure 4.14).

System Privileges

To maintain parameters, the user must have the `INIFILE ADMIN` and `EXTENDED STORAGE ADMIN` system privileges.

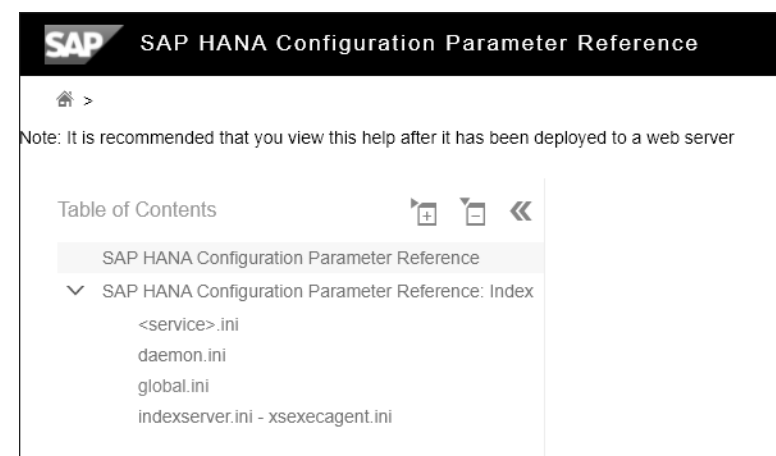


Figure 4.14 Parameter Reference Documentation

4.3.1 With SQL Statements

The SQL statement to change a parameter is `ALTER SYSTEM ALTER CONFIGURATION`. Use the keyword `SET` to set a parameter for the indicated layer and use `UNSET` to remove it from that layer. The general syntax to set or change a parameter is as follows:

```
ALTER SYSTEM ALTER CONFIGURATION ('filename', 'layer' [, 'name'])
    SET ('section', 'parameter') = 'value' [WITH RECONFIGURE]
```

In the first pair of parentheses, you specify the name of the INI file and the layer; the second set contains the section (without the square brackets) and the parameter name. The optional `WITH RECONFIGURE` means that the change is both written to the INI file and applied immediately to the active instance. Without this option, the INI file is changed, but this change does not take effect until either the instance is restarted or the same parameter is altered again, but now using `WITH RECONFIGURE`. The majority of parameters, but not all of them, can be changed online.

For the DATABASE layer, the change applies by default to the database you are currently connected to. In some cases, however, a change for a tenant database can only be made when connected to the system database. For the HOSTS layer, you have to specify an explicit host; we show this in one of the examples ahead. In both cases, you must use the 'name' argument, the optional third argument in the first set of parentheses. Parameter values must always be in single quotes, even if they are purely numeric.

For UNSET, the syntax is similar—except of course that no parameter value is assigned, as follows:

```
ALTER SYSTEM ALTER CONFIGURATION (filename, layer)
  UNSET (section, parameter) [WITH RECONFIGURE]
```

Let's walk through a fully worked out example, which also demonstrates the catch with the database and system layers we mentioned earlier. We want to set a parameter that deals with table repartitioning for tenant database P50 in SAP HANA instance HQ1 (for now, don't worry about what this means; we'll come back to this parameter again in Chapter 12). The `max_rows_per_partition` parameter is found in the `[table_placement]` section of the `global.ini` file.

Before making the change, we inspect the INI files on the server, first at the system level, as follows:

```
$ cd /usr/sap/HQ1/SYS/global/hdb/custom/config
$ grep max_rows_per_partition global.ini
(nothing found)
```

For the system layer, `global.ini` exists but it doesn't contain a setting for our parameter. We next look at the tenant database layer:

```
$ cd DB_P50
$ cat global.ini
global.ini: No such file or directory
```

No `global.ini` file even exists for the tenant database yet.

We now connect to the tenant and set the parameter at the database layer, as follows:

```
$ hdbsql -d P50 -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'DATABASE')
  SET ('table_placement','max_rows_per_partition') =
  '500000' WITH RECONFIGURE;
```

We then check the tenant database INI file directory again, as shown in Listing 4.6.

```
$ pwd
/usr/sap/HQ1/SYS/global/hdb/custom/config/DB_P50
$ cat global.ini
```

```
# global.ini last modified <date> <time> by hdbindexserver -port 30158
[table_placement]
max_rows_per_partition = 500000
```

Listing 4.6 New INI File Created

The system correctly created `global.ini` at the database layer and registered the one parameter that we set.

We now decide that we want to make this setting apply to the entire system. We connect to the system database and use the same statement, but now for the system layer:

```
$ hdbsql -d SYSTEMDB -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
  SET ('table_placement','max_rows_per_partition') =
  '500000' WITH RECONFIGURE;
```

We check whether the change was applied to the system layer `global.ini`, as follows:

```
$ cd /usr/sap/HQ1/SYS/global/hdb/custom/config
$ grep max_rows_per_partition global.ini
max_rows_per_partition = 500000
```

The parameter now exists the system layer.

Later, after some testing, we decide that a better setting would be 1 million instead of 500,000. We want to make this change at the system layer. We connect (mistakenly, as we will see) to the tenant database instead of the system database, as follows:

```
$ hdbsql -d P50 -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
  SET ('table_placement','max_rows_per_partition') =
  '1000000' WITH RECONFIGURE;
```

Let's verify that the change has indeed been applied in the INI file of the system layer, as follows:

```
$ cd /usr/sap/HQ1/SYS/global/hdb/custom/config
$ grep max_rows_per_partition global.ini
max_rows_per_partition = 500000
```

Somehow the system seems to have ignored our command and the parameter is unchanged. What about the tenant database layer?

```
$ cd /usr/sap/HQ1/SYS/global/hdb/custom/config/DB_P50
$ grep max_rows_per_partition global.ini
max_rows_per_partition = 1000000
```

This is the catch we talked about: we specified 'SYSTEM' as the layer in the SQL statement, but because we were connected to a tenant database, the system handled this as if we had specified 'DATABASE'.

Before describing how you can display the current parameter settings, let's look at a few more examples. The next statement shows the use of the HOSTS layer. The first time we're connected to the tenant database, as follows:

```
$ hdbsql -d P50 -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'HOST', 'exphanap1')
  SET ('table_placement','max_rows_per_partition') =
  '1000000' WITH RECONFIGURE;
* 2: general error: Configuration layer HOST can only be used in SYSTEMDB
```

No catch here: for the HOSTS layer, the system wants you to connect specifically to the system database. We will oblige, as follows:

```
$ hdbsql -d SYSTEMDB -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'HOST', 'exphanap1')
  SET ('table_placement','max_rows_per_partition') =
  '1000000' WITH RECONFIGURE;
```

Check the INI file directory for the host layer, as follows:

```
$ cd /usr/sap/HQ1/HDB01/exphanap1
$ grep max_rows_per_partition global.ini
max_rows_per_partition = 1000000
```

Finally, let's get rid of the parameter change at all levels, as shown in Listing 4.7.

```
$ hdbsql -d SYSTEMDB -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'HOST', 'exphanap1')
  UNSET ('table_placement','max_rows_per_partition') WITH RECONFIGURE;
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
  UNSET ('table_placement','max_rows_per_partition') WITH RECONFIGURE;
-- Switch to the tenant
\co -d P50 -u ...
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'DATABASE')
  UNSET ('table_placement','max_rows_per_partition') WITH RECONFIGURE;
```

Listing 4.7 Unset Parameter at All Levels

With parameter changes, it's often useful to document the reason for the change. You can do this by adding a COMMENT clause to the SQL statement, as follows:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'SYSTEM')
  SET ('mergedog', 'check_interval') = '30000' WITH RECONFIGURE
  COMMENT 'Recommended SAP EarlyWatch Alert Apr-2021';
```

To display the current parameter settings, you can query the M_INIFILE_CONTENTS monitoring view. The view contains one record per parameter per layer where it's defined. Figure 4.15 shows an example. The HOST column is filled in only for parameters of the HOSTS layer. The TENANT_NAME column is no longer in use.

FILE_NAME	LAYER_NAME	TENANT_NAME	HOST	SECTION	KEY	VALUE
indexserver.ini	DEFAULT			indexing	parallel_merge_threads	2
indexserver.ini	DEFAULT			joins	translator_cache_size	2000
indexserver.ini	DEFAULT			jwt_identity_provider	expiration_time	10
indexserver.ini	DATABASE			jwt_identity_provider	issuer	hana://exphanap1/HQ1/P50
indexserver.ini	SYSTEM			jwt_identity_provider	issuer	hana://exphanap1/HQ1/P50
indexserver.ini	DEFAULT			jwt_identity_provider	issuer	
indexserver.ini	DEFAULT			jwt_identity_provider	not_before_time	-5
indexserver.ini	DEFAULT			jwt_identity_provider	sign_algorithm	rs256
indexserver.ini	DEFAULT			jwt_identity_provider	user_claim	user
indexserver.ini	DEFAULT			linked_database	linked_database_cleanup_interval	0

Figure 4.15 Parameter Values in Monitoring View M_INIFILE_CONTENTS

Note that if you document the parameter changes with the COMMENT option, you won't see this comment in M_INIFILE_CONTENTS (nor will you find it in the INI file). To see these comments, you must query another monitoring view, M_INIFILE_CONTENT_HISTORY, as shown in Figure 4.16.

TIME	FILE_NAME	LAYER_NAME	SECTION	KEY	COMMENTS
2021-05-25 15:14:30.829000000	indexserver.ini	SYSTEM	mergedog	check_interval	Recommended SAP EarlyWatch Alert Apr-2021

Figure 4.16 Change History and Comments in M_INIFILE_CONTENT_HISTORY

4.3.2 With the SAP HANA Cockpit

In this section, we'll walk you through managing parameters with the SAP HANA cockpit. Look at Figure 4.17 and proceed as follows:

- Start in the Database Directory app and select the database and connection and user. As described earlier, connect to the system database if you want to change parameters of the SYSTEM and HOSTS layers; for parameters on the DATABASE layer, connect to the tenant.

- The Database Overview app opens. In the top-left corner, change the context to **Administration** or **All**. Scroll down to **Database Administration** section; there, click **Manage Database Configuration**.

The screenshot shows the SAP HANA Cockpit interface. At the top, there's a navigation bar with 'Administration' selected. Below it, the 'Database Administration' section is active, and 'Manage database configuration' is highlighted. The main area displays various database management tools like 'Capture Workload', 'Replay Workload', 'Database Backups', and 'Smart Data Integration'.

Figure 4.17 Maintain Parameters in SAP HANA Cockpit: Part 1

Now examine Figure 4.18, and proceed as follows:

- Initially all parameters for all layers are listed, which in our test system amounted to almost 2,000 entries. To make this manageable, use the filtering fields at the top to restrict the list to what you are interested in. Here we want to manage parameters in the [mergedog] section of *indexserver.ini*. To change a parameter, click **Override Value**.
- A pop-up window opens in which you can choose the layer (system, database, or host). Enable the checkbox for the layer you want to modify. You can now enter a value and an optional comment. If the value is expressed in a specific unit (here, milliseconds), then you will automatically be shown a list of acceptable units. Choose **OK** to save your change.
- The new setting is now listed along with the default (for which **Override Value** remains available in case you want to make a different setting for another layer). With the icons to the right of the custom setting, you can either change the value again or remove it (the equivalent of UNSET in the SQL statement).

The screenshot shows the 'Parameters' page in SAP HANA Cockpit. At the top, there are filters for Configuration File (indexserver.ini), Section (mergedog), Host (All), and Database (All). Below this is a table of 17 parameters. The 'check_interval' parameter is highlighted, and its 'Override Value' is visible. A pop-up window titled 'Override Values' is open, showing the 'check_interval' parameter details. The 'System Layer' checkbox is checked, and the new value '30000ms' is entered. The comment 'Demo only - return to default only' is also present.

Section	Parameter	Layer	Value	Override Value		
indexserver.ini	[] mergedog	+	active	DEFAULT	yes	Override Value
	auto_merge_decision_func	DEFAULT	((DRC*TMD > 3600*(MRC+0.000...		Override Value	
	auto_merge_priority_func	DEFAULT	1 / (7 + MMS)		Override Value	
	check_interval	DEFAULT	60000ms		Override Value	
	critical_merge_decision_func	DEFAULT	UPT > 43200 and LOADED and C...		Override Value	
	delta_merge_statistics_record_limit	DEFAULT	100000		Override Value	

Figure 4.18 Maintain Parameters in SAP HANA Cockpit: Part 2

4.3.3 With Transaction DBACOCKPIT

In DBA Cockpit, choose **Configuration • INI Files**. The list of INI files appears on the right. Open the file and section of the parameters you are interested in. The parameters in the sections are listed with their values at the different layers (see Figure 4.19).

To maintain a parameter, double-click its entry in the list. In the pop-up window shown in Figure 4.20, enter the new value. In this case, we're using the default connection to the tenant database, which means that we're only offered the option to change the system/database layer (remember that for tenant databases these layers are one and the same thing). Click **Save Change** to save the new value to the INI file and to change

the active setting for the instance if the parameter is dynamically changeable; as with the SAP HANA cockpit, the WITH RECONFIGURE clause is implicit and you cannot change this behavior. Unlike the SAP HANA cockpit, DBA Cockpit doesn't let you enter a comment.

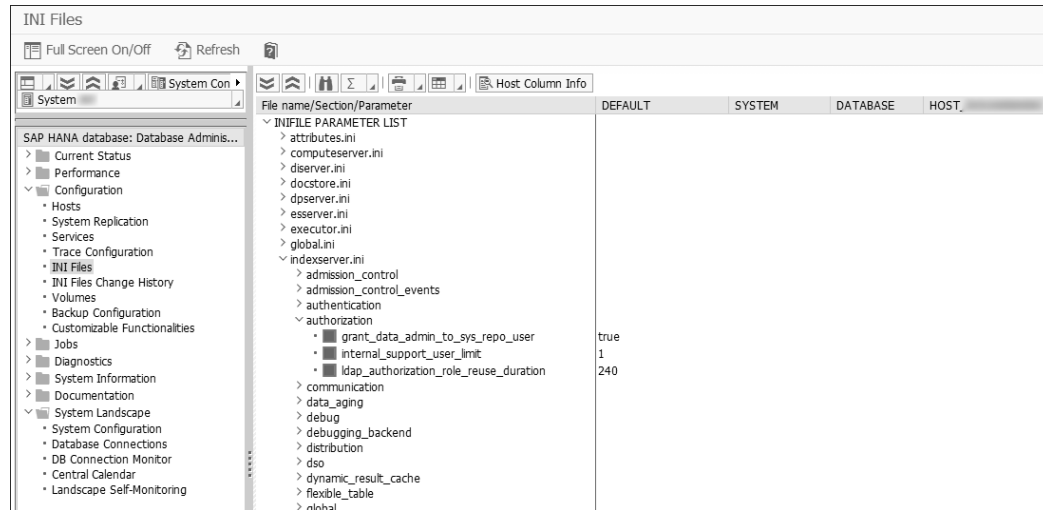


Figure 4.19 Display Parameters in DBA Cockpit

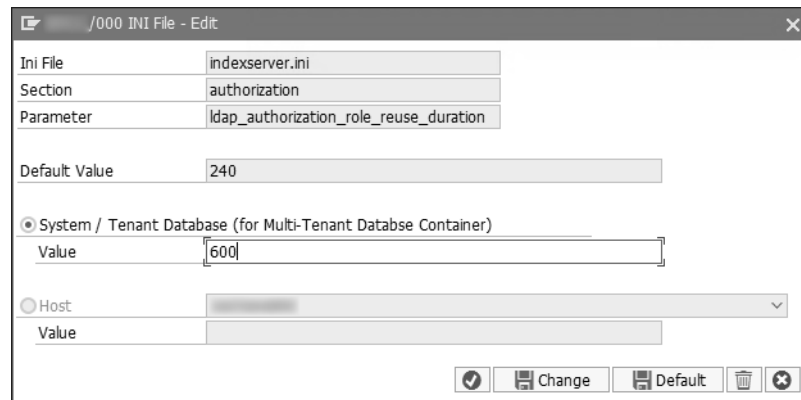


Figure 4.20 Maintain Parameters in DBA Cockpit

You must choose **Refresh** in the main SAP HANA cockpit window before the modified value is displayed (Figure 4.21).

To return the parameter to its default setting, double-click and then choose **Save Default**, shown in the window in Figure 4.20. Refresh the parameter list and the former change will no longer be visible.

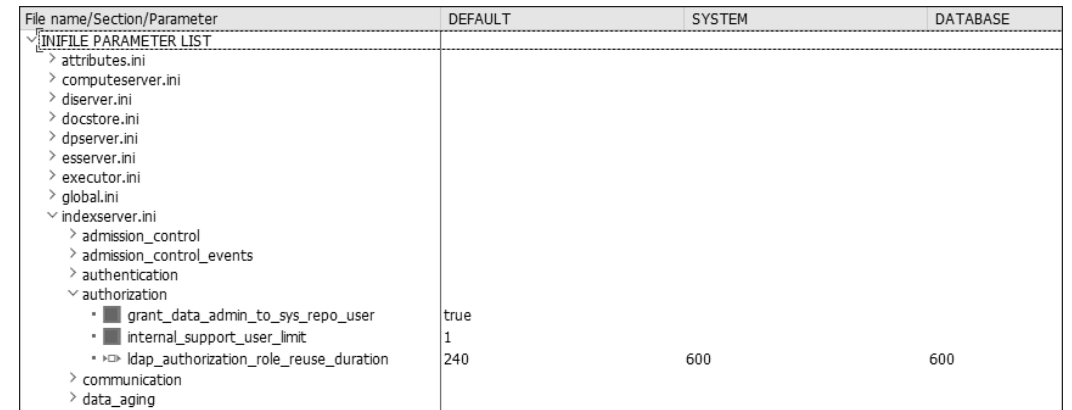


Figure 4.21 Display Changed Parameter in DBA Cockpit

4.3.4 With SAP HANA Studio

In SAP HANA Studio, open the Administration Console and go to the **Configuration** tab. Expand the list of INI files and the sections to get to the parameter you're interested in (see Figure 4.22). The gray diamonds mean that specific settings exist for the indicated layer.

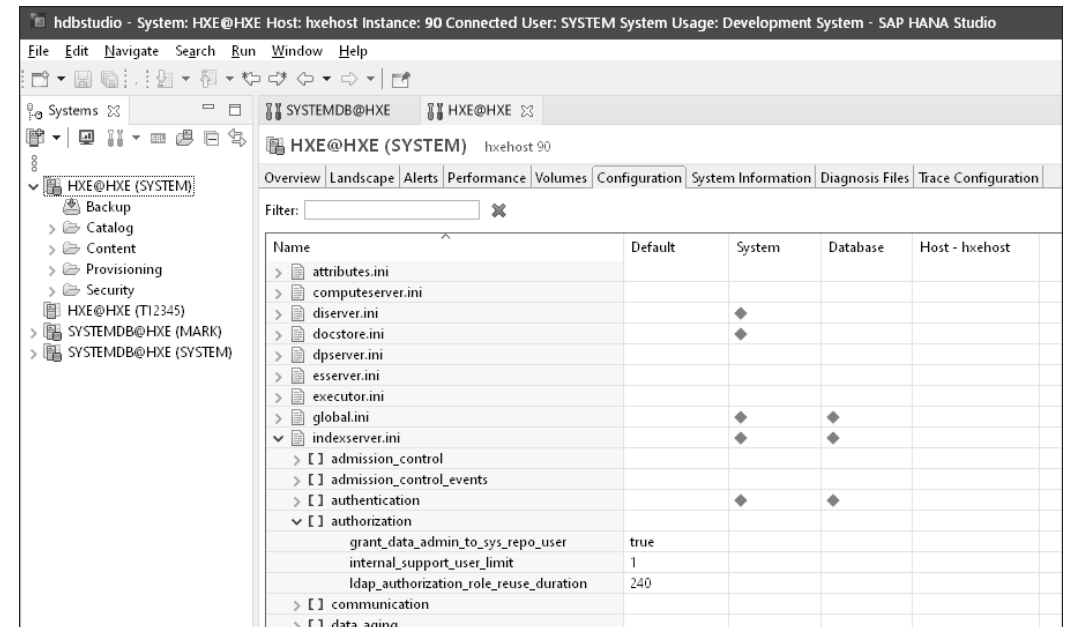


Figure 4.22 Display Parameters in SAP HANA Studio

Double-click a parameter to change its value, which opens the window shown in Figure 4.23. The layers that are available for input depend on the connection. In this case, we're connected to the tenant database, which only allows us to set the database layer (the hosts layer is also enabled for input, but entering a value there will be rejected with the error **Communication layer HOST can only be used in SYSTEMDB**).

The WITH RECONFIGURE option is used implicitly, so changes to dynamic parameters take effect immediately. It isn't possible to add a comment.

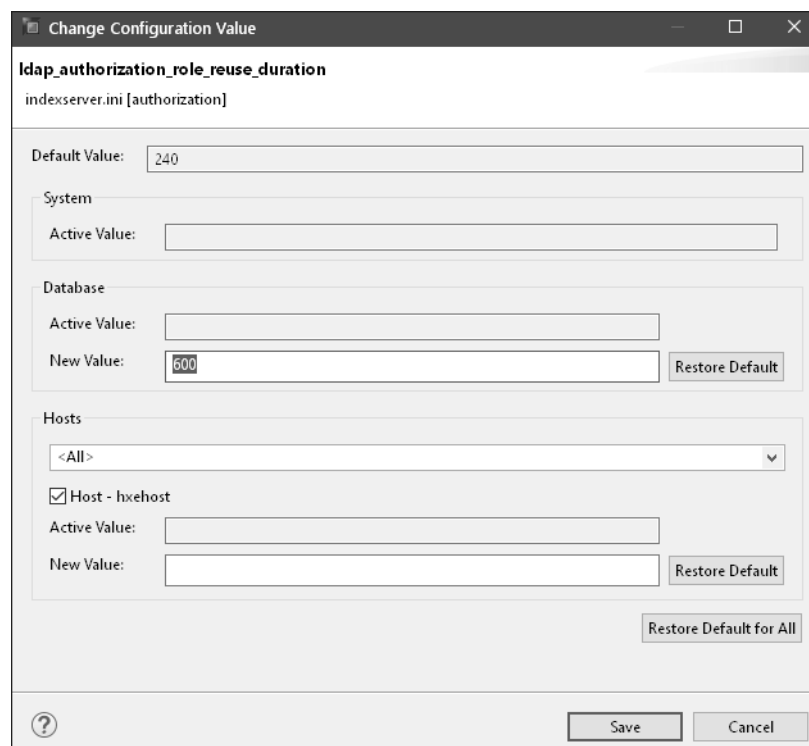


Figure 4.23 Maintain Parameters in SAP HANA Studio

4.3.5 With setParameter.py

This Python script is new in SAP HANA 2.0 SPS 05. You can use it to change parameters from the command line, regardless of whether the SAP HANA instance is online or offline. If the instance is offline, using this script replaces direct editing of the INI files (which is deprecated). For the parameters, you must specify the layer, file name, section name, and parameter name separated by forward slashes (/). Before running the script, use the `cdpy` alias to switch to the Python support directory—for example:

```
cdpy
python setParameter.py -set 'SYSTEM/indexserver.ini/mergedog/check_interval = 30000'
python setParameter.py -unset 'SYSTEM/indexserver.ini/mergedog/check_interval'
```

To set a parameter at the database layer, you must specify the name of the tenant database, as follows:

```
python setParameter.py \
  -set 'DATABASE:P50/indexserver.ini/mergedog/check_interval = 30000'
```

Backslash to Indicate More Lines to Follow

For long Linux shells spanning more than one line, use the backslash character (\) to indicate that more lines will follow.

For the hosts layer, you may omit the hostname, which then defaults to the current host, or specify a hostname explicitly, as shown in Listing 4.8.

```
python setParameter.py \
  -set 'HOSTS/indexserver.ini/mergedog/check_interval = 30000'
python setParameter.py \
  -set 'HOSTS:exphanap1/indexserver.ini/mergedog/check_interval = 30000'
```

Listing 4.8 Specifying HOSTS Layer with setParameter.py

By default, the script only changes the parameter in the INI file; for a dynamic parameter, the active value is not changed. This matches the behavior of the `ALTER SYSTEM ALTER CONFIGURATION` command, but it's contrary to the behavior in the SAP HANA cockpit, SAP HANA Studio, and Transaction DBACOCKPIT. Use the `-reconfigure` option to also change the parameter in the running instance, as follows:

```
python setParameter.py \
  -set 'SYSTEM/indexserver.ini/mergedog/check_interval = 30000' -reconfigure
```

You can also add comments, as follows:

```
python setParameter.py \
  -set 'SYSTEM/indexserver.ini/mergedog/check_interval = 30000' \
  -comment "For testing only, return to default later"
```

It's possible to change multiple parameters in one call of the script, as shown in Listing 4.9.

```
python setParameter.py \
  -set 'SYSTEM/indexserver.ini/password policy/minimal_password_length = 12' \
  -unset 'SYSTEM/indexserver.ini/password policy/minimum_password_lifetime' \
  -set 'DATABASE:P50/indexserver.ini/password policy/minimal_password_length = 16' \
  -set 'DATABASE:P50/indexserver.ini/password policy/last_used_passwords = 8' \
  -set 'DATABASE:P50/indexserver.ini/password policy/minimum_password_lifetime = 7' \
  -with reconfigure
```

Listing 4.9 Changing Multiple Commands at Once with setParameter.py



Adding the `-sapcontrol=1` option appends an extra line to the output in a format that can be easily parsed by text-processing commands, a very common practice in UNIX and Linux shell scripts, as shown in Listing 4.10.

```
python setParameter.py \
  -set 'SYSTEM/indexserver.ini/password policy/last_used_passwords = 8' \
  -sapcontrol=1
Setting parameters succeeded
SAPCONTROL-OK:

python setParameter.py \
  -set 'DATABASE:V10/indexserver.ini/password policy/last_used_passwords = 8' \
  -sapcontrol=1
SAPCONTROL-ERROR: DATABASE:V10/indexserver.ini/password policy/last_used_
passwords = 8 unknown database name
```

Listing 4.10 Using `-sapcontrol=1` Option with `setParameter.py`

4.3.6 Protecting against Unsupported Parameter Values

Until SAP HANA 2.0 SPS 03, it was possible to give any value to a parameter. SAP HANA would accept the setting without applying any form of consistency or even basic sanity check. Starting with SAP HANA 2.0 SPS 04, it's possible to protect against this by setting the `unsupported_configuration_change` parameter in the `[configuration]` section of the `global.ini` file.

Possible values are 'none', 'warning', and 'error'. The default is 'warning', which means you're still able to set a parameter to some absurd value, but at least the system will inform you that you are doing so. We recommend setting this parameter to 'error' for the SYSTEM layer, as follows:

```
-- Connected to SYSTEMDB:
alter system alter configuration('global.ini','SYSTEM')
set ('configuration','unsupported_configuration_change') =
  'error' with reconfigure;
```



Watch Your Spelling

Even with the parameter set to 'error', it's still possible to specify a nonexistent section or parameter name. Although this may sometimes be needed to implement SAP support advice, it also means that you must be extremely careful not to make mistakes. Consider these three parameter settings:

- `set ('password policy','minimal_password_length') = 16 <<< command 1`
- `set ('password_policy','minimal_password_length') = 16 <<< command 2`
- `set ('password policy','minimum_password_length') = 16 <<< command 3`

Only command 1 is correct, but in reality all three commands will work! Command 2 (with an underscore instead of a space in the section name) will create a new section by that name in the INI file. Command 3 (in which the parameter name is wrong) will create a parameter with the specified name alongside the "real" parameter, the value of which remains unchanged.

The graphical tools have an advantage here because they let you start from a list of parameters displayed by the tool. They still allow you to create new sections and parameters, but at least it's a conscious action by the user.

4.4 Managing Memory

When thinking about an in-memory database system, it's easy to understand that memory is its most fundamental resource. The way the SAP HANA system utilizes memory will to a large extent define how well that system is performing. Here we take a closer look at how memory is structured and managed. We first describe how SAP HANA organizes its memory and how the memory layout relates to that of the server. We then see how memory is allocated (Section 4.4.2) and what happens when there isn't enough of it (Section 4.4.3). Section 4.4.4 introduces heap memory, which is process-specific, and shared memory, which is shared among all processes in the instance. In Section 4.4.5, you'll learn how to limit the amount of memory SQL statements can use, which is quite useful if such a statement misbehaves. The last two sections deal with the monitoring of memory at the server level (Section 4.4.6) and for the SAP instance (Section 4.4.7).

4.4.1 Memory Layout

Figure 4.24 shows the general layout of memory on the server (left) and memory specific for SAP HANA (right). To start with the diagram on the left, the physical memory (RAM) installed in the database server is used partially by the operating system itself, partially by other processes unrelated to SAP HANA, and partially (a major part) by SAP HANA. The combined usage from all memory consumers might be smaller than the available RAM; in that case, part of the physical memory remains free. If demand for physical memory exceeds the available capacity, then the operating system will swap out part of the programs' memory pages to the swap area on disk according to a least recently used (LRU) algorithm. Swapping imposes a severe performance penalty on the affected processes, and in a correctly sized SAP HANA server it shouldn't occur at all.

The diagram on the right in Figure 4.24 shows the structure of the memory used by SAP HANA. Like every other application, SAP HANA is a collection of programs; these programs need memory to store their code instructions (also known as *program text*) and their stack (for data variables). All other memory allocated by SAP HANA is used on

behalf of the database system. This area is called the *SAP HANA memory pool*. The pool itself is made up of two major areas: the first and largest is used for the in-memory storage of tables. As we will see later in much more detail, database tables belong either to the row store or the column store, depending on the storage model that they follow. Normally, the column-store tables are both larger and far more numerous than the row-store tables, which means that they also occupy a significantly larger part of the pool. The second area holds the memory structures needed for database management. If the two areas combined take up less space than the total size of the pool, then the memory pool contains a free area. This space is free only from the perspective of SAP HANA; for the operating system, the memory is allocated and in use. SAP HANA will not normally return free memory in the pool to the OS; the memory is kept in the pool and serves to satisfy surges in demand, such as when the size of the table memory increases.

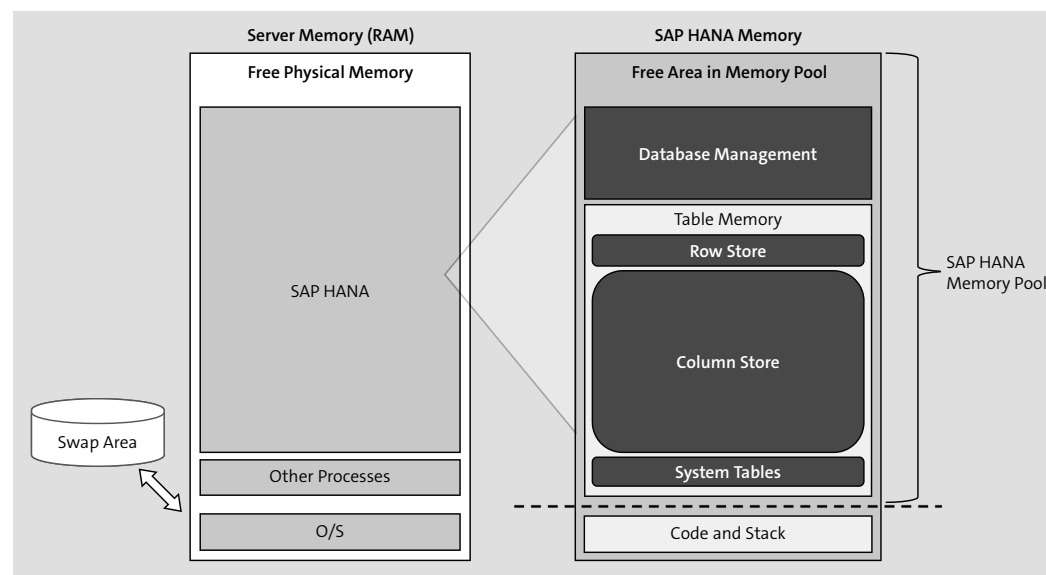


Figure 4.24 Memory Layout

4.4.2 When and How Memory Is Allocated

When SAP HANA is started, it doesn't allocate the entire memory pool in one go. It requests memory from the OS as demand increases, and this continues until it reaches a maximum quantity, the *global allocation limit*. This limit is set via the `global_allocation_limit` parameter in the `[memory]` section of the `global.ini` file.

The default setting is 0, which means that the global allocation limit is computed in the function of the physical memory of the server using the following rules:

- 90% of the first 64 GB
- 97% of the capacity above 64 GB

There is normally no reason to deviate from the standard setting, with two possible exceptions:

- If the server runs more than one SAP HANA system, then each system should have its global allocation limit set so that the combined limit doesn't exceed the capacity of the server.
- If the license sets a limit on the memory size, then the global allocation limit must of course respect the licensing conditions.

If you do set the global allocation limit explicitly, then you can express it either as an absolute number of megabytes or as a percentage of physical memory. The advantage of the latter is that if the installed physical memory changes, the global allocation limit adapts accordingly.

When the SAP HANA instance needs more memory and the pool size is below the global allocation limit, then the instance will request extra memory from the OS—but, as we stated in the previous paragraph, the opposite is not true: when memory in the pool becomes available again, such as when a query with a very large result set has ended, then this memory is not returned to the OS but is retained for later use.

In a multitenant installation, it's possible to set an allocation limit per tenant. The parameter for this is `allocationlimit` in the `[memory]` section of the `global.ini` file.

This setting can only be made from the system database; changing the allocation limit from inside the tenant is not allowed. This is one of the situations in which you must explicitly specify a database name in addition to the DATABASE layer in the ALTER SYSTEM ALTER CONFIGURATION statement. For example, the following command sets an allocation limit of 64 GB for the DEV tenant database:

```
hdbsql -d SYSTEMDB ...
alter system alter configuration('global.ini','DATABASE', 'DEV')
    set ('memorymanager','allocationlimit') = '65536' with reconfigure;
```

4.4.3 Memory Shortages

When SAP HANA runs out of available memory, it will use a series of increasingly invasive strategies to try and reclaim memory:

1. Return remaining free segments in the pool to the OS. It may seem contradictory to release memory when there isn't enough of it, but the reason is that the OS can do a better job at defragmentation. By requesting memory back after releasing it, SAP HANA may be able to satisfy large memory requests.
2. Perform garbage collection. The term *garbage collection* is well-known in the context of programming languages and denotes the process of freeing up memory areas that are no longer in use; in an object-oriented language, for instance, the garbage collector deletes objects that no longer have an active reference. In SAP HANA,

it covers a range of techniques that are applied to different memory areas; you can find more details in SAP Note 2169283. This note documents a long list of parameters that control the behavior of the various garbage collectors. Perhaps the most important of these is the memory garbage collector, which is lazy by default and only kicks in when the instance hits the global allocation limit. This behavior can be altered, but SAP recommends leaving it alone.

3. Shrink resource containers. One of the most important activities that takes place here is the unloading of column-store tables. The implication of this is of course that the next time the unloaded column is accessed, it must be reloaded from persistence, which causes disk I/O and penalizes performance. In deciding which columns to unload, SAP HANA uses a priority scheme (documented in SAP Note 2169283) aimed at keeping the impact as small as possible, but there will always be a price to pay.
4. Terminate transactions. If SAP HANA can no longer satisfy the memory requirements of active transactions, it can decide to forcibly terminate them.
5. Produce an out-of-memory (OOM) dump. Whatever action couldn't have its memory requirements met is terminated, and an OOM dump file is written to the trace directory. We introduced trace files in Section 4.1.8 and describe them in detail in Chapter 15. The OOM dump is not a dump of the entire memory (fortunately), but a text file containing the most relevant information about the conditions in which the dump occurred. The name of the file follows this format:

<processname>_<hostname>.<number>.rtedump.<number>.oom.trc

For example, it could look like this:

indexserver_exphanap1.12345.rtedump.98765.oom.trc

SAP Note 1984422 contains information on how to analyze OOM dumps.

4.4.4 Heap Memory and Shared Memory

Before we explain how you can monitor SAP HANA memory, we must introduce two new terms: heap memory and shared memory. *Heap memory* is memory that can only be accessed by the threads of a single process. For example, the heap memory allocated by the index server is only usable by the index server threads. *Shared memory*, on the other hand, is accessible to several processes. Data structures that must be used by several services are stored in shared memory. The memory SAP HANA uses can be divided into these two categories.

Heap memory is assigned through *allocators*. Each allocator belongs to a specific service; there are numerous allocators (more than 1,000 for the index server alone), organized in a hierarchical tree with a root allocator at the top. You can find the memory statistics per allocator in the M_HEAP_MEMORY monitoring view. Figure 4.25 shows heap usage statistics for an SAP S/4HANA tenant consolidated by service (DEPTH = 0 indicates

the top-level allocator record for the service; memory usage at a certain depth is the sum of usages of all subordinate allocators). As can be expected, the largest memory consumer, here with over 1 TB, is the index server. The INCLUSIVE_COUNT_IN_USE column shows the number of allocations since startup.

```

1 SELECT
2   s.service_name,
3   s.port,
4   h.host,
5   h.category,
6   h.depth,
7   to_integer(h.inclusive_size_in_use / 1048576) AS in_use_mb,
8   h.inclusive_count_in_use
9 FROM
10  M_HEAP_MEMORY AS h
11 INNER JOIN
12  m_services AS s
13 ON s.host = h.host
14    AND s.port = h.port
15 WHERE depth = 0;

```

	SERVICE_NAME	PORT	HOST	CATEGORY	DEPTH	IN_USE_MB	INCLUSIVE_COUNT_IN_USE
1	indexserver	30052		/	0	1079527	1191617480
2	diserver	30055		/	0	451	538587

Figure 4.25 Heap Memory by Service

Figure 4.26 shows the same view, but sorted by memory usage (keep in mind that the totals are consolidated at lower depths, so you must not add the usages together). Most of the allocator names in the screenshot are somewhat comprehensible, but if you descend further the names will quickly become much more cryptic.

```

1 SELECT
2   s.service_name,
3   s.port,
4   h.host,
5   h.category,
6   h.depth,
7   to_integer(h.inclusive_size_in_use / 1048576) AS in_use_mb,
8   h.inclusive_count_in_use
9 FROM
10  M_HEAP_MEMORY AS h
11 INNER JOIN
12  m_services AS s ON s.host = h.host AND s.port = h.port
13 ORDER BY h.inclusive_size_in_use DESC;

```

	SERVICE_NAME	PORT	HOST	CATEGORY	DEPTH	IN_USE_MB	INCLUSIVE_COUNT_IN_USE
1	indexserver	30052		/	0	1079351	1191490020
2	indexserver	30052		Pool	1	1065356	1164352413
3	indexserver	30052		Pool/ColumnStore	2	587219	176886468
4	indexserver	30052		Pool/ColumnStore/Main	3	570933	136345249
5	indexserver	30052		Pool/ColumnStore/Main/Dictionary	4	262481	69427673
6	indexserver	30052		Pool/ColumnStore/Main/Dictionary/RoDict	5	255778	69405548
7	indexserver	30052		Pool/PersistenceManager	2	235815	529760907
8	indexserver	30052		Pool/PersistenceManager/PersistentSpace	3	185792	24785570
9	indexserver	30052		Pool/PersistenceManager/PersistentSpace/Default	4	181711	22264304
10	indexserver	30052		Pool/PersistenceManager/PersistentSpace/Default	5	131918	8727319
11	indexserver	30052		Pool/L	2	124667	89731474
12	indexserver	30052		Pool/ColumnStore/Main/Index	4	120958	4154091
13	indexserver	30052		Pool/Ljit	3	89881	67027881

Figure 4.26 Heap Memory Allocators

It isn't necessary to know the functions of these allocators, but if you see one that apparently uses an abnormally large amount of memory, then it's best to do an SAP Note search using that allocator name as the search key. SAP Note 1999997 contains a table that lists known cases of excessive memory usage by specific allocators.

`M_HEAP_MEMORY` has an accompanying reset view, `M_HEAP_MEMORY_RESET`; a reset view lets you collect statistics relative to a chosen point in time instead of since startup. We describe reset views in Chapter 15. Usage history over a longer period (six weeks by default) is available in the `_SYS_STATISTICS.HOST_HEAP_ALLOCATORS` statistics server view (we see these views also in Chapter 15).

For shared memory, the monitoring view is `M_SHARED_MEMORY`. Figure 4.27 shows the shared memory statistics for the same SAP S/4HANA tenant. All shared memory areas in this database are owned by the index server (but are accessible to the other services). The `TABLE` category (which contains the row store) is by far the largest user of shared memory, but even then it's more than an order of magnitude smaller than the heap memory.

```

1) SELECT
2   s.service_name,
3   s.port,
4   m.host,
5   m.category,
6   to_integer(m.allocated_size / 1048576) AS allocated_mb,
7   to_integer(m.used_size / 1048576) AS used_mb,
8   to_integer(m.free_size / 1048576) AS free_mb
9 FROM
10  M_SHARED_MEMORY AS m
11 INNER JOIN m_services AS s ON s.host = m.host AND s.port = m.port
12 ORDER BY allocated_size DESC;
13

```

Result x Messages x History

Rows (7)	SERVICE_NAME	PORT	HOST	CATEGORY	ALLOCATED_MB	USED_MB	FREE_MB
1	indexserver	30052		TABLE	37509	25497	12011
2	indexserver	30052		CATALOG	1658	1588	70
3	indexserver	30052		PAGELIST	97	97	0
4	indexserver	30052		FREE	64	0	64
5	indexserver	30052		TOPOLOGY	3	0	3
6	indexserver	30052		INDEX	0	0	0
7	indexserver	30052		VERSION	0	0	0

Figure 4.27 Shared Memory

4.4.5 Memory for SQL Statements

Complex SQL statements, and especially multitable joins, sometimes do unexpected things. One of the more unpleasant of these is producing gigantic result sets because of some flaw in the join logic. Because result sets are memory objects, these out-of-control monsters can have a serious impact on the memory usage of the instance. Since version 1.0 SPS 08, it's therefore possible to set a limit on the amount of memory that SQL statements may use. There are two different limits, each controlled via an INI file parameter in the `[memorymanager]` section of the `global.ini` file. The first of these parameters sets the maximum memory for a single statement: `statement_memory_limit`.

The second parameter limits memory usage for all SQL statements combined: `total_statement_memory_limit`.

Neither parameter is set by default, which means that no limit on SQL statement memory applies.

Gigabytes, not Megabytes

Confusingly, the unit for these parameters is gigabytes, not megabytes like for the allocation limits.

Note that to make SAP HANA apply these limits, it's necessary to enable statement memory tracking in the `[resource_tracking]` section of the `global.ini` file using the following parameters:

- `enable_tracking = 'on'`
- `memory_tracking = 'on'`

With tracking enabled, the memory usage metrics are available in the `M_CONTEXT_MEMORY` monitoring view with reset companion `M_CONTEXT_MEMORY_RESET`.

Setting an absolute value as the memory limit for an SQL statement has a drawback. For example, suppose that the limit is set to 20 GB. Running a statement that potentially uses more than this amount of memory is certainly an issue when the system is very busy and the memory almost full, but running that same statement in the middle of the night with a terabyte of memory sitting unused shouldn't be a concern. To enable making a distinction between these two situations, SAP introduced an additional parameter: `statement_memory_limit_threshold` in the `[memorymanager]` section.

This parameter sets a percentage of the global allocation limit: only if the total used memory exceeds that percentage will the statement memory limit be honored. For example, if the global allocation limit is 500 GB and the threshold parameter is set to 80%, then SQL statements will be allowed to use more memory than the limit as long as the total used memory is less than 400 GB. Note that by default the parameter is set to 0, which means that the statement limit is applied unconditionally.

4.4.6 Host Memory Statistics

A final monitoring view we want to present is `M_HOST_RESOURCE_UTILIZATION`. This view contains capacity and resource usage information for the database host, as well as global memory usage data for the SAP HANA instance. Figure 4.28 shows some important metrics from this view. Here we can see that approximately 2,954 GB of physical memory was allocated, with 141 GB still free. The SAP HANA instance had allocated 3,094 GB, of which 1,396 GB was currently in use; peak usage since server start was 1,654 GB. It may seem surprising that the memory allocated by SAP HANA is greater than the



amount of physical memory in use, but this is normal because the relatively low usage rate of memory for SAP HANA means that not all its memory pages need to be present in main memory.

```

1 select to_integer(USED_PHYSICAL_MEMORY/(1024*1024)) as PHYS_USED_MB,
2        to_integer(FREE_PHYSICAL_MEMORY/(1024*1024)) as PHYS_FREE_MB,
3        to_integer(INSTANCE_TOTAL_MEMORY_ALLOCATED_SIZE/(1024*1024)) as INSTANCE_ALLOC_MB,
4        to_integer(INSTANCE_TOTAL_MEMORY_USED_SIZE/(1024*1024)) as INSTANCE_USED_MB,
5        to_integer(INSTANCE_TOTAL_MEMORY_PEAK_USED_SIZE/(1024*1024)) as INSTANCE_PEAK_USED_MB
6 from M_HOST_RESOURCE_UTILIZATION

```

PHYS_USED_MB	PHYS_FREE_MB	INSTANCE_ALLOC_MB	INSTANCE_USED_MB	INSTANCE_PEAK_USED_MB
2953988	141248	3094782	1395960	1653953

Figure 4.28 Memory Statistics for Host

Now we need to introduce two more terms (the last ones for now, we promise). First, a process running on Linux (and from the perspective of Linux, SAP HANA is nothing more than a bunch of processes) must reserve at the OS the memory that it needs for code, data, and other purposes (such as the pool, in the case of SAP HANA). The total size of these reservations is the *virtual memory* of that process (sometimes also called the virtual set size [VSS]). Second, not all of this needs to reside in the main memory of the server all the time. The part of the process's virtual memory that's effectively present in main memory at a given moment is the *resident memory* (or resident set size [RSS]) of that process. In the case shown in Figure 4.28, the virtual memory of the SAP HANA instance is 3,094 GB, but its resident memory (which the monitoring view doesn't show) is smaller. What we do know is that the combined resident memory of all processes on the server is 2,954 GB.

4.4.7 Monitoring Memory Usage

Now that we've introduced the main concepts of memory management in SAP HANA, let's look at the functions to obtain memory usage information in the various management tools and via SQL statements.

SAP HANA Cockpit

In the SAP HANA cockpit, you'll find the **Memory Management** card in the Database Overview app (Figure 4.29). On the card itself, you can toggle between **Used Memory** ❶ and **Resident Memory** ❷. Both views display a small histogram with the main categories; hovering the mouse over the histogram shows the corresponding numbers.

Click **Monitor Performance** ❸ to produce the more detailed report shown in Figure 4.30. The table on the left ❹ shows the main memory usage metrics for the server as a whole and for the index server of the current database (here, the SAP S/4HANA tenant). On

the right ❺ is a chart showing the evolution of the memory metrics over a time interval that you can configure at the top of the window. You can either enter a specific data range or use the **Presets** button ❻ to choose from a list of predefined intervals.

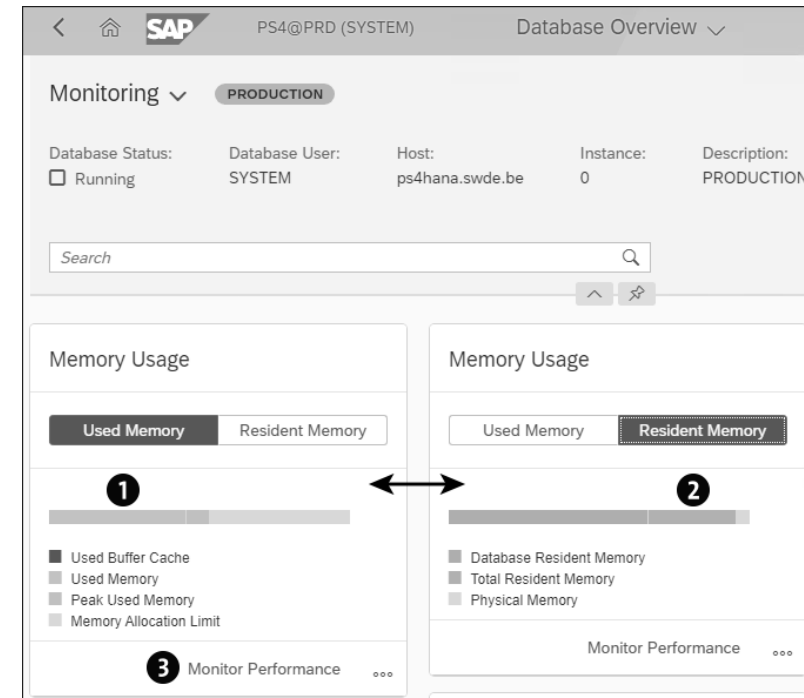


Figure 4.29 Memory Usage Card in SAP HANA Cockpit

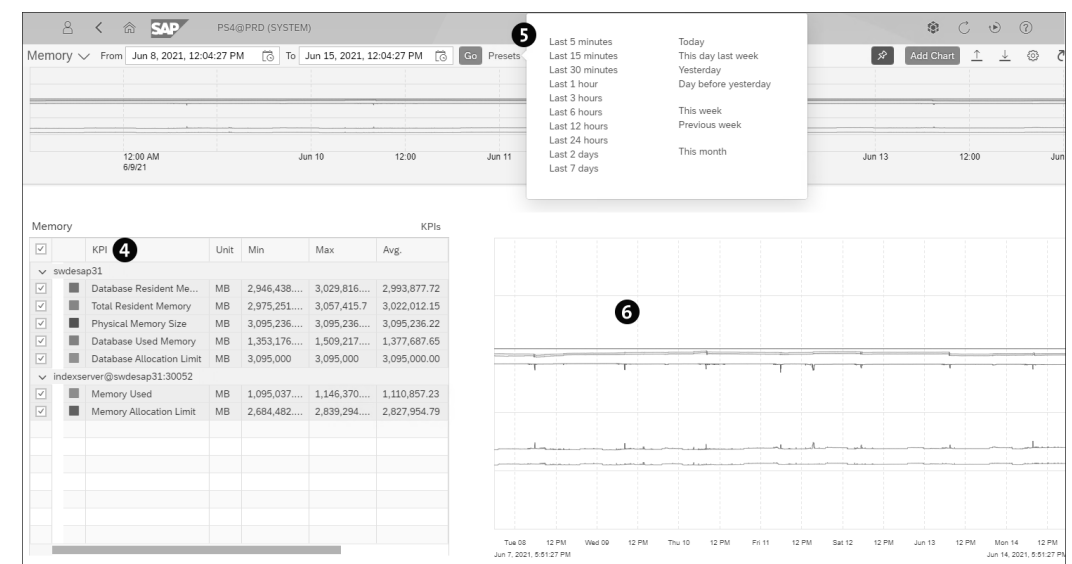


Figure 4.30 Memory Report in SAP HANA Cockpit

SAP HANA Studio

SAP HANA Studio shows summary memory statistics in the **Overview** tab of the Administration Console (Figure 4.31 ❶). The first histogram shows key metrics from the application perspective: the currently used memory, peak used memory (both are identical in the screenshot), and allocation limit. The second histogram gives a more server-oriented view, with the resident memory size of the current database, the total resident memory size, and the physical memory of the server. The instance in this example has two tenants of roughly equal size, which explains why the database resident memory is only a little over half the total resident size.

Click the **More Information** link below one of the histograms to open the **Landscape** tab ❷, which displays the resource usage per service for the current database. The memory histogram (enlarged, bottom of figure) shows bars with the current memory usage ❸ and peak memory usage ❹. The thin red line ❺ is the effective allocation limit for the service.

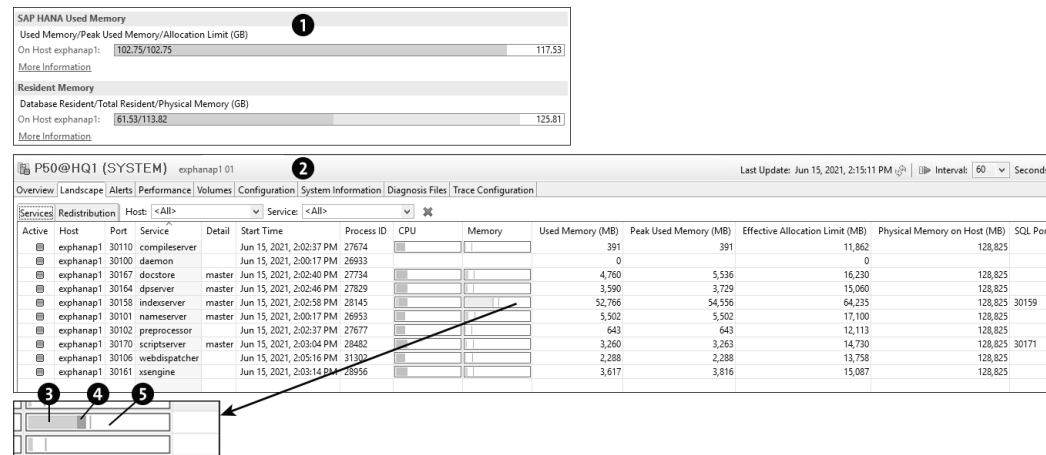


Figure 4.31 Memory Usage Statistics in SAP HANA Studio

DBA Cockpit

In SAP NetWeaver Application Server for ABAP and SAP S/4HANA systems, Transaction DBACOCKPIT shows summary metrics on the initial screen (see Figure 4.32). Under **Database Memory and CPU**, there are two histograms, but they have different meanings than those in SAP HANA Studio: the one at the top shows used and total memory for the entire multitenant database container (MDC); the bottom one shows this info only for the SAP tenant.

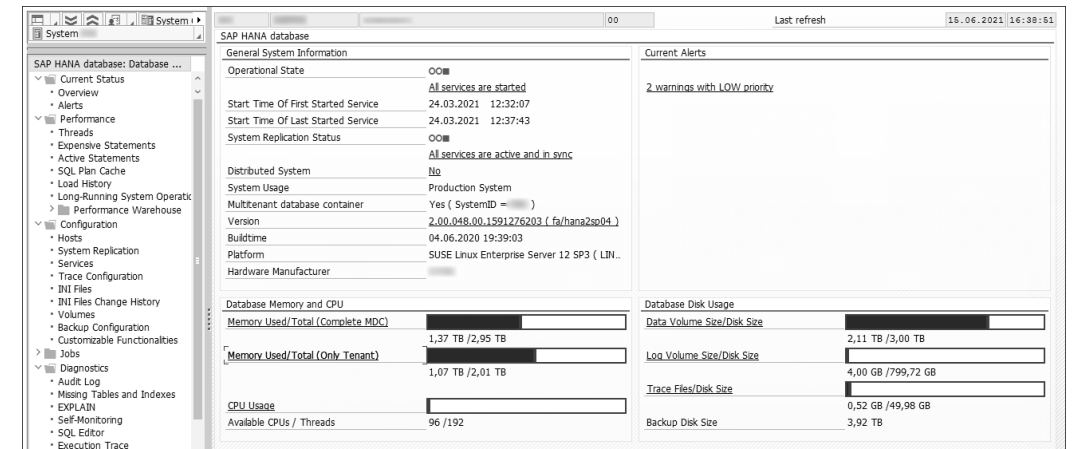


Figure 4.32 Memory Metrics in DBA Cockpit Overview

Click the title to the left of the histogram to open a detail screen (which is the same in both cases). This screen (see Figure 4.33) displays the memory statistics per service for the MDC. The list is so wide that we have split it into two parts. Notice that here you can see the sizes of the heap memory and shared memory (rightmost two columns of the lower section).

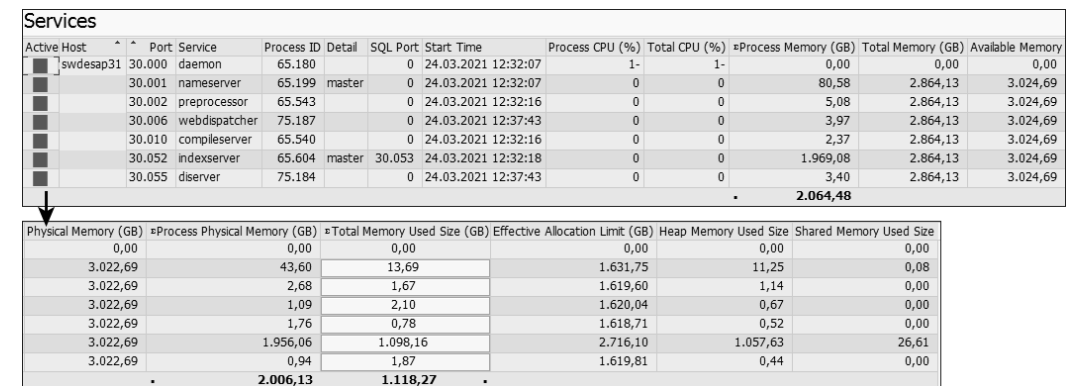


Figure 4.33 Resources per Service in DBA Cockpit

In contrast to the SAP HANA cockpit and SAP HANA Studio, DBA Cockpit allows you to drill down to a further level of detail. To do this, click an entry in the **Total Memory Used Size (GB)** column. For the selected service, this opens the screen shown in Figure 4.34. On the left, in tabular form and as a pie chart, you can see the main components that are using memory for the service. On the right is a detailed list with memory usage per allocator.

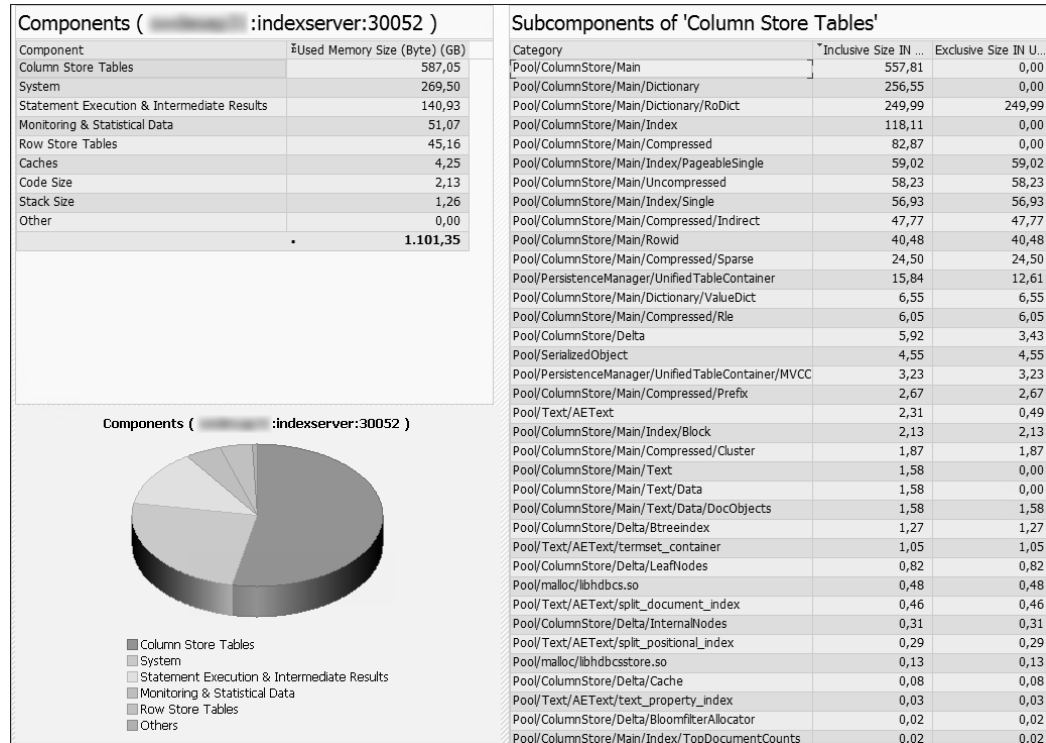


Figure 4.34 Memory Details in DBA Cockpit

SQL Statements

In Section 4.4.4, Section 4.4.5, and Section 4.4.6, we showed the monitoring and statistics views for SAP HANA memory. However, querying these views directly isn't always simple because in some cases, like, for example, the case of the allocator memory metrics in `M_HEAP_MEMORY`, the data is organized hierarchically and usage data is consolidated at higher levels. With a simple selection, there would be a risk of counting the same quantity several times over. To make life easier and to produce more reliable results, we advise you to use the standard scripts that SAP supplies with SQL statement collection. In Chapter 15, you'll learn where to find and how to install and use this collection. The relevant scripts have names starting with `HANA_Memory`; SAP Note 1999997 lists the most important of these.

4.5 Sessions and Transactions

All the work that application users do in the database takes place in the context of a *session*. To establish a session, the user connects via the database client software to the SAP HANA server, supplying connection parameters like the name of the tenant

database and valid credentials. After the parameters and credentials have been validated, the session is created; all the interaction between user and database then takes place in the context of this session. This interaction takes the form of a series of *transactions*. A transaction consists of one or more *commands*. A command is typically a SQL statement, but other types exist as well; we'll use *command* as a generic term for all requests the user sends to the database. The commands within a single transaction form a single, nondivisible (atomic) unit, which means that either all of them are executed or none of them are. A transaction is started implicitly when the first command is issued but must be ended explicitly with a commit or rollback.

Figure 4.35 schematically shows the lifetime of a database session. The session starts when the client connects to the database and a session context is set up. During the session, the client executes transactions consisting of commands; the session sends the command to the database server, which processes it and sends the result set, along with a return code (success or failure), back to the client. During the transaction, it will probably be necessary to lock areas of data against concurrent changes. These locks remain in force until the transaction is explicitly ended with a commit or a rollback. In case of a rollback, all changes made by the open transaction are undone. This process repeats until the user ends the session.

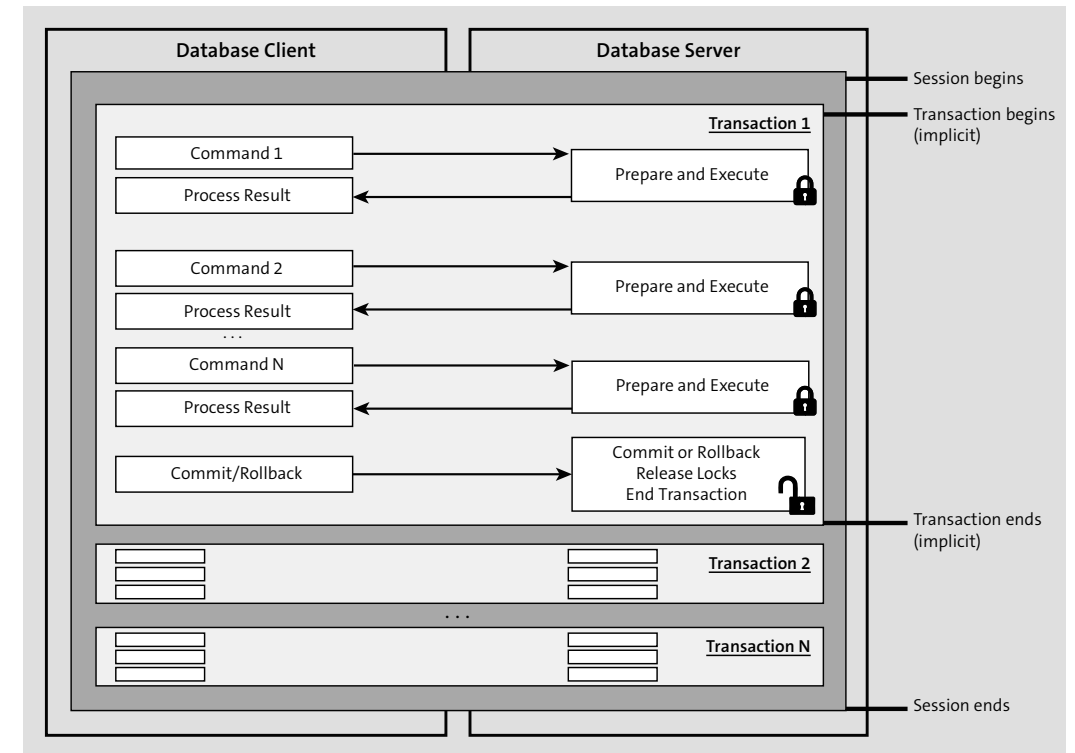


Figure 4.35 Sessions and Transactions



Autocommit Mode

A session may also operate in autocommit mode, which means that an implicit commit is executed after each command that makes a change in the database. While fine for simple tasks, autocommit is not suitable for more complex applications. The SQL console in the SAP HANA cockpit and SAP HANA Studio and the `hdbsql` command line tool work in autocommit mode by default, but this can be changed.

In Section 4.5.1 and Section 4.5.2, we'll explain how to monitor sessions and transactions. This is followed by a section about database locks.

4.5.1 Monitoring Sessions

You can find data about sessions in the `M_CONNECTIONS` and `M_CONNECTION_STATISTICS` monitoring views. These views provide a wealth of information and are extremely useful when you want to find out what's currently happening. Figure 4.36 shows an example in which we read some fields of `M_CONNECTIONS`.

