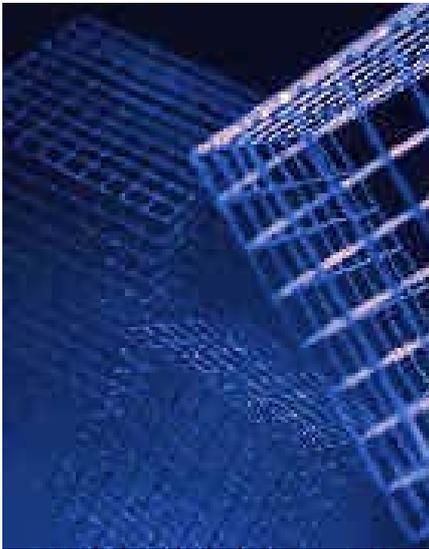


The SAP DB Database System



Copyright

© Copyright 2002 SAP AG.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation.

For more information on the GNU Free Documentaton License see <http://www.gnu.org/copyleft/fdl.html#SEC4>.

Icons

| Icon | Meaning |
|---|----------------|
|  | Caution |
|  | Example |
|  | Note |
|  | Recommendation |
|  | Syntax |

Typographic Conventions

| Type Style | Description |
|---------------------|--|
| <i>Example text</i> | Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths and options. Cross-references to other documentation. |
| Example text | Emphasized words or phrases in body text, titles of graphics and tables. |
| EXAMPLE TEXT | Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example, SELECT and INCLUDE. |
| Example text | Screen output. This includes file and directory names and their paths, messages, source code, names of variables and parameters as well as names of installation, upgrade and database tools. |
| EXAMPLE TEXT | Keys on the keyboard, for example, function keys (such as F2) or the ENTER key. |
| Example text | Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation. |
| <Example text> | Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries. |

| | |
|---|----|
| The SAP DB Database System:..... | 12 |
| Architecture of the Database System | 12 |
| Database Instance | 13 |
| Thread..... | 13 |
| User Kernel Thread (UKT) | 14 |
| Pager | 14 |
| Log Writer | 14 |
| Server Task | 15 |
| Timer Task..... | 15 |
| Trace Writer Task | 15 |
| User Task | 15 |
| Utility Task | 16 |
| Task State | 16 |
| Special Thread | 17 |
| Clock Thread | 17 |
| Console Thread | 17 |
| Coordinator | 17 |
| Dev Thread | 18 |
| Requester | 18 |
| Temporary Dev Thread | 18 |
| Timer..... | 18 |
| Cache..... | 19 |
| Catalog Cache | 19 |
| I/O Buffer Cache | 19 |
| Converter..... | 19 |
| Data Cache..... | 20 |
| Log Queue | 20 |
| Volume..... | 21 |
| Data Volume | 21 |
| Log Volume | 22 |
| Database Instance Type | 22 |
| SAP DB OLTP | 23 |
| liveCache | 23 |
| SAP DB Document Server..... | 23 |
| SAP DB OLAP | 23 |
| SAP DB E-Catalog..... | 24 |
| SAP DB Versions and Database Instance Types | 24 |
| Operating System Platforms | 25 |
| Multiprocessor Configuration..... | 25 |

| | |
|---|----|
| User Concept | 25 |
| SAP DB User Classes | 26 |
| Database Manager Operator (DBM Operator) | 26 |
| Authorizations | 27 |
| User Authorizations | 27 |
| Default Authorizations for the First DBM Operator | 28 |
| Operating System User Authorizations | 28 |
| Database User | 28 |
| Database User Classes | 29 |
| SYSDBA | 29 |
| DBA | 29 |
| DOMAIN | 30 |
| RESOURCE | 30 |
| STANDARD | 30 |
| User Groups | 30 |
| The Role Concept | 30 |
| User Data as Options | 31 |
| Required Options | 32 |
| User Data and XUSER | 33 |
| Using XUSER | 33 |
| XUSER Options | 34 |
| Generating XUSER Data in the Background | 34 |
| XUSER Data | 35 |
| Security Concepts | 37 |
| Log Settings | 37 |
| Log Mode | 38 |
| Overwrite Mode for the Log Area | 38 |
| Activating or Deactivating Redo Log Management | 39 |
| Availability | 40 |
| Backup Strategy | 40 |
| Backup | 41 |
| Data Backup | 42 |
| Complete Data Backup | 42 |
| Incremental Data Backup | 42 |
| Parallel Backup | 42 |
| Saving Data Backups | 43 |
| Log Backup | 43 |
| Automatic Log Backup | 43 |
| Interactive Log Backup | 44 |
| Saving Log Backups | 44 |

| | |
|---|----|
| External Backup Tool..... | 45 |
| Restartability..... | 45 |
| Log Concept..... | 45 |
| Log Entry | 45 |
| Redo Log Entry..... | 46 |
| Undo Log Entry..... | 46 |
| Online Logging | 46 |
| Redo Log Management..... | 47 |
| Log Queue | 47 |
| Log Page..... | 48 |
| Log Writer..... | 48 |
| Log Area..... | 49 |
| Undo Log Management..... | 49 |
| Undo Log File..... | 49 |
| History Management..... | 50 |
| History File | 50 |
| History List | 50 |
| Garbage Collector..... | 51 |
| Restart or Recovery | 51 |
| Redo Log Manager..... | 51 |
| Log Reader..... | 52 |
| Redo Log File..... | 52 |
| Redo List..... | 52 |
| Redo Task..... | 53 |
| Savepoint on Restart..... | 54 |
| Example: Restart | 54 |
| Database Tools..... | 55 |
| Architecture of the SAP DB Tools | 55 |
| Architecture of the Database Manager..... | 56 |
| Architecture of the SAP DB Loader | 57 |
| Architecture of the Query Tools..... | 58 |
| Architecture of the SAP DB Web Tools..... | 59 |
| X Server..... | 60 |
| DBM Server | 60 |
| Loader Server..... | 60 |
| Web Server..... | 61 |
| Database Manager..... | 61 |
| Database Manager GUI..... | 61 |
| Options (DBMGUI)..... | 62 |
| Database Manager CLI..... | 62 |

| | |
|--|----|
| Options (DBMCLI)..... | 63 |
| DBM Commands..... | 64 |
| Web DBM..... | 64 |
| SAP DB Loader | 65 |
| Options (LOADERCLI)..... | 66 |
| Loader Commands | 66 |
| Query Tools..... | 66 |
| SQL Studio..... | 67 |
| Options (SQL Studio)..... | 67 |
| Web SQL Studio | 67 |
| Directory Structure of the Database for SAP Systems | 68 |
| SAP DB Directories | 68 |
| Instance Data..... | 69 |
| Programs that Are Independent of the Database Software Version | 70 |
| Libraries for the Client Run-time Environment..... | 70 |
| Programs that Are Dependent on the Database Software Version..... | 70 |
| Client Tools..... | 71 |
| Example: SAP DB Directory Structure | 71 |
| Displaying SAP DB Directories..... | 71 |
| Define SAP DB Directories | 72 |
| Directory Structure of the Database System for Open Source..... | 72 |
| Performance Requirements | 73 |
| Example Configuration..... | 73 |
| Using Multiple Database Systems | 73 |
| SAP DB Directories | 74 |
| Displaying SAP DB Directories..... | 75 |
| Define SAP DB Directories | 75 |
| Database Files | 75 |
| Log Files | 75 |
| Classes of Log Files | 77 |
| Configuration Files | 77 |
| Database Parameters | 77 |
| General Database Parameters..... | 78 |
| Special Database Parameters (Extended)..... | 78 |
| Support Database Parameters..... | 79 |
| BACKUP_BLOCK_CNT | 79 |
| CACHE_SIZE | 79 |
| CAT_CACHE_SUPPLY | 79 |
| DATE_TIME_FORMAT | 79 |
| DEADLOCK_DETECTION..... | 79 |

| | |
|---------------------------------|----|
| DEFAULT_CODE..... | 79 |
| DEVNO_BIT_COUNT | 80 |
| INSTANCE_TYPE..... | 80 |
| JOIN_MAXTAB_LEVEL9 | 80 |
| JOIN_MAXTAB_LEVEL4 | 81 |
| JOIN_SEARCH_LEVEL..... | 81 |
| KERNELDIAGSIZE | 81 |
| KERNELVERSION..... | 82 |
| LOG_BACKUP_TO_PIPE..... | 82 |
| LOG_IO_QUEUE | 82 |
| LOG_SEGMENT_SIZE..... | 82 |
| LRU_FOR_SCAN..... | 82 |
| MAXARCHIVELOGS..... | 83 |
| MAXBACKUPDEVS | 83 |
| MAXCPU | 83 |
| MAXDATADEVSPACES | 83 |
| MAXDATAVOLUMES | 83 |
| MAXLOCKS..... | 83 |
| MAXLOGVOLUMES | 84 |
| MAXRGN_REQUEST | 84 |
| MAXSERVERTASKS | 84 |
| MAXUSERTASKS..... | 84 |
| MP_RGN_LOOP | 84 |
| OPTIM_MAX_MERGE | 84 |
| REQUEST_TIMEOUT..... | 85 |
| RESTART_SHUTDOWN | 85 |
| RUNDIRECTORY | 85 |
| SEQUENCE_CACHE..... | 85 |
| SESSION_TIMEOUT | 86 |
| UTILITY_PROT_SIZE | 86 |
| _DATA_CACHE_RGNS..... | 86 |
| _EVENT_ALIVE_CYCLE | 86 |
| _MAXEVENTS | 86 |
| _MAX_MESSAGE_FILES..... | 86 |
| _ROW_RGNS | 87 |
| _TAB_RGNS | 87 |
| _TRANS_RGNS..... | 87 |
| _TREE_RGNS..... | 87 |
| _UNICODE..... | 87 |
| SAP DB as UNICODE Database..... | 87 |

| | |
|---|-----|
| UNICODE | 88 |
| Installing a UNICODE-Enabled Database..... | 88 |
| Setting the Database Parameter _UNICODE..... | 89 |
| Setting Code Attribute UNICODE..... | 89 |
| UNICODE and SQL..... | 90 |
| Example 1 | 91 |
| UNICODE in Programming Languages | 93 |
| Example 2 | 94 |
| Data Management Using B* Trees | 97 |
| Concepts | 97 |
| Primary Key | 98 |
| Secondary Key | 98 |
| B* Tree..... | 98 |
| Root/Index Page | 99 |
| Leaf Page..... | 100 |
| Table Access..... | 100 |
| Table ID | 100 |
| B* Trees for Tables | 101 |
| B* Trees for Table with LONG Columns..... | 101 |
| B* Trees for Tables with Secondary Key..... | 102 |
| B* Trees for Tables with LONG Columns and Secondary Key | 102 |
| Table Access Using B* Tree | 103 |
| Table Access (SELECT) Using B* Tree | 103 |
| Table Access (INSERT) Using B* Tree | 105 |
| Table Access (DELETE) Using B* Tree | 106 |
| Table Access (UPDATE) Using B* Tree..... | 107 |
| Changes in the B* Tree Structure | 107 |
| Non-Uniform Distributions of Data Pages..... | 108 |
| Lock Behavior | 109 |
| Lock | 109 |
| Shared Lock..... | 110 |
| Exclusive Lock | 111 |
| Optimistic Lock | 111 |
| Requesting and Releasing a Lock | 111 |
| Isolation Level | 112 |
| Isolation Level 0..... | 113 |
| Isolation Level 1 or 10..... | 113 |
| Isolation Level 15..... | 114 |
| Isolation Level 2 or 20..... | 114 |
| Isolation Level 3 or 30..... | 114 |

| | |
|--|-----|
| Phenomena | 115 |
| Dirty Read | 115 |
| Non-Repeatable Read | 115 |
| Phantom | 115 |
| Creating a Homogeneous System Copy | 116 |
| Operating System Compatibility for Homogeneous System Copies | 117 |
| Homogeneous System Copy with the Database Manager CLI | 117 |
| Standby Databases with SAP DB | 118 |
| Setting Up a Standby Instance | 118 |
| Starting the Standby Instance as an Active Instance | 119 |
| Importing Log Backups up to a Specific Time | 120 |
| Importing Another Manual Log Backup | 121 |
| Copying the Log Volumes of the Original Instance | 121 |
| Example: Standby Database | 122 |
| SAP DB Version 7.4 | 123 |
| Requirements for a Database System | 123 |
| SAP DB Improvements Since 1997 | 123 |
| SAP DB Tools | 124 |
| Technical Specification of SAP DB Version 7.4 | 124 |
| New Developments in SAP DB Version 7.4 | 126 |
| Terms | 127 |
| Application Data | 128 |
| Backup History | 129 |
| Backup ID | 129 |
| Backup Medium | 129 |
| Checking the Database Structures | 130 |
| COMMIT | 131 |
| Data Area | 131 |
| Database Catalog | 131 |
| Database Console | 132 |
| Database Name | 132 |
| Database Session | 132 |
| Database Trace | 133 |
| DBM Operator | 133 |
| External Backup ID | 133 |
| External Backup Medium | 134 |
| Group of Parallel Backup Media | 134 |
| Instance Type | 134 |
| Kernel | 134 |
| Language Support (MapChar Sets) | 134 |