```
select
connection_id,
to_char(start_time,
start_time,
connection_status,
client_host,
client_pid,
connection_type,
auto_commit,
user_name,
current_statement_id
from m_connections;
```

CONNECTION_ID	START_TIME	START_TIME	CONNECTION_STATUS	CLIENT_HOST	CLIENT_PID	CONNECTION_TYPE	AUTO_COMMIT	USER_NAME	CURRENT_STATEMENT_ID
210.276	2021-06-16 09:19:14	16 jun. 2021 09:19:14.09083	IDLE		0	Local	FALSE	SVS	
210.225	2021-06-16 09:04:32	16 jun. 2021 09:04:32.657793	RUNNING				FALSE	SYSTEM	902911830960247
210.182	2021-06-16 08:52:57	16 jun. 2021 08:52:57.154764	IDLE	hvehost	9.933	Remote	TRUE	TI2345	
210.157	2021-06-16 08:46:25	16 jun. 2021 08:46:25.564486	IDLE	hvehost	9.166	Remote	TRUE	SYSTEM	
210.151	2021-06-16 08:44:32	16 jun. 2021 08:44:32.223281	IDLE	expwvs124	13.416	Remote	TRUE	SYSTEM	
210.150	2021-06-16 08:44:32	16 jun. 2021 08:44:32.172053	IDLE	expwvs124	13.416	Remote	TRUE	SYSTEM	
210.143	2021-06-16 08:44:26	16 jun. 2021 08:44:26.545447	IDLE	expwvs124	13.416	Remote	FALSE	SYSTEM	
200.145	2021-06-14 10:34:16	14 jun. 2021 10:34:16.622504	IDLE		0	Local	FALSE	SYSTEM	
200.144	2021-06-14 10:34:16	14 jun. 2021 10:34:16.610502	IDLE		0	Local	FALSE	SYSTEM	
200.127	2021-06-14 10:29:13	14 jun. 2021 10:29:13.882511	IDLE		0	Local	FALSE	SVS	
200.068	2021-06-14 10:19:16	14 jun. 2021 10:19:16.618197	IDLE		0	Local	TRUE	_SYS_WOR...	
200.067	2021-06-14 10:19:16	14 jun. 2021 10:19:16.606971	IDLE		0	Local	TRUE	_SYS_WOR...	
-210.049	2021-06-16 08:19:14	16 jun. 2021 08:19:14.104703			0	History (local)	FALSE	SVS	
-210.256	2021-06-16 09:14:14	16 jun. 2021 09:14:14.248378			0	History (local)	FALSE	SYSTEM	
-210.257	2021-06-16 09:14:19	16 jun. 2021 09:14:19.222264			0	History (local)	FALSE	SYSTEM	

Figure 4.36 Monitoring View `M_CONNECTIONS`

Each connection is identified by a numeric `CONNECTION_ID` code. This ID is positive for existing connections and negative for connections that no longer exist but whose history and statistics are kept for a limited period of time. The `connection_history_lifetime` parameter in the `[session]` section of the `indexserver.ini` file controls the length of this period.

The default is 60 minutes. For active sessions (`ID > 0`), the view also shows the connection status, which in most cases will be `RUNNING` or `IDLE`. For all sessions, you can see the database username; for remote sessions (those connected via the database client), the view also shows the hostname and process ID from where the session was started. For

active sessions, `CURRENT_STATEMENT_ID` identifies the active SQL statement, the details of which can be found in the SQL plan cache (covered in Chapter 14).

Instead of using SQL statements, you can also use the standard functionality of the SAP GUI tools to display session information. In the SAP HANA cockpit, go to the **Sessions** card in the Database Overview and click the **Total** or **Running** link. Figure 4.37 shows the first page of the session list for an ABAP system. An interesting extra piece of information here is the **Application Source** column, which shows the location in the ABAP source code where the SQL statement originates.

Logical Connection ID	Created At	Seconds Since Last Statement Start	Connection Status	Transaction Status	Blocked By Connection ID	Application Source	Client Process ID
200100	2021-03-24 12:39:18.661423	8887	IDLE	ACTIVE	<not blocked>	SAPLRHDB:2735 More	17627
200135	2021-03-24 12:39:19.183305	1	IDLE	INACTIVE	<not blocked>	RSM13000:9896 More	17651
200141	2021-03-24 12:39:19.234474	1367	IDLE	INACTIVE	<not blocked>	SAPLBSVU:464 More	17656
200142	2021-03-24 12:39:19.242978	5599	IDLE	INACTIVE	<not blocked>	SAPLCOZV:12225 More	17653
200147	2021-03-24 12:39:19.300268	772	IDLE	INACTIVE	<not blocked>	SAPLSWOD:3222 More	17675
200149	2021-03-24 12:39:19.306975	13497	IDLE	INACTIVE	<not blocked>	SAPLHRAC:1155 More	17620
200155	2021-03-24 12:39:19.354888	0	IDLE	ACTIVE	<not blocked>	SAPLFBK3:14160 More	17617
200167	2021-03-24 12:39:19.490667	482	IDLE	INACTIVE	<not blocked>	SAPLARFC:1783 More	17652
200170	2021-03-24 12:39:19.498266	7	IDLE	INACTIVE	<not blocked>	SAPLSWOD:3222 More	17676
200188	2021-03-24 12:39:20.500791	2812	IDLE	ACTIVE	<not blocked>	SAPLBUSS:759 More	1388

Figure 4.37 Session List in SAP HANA Cockpit

In SAP HANA Studio, you find the same information in the Administration Console in the **Performance • Sessions** tab, and in DBA Cockpit, it's in **System Information • Connections**.

4.5.2 Monitoring Transactions

The monitoring view for transactions has the hardly surprising name `M_TRANSACTIONS`. Figure 4.38 shows the result of a query on this view. `CONNECTION_ID` identifies the session executing the transaction. `TRANSACTION_ID` is an identifier for the transaction object;

this number remains associated with the session for the entire lifetime for that session and can be reused afterward. If an open transaction has effectively made changes to the database that have not yet been committed or rolled back, then you will see a nonzero value in UPDATE_TRANSACTION_ID; this number lasts only until the end of the transaction and is always incremented and never reused. Other columns selected in the example are for the isolation level (which we discuss in Chapter 12) and the number of locks the transaction is currently holding.

```
select
  host,
  port,
  connection_id,
  transaction_id,
  update_transaction_id,
  transaction_type,
  transaction_status,
  start_time,
  end_time,
  isolation_level,
  acquired_lock_count
from m_transactions
order by connection_id
```

HOST	PORT	CONNECTION_ID	TRANSACTION_ID	UPDATE_TRANSACTION_ID	TRANSACTION_TYPE	TRANSACTION_STATUS	START_TIME	END_TIME	ISOLATION_LEVEL	ACQUIRED_LOCK_COUNT
hsehost	39.003	-1	0	0	VERSION GARBAGE COLLECTION TRANSACTION	INACTIVE	-	-	READ COMMITTED	0
hsehost	39.003	-1	1	0	INTERNAL TRANSACTION	INACTIVE	2021-06-14 10:24:02	2021-06-14 10:24:02	READ COMMITTED	0
hsehost	39.003	-1	10	0	EXTERNAL TRANSACTION	INACTIVE	2021-06-14 16:50:22	2021-06-14 16:50:22	READ COMMITTED	0
hsehost	39.003	-1	5	0	EXTERNAL TRANSACTION	INACTIVE	2021-06-16 11:10:40	2021-06-16 11:10:40	READ COMMITTED	0
hsehost	39.003	-1	6	0	EXTERNAL TRANSACTION	INACTIVE	2021-06-16 11:10:40	2021-06-16 11:10:40	READ COMMITTED	0
hsehost	39.003	-1	4	0	EXTERNAL TRANSACTION	INACTIVE	2021-06-16 11:10:40	2021-06-16 11:10:40	READ COMMITTED	0
hsehost	39.003	200.067	2	0	USER TRANSACTION	INACTIVE	2021-06-14 10:19:16	2021-06-14 10:19:16	READ COMMITTED	0
hsehost	39.003	200.068	3	0	USER TRANSACTION	INACTIVE	2021-06-14 10:19:16	2021-06-14 10:19:16	READ COMMITTED	0
hsehost	39.003	200.127	8	0	USER TRANSACTION	INACTIVE	2021-06-16 11:09:14	2021-06-16 11:09:14	READ COMMITTED	0
hsehost	39.003	200.144	9	0	USER TRANSACTION	INACTIVE	-	-	READ COMMITTED	0
hsehost	39.003	200.145	11	0	USER TRANSACTION	INACTIVE	-	-	READ COMMITTED	0
hsehost	39.003	210.143	13	0	USER TRANSACTION	INACTIVE	2021-06-16 09:36:07	2021-06-16 09:36:08	READ COMMITTED	0
hsehost	39.003	210.150	14	0	USER TRANSACTION	INACTIVE	2021-06-16 08:44:32	2021-06-16 08:44:32	READ COMMITTED	0
hsehost	39.003	210.151	15	0	USER TRANSACTION	INACTIVE	2021-06-16 08:44:32	2021-06-16 08:44:32	READ COMMITTED	0
hsehost	39.003	210.157	12	821.792	USER TRANSACTION	ACTIVE	2021-06-16 11:10:36	-	READ COMMITTED	1
hsehost	39.003	210.182	17	821.789	USER TRANSACTION	ACTIVE	2021-06-16 11:03:47	-	READ COMMITTED	6
hsehost	39.003	210.225	16	0	USER TRANSACTION	ACTIVE	2021-06-16 09:08:38	-	READ COMMITTED	0
hsehost	39.003	210.495	7	0	USER TRANSACTION	INACTIVE	2021-06-16 11:09:14	2021-06-16 11:09:14	READ COMMITTED	0
hsehost	39.003	210.634	18	0	USER TRANSACTION	ACTIVE	2021-06-16 11:12:38	-	READ COMMITTED	0

Figure 4.38 Monitoring View M_TRANSACTION

To show the interplay between the different transaction fields, let's walk through a small demonstration in hdbsql. We'll connect to the database, switch off autocommit, and determine our connection ID (the system provides a CURRENT_CONNECTION variable for this purpose), as shown in Listing 4.11.

```
hdbsql -i <nr> -d <DB> -u <user> -p <password> -m
\a off
Auto commit mode switched OFF
select current_connection from dummy;
| CURRENT_CONNECTION |
| ----- |
| 210736 |
```

Listing 4.11 Connection ID in hdbsql

A transaction is implicitly active, and our session has been allocated a transaction object with ID = 17, as shown in Listing 4.12.

```
select * from icecream;
| ID          | FLAVOR      | PRICE      |
| ----- | ----- | ----- |
```

```
| 1 | Vanilla | 4.00 |
| 2 | Strawberry | 4.75 |
| 3 | Chocolate | 4.25 |
```

```
select transaction_id, update_transaction_id,
  transaction_status as status, acquired_lock_count
from m_transactions where connection_id = 210736;
| TRANSACTION_ID | UPDATE_TRANSACTION_ID | STATUS | ACQUIRED_LOCK_COUNT |
| ----- | ----- | ----- | ----- |
| 17 | 0 | ACTIVE | 0 |
```

Listing 4.12 Transaction ID in hdbsql

We only queried the database and didn't make any changes; the state of the transaction remains unchanged. Now we change a nonexistent row, as shown in Listing 4.13.

```
update icecream set price = 5.00 where flavor = 'Pistachio';
0 rows affected
select transaction_id, update_transaction_id,
  transaction_status as status, acquired_lock_count
from m_transactions where connection_id = 210736;
| TRANSACTION_ID | UPDATE_TRANSACTION_ID | STATUS | ACQUIRED_LOCK_COUNT |
| ----- | ----- | ----- | ----- |
| 17 | 821827 | ACTIVE | 0 |
```

Listing 4.13 Update Transaction ID Allocated

Because of the update statement, an update transaction ID has been assigned. Because the statement didn't actually change any data, the transaction isn't yet holding any locks. Now let's make a real change, as shown in Listing 4.14.

```
update icecream set price = price + 0.25;
3 rows affected
select transaction_id, update_transaction_id,
  transaction_status as status, acquired_lock_count
from m_transactions where connection_id = 210736;
| TRANSACTION_ID | UPDATE_TRANSACTION_ID | STATUS | ACQUIRED_LOCK_COUNT |
| ----- | ----- | ----- | ----- |
| 17 | 821827 | ACTIVE | 3 |
```

Listing 4.14 Transaction Holding Locks

The transaction now holds a row lock on the three modified rows. The last step is to commit the changes, as shown in Listing 4.15.

```

commit;
select transaction_id, update_transaction_id,
       transaction_status as status, acquired_lock_count
   from m_transactions where connection_id = 210736;

```

TRANSACTION_ID	UPDATE_TRANSACTION_ID	STATUS	ACQUIRED_LOCK_COUNT
-----	-----	-----	-----
17	0	ACTIVE	0

Listing 4.15 Status after Committing Transaction

The transaction ID remains unchanged, and the update transaction ID is zero again. All locks have been released.

4.5.3 Monitoring Locks

Locks are a crucial mechanism in every DBMS to manage concurrent access. Locks ensure that a transaction can never change data that's already being changed by another transaction. In this section, we begin by describing the lock types (the type indicates whether the lock applies to a row or an entire table), the lock modes (a lock can be exclusive or intentional), and the way these two attributes combine. After that, we discuss the monitoring views that let you monitor locking activity in the database.

Lock Types and Lock Modes

SAP HANA uses two lock types, distinguished by the scope: table locks and row locks. With respect to the impact of the lock on concurrent transactions, there are two lock modes:

- Exclusive (X) locks restrict the right to change the locked row or table to the transaction holding the lock. Other transactions can read the table or row but can't change it.
- Intentional exclusive (IX) locks, which only exist for tables, prevent other transactions from setting an exclusive lock on the same table. When a transaction executes a SQL statement that changes data, it will lock all the rows that it changes, as we saw in the example in Section 4.5.2. This row-level lock is always exclusive, and other transactions can't lock these rows. However, locking purely at the row level isn't watertight: another transaction could still set an exclusive lock on the entire table, causing a conflict. To prevent this, before the first transaction changes any rows, it sets an IX lock at the table level. This guarantees that no one else can lock the table exclusively.

Figure 4.39 shows the interaction between the two lock types in a multiconcurrency situation. Transaction 1 wants to update rows on table SALESDATA. The first step is to set an IX lock on the table. The WHERE clause of the UPDATE statement finds three matching rows, and an X lock is set on each of these rows. While transaction 1 is active,

transaction 2 executes an UPDATE on the same table. It also sets an IX lock on the table and then scans for rows that match the WHERE condition. The two matching rows do not conflict with those updated by transaction 1, and an X lock is set on these rows. Transaction 3 wants to create a new index on table SALESDATA. DDL statements like CREATE INDEX require an exclusive lock on the table. The request to set an X lock on SALESDATA is rejected because of the active IX locks, and transaction 3 is made to wait. Only when transactions 1 and 2 are both finished and their locks released will the index creation be allowed to proceed.

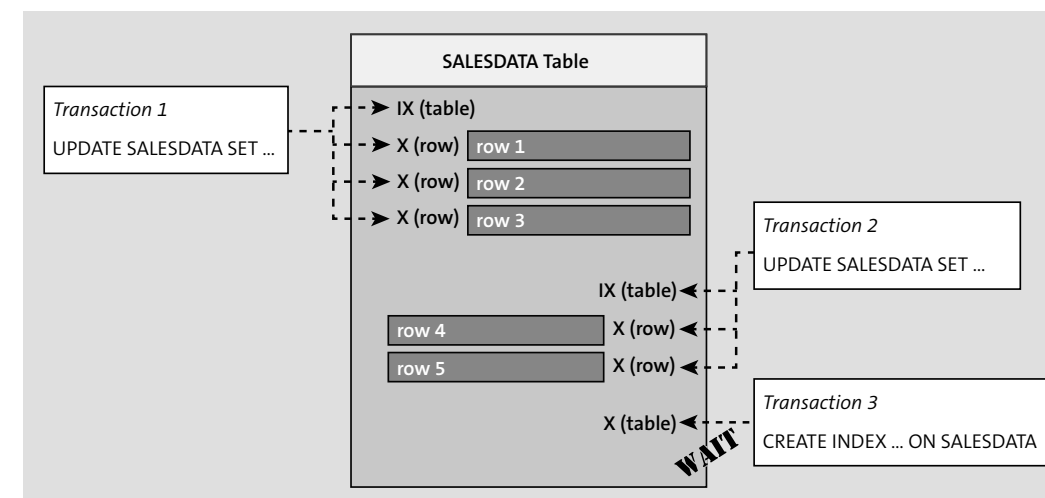


Figure 4.39 Exclusive and Intentional Exclusive Locks

Table 4.6 lists the combinations of lock types and lock modes, the circumstances when they are used, and the corresponding SQL statements.

Lock Type	Lock Mode	Use Cases
Table	X (exclusive)	<ul style="list-style-type: none"> ■ DDL statements ■ Truncation ■ With explicit table lock statements: ALTER TABLE, DROP TABLE, CREATE INDEX, DROP INDEX, TRUNCATE TABLE, LOCK TABLE
Table	IX (intentional exclusive)	<ul style="list-style-type: none"> ■ DML statements that change data ■ Delta merges ■ With explicit table lock statements: INSERT, UPDATE, UPSERT, DELETE, SELECT FOR UPDATE, MERGE DELTA, LOCK TABLE
Row	X (exclusive)	With DML statements that change data: INSERT, UPDATE, UPSERT, DELETE, SELECT FOR UPDATE

Table 4.6 Lock Types and Modes

Table locks can also be set explicitly using the `LOCK TABLE SQL` statement. This statement can be used to set both X and IX locks, as in the following examples:

- `LOCK TABLE SALESDATA IN EXCLUSIVE MODE`
- `LOCK TABLE SALESDATA IN INTENTIONAL EXCLUSIVE MODE`
- `LOCK TABLE SALESDATA IN EXCLUSIVE MODE NOWAIT`
- `LOCK TABLE SALESDATA IN EXCLUSIVE MODE WAIT 300`

The third and fourth statements specify the wait mode: `NOWAIT` means that the statement will fail immediately if the lock cannot be acquired, and `WAIT` specifies how many seconds to wait for the lock request before timing out.

Lock Waits and Timeouts

If a lock request can't be granted immediately, then the transaction will wait until the request either is granted or expires. The default behavior is controlled via the `lock_wait_timeout` parameter in the `[transaction]` section of the `indexserver.ini` file.

The default is 1800000, which is equal to 30 minutes. Careful: the unit for this parameter is milliseconds, whereas in `LOCK TABLE` it's seconds. Setting the timeout to 0 means that lock requests will wait indefinitely, meaning, in practice, until the transaction is canceled.

When a lock request, implicit or explicit, hits the timeout value, the following error is produced and the open transaction is rolled back:

```
Error: (dberror) [131]: transaction rolled back by lock wait timeout:
TrexColumnUpdate failed on table '<SCHEMA>'. '<TABLE>' with error: transaction
rolled back by lock wait timeout: Lock-wait time out exceeded
```

Monitoring Views for Locks

The monitoring views for locks are `M_OBJECT_LOCKS` and `M_RECORD_LOCKS` for table and row locks, respectively. In both views, the owner of the lock can be identified based on the update transaction ID, from which it's possible to derive the session.

4.6 Load Management and Distribution

Memory isn't the only critical resource in an SAP HANA installation. There wouldn't be much point in loading huge amounts of data into memory if that data couldn't then be processed efficiently. With that goal, your big strapping database server will come not just with a huge memory but also with an impressive battery of CPUs, and SAP HANA is designed to profit as much as possible from this CPU capacity, in terms of both speed and number.

When it comes to optimizing the distribution of the workload, an administrator can choose to go with the defaults and let SAP HANA figure things out by itself. In an

environment with a relatively homogeneous workload, that can be a viable option because SAP HANA derives many of its default settings from the actual characteristics of the server. Often, however, the workload is not homogeneous at all. A good example is a setup that runs both transactional (OLTP) and analytical (OLAP) loads. On the OLAP side, there is a further distinction between queries and extract, transform, and load (ETL) operations, both of which have substantial, and not always compatible, resource requirements. Another example is an SAP HANA system serving multiple tenants, where you must prevent one database from monopolizing the resources of the system at the expense of the others.

In this section, we cover several features and mechanisms to control how the workload is distributed. With admission control (Section 4.6.1), new requests can be queued or even rejected when the server is running near its maximum capacity. By configuring processor affinity (Section 4.6.2), a process can be bound to a processor or group of processors. It's possible to control to what degree SQL statements can execute in parallel (Section 4.6.3). You also can define workload classes (Section 4.6.4) and assign specific activities to these classes. Resource restrictions also can be set for individual users (Section 4.6.5). Finally, we look at the available metadata for load management in Section 4.6.6.

4.6.1 Admission Control

Admission control is a feature that makes it possible to control how new requests are handled when the system is at or near peak capacity of either CPU or memory. It uses two thresholds, soft and hard:

1. When the soft threshold is reached, new requests are queued rather than scheduled immediately. A request remains in the queue until sufficient capacity is available or until it times out in the queue, in which case it is rejected.
2. When the instance hits the hard threshold, new requests are rejected. The following error is returned:

```
1038: rejected as server is temporarily overloaded
```

The thresholds are controlled using parameters: one global on/off, two for the CPU, and two for memory. The global switch is set via the `enable = 'true' | 'false'` parameter found in the `[admission_control]` section of the `indexserver.ini` file.

Don't Turn Off Admission Control

Turning off admission control can be tempting when users complain about waiting or rejected commands, but this makes the problem worse and causes even more severe performance problems.

Soft and hard thresholds for the CPU are handled via the `queue_cpu_threshold` and `reject_cpu_threshold` parameters in the `[admission_control]` section of the `indexserver.ini` file.

Similarly for memory, they are handled via the `queue_memory_threshold` and `reject_memory_threshold` parameters.

All thresholds are expressed as a percentage. The default for the soft threshold is 90, which means that queuing will begin when 90% of the resource capacity is used. A value of 0 or 100 means that no queuing takes place. The default for the hard threshold is 0; this implies that normally no requests will be rejected.



Soft Threshold for Memory

SAP recommends setting the soft threshold for memory (not CPU!) to 0. This is because high memory consumption is rarely a cause for concern. As of version 2.00.054, 0 is the default value, meaning that admission control becomes a function of CPU usage only. See SAP Notes 2222250 and 2600030.

The instance measures CPU and memory utilization at fixed intervals; the value used is normally not the latest one but a smoothed moving average. The polling interval and the degree of smoothing are again controlled with the `statistics_collection_interval` parameter in the `[admission_control]` section of the `indexserver.ini` file.

The unit is milliseconds and the default is 1000 (one second). With this frequency, the performance impact of the check is negligible. It's possible to set a shorter interval, down to 100 ms.

In the same section, the smoothing factor can be controlled using these three parameters:

- `averaging_factor`
- `averaging_factor_cpu`
- `averaging_factor_memory`

The first parameter sets the overall smoothing factor; it's possible to deviate from this by setting a specific factor for CPU or memory using the second and third parameters, respectively. The unit is always percent. The lower the percentage, the more smoothing is applied. An averaging factor of 100 suppresses smoothing, and only the last capacity measurement is then considered. The overall averaging factor has a default of 70; the resource-specific factors are 0 by default, which means the global factor is used.

There are numerous parameters for the management of the request queue. They are documented in the *SAP HANA Administration Guide* and in SAP Note 2222250. We won't describe these parameters here—with two important exceptions. The `queue_timeout_check_interval` parameter defines how long a request can remain on the queue before being finally rejected. The unit is seconds, and the default is 600 (10 minutes).

Use the following parameter to set the maximum length of the queue: `max_queue_size`. When the queue is full, new requests are rejected. The default queue length is 10,000.

There are three monitoring views that provide useful data about admission control, which are described in Table 4.7.

Monitoring View	Description
M_ADMISSION_CONTROL_EVENTS	Event log for all recent admission control events. An event is only recorded if the request has been on the queue for five seconds at least (this time is configurable; see SAP Note 2222250). The maximum number of logged events is also configurable (see same note). It used to be one million but has been lowered to 10,000.
M_ADMISSION_CONTROL_QUEUES	Information about currently queued requests.
M_ADMISSION_CONTROL_STATISTICS	Admission control statistics. In this view you can, for example, find the latest CPU and memory measurements, the current queue size, and the average wait time of requests currently in the queue.

Table 4.7 Monitoring Views for Admission Control

Instead of querying the monitoring views, you can also monitor, as well as manage, the admission control settings in the SAP HANA cockpit (not in SAP HANA Studio). This is shown in Figure 4.40. In the Database Overview app, in the **Monitoring** view, locate the **Admission Control** card and choose **Monitor**. The monitor window shows a graph ❶ for the last 60 minutes at 10-second intervals (use the horizontal scroll bar to move through the displayed data series). Summary data can be found in the table ❷ below the graph. Via **Manage Admission Control** ❸, you can change the settings online.

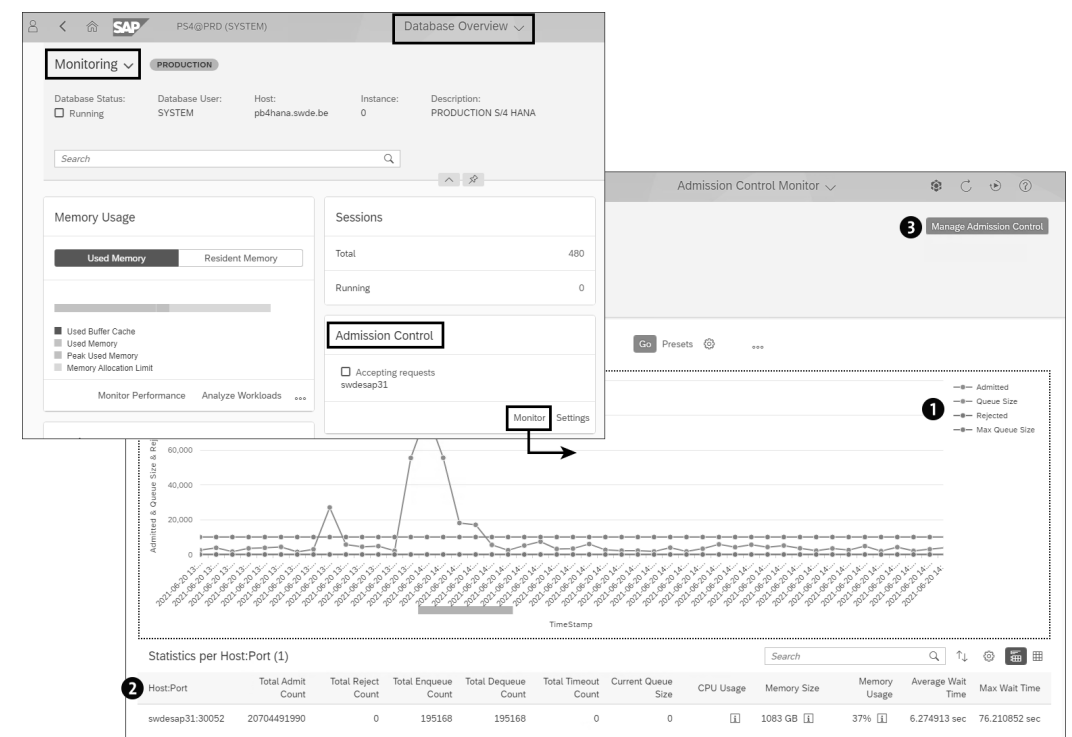


Figure 4.40 Admission Control Monitor in SAP HANA Cockpit

Figure 4.41 shows the screen for the admission control settings. The **Enable admission control...** checkbox ❶ controls the global on/off switch. Initially, only the two CPU and two memory thresholds are displayed ❷. Click **Advanced Options** ❸ to show an additional set of parameters; here you can change things like the averaging factor and collection interval, as well as various queue parameters (although not the maximum time a request can spend in the queue before it's rejected). Choose **Save** ❹ to make the new settings operational.

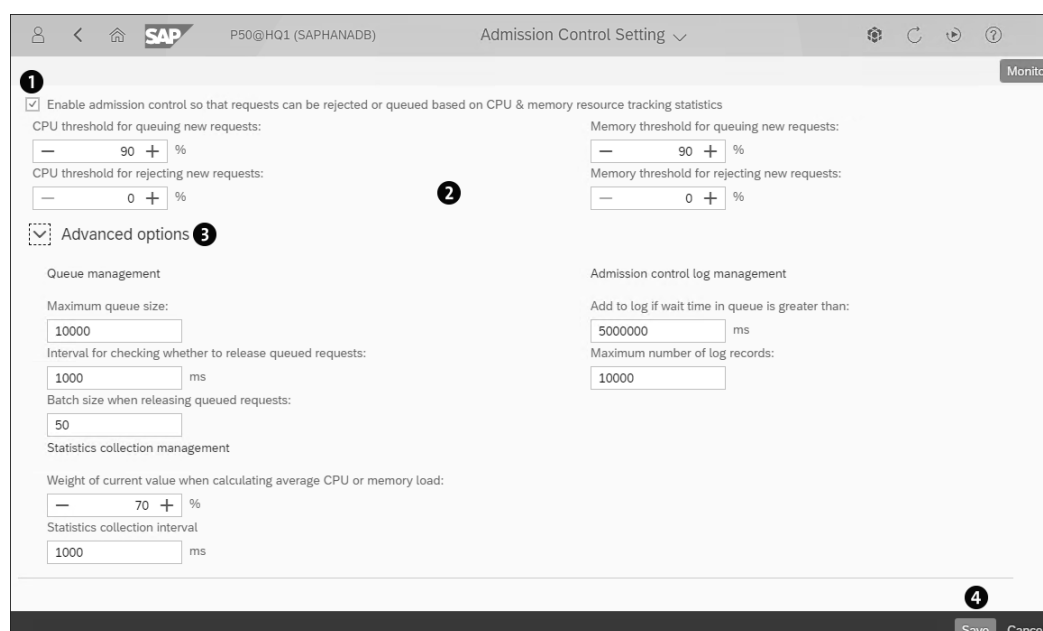


Figure 4.41 Admission Control Settings in SAP HANA Cockpit

4.6.2 Processor Affinity

The term *processor affinity* means binding a process to a specific CPU or group of CPUs. Normally, all the CPUs of a server form one pool and any process or thread can run on any CPU. While this is generally good to achieve a balanced use of the processor capacity, there are circumstances when it can be advantageous to take firmer control of the CPU scheduling process. For example, processors have a local cache memory that preserves data about the processes that previously ran on them. When the same process runs again on the same CPU, the presence of the data in the cache will speed up execution. If any process on the server can use that CPU, the probability of an (expensive) cache miss increases. The advantage of process affinity becomes even more significant with nonuniform memory access (NUMA) processors; a NUMA processor has a local memory of its own, which it can access faster than the memory shared among all the processors. In the next two sections, we first see how to determine the processor topology and then we explain how processor affinity can be configured.

Processor Topology

Before considering affinity, it's necessary to know what number and type of processors the server has and how they are identified (the processor topology). A good way to find out is to run the `lscpu` Linux command, which displays all relevant information about the processors. Figure 4.42 shows an example of the output. Here you can see the following elements:

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit ❶
Byte Order:            Little Endian ❶
Address sizes:         46 bits physical, 48 bits virtual
CPU(s):                112 ❷
On-line CPU(s) list:   0-111 ❷
Thread(s) per core:    2
Core(s) per socket:    28 } ❸
Socket(s):              2 }
NUMA node(s):          2 }
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 85
Model name:             Intel(R) Xeon(R) Platinum 8280M CPU @ 2.70GHz
Stepping:              7
CPU MHz:               2700.000
CPU max MHz:           4000.0000
CPU min MHz:           1000.0000
BogoMIPS:              5400.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              39424K
NUMA node0 CPU(s):    0-27,56-83 ❹
NUMA node1 CPU(s):    28-55,84-111
```

Figure 4.42 CPU Topology with `lscpu` Command

- ❶ The processor architecture is `x86_64` with little-endian bit order (that last fact is logical because SAP HANA 2.0 no longer supports big-endian).
- ❷ There are 112 logical CPUs, numbered from 0 to 111.
- ❸ There are two sockets, each with 28 processor cores. The cores are multithreaded with two threads per core; the total number of threads must of course match the number of logical processors ($2 \times 2 \times 28 = 112$). There are two NUMA nodes.
- ❹ The NUMA nodes each contain half of the number of logical processors. The first node includes processors 0–27 and 56–83; the second node consists of processors 28–55 and 84–111.

Configuring Affinity

Processor affinity for a service process is configured by setting a parameter called `affinity` in the section of that service in the `daemon.ini` file. Affinity can be set for the following services, where the first three run only in the system database and last two in the tenant databases.

- nameserver
- compileserver
- preprocessor
- indexserver
- xsengine

Suppose that the SAP HANA instance on the 112 logical CPU server shown in Figure 4.42 has two tenant databases, PR1 and PR2. PR1 is *used somewhat* more heavily than PR2, and we decide to set up affinity based on the distribution shown in Table 4.8.

Service	No. of CPUs	Assigned CPUs
nameserver	2	0 and 56 (NUMA group 1)
compileserver preprocessor	6	1–3 and 57–59 (NUMA group 1)
indexserver PR1 xsengine PR1	56	28–55 and 84–111 (complete NUMA group 2)
indexserver PR2 xsengine PR2	48	4–27 and 60–83 (remainder of NUMA group 1)

Table 4.8 Example of Processor Affinity and NUMA Groups

Listing 4.16 shows the corresponding SQL commands; you must be in the system database to execute them. Notice the use of the `.SID` suffix to refer to the service in a specific tenant.

```
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('nameserver', 'affinity') = '0,56';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('compileserver', 'affinity') = '1-3,57-59';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('preprocessor', 'affinity') = '1-3,57-59';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('indexserver.PR1', 'affinity') = '28-55,84-111';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('xsengine.PR1', 'affinity') = '28-55,84-111';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('indexserver.PR2', 'affinity') = '4-27,60-83';
alter system alter configuration ('daemon.ini','SYSTEM')
  SET ('xsengine.PR2', 'affinity') = '4-27,60-83';
```

Listing 4.16 Setting Processor Affinity

Restart Needed to Enable Affinity Settings

The affinity settings are not dynamic; they only take effect when the processes are restarted.



4.6.3 Parallel Execution of SQL Statements

When an SQL statement is submitted to SAP HANA, that statement can be picked up by two types of thread:

- An `SqlExecutor` thread handles incoming requests and executes simple SQL statements, defined as statements with short run times and limited resource usage, common in OLTP applications. When such a statement is received, it's picked up by a single `SqlExecutor` thread from a precreated thread pool.
- For complex operations (OLAP but also sometimes OLTP), the request is passed to the `JobExecutor`, which manages a pool of `JobWorker` threads. Rather than scheduling one thread, the `JobExecutor` will assign the task to multiple `JobWorkers`.

Figure 4.43 shows this schematically. Both incoming requests are handled by an `SqlExecutor`. If it's a simple statement, the `SqlExecutor` thread executes the statement itself. A complex statement, on the other hand, is passed on to the `JobExecutor`, which launches a series of `JobWorkers` that process the statement in parallel.

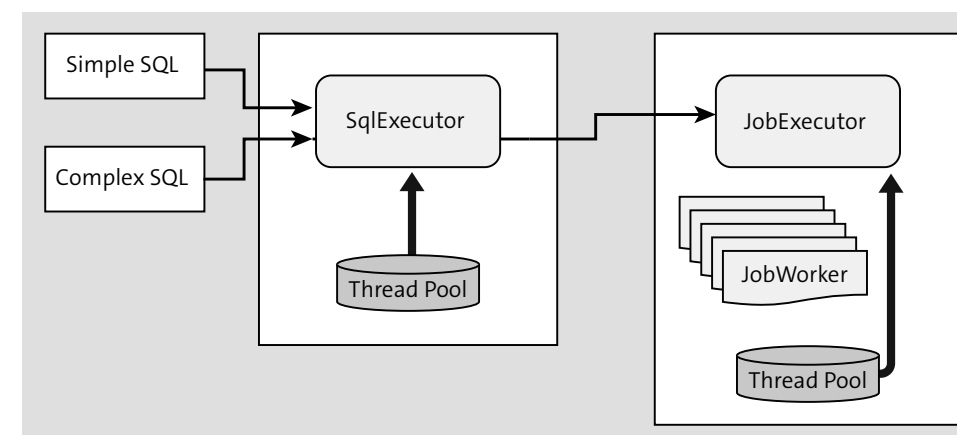


Figure 4.43 `SqlExecutor` and `JobExecutor`

You can influence parallelism by controlling the size of the two thread pools. For instance, in a system with a heterogeneous workload comprising both OLAP and OLTP, you might find that the OLAP load tends to dominate the system, starving the OLTP load of resources. To remedy this imbalance, you could reduce the size of the `JobExecutor` thread pool.

As you probably expect, thread pools are managed using parameters. By default none of these parameters are set (but see the note below referring to SAP Note 3011056), which means that either pool is allowed to use as many processors as it needs. For the `SqlExecutor`, you can set following parameters in the `[sql]` section of the `indexserver.ini` file; the value of both is the number of logical CPUs:

- `sql_executors`
- `max_sql_executors`

The first parameter sets a soft limit: the instance precreates this number of `SqlExecutor` threads in the thread pool. If demand exceeds the size of the pool, then extra threads are created and deleted as needed. If the parameter is not set (the default situation), then the number of threads in the pool is equal to the number of logical CPUs. While setting a smaller pool will not directly affect parallelism because extra threads can be created, it does reduce the overhead (e.g., memory use) of the idle threads.

The second parameter is a hard limit: the number of `SqlExecutor` threads can't exceed this value. Incoming requests for which no thread is available are rejected.

The parameters for the `JobExecutor` pool can be set in `global.ini` or at tenant level in `indexserver.ini` in the `[execution]` section. There are three of them, and like the `SqlExecutor` parameters, the value represents a number of logical processors:

- `max_concurrency`
- `max_concurrency_hint`
- `default_statement_concurrency_limit`

The first parameter sets the soft limit for the size of the thread pool. If the `JobExecutor` receives additional requests beyond the capacity of the pool, it creates and delete extra threads as needed. There is no hard limit for the `JobExecutor`; this is because its threads are also used for other operations, such as garbage collection, and forbidding the creation of extra threads might cause the instance to malfunction.

The second parameter defines the number of `JobWorker` threads to handle one parallelized operation. Note that this is the number per operation and not per statement; a single SQL statement may consist of multiple operations.

The third parameter is the maximum number of parallel threads for a single database request. If set, it should have a value equal to or greater than the per-operation limit set with `max_concurrency_hint`.



Default Value of `default_statement_concurrency_limit`

As of version 2.00.056, the default for the `default_statement_concurrency_limit` parameter is no longer 0 but 25% of the number of logical processors (if at least 16 logical CPUs are available). See SAP Note 3011356 for details. This stricter setting prevents highly parallelized statements from consuming all available CPU resources.

4.6.4 Workload Classes

Workload classes are a means to limit the resources that a related group of activities in SAP HANA are able to use at any given time. When using workload classes, the first step is to create classes and set resource limits for them; the second step is to link database activities to these classes using workload mappings.

Creating Workload Classes

To create a workload class, you use the `CREATE WORKLOAD CLASS SQL` statement. When you create a workload class, you must assign properties to it. There are eight properties, which we list in Table 4.9.

Property	Description
STATEMENT THREAD LIMIT	The maximum number of <code>JobWorker</code> threads that can work on a statement in parallel. A value of 0 removes any limit on parallelism. If set, the value must be between 1 and the number of logical CPUs. If the parameter is not set for the workload class, the value of the <code>default_statement_concurrency_limit</code> parameter (Section 4.6.3) is used.
STATEMENT MEMORY LIMIT	Maximum amount of memory a statement can use; the unit is gigabytes.
TOTAL STATEMENT THREAD LIMIT	Instead of setting a limit per statement, you can also set a global limit on all statements in the class. This creates an application-level workload class as opposed to a statement-level class. Using <code>STATEMENT THREAD LIMIT</code> and <code>TOTAL STATEMENT THREAD LIMIT</code> are mutually exclusive; you can only set one or the other. If you set this property, you must also set <code>PRIORITY</code> and <code>TOTAL STATEMENT MEMORY LIMIT</code> .
TOTAL STATEMENT MEMORY LIMIT	Memory limit for an application-class group. The same considerations apply as for <code>TOTAL STATEMENT THREAD LIMIT</code> .
PRIORITY	Priority given to the execution of the statement. The priority goes from 0 (lowest) to 9 (highest). The default is 5. Specifying this property is mandatory for an application-class group.
STATEMENT TIMEOUT	<i>Statement</i> here specifically means <i>query</i> . This parameter sets the maximum runtime for a query. If the query hasn't completed within the timeout interval, it is aborted. The unit is seconds and the default is 0 (no timeout).
WRITE TRANSACTION LIFETIME	Maximum runtime for uncommitted write transactions. After the specified time, the system will terminate the connection. Unlike the statement timeout, the unit here is minutes. The default is 0 (no limit). The system only checks for write transaction exceeding the lifetime limit every hour; therefore a transaction may run for up to 60 minutes longer than the value set in the parameter.

Table 4.9 Workload Class Properties

Property	Description
IDLE_CURSOR_LIFETIME	Maximum lifetime for an idle database cursor. This parameter is also specified in minutes and the default is 0 (no limit). Like write transactions, the actual cursor lifetime may exceed the parameter setting by up to 60 minutes.

Table 4.9 Workload Class Properties (Cont.)

Let's go through a few examples. First, we create a workload class simply for requests with the highest priority:

```
create workload class HIGH_PRIORITY set 'PRIORITY' = '9';
```

A statement-level class with limits for parallel threads and memory usage per statement and a query timeout is set in Listing 4.17.

```
create workload class myclass1
  set 'STATEMENT_THREAD_LIMIT' = '40',
     'STATEMENT_MEMORY_LIMIT' = '16',
     'STATEMENT_TIMEOUT' = '3600';
```

Listing 4.17 Create Workload Class

An application-level class with overall limits for parallel threads and memory usage is set in Listing 4.18.

```
create workload class appl_cls1
  set 'TOTAL_STATEMENT_THREAD_LIMIT' = '64',
     'TOTAL_STATEMENT_MEMORY_LIMIT' = '100';
```

Error: (dberror) [8]: invalid argument: application-level workload class always has all properties

-- For application-level class we must also specify PRIORITY

```
create workload class appl_cls1
  set 'TOTAL_STATEMENT_THREAD_LIMIT' = '64',
     'TOTAL_STATEMENT_MEMORY_LIMIT' = '100',
     'PRIORITY' = '5';
```

Listing 4.18 Application-Level Workload Class

Changing and Dropping Workload Classes

You can use ALTER WORKLOAD CLASS to change properties of the class or to enable/disable the class. First, let's add a write transaction timeout of two hours to the statement-level class:

```
alter workload class myclass1 set 'WRITE_TRANSACTION_LIFETIME' = '120';
```

Then, let's decide to shorten the write transaction lifetime to 60 minutes and to remove the statement thread and memory limits:

```
alter workload class myclass1
  set 'WRITE_TRANSACTION_LIFETIME' = '60'
  unset 'STATEMENT_MEMORY_LIMIT', 'STATEMENT_THREAD_LIMIT';
```

By default, a workload class becomes active immediately when it's created, but you can also create it in an initially disabled state:

```
create workload class HIGH_PRIORITY set 'PRIORITY' = '9' DISABLE;
```

To enable the class later:

```
alter workload class HIGH_PRIORITY ENABLE;
```

It's also possible to enable and disable all classes collectively:

```
alter workload class all disable;
alter workload class all enable;
```

Dropping the class is easy, but do read the warning ahead if the class has any workload mappings:

```
drop workload class myclass1;
```

Deleting Workload Classes

When you delete a workload class with DROP WORKLOAD CLASS, all mappings to that class are silently deleted as well.

Creating Workload Mappings

A workload mapping defines the link between a specific database activity and the workload class to which that activity must be assigned. Workload mappings also have their own SQL statement, CREATE WORKLOAD MAPPING, with companion statements ALTER WORKLOAD MAPPING and DROP WORKLOAD MAPPING. We know of no better way to explain this topic than to begin straight away with an example. Suppose we have a SAP_ERP workload class and we want to map all connections by the SAP ERP system PR1 to this class. An ABAP system identifies itself to SAP HANA with the application name SAP:<SID> (we'll explain in a moment how you can find out attributes like application names). Do this as follows:

```
create workload mapping SAP_ERP_WLMAP workload class SAP_ERP
  set 'APPLICATION_NAME' = 'SAP:PR1';
```

Here the mapping is based on just one attribute, the application name, but there are others as well. Listing 4.19 shows the complete list.



APPLICATION NAME
 APPLICATION COMPONENT NAME
 APPLICATION COMPONENT TYPE
 APPLICATION USER NAME
 CLIENT
 USER NAME (see note 1)
 USERGROUP NAME (see note 1)
 SCHEMA NAME (see note 2)
 OBJECT NAME (see note 2)
 XS APPLICATION USER NAME

Listing 4.19 Attributes of Workload Mappings

Note the following for the list:

- USER NAME and USERGROUP NAME are mutually exclusive.
- SCHEMA NAME and OBJECT NAME must always be specified together. The object here is a stored procedure, package, or application function library.

This looks very flexible, enabling you to define the contents of a workload class with great precision—but how can you find out the values of these attributes for a given session? Most of the information can be found in monitoring view M_SESSION_CONTEXT, which contains the session variables of every session. Figure 4.44 shows an example for a session opened by an ABAP work process. The session variable APPLICATION has the value SAP:<SID>; APPLICATIONCOMPONENT contains the name of the ABAP report (here RSM13000, the program used by update processes). Other attributes that can be used for the workload mapping are APPLICATIONUSER (here the SAP User ID), CLIENT, and XS_APPLICATIONUSER.

```

1- select * from m_session_context
2  order by connection_id, key
    
```

HOST	PORT	CONNECTION_ID	KEY	VALUE	SECTION
30052	200128	200128	ABAPVARCHARMODE	TRUE	SYSTEM
30052	200128	200128	APPLICATION	ABAP	SYSTEM
30052	200128	200128	APPLICATIONCOMPONENT	RSM13000	SYSTEM
30052	200128	200128	APPLICATIONCOMPONENTTYPE	V	SYSTEM
30052	200128	200128	APPLICATIONSOURCE	SAPLBSVU:492	SYSTEM
30052	200128	200128	APPLICATIONUSER		SYSTEM
30052	200128	200128	APPLICATIONVERSION	777 PL 200	SYSTEM
30052	200128	200128	APPLICATION_SERVER_INSTANCE_NAME		USER
30052	200128	200128	APPLICATION_SERVER_WORKPROCESS_INDEX	50	USER
30052	200128	200128	CDS_CLIENT	600	USER
30052	200128	200128	CLIENT	600	SYSTEM
30052	200128	200128	DRIVERVERSION	2.4.177	USER
30052	200128	200128	SAP_SYSTEM_DATE	20210622	USER
30052	200128	200128	TEST_ALL_SOLDBC_PCONN	TRUE	USER
30052	200128	200128	XS_APPLICATIONUSER	SAP	SYSTEM

Figure 4.44 Session Context for ABAP Connection

Note that not every application defines the same set of session variables, as we'll explain ahead. It's therefore important always to inspect the record of a session in M_SESSION_CONTEXT to determine which attributes to use for a mapping.

CLIENT Attribute

The CLIENT attribute refers to the client number in ABAP systems. It has nothing to do with the hostname or network address of the database client.

In the following example, we want to create a workload class with the appropriate mappings for the management tools (the SAP HANA cockpit, SAP HANA Studio, and hdbsql). Statements executed from these three programs must run at the highest priority. We begin by creating the workload class:

```
create workload class MGMT_CLASS set 'PRIORITY' = '9';
```

The next step is to map the three management tools to the new class. To do this, we first run each tool, open an SQL console, and examine the session context. The results are shown in Figure 4.45 for the SAP HANA cockpit ❶, SAP HANA Studio ❷, and hdbsql ❸.

HOST	PORT	CONNECTION_ID	KEY	VALUE	SECTION
hxehost	39003	218555	APPLICATION	sap_xsac_hrtt	
hxehost	39003	218555	APPLICATIONUSER	COCKPIT_ADMIN	
hxehost	39003	218555	XS_APPLICATIONUSER	SYSTEM	
hxehost	39003	218555	PROTOCOL_VERSION	4.1 (6, 8)	
hxehost	39003	218555	APPLICATIONVERSION	2.12.20381	
hxehost	39003	218555	DRIVERVERSION	2.4.167	

HOST	PORT	CONNECTION_ID	KEY	VALUE	SECTION
hxehost	39.003	218.543	APPLICATION	HDBStudio	SYSTEM
hxehost	39.003	218.543	APPLICATIONUSER	Mark	SYSTEM
hxehost	39.003	218.543	XS_APPLICATIONUSER	T12345	SYSTEM
hxehost	39.003	218.543	PROTOCOL_VERSION	4.1 (6, 8)	SYSTEM
hxehost	39.003	218.543	APPLICATIONVERSION	2.3.59	SYSTEM
hxehost	39.003	218.543	DRIVERVERSION	2.7.10-e3bb933fc18c4a149fb1b429e3bd3eba3...	USER
hxehost	39.003	218.543	APPLICATIONSOURCE	csns.sql.editor.SQLExecuteFormEditor\$4\$1.r...	SYSTEM

HOST	PORT	CONNECTION_ID	KEY	VALUE	SECTION
hxehost	39003	215141	APPLICATION	hdbsql	SYSTEM
hxehost	39003	215141	APPLICATIONUSER	hxeadm	SYSTEM
hxehost	39003	215141	XS_APPLICATIONUSER	SYSTEM	SYSTEM
hxehost	39003	215141	PROTOCOL_VERSION	4.1 (6, 8)	SYSTEM
hxehost	39003	215141	DRIVERVERSION	2.4.171	USER

Figure 4.45 Session Context for Management Tools

The APPLICATION variable clearly is suitable to identify the program, so we use that as the attribute for the workload mapping of each tool (see Listing 4.20).

```
create workload mapping MAP_COCKPIT workload class MGMT_CLASS
  set 'APPLICATION NAME' = 'sap_xsac_hrtrt';
create workload mapping MAP_STUDIO workload class MGMT_CLASS
  set 'APPLICATION NAME' = 'HDBStudio';
create workload mapping MAP_HDBSQL workload class MGMT_CLASS
  set 'APPLICATION NAME' = 'hdbsql';
```

Listing 4.20 Workload Mappings Based on Application Name

To check whether the mapping works and sessions are indeed assigned to the workload class, we open a new session (the mapping applies only to sessions opened after the workload mapping was created) and check the WORKLOAD_CLASS_NAME column in M_CONNECTIONS. We do this in all three programs, and the workload class is as expected (Figure 4.46). For sessions not mapped to a workload class, the column has the value _SYS_DEFAULT.

The figure consists of two screenshots. The top screenshot shows a SQL query result in a table with one row: 'MGMT_CLASS'. The bottom screenshot shows a terminal window with a SQL query and its result, also showing 'MGMT_CLASS'.

Figure 4.46 Workload Class Mapping

You're also allowed to use wildcards in an attribute string (but only at the end of the string, not at the beginning or in the middle). For example, you could create a workload

mapping for all ABAP systems like this. You must add the option WITH WILDCARD to indicate that a character (default is %) must be interpreted as a wildcard:

```
create workload mapping MAP_ABAP workload class SAP_ABAP_CLASS
  set 'APPLICATION NAME' = 'SAP:%' WITH WILDCARD;
```

This matches any application name that begins with SAP:. You can change the wildcard character, which might be useful if the attribute value already contains %. In the following example, a mapping is created for an application xyz%agent followed by a suffix (like xyz%agent1, xyz%agent2, etc.):

```
create workload mapping MYMAP1 workload_class MYCLASS1
  set 'APPLICATION NAME' = 'xyz%agent*' with wildcard '*';
```

Changing and Dropping Workload Mappings

Use ALTER WORKLOAD MAPPING to change the definition of a mapping. In the following example, we decided that it would be better to identify use of the SAP HANA cockpit by a username rather than an application name, as follows:

```
alter workload mapping MAP_COCKPIT workload class MGMT_CLASS
  unset 'APPLICATION NAME' set 'USER NAME' = 'HDBCCKPIT';
```

This disables the attribute APPLICATION NAME and replaces it with USER NAME (if we hadn't disabled the application name, then both attributes would be used). It may seem surprising that the workload class has to be specified, but this is indeed mandatory, although redundant (you're not allowed to create a mapping with the same name in a different class). Consider this a quirk of the statement.

To drop a workload mapping, use the following statement:

```
drop workload mapping MAP_COCKPIT;
```

In contrast to the ALTER statement, there's no need to also specify the workload class in the DROP. (We're sure there must be a logic to this, but it's evading us!)

Workload Hints

Even if a session doesn't belong to a workload class based on its attributes, it can still request executing a statement in that class by adding a statement hint, as follows:

```
select * from bigtable WITH HINT(WORKLOAD_CLASS("WLCLASS1")) ;
```

By default, however, this is allowed only if it results in *more* restrictive conditions for the statement. We'll explain this with an example. We have two workload classes, SMALL and LARGE, distinguished by the maximum statement memory, as follows:

```
create workload class SMALL set 'STATEMENT MEMORY LIMIT' = '10';
create workload class LARGE set 'STATEMENT MEMORY LIMIT' = '100';
```

We map a user `MOUSE` to the `SMALL` class, as follows:

```
create workload mapping MAP_MOUSE workload class SMALL set 'USER NAME' =
'MOUSE';
```

We connect to the database as `MOUSE` and execute a query that consumes more than 10 GB of memory, as follows:

```
select * from bigtable;
Error: (dberror) [4]: cannot allocate enough memory: OOM. Please check traces
for further information.
```

`MOUSE` is aware that the `LARGE` class has a higher memory quota, and thinks there's a neat way to circumvent the limitation—but tough luck:

```
select * from bigtable WITH HINT(WORKLOAD_CLASS("LARGE")) ;
Error: (dberror) [4]: cannot allocate enough memory: OOM. Please check traces
for further information.
```

Because the workload class specified in the hint has weaker restrictions than the user's natural class, SAP HANA ignores the hint and still applies the conditions of the `SMALL` class. The opposite would have been allowed: a session mapped to the `LARGE` class could specify `WITH HINT(WORKLOAD_CLASS("SMALL"))`, for instance, to ensure that a query that is known to sometimes get out of control is properly reined in.

It's possible to disable this rule and allow workload hints even if these cause the statement to use less restrictive conditions than it would normally be subject to in its own class. This is done by setting the `allow_more_resources_by_hint` parameter to `true` in the `[session_workload_management]` section of the `indexserver.ini` file.

`MOUSE` persuades the database administrator to change the parameter and retries the statement with the hint to use the `LARGE` workload class—and this time it works:

```
select * from bigtable WITH HINT(WORKLOAD_CLASS("LARGE")) ;
987654321 rows selected (overall time 1234.5 seconds) -- Success!
```

Whether this is a good idea is of course something we leave to the discretion of the administrator.



Invalid Hints Are Ignored

If the hint is against the rules, then it's silently ignored, and execution of the statement is still attempted. An incorrect or unacceptable hint by itself doesn't result in an SQL error. In our example, we used the statement memory limit because this is a case in which ignoring the hint results in a visible error (out of memory). Had we used a property like `PRIORITY`, then the statement would simply have run at the priority of the `SMALL` group without giving any feedback to the user that the hint hadn't been honored.