| | |
|--|-----|
| Log Area | 135 |
| Name of a Standard Backup Medium | 135 |
| Name of External Backup Medium..... | 135 |
| Operational State | 136 |
| Page | 136 |
| Page Pool..... | 136 |
| Restart..... | 136 |
| ROLLBACK | 137 |
| Run Directory..... | 137 |
| Savepoint..... | 137 |
| Single Backup Medium..... | 138 |
| SQL Mode | 138 |
| Task..... | 138 |
| Timeout Value | 138 |
| Transaction..... | 139 |
| Transaction File..... | 139 |
| Transaction List..... | 139 |
| Version File..... | 140 |
| Variables | 140 |
| SAP DB Documentation..... | 141 |
| SAP DB Software..... | 144 |
| SAP DB Version Notation | 144 |
| SAP DB Support | 144 |



The SAP DB Database System:

This manual gives you an overview of the database system SAP DB Version 7.4 and its tools.

[Architecture of the Database System \[Page 12\]](#)

[User Concept \[Page 25\]](#)

[Security Concepts \[Page 37\]](#)

[Log Concept \[Page 45\]](#)

[Database Tools \[Page 55\]](#)

[Directory Structure of the Database for SAP Systems \[Page 68\]](#)

[Directory Structure of the Database for Open Source \[Page 72\]](#)

[Database Files \[Page 75\]](#)

[Database Parameters \[Page 77\]](#)

[SAP DB as a UNICODE Database \[Page 87\]](#)

[Data Management Using B* Trees \[Page 97\]](#)

[Lock Behavior \[Page 109\]](#)

[Creating a Homogeneous System Copy \[Page 116\]](#)

[Standby Databases with SAP DB \[Page 118\]](#)

[SAP DB Version 7.4 \[Page 123\]](#)

[Terms \[Page 127\]](#)

[Variables \[Page 140\]](#)

[SAP DB Documentation \[Page 141\]](#)

[SAP DB Software \[Page 144\]](#)

[SAP DB Version Notation \[Page 144\]](#)

[SAP DB Support \[Page 144\]](#)



Architecture of the Database System

You can find an overview of the main architecture characteristics of the SAP DB relational database system in the **Fact Sheet** on the SAP DB homepage www.sapdb.org. Some aspects of the SAP DB architecture are described in more detail below:

- [Database instance \[Page 13\]](#)
- [Database instance type \[Page 22\]](#)
- [SAP DB versions and database instance types \[Page 24\]](#)
- [Operating system platforms \[Page 25\]](#)
- [Multiprocessor configuration \[Page 25\]](#)

See also:

- [User concept \[Page 25\]](#)
- [Security concepts \[Page 37\]](#)
- [Log concept \[Page 45\]](#)

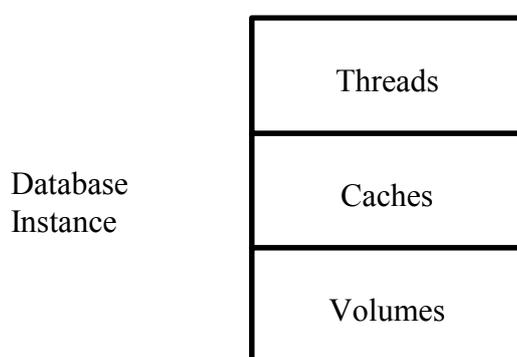
- [Database tools \[Page 55\]](#)
- [Directory structure of the database for Open Source \[Page 72\]](#)
- [Directory structure of the database for SAP Systems \[Page 68\]](#)
- [Database files \[Page 75\]](#)
- [SAP DB as a UNICODE Database \[Page 87\]](#)
- [Data management using B* trees \[Page 97\]](#)
- [Lock behavior \[Page 109\]](#)



Database Instance

The SAP DB database can be installed and run on a computer in one mode (database instance) or several modes (database instances) ([Database Instance Type \[Page 22\]](#), **see also:** [SAP DB Versions and Database Instance Types \[Page 24\]](#)).

Every database instance consists of [threads \[Page 13\]](#), main memory structures ([caches \[Page 19\]](#)) and [volumes \[Page 21\]](#).



Each database instance differentiates between the following areas for the logical storage of data:

- [Database catalog \[Page 131\]](#)
- [Application data \[Page 128\]](#)



Thread

A whole series of operating system threads (often referred to together as the kernel) belongs to a [database instance \[Page 13\]](#).

We differentiate between **UKTs** (user kernel threads) and **special threads**.

The required number of UKTs and of special threads depends on the hardware configuration, the number of [volumes \[Page 21\]](#) used, and the database parameters that were set.

- [User Kernel Thread \(UKT\) \[Page 14\]](#)
- [Special Thread \[Page 17\]](#)



User Kernel Thread (UKT)

A [database instance \[Page 13\]](#) contains a series of [threads \[Page 13\]](#). A user kernel thread (UKT) forms a subset of all [tasks \[Page 138\]](#) (for internal tasking).

The following types of tasks exist:

- [Pager \[Page 14\]](#)
- [Log Writer \[Page 48\]](#)
- [Server Task \[Page 15\]](#)
- [Timer Task \[Page 15\]](#)
- [Trace Writer Task \[Page 15\]](#)
- [User Task \[Page 15\]](#)
- [Utility Task \[Page 16\]](#)

Each user kernel thread can contain one or more [tasks \[Page 138\]](#). The [task states \[Page 16\]](#) give you information on the corresponding thread.



Pager

Pagers (data writers) is part of a [user kernel thread \(UKT\) \[Page 14\]](#). A pager is a [task \[Page 138\]](#) that is responsible for writing data from the [data cache \[Page 20\]](#) to the [data volumes \[Page 21\]](#). Pagers become active when a [savepoint \[Page 137\]](#) is performed. In a [restart \[Page 136\]](#), the pagers are responsible for reading the [converter \[Page 19\]](#).

Writing savepoints can take a long time for a large data cache. In this case, the pagers also become active between the end of a savepoint and the start of the next one, to write changed data asynchronously from the data cache to disk.

The system calculates the number of pagers. It depends primarily on the data cache size ([CACHE_SIZE \[Page 79\]](#)) and the number of data volumes ([MAXDATAVOLUMES \[Page 83\]](#)).



Log Writer

One active [task \[Page 138\]](#) of [redo log management \[Page 47\]](#) is the log writer.

The log writer is part of a [user kernel thread \(UKT\) \[Page 14\]](#). When the database system is started, it is initialized using permanently stored, internal configuration information. This configuration information is written to the log area at regular intervals, in particular at a [savepoint \[Page 137\]](#).

- A [log queue \[Page 47\]](#) is assigned to the log writer.
- The log writer writes the [log pages \[Page 48\]](#) that are full, or have to be written as a result of a [COMMIT \[Page 131\]](#) or [ROLLBACK \[Page 137\]](#), from the log queue to the [log area \[Page 135\]](#). The log pages are numbered (log sequence number), so that it is possible to check that all log pages were written, and to ensure the correct working sequence in the case of a [restart \[Page 136\]](#) or recovery. The log writer then notifies the transactions that were waiting for their [redo log entries \[Page 46\]](#) to be written.
- Log pages of the log queue that were not full when a write operation was performed remain in the log queue and continue to be filled, and are written to the log area in a

subsequent write operation. The log writer is configured so that it always writes one and the same log page to the same physical place.

- The log writer regularly checks the state of the log area.
If the log area is full, the log writer locks the log queue so that all transactions that want to enter [redo log entries \[Page 46\]](#) into the log queue are stopped.
When the [automatic log backup \[Page 43\]](#) is active, the log writer ensures that the redo log entries from the log area are backed up automatically.
When a certain number of log entries have been written, the administrative information is copied to the log area, and savepoints are requested, if required. In the case of a restart, this reduces the restart time.



Server Task

A server task is part of a [user kernel thread \(UKT\) \[Page 14\]](#). Server tasks are [tasks \[Page 138\]](#) that are used primarily for the running database functions in parallel, for example:

- Making backups to a group of parallel backup media
- Recovering multiple backup media in parallel
- Setting up indexes

When the database parameters are being configured, the number of server tasks is determined automatically from the number of [data volumes \[Page 21\]](#) and the number of data backup devices in use.

The maximum number of server tasks available is defined by the [MAXUSERTASKS \[Page 84\]](#) database parameter.



Timer Task

A timer task is part of a [user kernel thread \(UKT\) \[Page 14\]](#). A timer is a [task \[Page 138\]](#) that handles all types of timeout situations.



Trace Writer Task

The trace writer task is part of a [user kernel thread \(UKT\) \[Page 14\]](#). The database system enables the [database trace \[Page 133\]](#) to be activated for diagnosis purposes. The special trace writer [task \[Page 138\]](#) is provided for this purpose.



User Task

A user task is part of a [user kernel thread \(UKT\) \[Page 14\]](#). When connecting to the [database instance \[Page 13\]](#), each user of the instance or each application is assigned precisely one fixed user task. This [task \[Page 138\]](#) is responsible for processing SQL statements for the database session.

The number of user tasks available is defined by the [MAXUSERTASKS \[Page 84\]](#) database parameter.



Utility Task

A utility task is part of a [user kernel thread \(UKT\) \[Page 14\]](#). The utility task is a [task \[Page 138\]](#) reserved solely for managing the [database instance \[Page 13\]](#).

Since there is only one utility task for each database instance, no management actions can be performed in parallel.

An exception to this rule is the [automatic log backup \[Page 43\]](#). This can be performed in parallel with other management actions.



Task State

The following is a list of all possible [task \[Page 138\]](#) states:

| | |
|----------------|---|
| AsynClose | Closing input/output channels after backup/restore |
| AsynCntl | Task determining parameters or initializing backup device |
| AsynOpen | Opening input/output channels for backup/restore |
| AsynWait | Task waiting for execution of input/output activity when making backup/restore |
| Command reply | Task sending result of request to application |
| Command wait | Task is a database session without request |
| Connect wait | Free database session |
| Diaginit | Initializing trace volume |
| DcomObjCalled | Task is executing database procedure/COM object |
| Inactive | Task is in initial state and has no resources (such as stack) |
| InsertEvent | Task generating an event |
| IO wait | Waiting for I/O (W:write, R:read) |
| RescheduleMsec | Short-term wait situation; starts running again automatically after specified period (in microseconds) |
| Runnable | Can run now, but swapped due to long runtime or priority of another task |
| Running | Running, using CPU |
| Sus... | Special suspend situations |
| Terminated | Ending task/terminating database instance, or instance crashing |
| Vattach | Opening input/output channels (volumes, normal operation) |
| Vbegexcl | Waiting for protected memory access |
| Vdetach | Closing input/output channels (volumes, normal operation) |
| Vdevsize | Determining size or formatting of volume |
| Vdexccl | Can run again after protected memory access that caused lock collision |
| Vopmsg | Writing a message to the files <code>knldiag</code> , <code>knldiag.err</code> and/or <code>opmsg[n]</code> |
| Vrelease | Ending database session between database instance and application |

| | |
|--------------|--|
| Vshutdown | Operational state of database instance changing from ONLINE to ADMIN |
| Vsleep | Short-term wait situation; starts running again automatically after specified period |
| Vsuspend | Waiting for B* tree lock (very short) or log I/O, awoken explicitly |
| Vwait | Waiting for SQL lock |
| WaitForEvent | Task waiting for generation of events |

You can display the task states with the console tool, for example. In SAP systems, you can use the CCMS to call the console. For more information, see the documentation *Database Administration in CCMS: SAP DB OLTP* or *Database Administration in CCMS: liveCache*.