4.6.5 Workload Settings for Users

Some of the restrictions set for a workload class can also be set for an individual user. This applies to the statement memory limit, statement thread limit, and priority. For an example, see Listing 4.21.

```
alter user saphanadb
set parameter statement memory limit = '1',
statement thread limit = '4',
priority = '3';
```

Listing 4.21 Workload Restrictions for User

To remove restrictions, use `CLEAR` instead of `UNSET`, as follows:

```
alter user saphanadb
clear parameter statement memory limit, statement thread limit, priority;
```

Simple Models Are Best

The interactions between the different levels where workload limits can be configured (global, workload class, user) can become quite intricate, which may lead to situations that are hard to comprehend without forensic analysis. We generally recommend keeping the model as simple as possible—for instance, by only using workload classes.



4.6.6 Metadata

Metadata regarding workload management and workload classes is available in the catalog and monitoring views listed in Table 4.10.

View	Contents
<code>WORKLOAD_CLASSES</code>	Workload classes.
<code>WORKLOAD_MAPPINGS</code>	Workload mappings.
<code>M_CONNECTIONS</code>	Column <code>WORKLOAD_CLASS_NAME</code> shows the workload class of the session. Contains <code>_SYS_DEFAULT</code> if the session is not assigned to a workload class.
<code>M_ACTIVE_STATEMENTS</code> <code>M_EXPENSIVE_STATEMENTS</code> <code>M_PREPARED_STATEMENTS</code>	Column <code>WORKLOAD_CLASS_NAME</code> shows the workload class passed as a hint for the statement. If the statement was executed without hint from a session assigned to a workload class, then this column contains <code>_SYS_DEFAULT</code> and not the class of the session.
<code>USER_PARAMETERS</code>	Lists all user parameters including those related to workload control (priority, thread limit, and memory limit).

Table 4.10 Metadata for Workload Management

4.7 Special Utilities

In this section, we take a brief look at three utilities in the SAP HANA kernel: the database server management console `hdbcons`; the program `hdbrsutil`, which keeps row-store tables in memory when the instance is stopped; and its near namesake, `hdbn-sutil`, which is used specifically in the context of replication.

4.7.1 hdbcons

The `hdbcons` program is the SAP HANA database server management console. It's an expert tool that lets you do uncommon things (and suitably impress your colleagues, provided of course that you use it correctly). The tool has its own FAQ in SAP Note 2222218. That note begins by warning you that you should only run it under guidance of SAP, but then continues to describe it in detail; there are numerous mentions of `hdbcons` in other SAP Notes as well. Nevertheless, we recommend heeding SAP's advice and only using it when SAP asks you to do so or when you find its usage properly described in SAP documentation.

There are two ways to run `hdbcons`: from the command line and from SAP HANA Studio. To learn how to run it from the command line, use the `-help` argument, as shown in Listing 4.22.

`hdbcons -help`

```
SAP HANA DB Management Client Console (type '\?
' to get help for client commands)
Usage: hdbcons [<options>] [<command> [<repeat_freq_in_seconds>]]
Default connection: hdbindexserver, system HXE, instance 90
Available options:
  -h: print the help
  -n: prevent autoconnect
  -s: set system name to use (default read from environment)
  -i: set instance ID to use (default read from environment)
  -e: set executable to connect to (default hdbindexserver)
  -d <dirname>: open persistency in <dirname> for diagnose
  -p <pid>: attach to process with given <pid>
  -t <timeoutSec>: connection timeout in seconds
  -r remove IPMM shared memory
```

Listing 4.22 Arguments of `hdbcons` Program

What this already tells you is that `hdbcons` connects to a database process, which is `hdbindexserver` by default (the index server process). If you want to connect to a different service, you can do so by specifying either its name with `-e <program>` or its process ID with `-p <pid>`. You can find both in the output of `HDB info`. For example, if you want to connect to the name server of the system database and you find that its PID is 1840, then use either of these commands:

```
hdbcons -e hdbnameserver
hdbcons -p 1840
```

This opens an interactive dialog in which you can enter commands; type `help` for a complete list or `help <command>` for help on a specific command.

You can also pass the `hdbcons` command directly in the OS command line, as follows:

```
hdbcons -p 5413 'transaction c 13'
```

Obviously, we aren't going to describe all the commands here (nor are we going to pretend that we know what they're all doing). Refer to the "What Are the Most Important `hdbcons` Functionalities?" section in SAP Note 2222218 for a list of the most common or useful commands. Whenever it's relevant to mention `hdbcons` (as we did in the sections about canceling or disconnecting sessions or aborting transactions earlier in this chapter, for instance), we will of course do so.

SAP HANA Studio has a dedicated tab for `hdbcons`, but it isn't enabled by default. To activate it, proceed as follows (see Figure 4.47):

- 1 Choose **Window • Preferences**.
- 2 In the **Preferences** window, open **SAP HANA • Global Settings** and enable the **Show Management Console for hdbcons** option.
- 3 The Administration Console now shows a new **Console** tab where you can type the `hdbcons` commands.

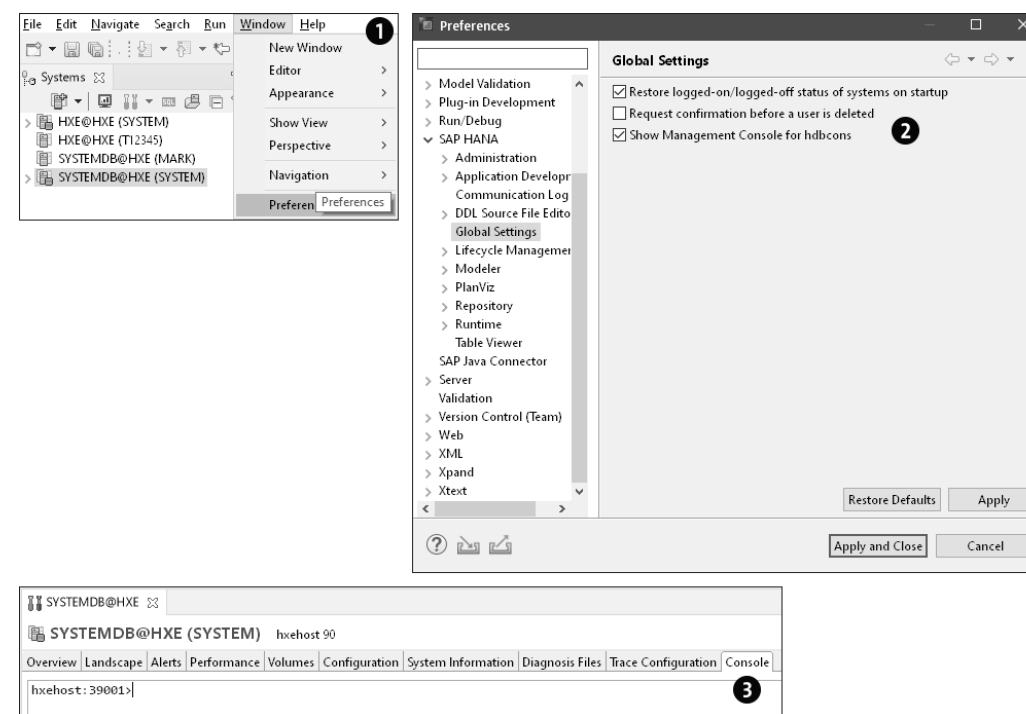


Figure 4.47 Enable `hdbcons` Console in SAP HANA Studio

4.7.2 hdbrsutil

Looking at the output of HDB info, you may spot a process called hdbrsutil. That process is there regardless of whether SAP HANA is running (Figure 4.48 ❶) or stopped ❷.

```

hqladm@exphanap1:/usr/sap/HQ1/HDB01> HDB info
USER      PID      PPID     %CPU     VSZ      RSS      COMMAND
hqladm    4417     4416     0.0      17748    6100     -sh
hqladm    23744    4417     0.0      14260    3784     \_ /bin/sh /usr/sap/HQ1/HDB01/HDB info
hqladm    23779    23744    0.0      34892    3436     \_ ps fx -U hqladm expdb p50db s14db hqldb -o user:8,pid:8,ppid:8,pcpu:5
hqladm    4979     1        0.0      781528   51388    hdbrsutil --start --port 30101 --volume 1 --volumesuffix mnt00001/hdb00001 -
hqladm    4627     1        0.0      23048    3080     sapstart pF=/usr/sap/HQ1/SYS/profile/HQ1_HDB01_exphanap1
hqladm    4635     4627     0.0      483624   90492    \_ /usr/sap/HQ1/HDB01/exphanap1/trace/hdb.sapHq1_HDB01 -d -nw -f /usr/sap/HQ
hqladm    4656     4635     8.4      8125456  5061948  \_ hdbnameserver
hqladm    5309     4635     0.4      1952592  177032   \_ hdbcompileserver
hqladm    5312     4635     0.5      2292860  211372   \_ hdbpreprocessor
p50db     5489     4635     1.4      6268892  2636640  \_ hdbdocstore -port 30167
s14db     5502     4635     1.4      6814944  2712424  \_ hdbdocstore -port 30152
p50db     5584     4635     1.4      5888928  2156700  \_ hdbdpserver -port 30164
s14db     5645     4635     1.3      5693600  1400544  \_ hdbdpserver -port 30155
expdb     5711     4635     4.6      7236564  3882364  \_ hdbindexserver -port 30191
hqladm    5774     4635     4.9      7635408  4535508  \_ hdbindexserver -port 30103
p50db     5897     4635     32.5     45563372 42482836 \_ hdbindexserver -port 30158
s14db     6021     4635     31.8     40068564 36899180 \_ hdbindexserver -port 30140
p50db     6277     4635     1.6      4821476  1286356  \_ hdbscriptserver -port 30170
s14db     6387     4635     1.6      4770024  1312072  \_ hdbscriptserver -port 30146
hqladm    6631     4635     1.2      5292532  1354948  \_ hdbxsengine -port 30107
p50db     6757     4635     1.4      5298936  1460284  \_ hdbxsengine -port 30161
s14db     6882     4635     1.4      5575156  1440952  \_ hdbxsengine -port 30149
s14db     9004     4635     0.5      3991012  414632   \_ hdbdiindexserver -port 30143

```

```

hqladm@exphanap1:/usr/sap/HQ1/HDB01> HDB info
USER      PID      PPID     %CPU     VSZ      RSS      COMMAND
hqladm    4417     4416     0.0      17748    6100     -sh
hqladm    25801    4417     0.0      14260    3800     \_ /bin/sh /usr/sap/HQ1/HDB01/HDB info
hqladm    25837    25801    0.0      34892    3664     \_ ps fx -U hqladm -o user:8,pid:8,ppid:8,pcpu:5,vsz:10,rsz:10,args
hqladm    4979     1        0.0      781528   51388    hdbrsutil --start --port 30101 --volume 1 --volumesuffix mnt00001/hdb00001 -
hqladm    2691     1        0.0      1162816  32348    /usr/sap/HQ1/HDB01/exe/sapstartsv pF=/usr/sap/HQ1/SYS/profile/HQ1_HDB01_exph
hqladm    2537     1        0.0      72360    8112     /usr/lib/systemd/systemd --user
hqladm    2538     2537     0.0      117416   2924     \ (sd-pam)

```

Figure 4.48 hdbrsutil Process

The role of this process is to keep the row store in memory even when SAP HANA is down. Because the row store is always loaded in its entirety at startup (we discuss the row and column store extensively in Chapter 12), having it already in memory will reduce start times. There are parameters that control this process, but the functionality is enabled by default and should remain so. There are more details in SAP Note 2159435.

Sometimes it's necessary to manually shut down hdbrsutil, such as when you have to unmount the file systems that contain the SAP HANA persistent data. In that case, use the following command:

```
(as user <hsid>adm)
hdbrsutil --stop --port <port>
```

Here, <port> is the port number used by the active process (30101 in the example shown in Figure 4.48).

4.7.3 hdbnsutil

There's only a one letter difference from hdbrsutil, but hdbnsutil is a completely different program. It's used in the context of system replication, providing functionality such as registering, enabling, and disabling replication or initiating a takeover between replication nodes. Run the command without parameters to display a complete list of

all parameters. A commonly used command is the one that checks the current replication status of the system:

```
hdbnsutil -sr_state
```

System replication is covered in Chapter 6.

4.8 Managing Redo Logs

Redo logs are database objects, and in that sense, they really belong in the next chapter on database management. However, monitoring the logs is one of the routine tasks of the SAP HANA administrator, and for that reason we decided to encroach a little on database territory and make this subject part of the chapter on instance management.

Together with the data, the redo logs form the persistence layer (disk files) of SAP HANA. Whenever a transaction that has changed data in the database commits, a record of those changes is written to the redo logs. These writes happen asynchronously (the session that has committed the transaction doesn't have to wait for confirmation that the log has been written), but they happen more frequently than data writes from memory to disk (which are grouped into periodic savepoints, something we describe in more detail in Chapter 12). Redo logs serve two important purposes:

- If the database or instance terminates abnormally, it's likely that not all data changes have been written to disk. In that case, the redo log will be used at startup to recover the changes made by committed transactions.
- With system replication, the logs are used to keep the standby copies of the primary system synchronized.

Terminology

Depending on the database system, the terms *transaction log* and *redo log* are commonly used to designate the logging system. In the context of SAP HANA, both terms appear, but there seems to be a preference for *redo log*. We will therefore use this term throughout the book.

In Section 4.1.5, we showed where the log volumes are located on disk and told you in no uncertain terms that you should leave those files well alone. That doesn't mean that redo logs must not be managed at all, only that you should do so using only the appropriate instruments. In this section, we examine SAP HANA logging and redo logs in detail. In Section 4.8.1, we introduce log segments, which are really storage units for redo log data. We then discuss in Section 4.8.2 the very important concept of log modes; the log mode determines whether in case of trouble a database can be restored up to the latest committed change or only up to the time of the most recent data backup. In exceptional circumstances, it's possible to completely disable logging,

something we explain in Section 4.8.3. Section 4.8.4 shows how to deal with the situation where the redo log becomes full, which you might hope will not happen to you (spoiler: one day it will). The problem isn't hard to solve, but mistakes can be fatal. In Section 4.8.5, we see how you can monitor redo logging in the various administration tools.

4.8.1 Log Segments

Redo log data is written to files on disk called log segments. We already saw in Section 4.1.5 that these are numbered sequentially; when a log segment reaches its configured size limit, the system automatically creates the next one in the series.

The size of a log segment is configured with the `log_segment_size_mb` parameter in the `[persistence]` section of the service INI file.

Here, the service INI file is either *global.ini* or the INI file of the service that performs redo logging (which several services do, not just the ones that serve user transactions). The default values are as follows:

- *global.ini*: 1,024 MB
- *nameserver.ini*: 64 MB
- *dpserver.ini*, *scriptserver.ini*, *xsengine.ini*: 8 MB

Notice that the parameter isn't defined for the index server—that is, for transactions in the tenant databases. The index server uses the size set in *global.ini*.

4.8.2 Log Modes

A crucial aspect of the configuration of a database is its log mode. There are two different log modes, *overwrite* and *normal*. The all-important parameter that sets the log mode is the `log_mode` global parameter found in the `[persistence]` section of the *global.ini* file.

Possible values are 'normal' and 'overwrite'. The default is 'normal'. A newly installed system initially runs in log overwrite mode, but this changes automatically to normal mode after the first backup.

In overwrite mode, data in log segments is overwritten as soon as the periodic savepoint (described in Chapter 12) has written all changed data in memory to the data volume. In overwrite mode, the log segments can't be backed up. In case of a crash, it's still possible to bring the database to a consistent state, but it will be the state at the time of the most recent data backup. It isn't possible to do a forward recovery to a more recent point in time because the redo logs needed for this no longer exist, which inevitably implies that some data is lost.

Log overwrite mode should only be used for databases in which the possible loss of recent data doesn't present a problem, which is typically true for test and sandbox

systems (perhaps less so for development systems). According to SAP, running a production system in log overwrite mode is not recommended—but we would go further and say that a production system should *never* run in this mode.

In normal log mode, log segments are only overwritten when they have been backed up and no longer need to be kept online to use for instance recovery. When a log segment fills up, the system closes that segment and switches to another one that is free (or creates a new one if none is currently free). If log backups are enabled (which in normal log mode they should be), the segment is backed up and removed from the log directory. Normal log mode permits recovering the database to any point in time after a data backup, because an uninterrupted sequence of redo log records is available to reconstruct the data exactly as it was at that point in time.

If there is an exceptionally high number of changes and segments cannot be freed up in a timely fashion, then segments will accumulate in the file system that contains the log directories. The same can occur if there is some problem, such as log backups failing. In the worst case, this can lead to the file system becoming full, preventing further logging. When that happens, the database no longer accepts requests and all activity freezes. Later in this section, we'll explain what to do if such a situation arises.

Backup Required after Switch to Normal Log Mode

If you've manually switched the system to overwrite log mode and now want to switch back to normal, then it's mandatory to immediately make a full data backup; otherwise, it won't be possible to recover to the most recent point in time.

4.8.3 Running with Logging Disabled

It's possible to completely disable logging with the following SQL statement:

```
alter system LOGGING OFF;
```

With logging switched off, no redo log writes occur at all and only savepoints write changes to disk. This means that if the system fails between savepoints, data from committed transactions will almost certainly be lost, forcing you to restore the latest data backup. Running with logging off should only be done in very rare circumstances; a typical case is a very large data upload done without concurrent activity in the system. In such a context, logging would not be useful because if the load fails, the most likely course of action is to truncate the data (or restore a backup) and to start over. In normal multiuser operation, though, you should never run the system without logging.

The statement may take some time because it will wait for all active write transactions to commit or roll back (on the other hand, if there *are* any active transactions, then disabling logging is probably not a good idea).

To reenabling logging use, use this statement:

```
alter system LOGGING ON;
```

After this you must immediately trigger a savepoint and start a data backup to ensure that the database is again recoverable.

4.8.4 Handling a Full Log Situation

With a properly sized file system for the redo logs and a correctly configured log backup, full log situations shouldn't occur—but bad things do happen sometimes, so it's best to be prepared. For this section, we artificially created a full log state in one of our test systems, and we start by explaining how we did this. Now, why would we tell you how to *create* a problem rather than how to *solve* it? The reason is that if you have a test environment available where you can play around with things (and occasionally break them), then we recommend that you use that test system to practice handling a full log situation. The procedure isn't difficult, but you'll feel much more comfortable applying it in a critical system if you have prior experience.

There are many ways to create a full log, but this is what we did:

1. Make sure the database is in normal log mode, not overwrite.
2. Disable automatic log backups, as follows:

```
alter system alter configuration ('global.ini','SYSTEM') set ('persistence',
enable_auto_log_backup') = 'false' with reconfigure;
```

3. Determine the free space in the file system of the log volumes, as follows:

```
df -m <log_directory>
```

4. Create a large dummy file that almost fills the file system, leaving less space than the size of a new log segment (see Section 4.8.1). Linux has a very nice command for this, as follows:

```
fallocate -l <file_size> <file_name>(example: "fallocate -l 6500M BIGFILE")
```

5. Create database activity. You can do this, for instance, by creating a dummy table and then repeatedly filling it with a large number of rows and deleting those rows again. Listing 4.23 shows an easy way to do this.

```
create table TESTTABLE (COL varchar(70));
insert into TESTTABLE select top 1000000 sysuuid||sysuuid
from objects cross join objects;
delete from TESTTABLE;
```

Listing 4.23 Causing Massive Database Changes

6. Repeat these INSERT and DELETE statements continuously until the log fills up, at which point the statement will stop responding. To check how close you are to running out of log space, you can use the query on M_LOG_SEGMENTS described in Section 4.8.5.

When the log becomes full and SAP HANA is no longer able to write redo log data to an available segment, the visible symptom is that activities will freeze: statements that change data hang and it's impossible to open a new connection to the database. When users experience such behavior, your first reaction should be to suspect that redo logging is stuck and to follow the procedure described in the next points.

Figure 4.49 shows the environment for the example. The `/hana/log/HDB` file system is 100% full; at least 2 GB of space is needed in another file system, and in this case that space is available in the `/hana/data/HDB` file system (other file systems would do as well).

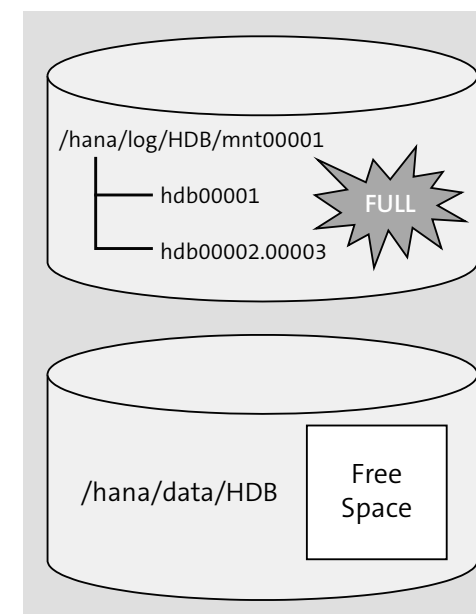


Figure 4.49 Environment for Full Log Example

The procedure frees up space in the full file system by moving a log volume to a temporary location and then backing up the logs and reclaiming log space. When this is done, the log volume is moved back to its standard location, as follows:

1. Verify free space in the file system of the log volumes:

```
df -m /hana/log/HDB
```

There are other circumstances that can make the database freeze up, but a log full situation is by far the most probable.

2. There should also be "log full" messages in the trace files of the tenant database:

```
cdtrace
cd DB_<tenant>
grep "DiskFull" *.trc
```

3. Log into the database server as <hsid>adm and stop the system. Use the sapcontrol command directly, not HDB stop:

```
-- Caution: "HDB" is a literal string, not the system name
sapcontrol -nr $TINSTANCE -function StopSystem HDB
```

4. Locate a file system that has at least 2 GB free space and create a temporary directory there. In our example, we have enough space in the data file system:

```
mkdir /hana/data/HDB/log_temp
```

5. Move at least 2 GB from the log file system to this temporary directory; this is necessary to have enough free space for the database to start. To find the space occupied by each log volume, use this command:

```
du -m /hana/log/HDB/mnt00001/*
```

6. Here we move the log volume *hdb00002.00003*:

```
mv /hana/log/HDB/mnt00001/hdb00002.00003 /hana/data/HDB/log_temp
```

7. Create a symbolic link so that the system is still able to locate the log segments in the volume just moved:

```
ln -s /hana/data/HDB/log_temp/hdb00002.00003 /hana/log/HDB/mnt00001/
hdb00002.00003
```

8. Do another `df -m /hana/log/HDB` execution to ensure there is now enough free space.

9. Start the system:

```
-- Caution: "HDB" is a literal string, not the system name
sapcontrol -nr $TINSTANCE -function StartSystem HDB
```

10. Check that automatic log backups are enabled (set 'enable_auto_log_backup' = 'true'; see Section 4.8.2 for the parameter location).

11. Wait for the log backups to complete:

```
cdtrace
cd DB_<tenant>
tail backup.log
```

12. Clean up the log volumes:

```
hdbsql -I $TINSTANCE -d <tenant> -u SYSTEM -p <passwd>
ALTER SYSTEM RECLAIM LOG
```

13. It's now time to return to the normal directory structure. The system must again be stopped for this:

```
-- Caution: "HDB" is a literal string, not the system name
sapcontrol -nr $TINSTANCE -function StopSystem HDB
```

14. Move the temporary log volume directory back to its normal location. The symbolic link must first be removed:

```
cd /hana/log/HDB/mnt00001
rm -f hdb00002.00003
mv /hana/data/HDB/log_temp/hdb00002.00003 .
```

15. Remove the temporary directory:

```
rm -rf /hana/data/HDB/log_temp
```

16. Restart the system:

```
-- Caution: "HDB" is a literal string, not the system name
sapcontrol -nr $TINSTANCE -function StartSystem HDB
```

Do Not Delete Log Segments

Please don't think we're hectoring you, but this bears repeating: *do not* delete log segments!



4.8.5 Information about Log Segments

In this section, we first look at the `M_LOG_SEGMENTS` monitoring view, which provides important data about the redo logging process, and then we describe how to monitor log segments in the SAP HANA cockpit and in SAP HANA Studio and in Transaction DBACOCKPIT.

M_LOG_SEGMENTS Monitoring View

The `M_LOG_SEGMENTS` monitoring view is the most important source of information about the state of the redo log. This view contains one record per log segment for all services in the current database that use redo logging. A useful query is the one shown in Listing 4.24; we join `M_LOG_SEGMENTS` with `M_SERVICES` in order to display not only the port but also the associated service name.

```
SELECT
  L.PORT, S.SERVICE_NAME, L.FILE_NAME, L.STATE,
  to_int(L.TOTAL_SIZE / ( 1024 * 1024 )) AS "Total Size (MB)",
  to_decimal(L.USED_SIZE / ( 1024 * 1024 ), 6, 1) AS "Used Size (MB)"
FROM
```

```
m_log_segments AS L
INNER JOIN m_services AS S ON s.host = l.host AND s.port = l.port
ORDER BY l.port ASC;
```

Listing 4.24 Querying M_LOG_SEGMENTS

Figure 4.50 shows the output of the statement. We describe the segment states in the STATE column later at the end of Section 4.8.5. The TOTAL_SIZE and USED_SIZE columns give the total size of the segment and the space currently used inside it. As mentioned earlier, the `indexserver` service, which is by far the most important in terms of data volume handled, takes its segment size from the `global.ini` setting, which is 1,024 MB by default. The other services (except for `docstore`, which also uses the global value) allocate segments of a size set in their own parameter files.

You can also see in the output that the effectively used space is in almost all cases far smaller than the physical size of the segments. For instance, the `indexserver` log segment currently in use is segment 8, but the previous segments 1–7 hardly contain any data. This is because the system will switch to a new segment not only when the current segment is full but also at regular time intervals. The interval is set with the `log_backup_timeout_s` parameter in the `[persistence]` section of the `global.ini` file.

The default is 900 seconds (15 minutes). This is an important fact to keep in mind because it implies that even in databases with very little activity, log segments of possibly large sizes are created continuously.

PORT	SERVICE_NAME	FILE_NAME	STATE	Total Size (MB)	Used Size (MB)
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000000.dat	Free	1024	7.5
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000008.dat	Writing	1024	578.2
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000007.dat	Free	1024	2.2
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000006.dat	Free	1024	7.5
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000005.dat	Free	1024	7.5
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000004.dat	Free	1024	9.7
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000003.dat	Free	1024	7.4
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000002.dat	Free	1024	7.6
30158	indexserver	/hana/log/HQ1/mnt00001/hdb00003.00005/logsegment_000_00000001.dat	Free	1024	9.8
30161	xsengine	/hana/log/HQ1/mnt00001/hdb00002.00005/logsegment_000_00000001.dat	Free	8	0.1
30161	xsengine	/hana/log/HQ1/mnt00001/hdb00002.00005/logsegment_000_00000000.dat	Writing	8	0.4
30164	dpserver	/hana/log/HQ1/mnt00001/hdb00004.00005/logsegment_000_00000000.dat	Writing	8	0.4
30164	dpserver	/hana/log/HQ1/mnt00001/hdb00004.00005/logsegment_000_00000001.dat	Free	8	0.1
30167	docstore	/hana/log/HQ1/mnt00001/hdb00005.00005/logsegment_000_00000000.dat	Free	1024	0.0
30167	docstore	/hana/log/HQ1/mnt00001/hdb00005.00005/logsegment_000_00000002.dat	Writing	1024	0.4
30167	docstore	/hana/log/HQ1/mnt00001/hdb00005.00005/logsegment_000_00000001.dat	Free	1024	0.1
30170	scriptserver	/hana/log/HQ1/mnt00001/hdb01024.00005/logsegment_000_00000000.dat	Writing	8	0.4
30170	scriptserver	/hana/log/HQ1/mnt00001/hdb01024.00005/logsegment_000_00000001.dat	Preallocated	8	0.0

Figure 4.50 Redo Log Information in M_LOG_SEGMENTS

Monitoring Log Segments in SAP HANA Cockpit

In the SAP HANA cockpit, open the **Database Overview** (Figure 4.51 1); select the **Monitoring** view 2. In the **Disk Usage** section 3, choose **Monitor Disk Volume** 4.

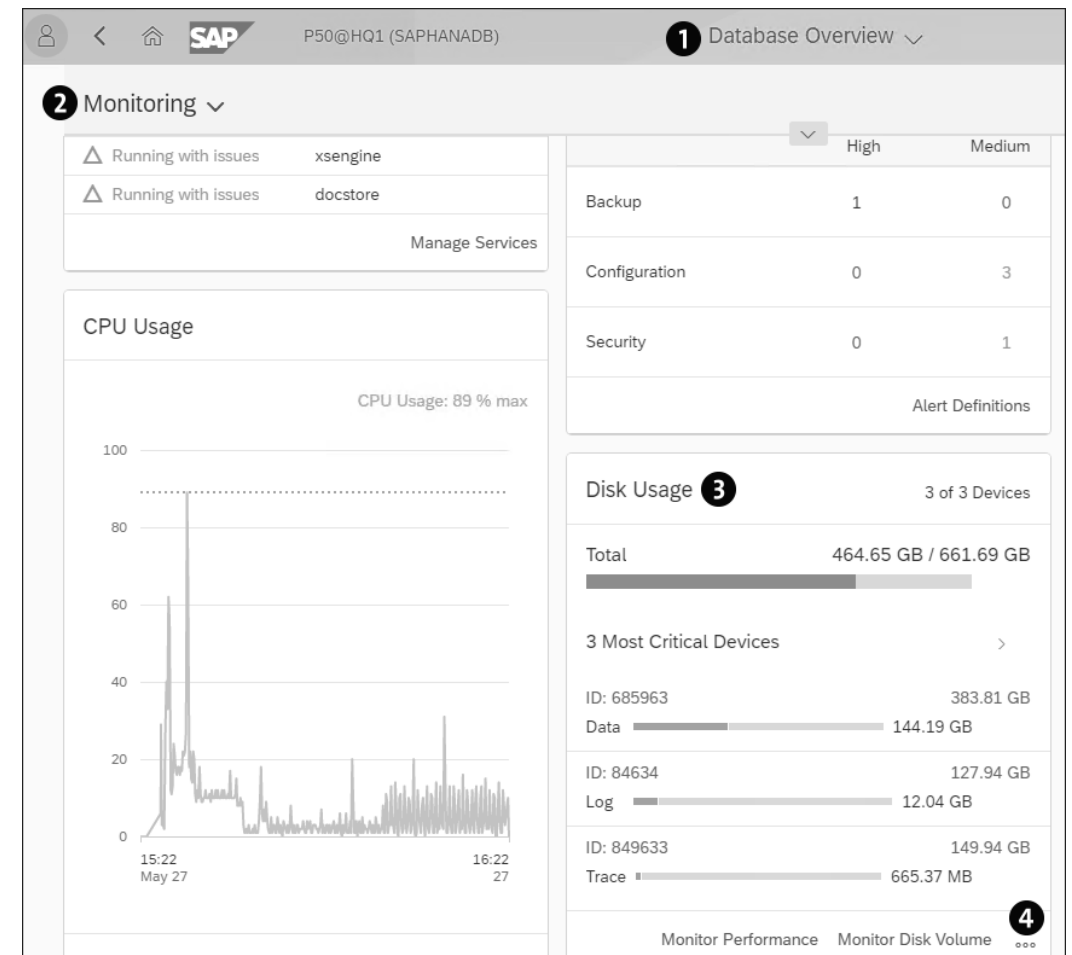


Figure 4.51 Monitor Disk Volume in SAP HANA Cockpit

The upper part of the next window (not shown) is a graph of the total data and log space usage. The part we are interested in is the report at the bottom of the screen, shown in Figure 4.52. Initially, the report lists all services, but in Figure 4.52 we have filtered on the `indexserver` service only.