Special Thread

A [database instance \[Page 13\]](#) has the following special [threads \[Page 13\]](#):

- [Console Thread \[Page 17\]](#)
- [Coordinator \[Page 17\]](#)
- [Dev Threads \[Page 18\]](#)
- [Requester \[Page 18\]](#)
- [Temporary Dev Threads \[Page 18\]](#)
- [Timer \[Page 18\]](#)

There is also a range of operating system-specific special threads, such as the [clock thread \[Page 17\]](#).

Clock Thread

The clock thread is an operating system-dependent [special thread \[Page 17\]](#). The clock thread is only used under Windows NT.

It computes internal times; for example, to determine the time needed to execute an SQL statement.

Console Thread

The clock thread is a [special thread \[Page 17\]](#).

The console thread processes requests from the [Database Console \[Page 132\]](#).

Coordinator

The coordinator is a [special thread \[Page 17\]](#). The coordinator monitors all [threads \[Page 13\]](#) in the [database instance \[Page 13\]](#).

When the database instance is started, the coordinator is the first active thread. It coordinates the starting processes of the other threads.

- UNIX: If a thread fails while a UNIX operating system is running, the coordinator terminates all other threads.
- Windows: If a thread fails while a Windows operating system is running, an exception handler becomes responsible for terminating all the other threads in an orderly way.



Dev Thread

Dev threads are [special threads \[Page 17\]](#). Dev threads handle the read and write commands that read and write [tasks \[Page 138\]](#) ask to have performed.

The number of dev threads is primarily dependent on the number of [volumes \[Page 21\]](#) in the [database instance \[Page 13\]](#). Under normal circumstances, two dev threads are activated for every [data volume \[Page 21\]](#) and every [log volume \[Page 22\]](#). One dev thread is activated for writing the kernel trace, if it is enabled.

The dev thread **dev0** plays a special role; **dev0** coordinates and monitors the dev threads.

- For example, if a log volume fails while the database is running (bad volume), dev0 ensures that the corresponding dev threads are terminated. The database instance is transferred to OFFLINE [mode \[Page 136\]](#).
- If the database is enlarged while running by adding another data volume, dev0 ensures that new dev threads are generated.

All the other **dev<i>** threads write data to or read data from the volumes.



Requester

The requester is a [special thread \[Page 17\]](#). The requester receives both local communication requests (CONNECT) and requests from the network and assigns them to a [user kernel thread \(UKT\) \[Page 14\]](#).



Temporary Dev Thread

Temporary dev threads (**asdev<i>**), which are [special threads \[Page 17\]](#), are activated to read and write data for [data backups \[Page 42\]](#).



Timer

The timer is a [special thread \[Page 17\]](#). The timer monitors time for timeout control.



Cache

To reduce the number of times that the disk is accessed, read and write operations on the data of a [database instance \[Page 13\]](#) are buffered.

The relevant main memory structures are called caches. They can be dimensioned appropriately.

The database system recognizes the following caches, among others:

- [Catalog Caches \[Page 19\]](#)
- [I/O Buffer Cache \[Page 19\]](#)
[Data Cache \[Page 20\]](#)
[Converter \[Page 19\]](#)
- [Log Queue \[Page 47\]](#)



Catalog Cache

The catalog [cache \[Page 19\]](#) of a [database instance \[Page 13\]](#) stores the catalog objects most recently used by a [database session \[Page 132\]](#) and the internal representation (execution plans) of the most recently executed SQL statements.

Data which is expelled from the catalog cache is moved for the time being to the [data cache \[Page 20\]](#).

A catalog cache exists once for each database user session.



I/O Buffer Cache

One important [cache \[Page 19\]](#) of a SAP DB database is the I/O buffer cache.

When the database system is started, the I/O buffer cache is created in the main memory in accordance with the size entered in the database parameter [CACHE_SIZE \[Page 79\]](#) and is managed via the [page pool \[Page 136\]](#).

A certain number of pages of the I/O buffer cache are made available to the [converter \[Page 19\]](#). The remaining pages are made available to the [data cache \[Page 20\]](#).

If the converter requires more pages during database operation, the distribution of all I/O buffer cache pages between the converter and data cache is changed dynamically to meet this requirement.



Converter

The converter is used for the assignment of logical to physical data pages. When data pages that are not located in the [data cache \[Page 20\]](#) are accessed, their logical position is first searched for in the converter. The converter then determines the corresponding physical position of the data pages in the [data volumes \[Page 21\]](#).

The converter is a main memory structure that is shared simultaneously by all active users. Only the converter pages that contain a mapping of permanent data pages are written to the data volumes with each [savepoint \[Page 137\]](#). In a [restart \[Page 136\]](#), the [pager \[Page 14\]](#) can use these pages to restore the converter.

The converter is dimensioned dynamically, which means that you cannot set the size of the converter. The required converter pages are taken from the [I/O buffer cache \[Page 19\]](#), which is used jointly by the converter and the data cache. The size of the I/O buffer cache is determined by the database parameter [CACHE_SIZE \[Page 79\]](#).

If the converter requires more pages than were originally assigned, the number of data cache pages is reduced accordingly. If the converter requires fewer pages, the free pages are managed via the [page pool \[Page 136\]](#), and can be used again by the converter if it grows.



Data Cache

The data cache contains the last read- or write-accessed pages of the [data volumes \[Page 21\]](#).

The data cache is a main memory structure that is shared simultaneously by all active users.

The data cache is dimensioned dynamically, which means that you cannot directly influence the size of the data cache. The required data cache pages are taken from the [I/O buffer cache \[Page 19\]](#), which is used jointly by the [converter \[Page 19\]](#) and the data cache. The size of the I/O buffer cache is determined by the database parameter [CACHE_SIZE \[Page 79\]](#). If the converter grows, the number of data cache pages is reduced accordingly.

The hit rate, that is the proportion of successful accesses in the total number of accesses, is a crucial measure of performance. Successful access means that the required data was already available in the data cache when it was accessed.



Log Queue

The area of the main memory required for [redo log management \[Page 47\]](#) is called the log queue. The size of a log queue (in [log pages \[Page 48\]](#)) is determined by the database parameter [LOG_IO_QUEUE \[Page 82\]](#).

A [transaction \[Page 139\]](#) uses a log queue to obtain a main memory area for a [redo log entry \[Page 46\]](#). The transaction writes the redo log entry to the log pages of the log queues.

Writing of the log pages to the [log area \[Page 135\]](#) is carried out by the [log writer \[Page 48\]](#).

Process Flow

1. The [user task \[Page 15\]](#) of the transaction reserves main memory space for a redo log entry in the log queue.
2. The transaction writes the redo log entry to the reserved area of the log queue. The time at which the redo log entry is written to the log queue is assigned to the relevant log page (log queue sequence number).
3. The transaction releases the reserved area of the log queue for processing by the log writer, and provides information on whether it wants to wait for the log page from the log queue to be written to the log area. This behavior is always required for [COMMIT \[Page 131\]](#) and [ROLLBACK \[Page 137\]](#) operations.
If a transaction does wait for the redo log entry to be written, the log writer notifies the transaction once the relevant page has been written from the log queue to the log area, and informs the transaction of the log sequence number that was assigned when the log page was written to the log area.



Volume

A volume is a logical grouping of physical storage units (disks). A volume can be a part of a physical disk, a complete physical disk, or a completely structured storage system consisting of several storage units.

The disks used should have identical performance parameters (specifically access speeds) to ensure an even filling of the volumes.

If database management tools are used, a volume is addressed via a directory path.

The database system differentiates between [data volumes \[Page 21\]](#) and [log volumes \[Page 22\]](#).

Every [database instance \[Page 13\]](#) has one or more log volumes and data volumes. You configure the maximum numbers of data and log volumes that are possible when installing the database instance.

If necessary, you can add data or log volumes to a database instance while the database is running. Directory paths for data and log volumes can be changed.



Data Volume

A [database instance \[Page 13\]](#) has [volumes \[Page 21\]](#), in which the [pager \[Page 14\]](#) records the database contents. These volumes are called data volumes. All data volumes together form the [data area \[Page 131\]](#).

Among other things, the data volumes contain the [application data \[Page 128\]](#) and the metadata of the [database catalog \[Page 131\]](#).

Managing Application Data

The application data of a table or index can use just one [page \[Page 136\]](#) in the data area as a minimum. A table can extend across all data volumes of the data area as a maximum. A table increases or decreases in size automatically without administrative intervention. As a rule, a database-internal striping algorithm distributes the data belonging to a table evenly across all the data volumes.

It is therefore not possible or necessary to assign tables to the individual data volumes.

See: [Data Management Using B* Trees \[Page 97\]](#)

Managing Data Volumes

You can configure one or more data volumes when you install a new database instance. The disk storage space defined by all the data volumes is the total size of the database, and must not fall below 2000 pages.

During database operation, you can add new data volumes. The number of data volumes is limited by the database parameter [MAXDATAVOLUMES \[Page 83\]](#).

See:

Database Manager GUI: SAP DB 7.4, Section [Managing Database Instances \[See SAP DB Library\]](#)

Database Manager CLI: SAP DB 7.4, Section [Adding a Volume \[See SAP DB Library\]](#)

Data backups

To ensure safe database operation, it is necessary to back up the data area at regular intervals.

See: [Data Backups \[Page 42\]](#)



Log Volume

A [database instance \[Page 13\]](#) has [volumes \[Page 21\]](#), in which the [log writer \[Page 48\]](#) records all changes to the database contents. These volumes are called log volumes. All log volumes form the [log area \[Page 135\]](#).

Managing Log Volumes

You can configure one or more log volumes when you install a new database instance.

During database operation, you can add new log volumes. The number of log volumes is limited by the database parameter [MAXLOGVOLUMES \[Page 84\]](#).

See:

Database Manager GUI: SAP DB 7.4, Section [Managing Database Instances \[See SAP DB Library\]](#)

Database Manager CLI: SAP DB 7.4, Section [Adding a Volume \[See SAP DB Library\]](#)

Log Settings

The log entries are needed to ensure that the database content can be restored if a media device fails. Choose [log settings \[Page 37\]](#) that guarantee secure operation of your database: mirror the log area, only overwrite the log area after the [log backup \[Page 43\]](#), always leave [redo log management \[Page 47\]](#) activated.

Log Backups

So that your database operates securely, choose log settings that specify that log entries are made (mirrored, if possible), and that log backups are required.

- To ensure uninterrupted database operation, you must ensure that there is always enough space available in the log area.
- For safe database operation, it is necessary for the log entries to be backed up at regular intervals (even when you mirror the log area).

Perform log backups at regular intervals. You can best ensure uninterrupted database operation by activating [automatic log backups \[Page 43\]](#).

The [log segment size \[Page 82\]](#), the size of the log volume, and the capacity of the [backup medium \[Page 129\]](#) must be matched to each other. The log segment size should be selected so that enough space remains in the log area for any changes made during the backup of a log segment.



Database Instance Type

The SAP DB database system supports different application areas. The SAP DB [database instance \[Page 13\]](#) has different characteristics depending on the application area. The following database instance types exist:

- [SAP DB OLTP \[Page 23\]](#)
- [liveCache \[Page 23\]](#)
- [SAP DB Document Server \[Page 23\]](#)
- [SAP DB OLAP \[Page 23\]](#)

- [SAP DB E-Catalog \[Page 24\]](#)

See also:

[SAP DB Versions and Database Instance Types \[Page 24\]](#)



SAP DB OLTP

SAP DB OLTP is a [database instance type \[Page 22\]](#) of the SAP DB database system. SAP DB is a relational database system that was developed for OLTP (Online Transaction Processing). The database system is optimized to process individual transactions fast in environments with a high number of users and large databases.



liveCache

liveCache is a [database instance type \[Page 22\]](#) of the SAP DB database system. This database instance type is used for SAP applications only.

In Supply Chain Management, large volumes of data must be permanently available and changeable. For this reason, an addition has been made to the [SAP DB OLTP \[Page 23\]](#) relational database system to enable actual data structures and data flows (such as networks and relationships) to be mapped more easily and effectively. This product is called liveCache. liveCache is object-oriented and, unlike SAP DB OLTP, operates with its data only in the main memory of the database system, if configured optimally.



SAP DB Document Server

SAP DB Document Server is a [database instance type \[Page 22\]](#) of the SAP DB database system.

In today's information landscape, a large amount of data must be processed that does not have the typical format of a relational database, but is "unstructured" (such as videos, XML documents). SAP DB Document Server was developed on the basis of the [SAP DB OLTP \[Page 23\]](#) relational database system to ensure that as much of this unstructured data as possible can be processed outside the OLTP database. This improves the performance of the SAP DB OLTP database.