Volume ID	Service	Type	Size (MB)	Used (MB)	Used (%)	State	Files	Path
3	indexserver	DATA	147,072.0	135,436.4	92.0		1	/hana/data/HQ1/mnt00001/hdb00003.00005
3	indexserver	LOG	8,192.0	8,192.0	100.0	Free	8	/hana/log/HQ1/mnt00001/hdb00003.00005
3	indexserver	LOG	1,024.0	1,024.0	100.0	Writing	1	/hana/log/HQ1/mnt00001/hdb00003.00005
3	indexserver	LOG	1.2	1.2	100.0		1	/hana/log/HQ1/mnt00001/hdb00003.00005

Figure 4.52 Data and Log Report in SAP HANA Cockpit

The list shows one line for the data and one line for all log segments in a given state (the last line, with the **State** column empty, is for the *logsegment_000_directory.dat* file). This is a useful summary, although you have to be careful with the **Used(%)** column, which always shows 100% here, whereas we clearly saw in Figure 4.50 (a screenshot from the same database and made at the same time) that the log segments were anything but full. The reason is that this used space metric comes not from `M_LOG_SEGMENTS` but from another monitoring view, `M_DISKS`, which provides disk space data. In `M_DISKS`, the physical size and used space only differ for data volumes; for log segments they're identical, resulting in a 100% usage rate.

Monitoring Log Segments in Transaction DBACOCKPIT

In DBA Cockpit, choose **Volumes** in the left-hand pane (Figure 4.53 ❶). Under **Disks**, double-click the **LOG** entry ❷. The log segments are then listed at the bottom of the screen. In contrast with the cockpit, the information is shown per log segment instead of being summarized. The column for the used size shows 100% here as well, but you know the cause of that now.

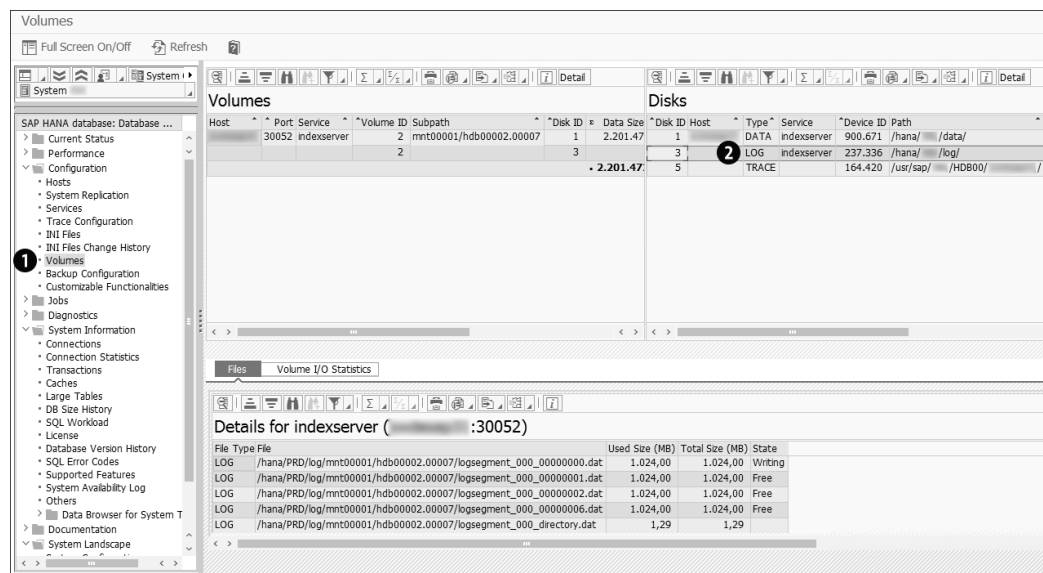


Figure 4.53 Data and Log Report in Transaction DBACOCKPIT

Monitoring Log Segments in SAP HANA Studio

In SAP HANA Studio, you can find information about the log segments under the **Volumes** tab of the Administration Console (Figure 4.54). Click a service in the upper part of the window to list the log segments that belong to it.

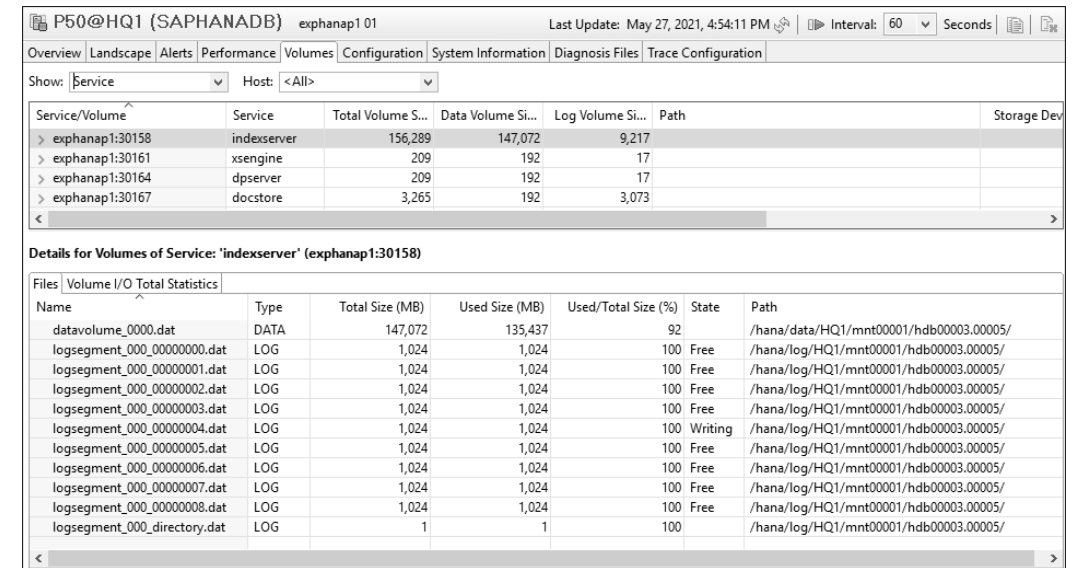


Figure 4.54 Data and Log Report in SAP HANA Studio

As in DBA Cockpit, the list is per log segment and not summarized, with the **Used/Total Size (%)** metric again at 100% for all log segments.

Log Segment States

We still need to look at the different states of a log segment, as shown in the **STATE** column of `M_LOG_SEGMENTS`. The possible states are shown in Table 4.11.

Log Segment State	Description
Formatting	The segment is being formatted and not yet ready for use.
Preallocated	The segment has been allocated but has not yet been used.
Writing	The log segment is being written. This state identifies the currently active log segment of the service.
Closed	The log segment is no longer in active use. It has not yet been backed up and is still required in case of a restart.
Truncated	The log segment is no longer in active use. It has not yet been backed up and is no longer required for restart.
BackedUp	The log segment has been backed up but cannot be removed yet because it is still needed in case of a restart.

Table 4.11 Log Segment States

Log Segment State	Description
RetainedFree	This state only appears in replicated systems. This means that the segment has been backed up and is no longer needed for a restart, but it is required for the synchronization of the replication sites.
Free	The log segment has been backed up and is no longer required for restart. Its present contents can be overwritten.

Table 4.11 Log Segment States (Cont.)

4.9 Summary

In this chapter, we made a detailed profile of the SAP HANA instance: how it's organized, which processes it runs, and where it stores its critical files. You've learned how to start and stop the instance and how to maintain its parameters. We presented the memory management model and explained how you can manage and monitor the way SAP HANA uses its ample memory. We took a closer look at sessions, transactions, and locks, described several features that enable you to manage the workload in the SAP HANA system, and talked about some special utility programs. Finally, we introduced transaction logging and log modes. That last subject brought us to the gates of the database realm, and that is where the next chapter will take you.

Contents

Preface	23
1 SAP HANA Overview	33
1.1 The Column Store	33
1.2 Hardware Requirements	36
1.3 Editions	37
1.3.1 SAP HANA, Platform Edition	37
1.3.2 SAP HANA, Express Edition	39
1.3.3 SAP HANA, Developer Edition	39
1.3.4 SAP HANA Cloud	39
1.4 Use Cases	40
1.4.1 SAP HANA as the Primary Database for the ABAP Platform and SAP NetWeaver	41
1.4.2 SAP HANA as a Data Mart	42
1.4.3 SAP liveCache on SAP HANA	43
1.4.4 SAP HANA Accelerators	43
1.4.5 SAP HANA as a Development Platform	44
1.5 Data Provisioning	46
1.5.1 SAP Landscape Transformation Replication Server	46
1.5.2 Mobilink	48
1.5.3 Direct Extractor Connection	49
1.5.4 SAP Data Services for ETL-Based Replication	49
1.5.5 SAP Replication Server for Log-Based Replication	50
1.6 Advanced Analytics	50
1.6.1 Machine Learning and SAP HANA, Predictive Option	51
1.6.2 SAP HANA, Spatial and Graph Option	51
1.6.3 Text Analytics and Search	52
1.6.4 SAP HANA, Streaming Analytics Option	53
1.7 Data Integration	54
1.7.1 SAP HANA Smart Data Access	54
1.7.2 SAP HANA Smart Data Integration and SAP HANA Smart Data Quality	55
1.8 Summary	55

2	Architecture of the SAP HANA System	57
2.1	The Basics	57
2.1.1	System Database	58
2.1.2	Tenant Databases	59
2.1.3	SAP HANA License Keys	61
2.1.4	Cross-Database Access	62
2.1.5	Database Isolation	62
2.2	Services and Processes	63
2.3	SAP HANA XSA Runtime Platform	67
2.4	Data Persistency	70
2.5	Multihost or Distributed SAP HANA Systems	71
2.5.1	Data and Table Partitioning	73
2.5.2	Single-Level Partitioning	75
2.5.3	Multilevel Partitioning	75
2.5.4	Heterogeneous Partitioning	76
2.6	High Availability	76
2.7	Backup and Recovery	78
2.8	Summary	80
3	Administration Tools	81
3.1	SAP HANA Cockpit	81
3.1.1	Navigation	81
3.1.2	Setup and Configuration	85
3.1.3	Monitoring and Administration	91
3.1.4	Starting and Stopping Instances and Databases	94
3.1.5	Configuring the System	96
3.1.6	Monitoring Performance	99
3.1.7	Installing the SAP HANA License	101
3.1.8	Executing SQL Statements	102
3.2	SAP HANA Studio	103
3.2.1	Setup and Configuration	104
3.2.2	Monitoring and Administration	105
3.2.3	Starting and Stopping Instances and Databases	108
3.2.4	Configuring the System	109
3.2.5	Monitoring Performance	111

3.2.6	Installing the SAP HANA License	112
3.2.7	Executing SQL Statements	112
3.3	Command-Line Tools	113
3.3.1	Starting and Stopping Instances and Databases	114
3.3.2	Monitoring and Administration	115
3.3.3	Configuring the System	117
3.3.4	Executing SQL Statements	119
3.3.5	Using hdbsql	121
3.4	DBA Cockpit for SAP HANA	124
3.5	Additional Tools	127
3.5.1	SAP Solution Manager	127
3.5.2	SAP Landscape Management	128
3.6	Summary	128
4	Instance Administration	131
4.1	Structure of an SAP HANA Instance	132
4.1.1	SAP HANA System ID and Instance Number	133
4.1.2	Users and Groups	133
4.1.3	Directory Structure	134
4.1.4	Process Tree	135
4.1.5	Services and Ports	136
4.1.6	SAP HANA Database Directories	137
4.1.7	INI Files: Database Parameters	141
4.1.8	Trace Files	143
4.1.9	Command Aliases for Directory Navigation	144
4.2	Starting and Stopping the Instance	145
4.2.1	With the Command Line	145
4.2.2	With SAP HANA Studio	145
4.2.3	With SAP HANA Cockpit	148
4.2.4	Starting and Stopping Tenant Databases	148
4.2.5	Checking the Instance and Database Status	150
4.2.6	Managing the SAP Start Service	150
4.3	Maintaining Parameters	152
4.3.1	With SQL Statements	153
4.3.2	With the SAP HANA Cockpit	157
4.3.3	With Transaction DBACOCKPIT	159
4.3.4	With SAP HANA Studio	161

4.3.5	With setParameter.py	162
4.3.6	Protecting against Unsupported Parameter Values	164
4.4	Managing Memory	165
4.4.1	Memory Layout	165
4.4.2	When and How Memory Is Allocated	166
4.4.3	Memory Shortages	167
4.4.4	Heap Memory and Shared Memory	168
4.4.5	Memory for SQL Statements	170
4.4.6	Host Memory Statistics	171
4.4.7	Monitoring Memory Usage	172
4.5	Sessions and Transactions	176
4.5.1	Monitoring Sessions	178
4.5.2	Monitoring Transactions	179
4.5.3	Monitoring Locks	182
4.6	Load Management and Distribution	184
4.6.1	Admission Control	185
4.6.2	Processor Affinity	188
4.6.3	Parallel Execution of SQL Statements	191
4.6.4	Workload Classes	193
4.6.5	Workload Settings for Users	201
4.6.6	Metadata	201
4.7	Special Utilities	202
4.7.1	hdbcons	202
4.7.2	hdbrsutil	204
4.7.3	hdbnsutil	204
4.8	Managing Redo Logs	205
4.8.1	Log Segments	206
4.8.2	Log Modes	206
4.8.3	Running with Logging Disabled	207
4.8.4	Handling a Full Log Situation	208
4.8.5	Information about Log Segments	211
4.9	Summary	216
5	Database Administration	217
5.1	Managing Tenant Databases	218
5.1.1	Creating a Tenant Database	218
5.1.2	Starting and Stopping Tenants	220

5.1.3	Renaming a Tenant	221
5.1.4	Deleting Tenant Databases	222
5.1.5	Working with Fallback Snapshots	223
5.1.6	Working with Services	224
5.1.7	Working with Alerts	226
5.1.8	Preventing System Changes in Tenant Databases	228
5.1.9	Restricting and Disabling Features on Tenant Databases	231
5.2	Managing Data and Log Volumes	231
5.2.1	Data and Log Volumes Overview	232
5.2.2	Savepoint Operation	233
5.2.3	Data Volumes	235
5.2.4	Partitioning of the Data Volumes	238
5.2.5	Log Volumes	241
5.2.6	Reclaiming Space	246
5.3	Database Corruptions	248
5.3.1	Verifying Logical Consistency	249
5.3.2	Verifying Physical Consistency	254
5.4	Backup and Recovery	256
5.4.1	What Is Disaster Recovery?	256
5.4.2	Savepoints and Redo Logs	258
5.4.3	Backup Types	259
5.4.4	Backup Targets	273
5.4.5	Backup Configuration	282
5.4.6	Running Backups	284
5.4.7	Database Recoveries	290
5.4.8	Diagnostic Files for Backup and Recovery	315
5.5	Landscape Management	316
5.5.1	Moving or Cloning an SAP HANA System Using SAP HANA Lifecycle Manager Tools	316
5.5.2	Renaming an SAP HANA System	318
5.5.3	Copying a System Using System Replication	318
5.5.4	Copying and Moving Tenant Databases Using Database Replication	319
5.5.5	Copying a Database Using Backup and Restore	330
5.6	Network Configuration	334
5.6.1	Network Zones	334
5.6.2	Ports and Connections	335
5.6.3	Client Connections	338
5.7	Summary	339

6	System Replication	341
6.1	Architecture	343
6.1.1	Storage Replication	343
6.1.2	System Replication Overview	345
6.1.3	Storage versus System Replication	354
6.1.4	Backup and Recovery	355
6.1.5	Hostname Resolution	356
6.1.6	System Replication Parameters	360
6.2	Configuring and Monitoring System Replication	365
6.2.1	Prerequisites and Process Overview	366
6.2.2	Using the SAP HANA Cockpit	369
6.2.3	Using SAP HANA Studio	373
6.2.4	Using the Command Line	376
6.3	Takeover and Failback	381
6.3.1	Using the SAP HANA Cockpit	383
6.3.2	Using SAP HANA Studio	384
6.3.3	Using the Command Line	385
6.4	Disable System Replication	386
6.4.1	Using the SAP HANA Cockpit	387
6.4.2	Using SAP HANA Studio	387
6.4.3	Using the Command Line	389
6.5	Multitier System Replication	390
6.5.1	Architecture	390
6.5.2	Multitier via the SAP HANA Cockpit	394
6.5.3	Multitier via SAP HANA Studio	397
6.5.4	Multitier via the Command Line	399
6.5.5	Performing a Takeover	401
6.5.6	Restore to the Original Configuration	404
6.6	SAP HANA, Active/Active Read-Enabled Option	406
6.7	Secondary Time Travel	410
6.8	Zero-Downtime Maintenance with SAP Solutions	414
6.9	Hardware Exchange via System Replication	417
6.10	Summary	418

7	Scale-Out Systems and High Availability	419
7.1	Architecture	419
7.1.1	Scale Up Versus Scale Out	419
7.1.2	Multiple Node System Overview	421
7.1.3	Host Grouping to Separate Hot and Warm Data	425
7.1.4	Host Roles	425
7.1.5	File System Layout	427
7.2	Network Layout	428
7.2.1	Host Name Resolution	428
7.2.2	Client Connections	431
7.2.3	Application Connection after Takeover	432
7.3	Host Autofailover	433
7.3.1	Network-Attached Storage	434
7.3.2	Storage Area Network	435
7.4	Administration and Monitoring	436
7.4.1	Starting and Stopping	436
7.4.2	Monitoring	437
7.4.3	Configure Host Failover	443
7.5	Configuration and Setup	445
7.5.1	Adding Hosts	445
7.5.2	Distributing Tenant Databases	449
7.5.3	Data Redistribution	451
7.5.4	Removing Hosts	455
7.5.5	Adding/Removing Host Roles	457
7.6	Takeover and Failback	457
7.6.1	Host Takeover Detection	458
7.6.2	Takeover Execution	459
7.6.3	Failback Execution	464
7.7	Summary	464
8	Data Tiering	465
8.1	Data Tiering Options	466
8.2	Persistent Memory	470
8.2.1	Persistent Memory Overview	470
8.2.2	Configuration Parameters	472
8.2.3	Enabling Persistent Memory on Database Objects	473

8.2.4	Unloading Tables from Memory	474
8.2.5	Monitoring Views	474
8.2.6	Persistent Memory Monitor	475
8.3	Native Storage Extensions	476
8.4	SAP HANA Extension Node	480
8.4.1	Architecture	481
8.4.2	Configuration	482
8.4.3	Backup and Recovery	483
8.4.4	High Availability	483
8.5	SAP HANA Dynamic Tiering	484
8.5.1	Dynamic Tiering Overview	484
8.5.2	Architecture	485
8.5.3	Configuration	487
8.5.4	Database Backup and Recovery	488
8.5.5	High Availability	488
8.6	Integration with SAP IQ	489
8.7	Integration with Hadoop	491
8.8	Summary	493

9 Planning and Setting Up the SAP HANA Landscape 495

9.1	Architecture	495
9.2	Cloud Deployments	496
9.3	On-Premise Deployments	498
9.3.1	The Appliance	500
9.3.2	SAP HANA Tailored Data Center Integration	501
9.3.3	Appliance versus SAP HANA Tailored Data Center Integration	503
9.3.4	Scale Up versus Scale Out	505
9.3.5	SAP HANA Hardware Directory	505
9.3.6	Deployment Options	506
9.3.7	Software and Hardware Virtualization	512
9.3.8	Codeployment of SAP HANA with SAP NetWeaver AS ABAP/Java on One System	515
9.3.9	Deployment Options Compared	517
9.4	Capacity Planning	519
9.4.1	Quick Sizer	519

9.4.2	Sizing Reports	520
9.4.3	T-Shirt Sizing	533
9.4.4	Easy and Lazy Sizing	533
9.5	Infrastructure Design	536
9.5.1	Appliance	536
9.5.2	SAP HANA Tailored Data Center Integration	537
9.6	Network Layout	540
9.6.1	Client Zone	542
9.6.2	Internal Zone	543
9.6.3	Storage Zone	545
9.6.4	Backup Zone	546
9.7	High-Availability Clusters	547
9.8	Virtualization	550
9.8.1	Virtualization on VMware	551
9.8.2	Virtualization on IBM Power VM	555
9.9	Operating System Platforms	557
9.10	Reference Architecture	557
9.11	Summary	561

10 Installation and Updates 563

10.1	Release Cycle of SAP HANA	563
10.2	Preparing for Installation and Upgrades	565
10.2.1	Hardware Requirements	566
10.2.2	Software Requirements	566
10.2.3	SAP Hardware and Cloud Measurement Tool	584
10.3	Using Lifecycle Management Tools	588
10.3.1	SAP HANA Lifecycle Manager	589
10.3.2	Graphical User Interface	591
10.3.3	Command Line Interface	592
10.3.4	Web User Interface	593
10.3.5	Executing Tasks in a Distributed System	596
10.3.6	Optimized or Flexible Downtime	597
10.4	Downloading the Software	598
10.5	Installing an SAP HANA System	600
10.5.1	Users and Configuration Files Created during the Installation	600
10.5.2	Preparing the Installation Media	601

10.5.3	Installing a Single System	603
10.5.4	Installing a Multiple-Host System	612
10.5.5	Postprocessing Actions	617
10.6	Updating an SAP HANA System	620
10.6.1	Preparing the Update Media	621
10.6.2	Updating a Single System	626
10.6.3	Updating a Multiple Host System	634
10.6.4	Updating Using System Replication and Zero-Downtime Maintenance	635
10.6.5	Postprocessing Actions	637
10.7	Uninstalling SAP HANA	639
10.8	Installing and Updating the SAP HANA Client	640
10.8.1	Installation on UNIX, Linux, and macOS	641
10.8.2	Installation on Microsoft Windows	642
10.9	Installing and Updating the SAP HANA Cockpit	644
10.10	Installing and Updating SAP HANA Studio	648
10.11	Summary	651
11	Database Objects	653
11.1	Object Attributes	654
11.1.1	Database Catalog	654
11.1.2	Schemas, Users, and Ownership	655
11.1.3	Object Names	657
11.1.4	The Object ID	658
11.1.5	Object Dependencies	658
11.2	Tables	659
11.2.1	Row Store and Column Store	660
11.2.2	Table Columns	662
11.2.3	Displaying the Structure of a Table	665
11.3	Indexes	667
11.4	Triggers	668
11.5	Constraints	669
11.6	Views	670
11.6.1	SQL Views	670
11.6.2	Column Views	674

11.7	Sequences	676
11.7.1	Usage Example	677
11.7.2	Numbering Gaps and Chronology	679
11.8	Stored Procedures	680
11.8.1	Creating a Stored Procedure	680
11.8.2	Calling the Stored Procedure	682
11.9	Functions	682
11.9.1	Creating a Function	683
11.9.2	Using the Function	684
11.10	Synonyms	686
11.11	Summary	687
12	Working with Tables	689
12.1	The Storage Models: Row and Column Stores	691
12.1.1	Row-Store Tables	691
12.1.2	Column Store Tables	693
12.1.3	Usage Guidelines: Row Store or Column Store?	694
12.2	Compression	695
12.2.1	Dictionary Compression	696
12.2.2	Optimized Compression	698
12.2.3	When Is Compression Used?	702
12.3	Delta Store and Delta Merge	703
12.3.1	The Delta Merge Process	704
12.3.2	When Does a Delta Merge Happen?	706
12.4	Partitioning	711
12.4.1	Partition Placement	712
12.4.2	Single-Level Partitioning	712
12.4.3	Multilevel Partitioning	723
12.4.4	Special Operations on Partitioned Tables	727
12.4.5	Heterogeneous Partitioning	732
12.4.6	How Many Partitions Should a Table Have?	733
12.5	Table Classification and Distribution	734
12.5.1	Table Classification	734
12.5.2	Table Placement Rules	736
12.5.3	Table Placement Location	737
12.5.4	Example of Table Placement	739

12.6 Memory and Persistence Layer	745
12.6.1 Loading Data into Memory	746
12.6.2 Preloading Tables	746
12.6.3 Loading and Unloading Tables	747
12.6.4 Savepoints	748
12.7 Special Table Types	749
12.7.1 Temporary Tables	749
12.7.2 Temporal Tables	756
12.7.3 Flexible Schema Tables	764
12.7.4 Table DUMMY	766
12.8 Indexes	768
12.8.1 Row-Store Index Types	769
12.8.2 Column-Store Index Types	771
12.8.3 Index Persistence to Disk	777
12.9 Multiversion Concurrency Control	778
12.9.1 Read Stability and MVCC	779
12.9.2 Row-Store Garbage Collection	780
12.9.3 MVCC in the Column Store	781
12.10 Export and Import	781
12.10.1 Output Formats	782
12.10.2 Export	782
12.10.3 Import	787
12.10.4 Export and Import with SAP HANA Cockpit	787
12.10.5 Export and Import with SAP HANA Studio	790
12.11 Consistency Checks	794
12.11.1 Stored Procedures	794
12.11.2 Manual Consistency Checks	795
12.11.3 Automatic Table Consistency Check	796
12.11.4 Results of the Checks	796
12.11.5 Consistency Checks at Startup	797
12.12 Metadata	797
12.12.1 Row and Column Store	797
12.12.2 Compression	797
12.12.3 Delta Merge	798
12.12.4 Partitioning	799
12.12.5 Table Classification and Distribution	800
12.12.6 Persistence	801
12.12.7 Temporary Tables	801
12.12.8 Temporal Tables	802
12.12.9 Indexes	803

12.12.10 Multiversion Concurrency Control	803
12.12.11 Export and Import	804
12.12.12 Consistency Checks	805
12.13 Summary	805
13 Security	807
13.1 Security Patches	809
13.2 User Management	810
13.2.1 Standard Database Users	811
13.2.2 Restricted User Accounts	812
13.2.3 Internal User Accounts	812
13.2.4 Creating and Managing User Accounts	814
13.2.5 The SYSTEM User	822
13.2.6 User Groups	824
13.3 Database Privileges	826
13.3.1 Privileges Managed in the Catalog	828
13.3.2 Privileges Managed in the Repository of SAP HANA	833
13.3.3 Privilege Types	834
13.4 Database Roles	842
13.4.1 Creating and Maintaining Roles	843
13.4.2 Catalog and Repository Roles	844
13.4.3 Standard Catalog Roles	847
13.5 Troubleshooting Authorization Issues	848
13.5.1 Finding Information on Granted Privileges and Roles	848
13.5.2 Tracing Missing Authorizations	849
13.6 Authentication Methods	854
13.6.1 Username and Password Authentication	855
13.6.2 LDAP Authentication	861
13.6.3 Supported Third-Party Authentication Providers	862
13.7 Auditing the Database	863
13.7.1 Audit Policies	864
13.7.2 System-Wide Audit Policies	866
13.7.3 Default Audit Policies	867
13.7.4 Configuration and Audit Policy Management	867
13.7.5 Recommended Audit Policies and Best Practices	875
13.8 Encryption	877
13.8.1 Server-Side Data Encryption	877

13.8.2	Persistency Encryption	890
13.8.3	Client Encryption Settings	894
13.8.4	Managing Certificates	900
13.9	Increase the System Isolation Level	902
13.10	Cross-Database Authorizations	906
13.11	Summary	909

14 Performance Analysis 911

14.1	System Performance Monitoring	912
14.1.1	SAP HANA Cockpit	912
14.1.2	SAP HANA Studio	915
14.1.3	DBA Cockpit	915
14.1.4	SQL Statements	916
14.2	SQL Statement Analysis	917
14.2.1	EXPLAIN PLAN	918
14.2.2	SQL Plan Cache and Expensive SQL Statements	922
14.2.3	PlanViz	930
14.2.4	DBA Cockpit	937
14.2.5	SQL Trace: Transaction ST05	939
14.3	Optimizer Statistics	946
14.4	Summary	948

15 Monitoring and Troubleshooting 949

15.1	Monitoring Views	950
15.1.1	Views in the SYS_DATABASES Schema	951
15.1.2	Reset Points for Monitoring Views	952
15.2	Statistics Service Views	953
15.3	Displaying System Views	955
15.4	Diagnostic Files	955
15.4.1	Trace Files	955
15.4.2	Dump Files	958
15.4.3	Additional Files	960
15.4.4	Viewing Diagnostic Files	960

15.5	Alerts	966
15.5.1	Statistics Views for Alerts	966
15.5.2	Displaying and Managing Alerts	968
15.6	SAP's SQL Statement Collection	978
15.6.1	Installing the Scripts	979
15.6.2	Structure of the SQL Scripts	980
15.6.3	Editing Scripts	981
15.6.4	Version and Feature-Dependent Scripts	983
15.6.5	Running the Scripts	983
15.7	Mini Checks	991
15.8	SQL Trace	994
15.8.1	Activating the Trace	995
15.8.2	Displaying the Trace	998
15.9	Additional Traces	1002
15.9.1	Database Trace	1002
15.9.2	User-Specific and End-to-End Traces	1004
15.9.3	Kernel Profile Trace	1004
15.9.4	Kernel Sentinel	1004
15.9.5	Client-Side Traces	1004
15.10	Summary	1005

Appendices 1007

A	Setting Up Your Trial System	1009
B	References and Further Reading	1023
C	The Authors	1035

Index	1037
-------------	------

Index

_SYS_STATISTICS schema	953	Application time-period tables	761
A		Architecture	57
ABAP	41	ARRAY type	665
ABAP parameters		Assign Privileges app	842
/dbshdb/quiesce_check_enable	415	Atomicity, consistency, isolation, and	
/dbshdb/quiesce_sleep_time	415	durability (ACID)	36
rdisp/max_wprun_time	524	Attributes	654
ABAP platform	42	Auditing	864
ABAP reports		base setup	869
/SDF/HANA_BW_SIZING	527	best practices	875
/SDF/HDB_SIZING	521	configuration	867, 874
ZNEWHDB_SIZE	521	default policies	867
Accelerators	43	enable	872
Active statements	939	grants and revokes	876
Admission control	185, 188	policies	864, 873
Advanced analytics	50	policy management	872
Advanced DSOs (aDSOs)	483	system-wide policies	866
Alert Definitions app	228	trails	865, 870, 874
Alerts	94, 226, 966	Authentication	854
alter schedule	977	JSON	863
change threshold	978	Kerberos	862
check times	974	LDAP	861
command line	976	SAML	862
configure	975	SAP Logon	863
configure e-mail	972	third-party	862
configure properties	971	username and password	855
DBA Cockpit	974	X.509	863
deactivate	977	Authorizations	848
definition	227, 969	cross-database	906
display	968	missing	849
display and manage	968	Auto restart	342
email recipient	970	Autocommit mode	178
example	967	Automated Predictive Library (APL)	51
history	973	Automatic Bug Reporting Tool	575
SAP HANA cockpit	968	Automerger	707
SAP HANA Studio	973	B	
statistics view	966	Backint	80, 266, 275
thresholds	973	Backup and recovery	78
Alerts app	228	Backup catalog	266–267, 302
Aliases	144	settings	282
for cd	144	snapshot	280
for directory navigation	144	Backup Configuration app	282
Allocators	168	Backups	78, 256, 355, 620
Analytic privileges	829, 840	Backint	275
Appliance	500, 536	configuration files	272
Application privileges	838	consistency	296

Backups (Cont.)