One application example is the SAP Content Server.



SAP DB OLAP

SAP DB OLAP is a [database instance type \[Page 22\]](#) of the SAP DB database system.

Online Analytical Processing (OLAP) technologies enable you to perform flexible analyses from a variety of business perspectives. It is based on a multi-dimensional data model that is achieved using relational database tables.

One application example is the Business Warehouse System. In contrast to [SAP DB OLTP \[Page 23\]](#) systems, a Business Warehouse System is configured so that large quantities of historical and operative data can be formatted with acceptable response times.

SAP DB E-Catalog

SAP DB E-Catalog is a [database instance type \[Page 22\]](#) of the SAP DB database system.

In Internet catalog applications, a small number of hits must be determined from a large number of product descriptions. The functions of a [SAP DB OLTP \[Page 23\]](#) database instance are enhanced for this purpose, if necessary.

One application example is the BugsEye or eMerge product from Requisite.

For this application, the TREX search engine has been integrated into the SAP DB OLTP relational database system. You can use the TREX search engine to index product descriptions (long texts) and search for search terms (exact search, phrase search, fuzzy search, linguistic search).

SAP DB Versions and Database Instance Types

The following table contains a list of the SAP DB versions of the individual [database instance types \[Page 22\]](#).

| SAP DB Version | Database Instance Type | | | | |
|----------------|------------------------|-----------|------------------------|-------------|------------------|
| | SAP DB OLTP | liveCache | SAP DB Document Server | SAP DB OLAP | SAP DB E-Catalog |
| 7.2.05 | × | × | × | × | |
| 7.3.00 | × | | × | × | × |
| 7.3.32 | | | | | × |
| 7.4.01 | | × | | | |
| 7.4.02 | | × | | | |
| 7.4.03 | × | × | × | × | × |
| 7.4.04 | × | × | × | × | × |



Operating System Platforms

SAP DB Version 7.4 supports the following operating system platforms:

| Operating System Platform | Name in Installation Package: <os> <arch> |
|-------------------------------------|--|
| Compaq Tru64 Unix (Alpha) | tru64 alpha |
| HP-UX (HP-PA) | hpux hppa |
| IBM AIX (PowerPC) | aix ppc |
| Linux (Intel Pentium) | linux i386 |
| SUN Solaris (SPARC) | solaris sparc |
| Windows 2000 (Intel Pentium) | win i386 |
| Windows XP (Intel Pentium) | win i386 |
| Windows Server 2003 (Intel Pentium) | win i386 |
| Windows Server 2003 (Intel Itanium) | win ia64 |



liveCache does not support Windows XP.



Multiprocessor Configuration

To allow multiprocessor configurations to be used to the best advantage, the database system supports external/internal tasking that can be configured.

The aim here is to allow the maximum possible number of [database sessions \[Page 132\]](#) to be supported by the minimum possible number of operating system threads.

The configuration of the database system controls the degree of external/internal tasking via the two parameters [MAXUSERTASKS \[Page 84\]](#) and [MAXCPU \[Page 83\]](#).



On a computer, four processors are available for the [database instance \[Page 13\]](#). No more than 800 database sessions should be running simultaneously.

You set MAXCPU to four, and MAXUSERTASKS to 800.

The database instance can then utilize up to four processors by establishing four operating system threads, each of which performs internal tasking for up to 200 users.



User Concept

The SAP DB database system uses [SAP DB user classes \[Page 26\]](#) and supports different roles ([role concept \[Page 30\]](#)).

Setting Up a Database Session

A [database session \[Page 132\]](#) is started when the user logs on to the [database instance \[Page 13\]](#). In order to log on, certain user data needs to be transferred to the SAP DB component for identification purposes.

The following user data is needed to log on:

| | |
|-----------------|--|
| user_name | User name |
| password | The user's password |
| database_name | The name of the SAP DB instance you want to work on |
| database_server | The name of the server on which the database you called is running |

In addition to this generally required user data, you can enter further data, which is then transferred when you log on.

You can also transfer the user data to the SAP DB tools or the C/C++ Precompiler programs in the following ways:

- Enter [user data as options \[Page 31\]](#) (note that the [required options \[Page 32\]](#) must always be entered for each tool used)
- Enter user data on the logon screen (for the Database Manager GUI and SQL Studio tools)
Note the special features of entering user data in the SQL Studio ([Opening a Database Session \[See SAP DB Library\]](#)).
- [Use XUSER \[Page 33\]](#) to specify user data

Entering incomplete or incorrect user data when calling the C/C++ Precompiler programs, application programs, or the SAP DB tools causes the program to terminate with an error message.



When you call the C/C++ Precompiler and application programs that were generated using the SAP DB programming interface, you can also transfer user data directly in the program itself. For this reason, the program contains a special modified rule for the transfer of user data. Basically, the logon options for the C/C++ Precompiler and application programs are the same as for the SAP DB tools.



SAP DB User Classes

The SAP DB database system differentiates between two main user classes:

- [Database Manager Operator \(DBM Operator\) \[Page 26\]](#)
- [Database Users \[Page 28\]](#)



Database Manager Operator (DBM Operator)

Users working with the [Database Manager \[Page 61\]](#) are known as Database Manager Operators. The [SAP DB user class \[Page 26\]](#) is the DBM operator.

Depending on what [authorizations \[Page 27\]](#) the DBM operator has been given, a DBM operator is able to perform all kinds of Database Manager functions.

You create the **first** DBM operator when you install a [database instance \[Page 13\]](#).



The Database Manager stores the name and password of the DBM operator in uppercase characters.

When you register a database instance, you must specify the name and the password of the first DBM operator. In this way, you legitimize yourself as the DBM operator of the database instance. The DBM operator's password can be changed at a later date.

- This DBM operator is then responsible for managing and monitoring the database system and for running backups and recoveries. The DBM operator is also authorized to perform all Database Manager functions, regardless of what operational state the database instance is in.
- The DBM operator is also authorized to create additional DBM operators, and assign them some or all of his or her authorizations.
- The DBM operator can log on to the Database Manager more than once, which means the DBM operator can, for example, query operating parameters while functions that take a long time are still running.



DBM operators are **not** [database users \[Page 28\]](#). To be able to work with a database instance, you must create the database user (with the [SQL Studio \[Page 67\]](#), for example).