- diagnostic files* 315
- disk space* 539
- file systems* 273
- naming conventions* 262
- options compared* 281
- run* 284
- SAP HANA cockpit* 285
- SAP HANA Studio* 288
- schedule* 286–287
- sizing* 274
- snapshots* 276
- to file system* 79
- types* 259
- verify in verbose mode* 298

Base path 240

Basepath 138

BI Content 49

Binary data 665

Binary export/import 782

Binary exports 784

Bind variables 944

Bitemporal tables 763

Blocked Transactions app 99

Boolean 664

Buffer Cache Monitor app 479

Build packs 68

Bullion S 514

Business continuity 341

C

Capacity planning 519

Capture and replay 621

Cards 92

Catalog objects 102, 827, 829

- functions* 828
- indexes* 827
- procedures* 828
- schemas* 827
- sequences* 827
- synonyms* 827
- tables* 827
- triggers* 827
- views* 827

Catalog roles 829, 845, 847

Catalog views 132, 654

- CONSTRAINTS* 670
- OBJECT_DEPENDENCIES* 658
- REFERENTIAL_CONSTRAINTS* 670
- TABLE_COLUMNS* 665, 766, 797
- TABLE_GROUPS* 800

Catalog views (Cont.)

- TABLE_PARTITIONS* 799
- TABLE_PLACEMENT_RULES* 736
- TABLES* 665, 797, 799

CDS views 675

Central services instance 134

Certificate Collections app 901

Certificates 324, 900

Character datatype 766

Character strings 664

Charts 914

CHECK_CATALOG 795

CHECK_TABLE_CONSISTENCY 795

Client connections 338–339, 431

Client zones 335

Client-side traces 1004

Cloud deployments 496

Clustered encoding 699

Cockpit Manager app 86

Code pushdown 41

Codeployment of SAP HANA 515

Cold data 467–468, 490

Column store 33, 660–662, 691, 693

- efficiency* 35
- example* 35
- tables* 74
- when to use* 694

Column views 675

Command line

- add host* 447
- disable system replication* 389
- enable_session_recovery* 635
- executing SQL statements* 119
- GetSystemInstanceList* 117
- HDB* 113
- HDB info* 115
- hdbaddhost* 595
- hdbbackupcheck* 118, 296
- hdbbackupcheckpack* 118, 297
- hdbbackupdiag* 118, 270, 294
- hdbcons* 115, 118
- hdbinst* 595, 642–643, 648
- hdbkeystore* 118
- hdblcm* 592
- hdblcmweb* 593
- hdbblogdiag* 118
- hdbltracediag* 118
- hdbmcutil* 118, 301
- hdbmodify* 595
- hdbnsutil* 118, 356
- hdbodbc_cons* 118
- hdbpersdiag* 118, 256

Command line (Cont.)

- hdbreg* 595
- hdbremovehost* 595
- hdbrename* 595
- hdbsdutil* 118
- hdbsetup* 642–643, 648
- hdbsql* 118–119, 899
- hdbsqldbc_cons* 118
- hdbuninst* 595
- hdbupd* 595
- hdbupdprep* 595
- hdbuserstore* 118, 432, 898
- hdbwreplayer* 118
- install single system* 608
- libsapcrypto.so* 625
- monitor system replication* 379
- monitoring and administration* 115
- multitier system replication* 399
- register secondary system* 377
- rsecsfx* 883
- SAPCAR* 626
- sapcontrol* 114, 116
- start instance* 114
- start/stop instance* 145
- start/stop tenant database* 149
- system configuration* 117
- system replication* 376
- takeover and failback* 385
- tools* 113
- unregister secondary system* 389
- updates* 630
- verify replication* 377
- xs* 647

Commands 177

Comma-separated values (CSV) 782

CommonCryptoLib 644

Composite index 774

Compression 363, 695, 797

- clustered encoding* 699
- dictionary* 696
- indirect encoding* 700
- optimized* 698
- prefix encoding* 701
- RLE* 698
- sparse encoding* 700
- where to use* 702

Concat attribute 771, 775

Configuration parameters 472

- *_audit_trail_type* 866
- alert_audit_trail_type* 868
- allowed_sender* 360
- audit_statement_length* 868

Configuration parameters (Cont.)

- authentication_methods* 854
- backup_encryption* 888
- basepath_catalogbackup* 267, 276
- basepath_databackup* 261
- basepath_datavolume* 239
- basepath_datavolumes* 233, 584
- basepath_logbackup* 264, 276
- basepath_logvolumes* 233, 584
- basepath_persistent_memory_volumes* 472
- catalog_backup_parameter_file* 267
- catalog_backup_using_backint* 276
- check_max_concurrency* 253
- check_max_concurrency_percent* 253
- consistency_check_at_startup* 253
- critical_audit_trail_type* 868
- cross_failover_group* 441, 460
- data_backup_buffer_size* 271
- data_backup_max_chunk_size* 261
- database_isolation* 322, 904
- datashipping_logsize_threshold* 360
- datashipping_min_time_interval* 360
- datashipping_parallel_channels* 363
- datashipping_snapshot_max_retention_time* 350, 360
- datavolume_stripping_size_gb* 239
- default_audit_trail_path* 868
- default_audit_trail_type* 868, 872
- detailed_error_on_connect* 858
- emergency_audit_trail_type* 868
- enable* 365
- enable_auto_log_backup* 242, 246, 265
- enable_automatic_unload* 253
- enable_data_compression* 362
- enable_full_sync* 356, 361
- enable_log_compression* 361
- enable_log_retention* 350, 362, 392
- enable_parallel_backup_encryption* 271
- enable_session_recovery* 364
- enable_startup_consistency_check* 252
- encryption_config_control* 888
- enforce_ssl_database_replication* 322
- force_first_password_change* 858
- global_allocation_limit* 365, 619
- global_auditing_state* 868
- internal_check_max_concurrency* 253
- internal_hostname_resolution* 429
- large_job_threshold* 253
- last_used_passwords* 858
- listenerinterface* 323–324, 357, 429
- log_backup_interval_mode* 265

Configuration parameters (Cont.)

- log_backup_timeout_s* 242, 246, 265
- log_backup_using_backint* 276
- log_encryption* 888
- log_mode* 241, 246, 264, 325
- log_preformat_segment_count* 246
- log_recovery_resume_point_interval* 315
- log_segment_size_mb* 246
- logshipping_async_buffer_size* 361
- logshipping_async_wait_on_buffer_full* 361
- logshipping_max_retention_size* ... 350, 362
- logshipping_timeout* 361
- max_duration* 253
- max_log_backup_size* 265
- max_num_large_jobs* 253
- max_size* 477
- max_size_rel* 477
- max_trace_file_size* 316
- max_trace_files* 316
- maximum_invalid_connect_attempts* ... 858
- maximum_password_lifetime* 858
- maximum_unused_initial_password_lifetime* 858
- maximum_unused_productive_password_lifetime* 858
- minimal_password_length* 857
- minimal_retention_period* 865
- minimum_password_lifetime* 858
- operation_mode* 348, 362
- parallel_data_backup_backint_channels* 276
- password_expire_warning_time* 858
- password_layout* 857
- password_lock_for_system_user* ... 824, 858
- password_lock_time* 858
- PERSISTENCE_DATAVOLUME_PARTITION_MULTIPATH* 239
- persistence_encryption* 888
- preload_column_tables* 360
- reconnect_time_interval* 361
- replicate* 365
- reserved_instance_numbers* 336
- savepoint_interval_s* 232, 234
- sr_audit_trail_type_cstable_override* ... 868
- sr_enable_tracking* 408
- sr_memory_tracking* 408
- sr_total_statement_memory_limit* 408
- ssl* 322, 895
- startup_consistency_check_timeout* 253
- startup_error_restart_retries* 458
- startup_error_shutdown_instance* 458

Configuration parameters (Cont.)

- systemdb_separated_sql_port* 338
- table_default* 472
- table_unload_action* 473
- targets_for_'database name'* 908
- timetravel_call_takeover_hooks* 412
- timetravel_max_retention_time* 411
- timetravel_snapshot_creation_interval* 411
- usage* 82

Configuration System Properties app 282

Consistency checks 794, 805

- at startup* 797
- automatic table checks* 796
- manual* 795
- results* 796

Constraints 669

Converged infrastructure 499

Core-to-memory ratio 420

Corruptions 248

- logical* 249
- physical* 254

Cost calculator 1018

CPU 36, 94, 419

- monitoring* 912
- sizing* 535

Crash dump files 958

- SAP HANA Studio* 962

Cross-database access 62

Cross-takeover groups 441

Current Table Distribution app 451

D

Data aging 466

Data anonymization 901

Data archiving 466

Data Encryption app 271, 901

Data footprint 425

Data integration 54

Data Lifecycle Manager 485, 491

Data mart 42

Data masking 901

Data persistency 70, 342

Data provisioning 46

Data replication 46

Data statistics 946

Data tiering 465

- options* 466, 469

Data types 662

Data volumes 138, 231, 235, 238, 581

- disk space* 537

Data volumes (Cont.)

- partitioning* 238
- unusable* 292
- verify* 255

Data warehouses 480

Database administration 217

Database as a service (DBaaS) 39

Database backups 260

Database Backups app 268

Database catalog 60, 654

Database directory 60

Database Directory app 83, 94, 157

Database disk usage alert 970

Database Explorer app 315

Database exports 540

Database groups 87, 90

Database isolation 62

Database Licenses app 101

Database Management app 92–93, 95, 823

- activities* 96
- create tenant database* 97

Database objects 653

Database Overview app 91, 158, 426, 968

- database configuration* 97

Database recovery 78, 290

- command line* 310
- considerations* 291
- encrypted backup* 333
- point in time recovery* 299
- preparation* 294
- required files* 294
- resume canceled recovery* 314
- SAP HANA cockpit* 301, 307
- SAP HANA Studio* 304, 311
- scenarios* 292
- sequence* 292
- system database* 301
- tenant databases* 306

Database replication

- configure* 322
- data copied* 321
- postprocessing* 329
- prerequisites* 322
- select source and target* 323
- statuses* 328
- tenant databases* 325–326
- tenant replication* 324

Database status 93

Database trace 1002

Database-aware snapshot 278

Database-unaware snapshot 278

Date and time data 664

DBA Cockpit 124, 937

- functionalities* 125
- load monitor* 127
- manage parameteres* 159
- monitor memory usage* 174
- overview screen* 124
- SQL editor* 126
- threads* 937

DBA Planning Calendar 252

Delimited name 657

Delta backups 78, 260, 302, 308

Delta merge 74, 703, 740, 798

- automerger* 707
- critical merge* 708
- forced merge* 709
- hard merge* 709
- memory merge* 710
- new delta store* 704
- process* 704–705
- smart merge* 709

Delta shipping 348

Delta store 703

Deployment options 506

- compared* 517
- MCOD* 509
- MCOS* 510
- MDC* 508
- SCOS* 507
- virtualization* 512

Design-time objects 838

Development platform 44

Diagnostic files 955

- autorefresh* 962
- DBA Cockpit* 966
- merge* 962
- SAP HANA cockpit* 960
- SAP HANA Studio* 962
- support information* 964–965
- view* 960

Dictionary 697

Differential backup 78, 260, 263

Direct Extract Connection 49

Directories 134

- database* 137
- INI files* 142

Dirty read 779

Disaster recovery 256, 342

Disk usage 94

Disk Volume Monitor app 236, 242

Distributed scenario 357

Distributed systems 71, 427

- executing tasks* 596

- Documentation 1024, 1027
 - Double-buffering method 704
 - Download manager 1011
 - Downloading software 598
 - Dual-stack system 42
 - DUMMY table 684, 766–767
 - Dump files 958, 960
 - Dynamic random-access memory (DRAM) 470
- ## E
- Eager eviction 923
 - Eclipse 104
 - Editions 37
 - Embedded statistics server 252
 - Encryption 618, 877
 - activate root keys 886
 - backups 270
 - client settings 894
 - configure 881
 - configure default settings 888
 - data and log volumes 891
 - disable 893
 - enable services 888
 - JDBC and ODBC drivers 897
 - persistency 890
 - root keys 884
 - server-side 877
 - settings for tenant database 888
 - SSFS master keys 882–883
 - End-to-end trace 1004
 - Enforced license key 61
 - Enqueue replication server 558
 - Enqueue server 558
 - Enterprise network 502
 - Enterprise storage 502
 - Environment variables 642
 - HDB_INSTALLER_TRACE_FILE 603, 612
 - LD_LIBRARY_PATH 642
 - LIBPATH 642
 - SHLIB_PATH 642
 - Execution plans 918
 - display 944
 - example 919
 - how to read 920
 - result 920
 - size and cost 921
 - SQL plan cache 928
 - Expensive SQL statements 922, 925
 - DBA Cockpit 938
 - SAP HANA cockpit 928
 - Explain plan 917
 - EXPLAIN_PLAN_TABLE 918
 - Export data 781–782
 - catalog only 786
 - export multiple objects 784
 - identify objects 786
 - parallel 786
 - partial 786
 - scrambling 785
 - single object 783
 - to compressed archive 786
 - Extended tables 485
 - Extension nodes 480, 533
 - External Machine Learning Library (EML) 51
 - Extract, transform, and load (ETL) 49
- ## F
- Failback 381
 - Failback execution 464
 - Failover 549
 - Fallback snapshots 94, 223, 278, 367
 - FAQ notes 967
 - Fault tolerance 553
 - Fencing 433
 - Fiber Channel Storage Connector 435
 - File system layout 583
 - File system recommendations 581
 - File-based backups 261
 - Flexible schema tables 764
 - Flight data model 918
 - Fujitsu physical partitioning 514
 - Full backup 78
 - Full-text index 777
 - Functions 682
 - aggregation statements 685
 - create 683
 - DML 685
 - example 683
 - in queries 685
 - use 684
- ## G
- gawk command 957
 - General Data Protection Regulation (GDPR) 877
 - Geographic information system (GIS) 52
 - GET_CHECK_ACTIONS 795
 - GET_NUM_SERVERS() function 714
 - Global allocation limit 166, 619
 - Global temporary tables 750–751

- Globally unique ID (GUID) 853
 - Graph data processing 52
 - Graph database 52
 - Groups 133
- ## H
- Hadoop 491–492
 - Hadoop Cluster 493
 - Hadoop Distributed File System (HDFS) 491
 - Hard shutdown 147
 - Hard stop 94
 - Hardware requirements 36, 566
 - Hardware virtualization 513
 - Hash key 715
 - HDB
 - info 150
 - start 145
 - hdbcons 202
 - hdbdaemon process 458
 - hdblcm 137
 - hdbnsutil 204
 - hdbrsutil 204
 - hdbsql 119
 - aligned output 122
 - autocommit 124
 - command line 991
 - display session information 122
 - multiline input 123
 - turn off pager 123
 - use 121
 - Heap memory 168
 - Heartbeats 433
 - High availability 76, 341–342, 419
 - cluster 547
 - DNS binding 550
 - IP redirection 550
 - High isolation 134
 - High isolation mode 446
 - History tables 756, 759
 - Hitachi Advanced Server DS7000 514
 - Hitachi LPAR 513
 - Hive MetaStore 493
 - Host autofailover 72, 433
 - Host autotakeover 422, 433
 - Host failover 443
 - groups 448
 - Host Failover app 438
 - Host grouping 425
 - Host layer 98
 - Host name resolution 428
 - Host roles 425, 440
 - Host takeover detection 458
 - Host worker group 448
 - HOST_LOAD_HISTORY_HOST 916
 - HOST_LOAD_HISTORY_SERVICE 916
 - Hostname resolution 356, 358
 - Hosts 419
 - adding 445
 - layer 163
 - parameters 614
 - remove 455–456
 - Hot data 467
 - HPE nPartitions 514
 - Huawei Virtualization 513
 - Hyperconverged infrastructure 499
 - Hypervisor 1011
- ## I
- IBM Power VM
 - Live Partition Mobility 556
 - LPARS 513, 556
 - Import data 781, 787
 - Imported packages 839
 - Incremental backup 78, 260, 263
 - Index server 66, 426, 449–450
 - roles 440
 - Indexes 36, 667, 768, 803
 - BTREE 769
 - BTREE CPB 770
 - column store 771
 - full-text 776
 - inverted 771
 - number 668
 - persistence to disk 777
 - row-store 769
 - size 667
 - types 769
 - Indirect encoding 700
 - Infrastructure
 - architecture 495
 - design 536
 - standardization 504
 - Infrastructure as a service (IaaS) 495–496
 - Input and output (I/O) 667
 - Installation 563
 - check 617
 - create database admin 618
 - deactive SYSTEM user 619
 - disk space requirements 539
 - multiple host system 612–616
 - postprocessing actions 617
 - prepare 565
 - prepare media 601

- Installation (Cont.)
 - SAP HANA system* 600
 - single system* 603, 605–606, 609, 611–612
 - user and configuration files* 600
 - verify process list* 617
 - Instance 63, 131
 - administration* 131
 - check status* 150
 - number* 133
 - start and stop* 145
 - structure* 132
 - Inter Process Memory Management (IPMM) 959
 - Internal networks 430, 544
 - Internal user accounts 812
 - Inverted hash index 772
 - Inverted individual indexes 772
 - Inverted value index 772
 - Isolation level 780
 - high* 903
 - revert* 905
- ## J
- Java 41
 - Java Database Connectivity (JDBC) 44, 432
 - Java SDK 1010
 - JDBC trace 1005
 - JobExecutor 191
 - JobWorker thread 191
 - Join engine 935
 - Join statistics 946
- ## K
- Kernel profile trace 1004
 - Kernel sentinel 781, 1004
 - Key performance indicators (KPIs) 914
 - Knowledge base articles (KBAs) 1028
- ## L
- Landscape management 316
 - Large objects (LOBs) 665, 691
 - Latency 356
 - Layers 98
 - Lazy loading 747
 - LDAP 815
 - Least recently used (LRU) algorithm ... 165, 745
 - Lenovo FlexNode 514
 - License keys 61, 112, 291
 - after deregistering* 390
 - install* 619
 - Lifecycle management tools 588
 - Linux kernel parameters 578
 - Load balancing 73, 725
 - Load graph 915
 - Load management 184
 - Load Unit Configuration app 479
 - Loading 746
 - column store* 746
 - row store* 746
 - Local temporary tables 751
 - Locks 182
 - modes* 183
 - types* 182
 - waits and timeouts* 184
 - Log backups 264–265, 305
 - Log files
 - backint.log* 261
 - backintcheck.log* 297
 - backup.log* 261, 314
 - backupcheck.log* 297
 - Log modes 206
 - Log segments 140, 211, 243, 264
 - DBA Cockpit* 214
 - monitor* 212
 - SAP HANA cockpit* 212
 - SAP HANA Studio* 214
 - states* 215
 - Log volumes 139, 231, 241, 581
 - disk space* 538
 - full* 244
 - no free space* 245
 - unusable* 293
 - Log-based data replication 50
 - Logging 205
 - disable* 207
 - full log* 208
 - log modes* 206
 - log volumes* 209
 - normal mode* 207
 - overwrite mode* 206
 - segments* 206
 - Logical consistency 249
 - Logical error 293
 - Logreplay 348
 - lscpu (command) 189
- ## M
- Machine learning 51
 - Maintenance revisions 564
 - Manage Database Configuration app 230
 - Manage Services app 94, 224, 337, 451
 - Management zones 335
 - Master host takeover 461–462
 - Master name server 426, 458
 - Materialization 921
 - Memory 94, 745
 - allocation* 166, 605
 - allocators* 168
 - for SQL statement* 170
 - heap and shared* 168
 - host statistics* 171
 - layout* 165
 - loading data* 746
 - management* 165
 - monitor usage* 172
 - monitoring* 912
 - pool* 166
 - resident* 172
 - shortage* 167
 - sizing* 534
 - SQL scripts for monitoring* 176
 - threshold* 926
 - tracking* 926
 - virtual* 172
 - Merge motivations 706
 - Merge tokens 706
 - mergedog 702, 707
 - Message server 558
 - Metadata 797
 - Mini checks 991
 - configuration category* 993
 - multitenant configuration* 992
 - SAP Notes* 992
 - Mobilink 48
 - Monitoring 91, 949
 - Monitoring views 132, 241, 246, 916, 950
 - M_ADMISSION_CONTROL_EVENTS* 187
 - M_ADMISSION_CONTROL_QUEUES* 187
 - M_ADMISSION_CONTROL_STATISTICS* 187
 - M_CONFIGURATION_PARAMETER_VALUES* 138
 - M_CONNECTION_STATISTICS* 178
 - M_CONNECTIONS* 178, 198
 - M_CONTEXT_MEMORY* 171
 - M_CONTEXT_MEMORY_RESET* 171
 - M_CS_ALL_COLUMNS* 774
 - M_DATABASES* 150
 - M_DELTA_MERGE_STATISTICS* 798
 - M_EXPENSIVE_STATEMENTS* 927
 - M_HEAP_MEMORY* 168
 - M_HOST_RESOURCE_UTILIZATION* 171
 - M_INIFILE_CONTENT_HISTORY* 157

- Monitoring views (Cont.)
 - M_INIFILE_CONTENTS* 157
 - M_INIFILES* 143
 - M_LOAD_HISTORY_HOST* 916
 - M_LOAD_HISTORY_INFO* 917
 - M_LOAD_HISTORY_SERVICE* 916
 - M_LOG_SEGMENTS* 211
 - M_OBJECT_LOCKS* 184
 - M_RECORD_LOCKS* 184
 - M_SAVEPOINTS* 801
 - M_SEQUENCES* 676
 - M_SERVICES* 951
 - M_SHARED_MEMORY* 170
 - M_SYSTEM_DATA_STATISTICS* 948
 - M_TABLE_PARTITION_STATISTICS* 800
 - M_TABLE_PARTITIONS* 713, 726, 800
 - M_TABLES* 797, 799
 - M_TRANSACTIONS* 179
 - Monitoring views*
 - M_SQL_PLAN_CACHE* 924
- Multidatabase containers (MDCs) 507, 517
- Multi-Host Health app 437
- Multilevel partitioning 75, 723
- Multiple components on one system (MCOS) 507, 518
- Multiple components one database (MCOD) 507, 518
- Multiple node systems 421
 - architecture* 422
- Multiple-host systems 419–420
 - administration and monitoring* 436
 - file system layout* 427
 - monitoring* 437
 - start and stop* 436
- Multiplex grid option for SAP IQ 490
- Multitier system replication 390
 - architecture* 390
 - command-line interface* 399
 - monitor* 399
 - replication mode* 391
 - restore* 404
 - SAP HANA cockpit* 394
 - SAP HANA Studio* 397
 - takeover* 401
 - three-tier* 393
 - two-tier* 396
- Multitiered replication 77
- Multivalued types 665
- Multivolume concurrency control (MVCC) 690, 778, 803
 - column store* 781
 - read stability* 779
 - row store* 780

N

Name server 66, 426, 458
 roles 440
 Native catalog objects 826
 Native packages 839
 Near-line storage 489
 Network bandwidth 356
 Network configuration 334
 Network layout 428, 540
 for internal communication 429
 Network performance 356
 Network ports 335
 Network Time Protocol (NTP) 580
 Network zones 334, 540
 backup zone 546
 client zone 542
 internal zone 543
 storage zone 545
 Network-attached storage (NAS) 334, 422, 434
 Networked file system (NFS) 428, 435
 No-logging retention tables 754
 No-logging tables 752
 Nonuniform memory access
 (NUMA) 188, 575
 NULL 662
 Numeric data types 663
 Nutanix Acropolis Hypervisor 513

O

Object privileges 837
 CREATE ANY ON OWN SCHEMA 811
 SELECT 88
 Objects 653
 attributes 654
 dependencies 658
 indirect dependency 658–659
 naming rules 657
 OID 658
 owners 655–656
 ownership 814
 schemas 655
 ODBC trace 1005
 On-demand software 497
 Online analytical processing (OLAP) 33
 query 34
 Online transaction processing (OLTP) 33
 On-premise deployments 498
 Open Database Connectivity (ODBC) 44, 432
 Operating system platforms 557

Optimized or flexible downtime 597
 Optimizer statistics 946
 maintain 946
 metadata views 947
 plan pinning 947
 sketch 947
 statement hints 947
 Oracle VirtualBox 39
 OS commands 39
 getenforce 574
 groupadd 903
 ulimit 573
 unzip 601
 useradd 903
 OS process 63
 hdbcompilesrv 64
 hdbcontroller 68
 hdbdaemon 63
 hdbdisrv 68
 hdbdocstore 64
 hdbdpsrv 65
 hdbessrv 65
 hdbindexsrv 64
 hdbisrv 65
 hdbnamesrv 63
 hdbpreprocessor 64
 hdbscriptsrv 64
 hdbstreamingsrv 65
 hdbwebdispatcher 64
 hdbxcontroller 65
 hdbxengine 65
 hdbxexeagent 65
 hdbxsexecagent 68
 hdbxsuaasrv 65, 68
 sapstartsrv 64

P

Package privileges 829, 838
 Page management 234
 Page sizes 692
 Pages 692
 Parameters 141, 186
 DBA Cockpit 159
 layers 141
 maintenance 152
 SAP HANA cockpit 157
 SAP HANA Studio 161
 setParameter.py 162
 SQL statements 153
 unsupported values 164
 Partition handling 74

Partitioning 72
 hash 75
 heterogenous 76
 multilevel 75
 range 75
 round robin 75
 single level 75
 Partitioning key 716
 Partitions 711, 799
 add new 239
 adding 729
 dropping 731
 dynamic range partitioning 721
 hash partitioning 714
 hash-hash 726
 hash-range partitioning 724
 heterogeneous partitioning 732
 moving between nodes 730
 multilevel 723
 number of 733
 OTHERS (with dynamic range) 722
 OTHERS partition (range) 719
 placement 712
 pruning 711, 721
 range conditions 718
 range partitioning 718
 range-range 727
 repartitioning 730
 round-robin partitioning 712
 round-robin-range 726
 row migration 728
 SAP BW systems 734
 single-level partitioning 712
 tables 727
 Password policies 855
 blacklist 859
 user group level 856
 Passwords 618
 Performance 363
 Performance analysis 911
 Performance Monitor 99
 Persistence 801
 Persistence layer 745
 Persistency files 548
 Persistent memory 470–471, 878
 database objects 473
 monitoring views 474
 overview 470
 parameters 472
 sizing 471
 unloading tables 474
 Persistent Memory Monitor 475
 Personal security environment
 (PSE) 321, 900
 PKI SSFS store 367
 Placement rules 736
 Plan pinning 947
 PlanViz 929–930
 executed plan 933
 export plan to file 933
 export to graphics file 934
 operator list 936
 performance trace 937
 Plan operator 934
 PLV file 933
 sample query 930
 tables used 936
 time metrics 935
 timeline 935
 Platform as a service (PaaS) 495, 497
 Point-in-time recovery 303
 Port numbers 137, 337
 Ports 136
 Predictive Analysis Library (PAL) 51
 Predictive analytics 51
 Prefix encoding 701
 Primary application server 559
 Primary key constraints 758, 775
 Primary system 394
 automatically register 404
 Primary tables 759
 Privilege Assignment app 829
 Privileges 826
 analytic 840
 application 838
 CREATE SCHEMA 835
 granting 828
 managed in catalog 828
 managed in repository 833
 object 837
 on users 842
 package 839
 revoking 831
 system 834
 types 834
 Process list 617
 Process tree 135
 Processor affinity 188
 configure 189
 Processor topology 189
 Product Availability Matrix (PAM) 37, 567
 Program text 165
 Proxy schemas 381
 PUBLIC schema 655

- PUBLIC synonyms 686
 - Python scripts
 - convertMDC.py* 904
 - getTakeoverRecommendation.py* 382
 - hdbrecovercheck.py* 299
 - landscapeHostConfiguration.py* 381–382, 442
 - recoverSys.py* 303
 - setParameter.py* 119
 - systemReplicationStatus.py* 381–382
- Q**
- Query isolation 74
 - Quick Sizer 519
- R**
- READ COMMITTED 780
 - Reclaiming space 246
 - Recovery Database app 310
 - Recovery point objective (RPO) ... 256, 258, 342
 - zero* 379
 - Recovery procedures 290
 - Recovery time objective (RTO) 256, 258, 342
 - Red Hat Enterprise Linux for
 - SAP Solutions 566, 574, 1009
 - abrt* 575
 - abrt-addon-ccpp* 575
 - additional packages* 577
 - automatic NUMA balancing* 575
 - configure C-states for lower latency* 576
 - kernel samepage merging* 577
 - SELinux* 574
 - Transparent Huge Pages* 576
 - tuned-profiles-sap-hana* 574
 - Red Hat Enterprise Linux High Availability
 - add-on* 549
 - Red Hat Virtualization KVM 513
 - Redo logs 36, 205, 258
 - Reference architecture 557
 - Registered databases 86
 - Relational database management system
 - (RDBMS) 36, 46
 - Release cycles 563
 - Relocation 316
 - Remote data sources 54
 - Remote identity 908
 - Renaming 318
 - Reorganization plan 741
 - REPEATABLE READ 780
 - Repository catalog objects 827
 - Repository roles 820, 845
 - Reset points 952
 - create* 952
 - Reset views 952
 - Retention periods 865
 - Retention policy 284
 - Revisions 564, 809
 - Role Assignment app 816
 - Role Management app 843
 - Roles
 - catalog* 845
 - CONTENT_ADMIN* 847
 - create and maintain* 843
 - database* 842
 - design-time objects* 846
 - MODELLING* 848
 - MONITORING* 847
 - PUBLIC* 847
 - repository* 845
 - RESTRICTED_USER_JDBC_ACCESS* 848
 - RESTRICTED_USER_ODBC_ACCESS* 848
 - SAP HANA Studio* 844
 - SAP_INTERNAL_HANA_SUPPORT* 848
 - Root backup keys 333, 893
 - Root user 448, 573, 608
 - Round-robin partitioning 713
 - Row store 33, 35, 660, 691
 - when to use* 694
 - ROWID 692
 - Run-length encoding (RLE) 698
- S**
- SAP Business Technology Platform
 - (SAP BTP) 498
 - SAP Business Warehouse (SAP BW) 483, 735
 - SAP BusinessObjects 42, 49
 - SAP BW/4HANA 482
 - SAP Central Services 558
 - SAP Cloud Appliance Library 1013
 - accounts* 1017
 - cost forecast* 1018
 - instance operation* 1020
 - options* 1015
 - overview* 1014
 - private key* 1019
 - setup* 1016
 - solution overview* 1017
 - virtual machine* 1018
 - SAP control connection 88
 - SAP Data Services 49
 - SAP EarlyWatch Alert 83, 128

- SAP ERP 44
- SAP Fiori 81
- SAP HANA Application Function Library
 - (AFL) 67
- SAP HANA client 428, 640
 - installation on Linux and macOS* 641
 - installation on Windows* 642
- SAP HANA Cloud 39
- SAP HANA cockpit 60, 81
 - add roles* 457
 - add service* 225, 449
 - alerts* 226
 - auditing* 868
 - authorization dependency viewer* 838
 - backup schedules* 285
 - blacklisted parameters* 229
 - certificate store* 900
 - change port* 338
 - configuration* 85
 - configuration parameters* 98
 - configure host failover* 443
 - configure persistent memory* 475
 - configure tenant replication* 322
 - connection URLs* 647
 - copy database* 330
 - create database groups* 91
 - create empty tenant* 330
 - create tenant database* 219–220
 - create tenant using replication* 325
 - create users* 87
 - data encryption* 888, 892
 - database backup location* 331
 - database licenses* 619
 - Database Management page* 84
 - Database Overview page* 84
 - disable system replication* 387
 - disk volume monitor* 450
 - enable system replication* 369
 - executing SQL statements* 102
 - export data* 787
 - fallback snapshot* 223
 - import data* 789
 - install and update* 644
 - install license* 101
 - insufficient privilege details* 853
 - maintenance strategy* 645
 - manage database configuration* 239
 - Manage Landscape section* 83
 - manage parameters* 157
 - manage services* 94
 - memory paging monitor* 478
 - monitor disk volume* 236, 242
- SAP HANA cockpit (Cont.)
 - monitor memory usage* 172
 - monitor persistent memory usage* 475
 - monitoring and administration* 91
 - Monitoring Landscape section* 82
 - multiple-host system* 437
 - multitier system replication* 394
 - navigation* 81
 - network configuration* 647
 - network monitor* 431
 - password blacklist* 860
 - password policy* 856
 - performance monitoring* 99
 - platform lifecycle management* 595
 - reclaim space* 247
 - recover database* 301
 - recover tenant* 307
 - register database* 88
 - reinitialize secondary system* 372
 - remove services* 450
 - reset system password* 823
 - restricted features* 231
 - security and user management* 810
 - security checklist* 809
 - start/stop instance* 148
 - stop database* 94
 - system configuration* 96
 - system replication* 369
 - system replication overview* 370
 - table distribution* 451
 - takeover and fallback* 383
 - tenant using backup* 330
 - unregister tenant database* 223
 - user group management* 825
 - users* 87
- SAP HANA cockpit manager 84, 86
- SAP HANA data warehousing
 - foundation* 466, 485, 491
- SAP HANA database 70, 557
- SAP HANA database explorer 102, 960
 - trace configuration* 852
- SAP HANA deployment infrastructure
 - (HDI) 45
- SAP HANA dynamic tiering 76, 468, 484
 - architecture* 485
 - backup and recovery* 488
 - common database* 486
 - configuration* 487
 - dedicated server* 486
 - high availability* 488
 - overview* 484
 - use case* 485

- SAP HANA extension node 480
 - architecture 481
 - backup 483
 - configuration 482
 - hardware guidelines 483
 - high availability 483
- SAP HANA hardware and cloud measurement
 - tool 584
 - create system 587
 - install 585
 - measurement analysis 587
 - space requirements 586
 - start via command line 586
- SAP HANA Hardware Directory 505
- SAP HANA instance 57
- SAP HANA Interactive Education (SHINE) 45
- SAP HANA lifecycle manager 137, 316, 428, 430, 471, 589, 603, 623
 - command-line interface 592, 608
 - configuration options 591
 - GUI 591
 - installation medium 590
 - remove hosts 456
 - resident version 590
 - upload archives 624
 - web user interface 593
 - wrapper 595
- SAP HANA native storage extension ... 468, 476
 - advisor 479
 - hot and warm data 477
 - page loadable 477
 - SAP S/4HANA 478
- SAP HANA One 39
- SAP HANA remote data sync 48
- SAP HANA smart data access 54, 492–493
- SAP HANA smart data integration 55
- SAP HANA smart data quality 55
- SAP HANA start sequence 95
- SAP HANA Studio 65, 103
 - activities 107
 - administration 106
 - auditing 871
 - backup and recovery 288
 - backup console 268, 296
 - check for SAP HANA component updates 623
 - client encryption 895
 - configure host failover 444
 - configure system replication 373
 - connect third to secondary system 398
 - deprecated 651
 - diagnosis files 850
- SAP HANA Studio (Cont.)
 - disable system replication 387–388
 - editor area 107
 - enable system replication 398
 - export data 790
 - folder structure 105
 - hdbcons 203
 - hosts 439
 - import data 793
 - install and update 648
 - install licenses 112
 - install on Windows 648
 - logical system 373
 - manage parameters 161
 - manage services 108
 - monitor memory usage 174
 - monitoring and administration 105
 - multitier system replication 397
 - number of rows 931
 - parameter configuration 109
 - password blacklist 860
 - performance monitoring 111
 - platform lifecycle management 595
 - primary system 375
 - recover system database 304
 - recover tenant database 311
 - register secondary system 374
 - register system 104
 - restart instance 147
 - security editor 868
 - service market place 622
 - setup and configuration 104
 - SQL console 112
 - start instance 146
 - start/stop instance 145
 - stop instance 147
 - stop system 108
 - system configuration 109
 - system monitor 106
 - system replications 373
 - table distribution 451
 - takeover and failback 384
 - toolsets 103
 - trace configuration 849
 - trace files 148
 - unregister secondary system 387
 - users 817–818
 - workspace 650
- SAP HANA tailored data center
 - integration 501, 503, 537
- SAP HANA Web-Based Development
 - Workbench 817, 833

- SAP HANA XS 45, 67, 485, 838
 - engine 67
- SAP HANA XSA 45
 - application instances 69
 - backing services 70
 - controller 68
 - execution agent 68
 - platform router 69
 - runtime platform 67
 - scale out 68
 - service 87
 - user account and authentication 69
- SAP HANA, active/active read-enabled
 - option 38, 406, 636
 - activate 406
 - SAP HANA cockpit 408
 - secondary systems 410
 - system replication 409
- SAP HANA, developer edition 39
- SAP HANA, enterprise edition 38
- SAP HANA, express edition 39, 1009
 - install 1010
- SAP HANA, platform edition 37
 - components 37
- SAP HANA, predictive option 51
- SAP HANA, runtime edition 38
- SAP HANA, spatial and graph option 51
- SAP HANA, standard edition 38
- SAP HANA, streaming analytics
 - option 49, 53
- SAP Help Portal 1023
- SAP Host Agent 324, 583, 616
- SAP IQ 34, 489–490
- SAP Landscape Management 128, 504
- SAP liveCache 43, 599
- SAP LT Replication Server 46–47
 - advantages/disadvantages 47
- SAP MaxDB 43
- SAP NetWeaver 41
- SAP NetWeaver Application Server
 - for ABAP 43
 - zero downtime 414
- SAP Notes 1028
 - 1275776 570
 - 2593824 572
 - 2684254 570
 - 2788495 569
- SAP One Support 83
- SAP ONE Support Launchpad 1028
- SAP ONE Support portal 1023
- SAP Replication Server 50
- SAP S/4HANA 42
 - scale-out 423
- SAP S/4HANA sizing report 423
- SAP Solution Manager 127, 627
- SAP SQL Anywhere 48
- SAP start service 150
- SAP Test Data Migration Server
 - (SAP TDMS) 46
- SAP Vora 492
- SAP Web IDE for SAP HANA 45
- sapcontrol 151
- Savepoints 70, 232, 258, 472, 748
 - operation 233
- Scale-out systems 71–72, 419
 - add hosts 445
 - adding/removing host roles 457
 - architecture 419, 423
 - client connections 431
 - configuration 445
 - data redistribution 451
 - distributing tenant databases 449
 - HA/DR providers 459
 - host autofailover 433
 - host parameters 445
 - host roles 425
 - removing hosts 455
 - takeover and failback 457
 - vs. scale-up 505
 - worker groups 425
- Schema names 657
- Schemas 655
- Scrambling 785
- Secondary system 394
- Secondary time travel 410–411
 - system replication 414
- Secure store
 - local secure store (LSS) 880
 - system PKI SSFS 879
- Secure store and forward (SSF) 878
- Secure user store 898
- Security 60, 807
- Security and user management 91
- Security patch day 810
- Security patches 809
- Self-signed certificate 85
- Sequences 676
 - caching 677
 - chronology 679
 - decrementing 678
 - example 677
 - multitable model 678
 - numbering gaps 679
- Services 136
- Session autocommit 124

Sessions 176
 monitoring 178
 Sessions app 99
 setParameter.py 162
 Shared memory 168
 Shared network storage file system 71
 Shared nothing architecture 543
 Single component on one system
 (SCOS) 507, 517
 Single sign-on (SSO) 83
 Single-host systems 419
 Sizing 521
 disk 535
 easy and lazy 533
 for SAP Business Suite on SAP HANA 523
 for SAP BW 527
 for SAP BW/4HANA 527
 for SAP ERP 525
 reports 520
 Slave host takeover 459
 Snapshot pages 247
 Snapshots 80, 276
 create 279
 Soft stop 94
 Software as a service (SaaS) 497
 Software requirements 566
 Software virtualization 512
 Spark controller 491, 493
 Spark SQL Adapter 493
 Sparse encoding 700
 Spatial data 51
 Spatial data types 665
 Spatial database 52
 Split-brain situation 385, 463
 SQL analytic privileges 841
 SQL console 112
 SQL Database Connectivity (SQLDBC) 432
 SQL plan cache 100, 111, 922
 columns 924
 eviction 924
 manage 922
 querying 924
 SAP HANA Studio 929
 sizing rules 922
 statistics 923
 SQL statement analysis 917
 SQL statement collection 978
 command line 990
 DBA Cockpit 988
 editing 981
 folder structure 984
 import 984
 SQL statement collection (Cont.)
 installing 979
 personal folder 985
 running 983
 SAP HANA cockpit 987
 SAP HANA Studio 983
 structure 980
 version/feature dependent 983
 SQL statements
 execute 112
 monitor memory usage 176
 rules 112
 SQL Statements app 99–100
 SQL trace 939, 994
 activate 995
 command line 997
 connection and statement execution 999
 DBA Cockpit 996
 demo program 944
 display 941, 998
 execution error 1001
 file size 941
 header 999
 join statement 943
 levels 995
 multirow result 1000
 results 942, 1000
 SAP HANA cockpit 995, 1001
 SAP HANA Studio 996
 set up 939
 user filter 996
 with filter 940
 SQL views 62, 670
 restrict access 671
 SQLDBC trace 1005
 SqlExecutor thread 191
 Standard revisions 564
 Standby host 426, 459, 461
 Statement hints 947
 Statistics service views 953
 Statistics views 132
 HOST_HEAP_ALLOCATORS 170
 STATISTICS_ALERTS 966, 976
 STATISTICS_CURRENT_ALERTS 967, 976
 STATISTICS_EMAILRECIPIENTS 977
 STATISTICS_OBJECTS 954, 967
 STATISTICS_PROPERTIES 977
 STATISTICS_SCHEDULE 967, 976
 STATISTICS_THRESHOLDS 977
 STONITH call 434
 Storage area network (SAN) 334, 422,
 435, 502

Storage Connector API 422, 463
 Storage partition number 448
 Storage replication 343
 asynchronous 344
 point-in-time 344
 semi-synchronous 344
 storage level 345
 synchronous 344
 Stored procedures 680, 794
 call 682
 create 680
 example 681
 Support Package Stack (SPS) 563
 SUSE Enterprise High Availability 549
 SUSE Linux Enterprise hypervisors 513
 SUSE Linux Enterprise Server for SAP
 Applications 566, 568, 1009
 configure C-states 570
 kernel samepage merging 570
 libatomic1 572
 libgcc_s1 572
 libopenssl1_0_0 570
 libstdc++6 572
 sapconf 570
 Transparent Huge Pages 570
 workload memory protection 573
 xorg-x11-server 591
 SUSE packages
 libXssl 114
 xorg-x11-server 613
 Swap space 583
 Synchronous replication 402
 Synonyms 686
 SYS schema 950
 SYS_DATABASES schema 951
 System components 66
 System database 58, 66, 424
 limitations 59
 System ID (SID) 57, 133
 System isolation level 902
 System Landscape Optimizer 627
 System layer 98
 System performance 912
 DBA Cockpit 915
 metrics 912
 SAP HANA cockpit 912
 SAP HANA Studio 915
 SQL statements 916
 System privileges 828, 834
 ATTACH DEBUGGER 835
 AUDIT ADMIN 835, 867
 AUDIT ADMN 865
 System privileges (Cont.)
 AUDIT OPERATOR 835, 865
 AUDIT READ 835
 BACKUP ADMIN 835
 BACKUP OPERATOR 835
 CATALOG READ 88, 821, 835, 866
 CERTIFICATE ADMIN 835
 CREATE ANY 812
 CREATE STRUCTURED PRIVILEGE 835
 CREDENTIAL ADMIN 835
 DATA ADMIN 835, 866
 DATABASE ADMIN 221, 835, 867, 903
 DATABASE START 835
 DATABASE STOP 221, 835
 ENCRYPTION ROOT KEY ADMIN 835
 EXPORT 835
 EXTENDED STORAGE ADMIN 836
 IMPORT 836
 INFILE ADMIN 836
 INIFILE ADMIN 866
 LICENSE ADMIN 836
 LOG ADMIN 836
 MONITOR ADMIN 836
 OPTIMIZER ADMIN 836
 RESOURCE ADMIN 836
 ROLE ADMIN 836, 845
 SAVEPOINT ADMIN 836
 SERVICE ADMIN 836
 SESSION ADMIN 836
 SSL ADMIN 836
 STRUCTURED PRIVILEGE ADMIN 836
 TABLE ADMIN 836
 TRACE ADMIN 836, 866
 TRUST ADMIN 900
 USER ADMIN 619, 818, 821, 829, 836
 System replication 77, 341, 396, 549
 architecture 343
 asynchronous 347
 backup and recovery 355
 between primary and secondary 367
 client connections 363
 comparison table 351
 copy system 318
 data retention 350
 delta shipping 348
 deployment option 353
 disable 386
 distributed systems 427
 hardware exchange 417
 invisible takeover 364
 log retention 350
 logreplay 348

System replication (Cont.)

- logreplay read access* 348
- modes* 347
- monitor* 365
- multitier* 390
- overview* 345
- parameters* 360
- prerequisites* 366
- proxy schemas and views* 381
- replication parameter changes* 364
- resynchronization optimization* 349
- secondary replication* 345
- synchronous* 347
- synchronous in-memory* 347
- synchronous with full sync option* 347
- takeover and failback* 381
- time travel* 411
- update* 635
- vs storage* 354
- zero downtime maintenance* 414

System replication network 545

System views 654, 955

- _SYS_PASSWORD_BLACKLIST* 860
- _SYS_STATISTICS* 88
- ALL_AUDIT_LOG* 865
- AUDIT_READ* 865
- AUDIT_ACTIONS* 875
- AUDIT_LOG* 865, 875
- AUDIT_POLICIES* 867, 875
- CERTIFICATES* 901
- EFFECTIVE_PRIVILEGE_GRANTEES* 836
- EFFECTIVE_PRIVILEGES* 832–833, 849
- EFFECTIVE_ROLES* 833, 849
- ENCRYPTION_ROOT_KEYS* 885, 901
- GLOBAL_TABLE_CONSISTENCY_BASE* 248
- GRANTED_PRIVILEGES* 849
- GRANTED_ROLES* 849
- M_BACKUP_CATALOG* 267, 279
- M_BACKUP_CATALOG_FILES* 267
- M_BACKUP_PROGRESS* 267
- M_BACKUP_SIZE_ESTIMATIONS* 274
- M_BUFFER_CACHE_POOL_STATISTICS* 477
- M_BUFFER_CACHE_STATISTICS* 477
- M_CONVERTER_STATISTICS* 234, 274
- M_CS_ALL_COLUMNS* 474
- M_CS_COLUMNS* 474
- M_CS_TABLES* 474
- M_CUSTOMIZABLE_FUNCTIONALITIES* 231
- M_DATA_VOLUME_PAGE_STATISTICS* 234, 241
- M_DATA_VOLUME_PARTITIONS_STATISTICS* 240

System views (Cont.)

- M_DATA_VOLUME_STATISTICS* 238, 240
- M_DATA_VOLUME_SUPERBLOCK_STATISTICS* 241
- M_DATA_VOLUMES* 241
- M_DATABASE_REPLICAS* 328
- M_DISKS* 241, 246
- M_EFFECTIVE_PASSWORD_POLICY* 857
- M_ENCRYPTION_OVERVIEW* 891–892
- M_EVENTS* 238, 241, 365
- M_JOB_PROGRESS* 250
- M_LANDSCAPE_HOST_CONFIGURATION* 432, 438, 482
- M_LOG_BUFFERS* 246
- M_LOG_PARTITIONS* 246
- M_LOG_SEGMENTS* 243, 246
- M_PASSWORD_POLICY* 857
- M_PERSISTENT_MEMORY_VOLUME_DATA_FILES* 475
- M_PERSISTENT_MEMORY_VOLUME_STATISTICS* 475
- M_PERSISTENT_MEMORY_VOLUMES* 475
- M_REPLICA_STATISTICS* 328
- M_SAVEPOINT_STATISTICS* 241
- M_SAVEPOINTS* 234, 241
- M_SAVEPOINTS_STATISTICS* 234
- M_SERVICE_REPLICATION* 328, 372, 375, 379, 401, 403
- M_SERVICE_THREAD_SAMPLES* 250
- M_SERVICE_THREADS* 250, 639
- M_SERVICES* 225
- M_SNAPSHOTS* 248
- M_SYSTEM_REPLICATION* 372, 408
- M_SYSTEM_REPLICATION_TAKEOVER_HISTORY* 372, 412
- M_SYSTEM_REPLICATION_TIMETRAVEL* 413
- M_VOLUME_FILES* 254
- M_VOLUME_IO_TOTAL_STATISTICS* 241, 246
- M_VOLUMES* 241, 243, 246
- OWNERSHIP* 821
- PRIVILEGES* 849
- PSE_CERTIFICATES* 901
- REMOTE_USERS* 909
- REORG_OVERVIEW* 441, 451
- REORG_STEPS* 441, 451
- reserved_instance_numbers* 337
- ROLES* 849
- SCHEMAS* 821
- STRUCTURED_PRIVILEGES* 849
- SYS_DATABASES* 381
- System-versioned tables* 756

- USER_PARAMETERS* 818
- USERGROUP_PARAMETERS* 857
- USERS* 818, 822, 855, 908
- XSA_AUDIT_LOG* 865

T

- Table distribution 452
- Table Group Advisor app 454
- Table partitioning 72–73
- Table Placement Rules app 453
- Table preload 353
- Table Redistribution Execution History app 452
- Table Redistribution Plan Generator app 452
- Table replication 73
- Table Usage app 99
- Tables 420, 659, 689, 734
 - classification* 734
 - classification and distribution* 800
 - columns* 662
 - consistency* 249
 - display structure* 665
 - GLOBAL_TABLE_CONSISTENCY* 250
 - loading and unloading* 747
 - partitions* 742–744
 - placement example* 739
 - placement locations* 737
 - placement rules* 736, 740
 - preloading* 746
 - special types* 749
 - structure* 660
 - TABLE_PLACEMENT* 453
 - TABLE_PLACEMENT_LOCATIONS* 482
 - temporary* 749
 - worker groups* 737
- Takeover 381, 401
 - execution* 459
 - with handshake* 385
- Temporal tables 756, 802
- Temporary tables 749, 801
 - restrictions for SQL* 755
 - use in SAP systems* 756
- Tenant databases 59, 61–62, 66, 92, 218, 424
 - alerts* 226
 - automatic start* 148
 - copying* 330
 - create* 97
 - creation* 218
 - deleting* 222
 - distributing* 449
 - fallback snapshot* 223
 - isolated* 60

Tenant databases (Cont.)

- prevent system changes* 228
- renaming* 221
- replication* 320
- restrict and disable features* 231
- services* 224
- starting and stopping* 148, 220

TensorFlow 51

Text analytics and search 52

Text Retrieval and Information Extraction (TREC) 114

Threads app 99

Time-based partitioning specification 75

Timeouts 184

Time-travel query 756, 760, 762

Trace directory files 960

Trace files 143, 148, 955

- alert* 956
- display with SQL* 958
- formatting* 957
- location* 956
- message structure* 957
- specialized traces* 958

Transaction

- DBACOCKPIT* 124, 955, 988
- RZ11* 416, 942
- SM50* 940
- ST05* 939–940

Transactional savepoint 749

Transactions 176

- ID* 180
- monitoring* 179

Trial system 1009

Triggers 668–669

- warnings* 668

Troubleshooting 949

T-shirt sizing 533

U

- UltraLite 48
- Unenforced license key 61
- Unicode 48
- Uninstalling 639
 - select components* 640
- Updates 563, 620
 - multiple host system* 634
 - postprocessing actions* 637
 - prepare media* 621
 - prepare software archives* 626
 - SAP HANA lifecycle manager* 623
 - SAP HANA Studio* 622
 - single system* 626–634

Updates (Cont.)

<i>software authentication verification</i>	625
<i>using system replication</i>	635
<i>using zero-downtime maintenance</i>	635
<i>verify process list</i>	637
User account and authentication (UAA)	69
User and Role Management app	848
User groups	824
<i>create</i>	826
User management	810
User Management app	815, 818, 820
User parameters	
<i>CLIENT</i>	816
<i>LOCALE</i>	816
<i>PRIORITY</i>	816
<i>STATEMENT MEMORY LIMIT</i>	816
<i>STATEMENT THREAD LIMIT</i>	816
<i>TIME_ZONE</i>	816
Users	133, 655
<i>_SYS</i>	812
<i>_SYS_AFL</i>	813
<i>_SYS_DATA_ANONYMIZATION</i>	813
<i>_SYS_DI</i>	814
<i>_SYS_DI * CATALOG</i>	814
<i>_SYS_DI_SU</i>	814
<i>_SYS_DI_TO</i>	814
<i>_SYS_EPM</i>	813
<i>_SYS_PLAN_STABILITY</i>	813
<i>_SYS_REPO</i>	813
<i>_SYS_SQL_ANALYZER</i>	813
<i>_SYS_STATISTICS</i>	813
<i>_SYS_TABLE_REPLICAS</i>	813
<i>_SYS_TABLE_REPLICAT_DATA</i>	813
<i>_SYS_TASK</i>	813
<i>_SYS_WORKLOAD_REPLAY</i>	813
<i>_SYS_XB</i>	814
<sid> <i>adm</i>	601, 812, 881
<sid> <i>crypt</i>	881
<i>administration user</i>	133
<i>COCKPIT_ADMIN</i>	85
<i>create and manage accounts</i>	814
<i>delete</i>	838
<i>deleting</i>	819–820
<i>locking</i>	819
<i>modifying</i>	818
<i>properties</i>	606
<i>restricted</i>	812
<i>sapadm</i>	601
<i>service users</i>	811
<i>standard</i>	811
<i>SYSTEM</i>	601, 812, 822
<i>XSSQLCC_AUTO_USER_</i> <generated ID>	813
User-specific trace	1004

V

Value list	697
Variables	
<i>in SQL statements</i>	926
<i>value tracing</i>	927
Views	670
<i>aggregated</i>	672
<i>column views</i>	674
<i>dependencies</i>	674
<i>for table join</i>	673
<i>multitable</i>	672
<i>restricting</i>	672
<i>single table</i>	671
<i>SQL views</i>	670
Virtualization	518, 550
<i>IBM Power VM</i>	555
<i>VMware</i>	551
Virus scanners	580
Visualized plan	930
VMware	39, 513, 551
<i>fault tolerance</i>	553
<i>high availability</i>	552
<i>vMotion</i>	552, 554
<i>Workstation Player</i>	1012
Volatile tables	754
Volume ID	738

W

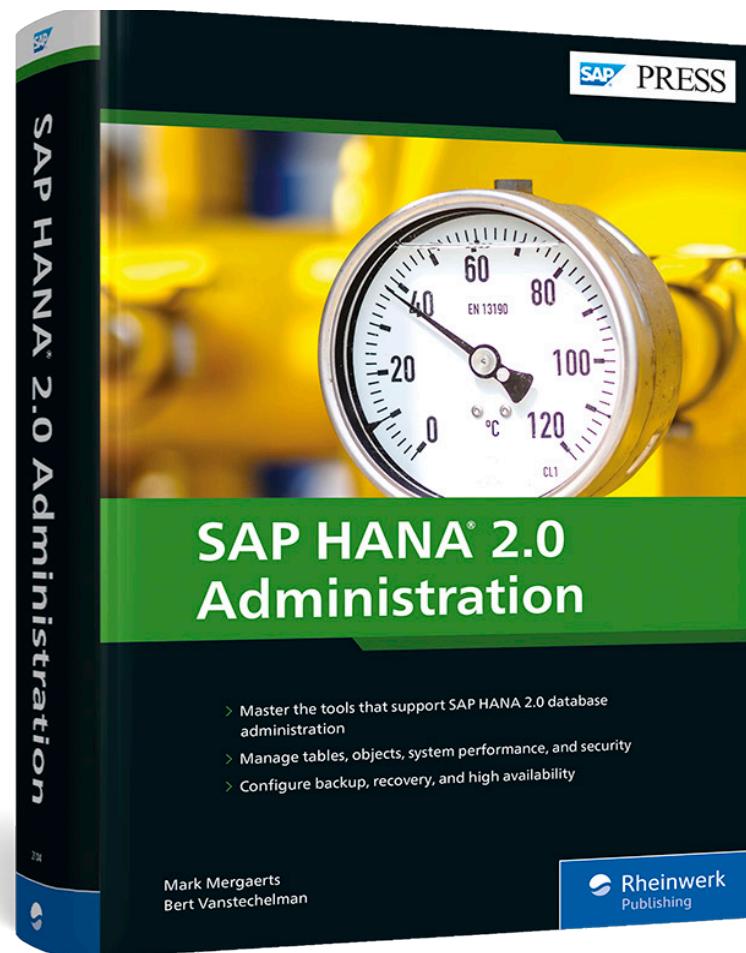
Warm data	467–468
Worker groups	482, 738
Workload classes	193
<i>changing and dropping</i>	194
<i>create</i>	193
<i>mapping</i>	198
<i>user settings</i>	201
Workload hints	199
Workload mapping	195
<i>change and drop</i>	199

X

X Windows	114, 591, 603, 627
XEN	513
XML analytic privileges	841

Z

Zero downtime	414
Zero-downtime maintenance	635



Mark Mergaerts is a principal technical consultant at Expertum and has more than 20 years of SAP experience. His activities concentrate on system administration, database management, performance, upgrades, and platform and Unicode migrations. He has written four other books for SAP PRESS.



Bert Vanstechelman is a partner of and principal technical consultant at Expertum and is the founder of Logos Consulting, now a part of Expertum. He has more than 20 years of SAP experience. Bert specializes in platform migrations, SAP installations, release upgrades, SAP Business Warehouse, SAP Supply Chain Management, SAP HANA, and OS/DB migrations. Bert is an advisor for SAP Professional Journal and an SAP Solution Manager Expert. He has written four other books for SAP PRESS.

Mark Mergaerts, Bert Vanstechelman

SAP HANA 2.0 Administration

1056 Pages, 2022, \$89.95

ISBN 978-1-4932-2104-2



www.sap-press.com/5287

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.