Authorizations

The database system SAP DB makes a distinction between two types of authorizations for the [Database Manager operator \[Page 26\]](#) (DBM operator):

- [User authorizations \[Page 27\]](#)
Authorizations of the DBM operator for executing the functions of the [Database Manager \[Page 61\]](#)
- [Operating system user authorizations \[Page 28\]](#)
Authorizations of the DBM operator for accessing remote servers via the Database Manager

User Authorizations

User authorizations are the [authorizations \[Page 27\]](#) of the [Database Manager operator \[Page 26\]](#) (DBM operator) that he/she needs to use the required functionality of the [Database Manager \[Page 61\]](#).

An authorization may cover more than one command and one command may have more than one authorization assigned to it.

Use the Database Manager to assign the relevant user authorizations to a DBM operator.

See also:

Database Manager CLI: SAP DB 7.4: [User Authorizations \[See SAP DB Library\]](#).

The entries in the following table refer to the relevant sections of the documentation on the Database Manager CLI.

| User authorization | Description |
|---|-------------------------------------|
| DBInfoRead [See SAP DB Library] | Displaying state information |
| ExecLoad [See SAP DB Library] | Running the LOAD program |
| SystemCmd [See SAP DB Library] | Executing operating system commands |

| | |
|--|--|
| Backup [See SAP DB Library] | Carrying out backups |
| InstallMgm [See SAP DB Library] | Installation management |
| LoadSysTab [See SAP DB Library] | Loading system tables |
| DBStart [See SAP DB Library] | Starting the database instance |
| DBStop [See SAP DB Library] | Stopping the database instance |
| UserMgm [See SAP DB Library] | Managing the DBM operators |
| Recovery [See SAP DB Library] | Restoring backups |
| DBFileRead [See SAP DB Library] | Accessing database files (read-only) |
| ParamCheckWrite [See SAP DB Library] | Accessing database parameters (checked write) |
| ParamFull [See SAP DB Library] | Accessing database parameters (read and write) |
| ParamRead [See SAP DB Library] | Accessing database parameters (read-only) |
| AccessSQL [See SAP DB Library] | Accessing an SQL session |
| AccessUtility [See SAP DB Library] | Accessing a utility session |

[Default Authorizations for the First DBM Operator \[Page 28\]](#)



Default Authorizations for the First DBM Operator

The first [DBM operator \[Page 26\]](#) you create when you register a database instance has all [user authorizations \[Page 27\]](#).



Operating System User Authorizations

The operating system user authorization is one of the [authorizations \[Page 27\]](#) of the [Database Manager operator \[Page 26\]](#) (DBM operator). To execute operating system commands on remote computers using the [Database Manager \[Page 61\]](#), the DBM operator must have operating system user authorizations on those computers.

Use the Database Manager functionality to assign a DBM operator the user identification for the operating system.



Database User

Database users log on to a [database instance \[Page 13\]](#) and work with database objects such as tables, views, and indexes (the [SAP DB user class \[Page 26\]](#) is database user).

SAP DB differentiates between various [database user classes \[Page 29\]](#).

Database users can be grouped into [user groups \[Page 30\]](#).



Database User Classes

[Database users \[Page 28\]](#) are divided into database user classes. The database user class that a database user belongs to defines which operations that user is allowed to execute.

SAP DB differentiates between the following database user classes:

- [SYSDBA \[Page 29\]](#)
- [DBA \[Page 29\]](#)
- [DOMAIN \[Page 30\]](#)
- [RESOURCE \[Page 30\]](#)
- [STANDARD \[Page 30\]](#)



SYSDBA

The SYSDBA user (database system administrator) is a special [database user \[Page 28\]](#) of the [database user class \[Page 29\]](#) SYSDBA. The SYSDBA user is the first database user that the [Database Manager \[Page 61\]](#) creates when a new [database instance \[Page 13\]](#) is installed. Enter a user name and password for this user.



When you enter the data for the SYSDBA user and when you use it in the SQL environment, remember that the Database Manager automatically converts all characters into uppercase.

Each database instance has a single SYSDBA user.

The SYSDBA user's password can be changed once the database registration has finished (with [SQL Studio \[Page 67\]](#), for example).

The SYSDBA user is very important, especially when database instances are installed. The SYSDBA user is responsible for setting up the system and for creating other database users. The SYSDBA user is the owner of system tables. When system tables are uploaded, the upload tool logs on to the database instance as SYSDBA.

The SYSDBA is able to define data and database procedures. The SYSDBA can also grant other users privileges for these database objects.



DBA

A DBA user (database system administrator) is a special [database user \[Page 28\]](#) in the [database user class \[Page 29\]](#) DBA. A database user must be created by [SYSDBA \[Page 29\]](#) (with the [SQL Studio \[Page 67\]](#), for example).

A DBA user is authorized to create [RESOURCE \[Page 30\]](#) and [STANDARD users \[Page 30\]](#). The DBA user can also define data and database procedures and grant other users all or some DBA user privileges for these database objects.

A DBA user can group users with identical access rights into [user groups \[Page 30\]](#).

A special DBA user is the user [DOMAIN \[Page 30\]](#).



DOMAIN

The DOMAIN user is a special [database user \[Page 28\]](#) of the [database user class \[Page 29\]](#) [DBA \[Page 29\]](#). The DOMAIN user is the owner of the system tables of the [database catalog \[Page 131\]](#).

Just like the [SYSDBA \[Page 29\]](#) user, the DOMAIN user is created when a [database instance \[Page 13\]](#) is installed. The DOMAIN user is created by the system under the pre-defined name DOMAIN. The password is the same as that of the SYSDBA user that was created earlier.

The DOMAIN user's password can be changed at a later date.



RESOURCE

RESOURCE users are special [database users \[Page 28\]](#) of the [database user class \[Page 29\]](#) [RESOURCE](#). RESOURCE users can be created by [SYSDBA users \[Page 29\]](#) and [DBA users \[Page 29\]](#).

RESOURCE users can define data and database procedures and grant other users privileges for these database objects.



STANDARD

STANDARD users are special [database users \[Page 28\]](#) of the [database user class \[Page 29\]](#) [STANDARD](#). STANDARD users only have access to data and database procedures that were defined by other users and for which they have privileges.

STANDARD users themselves can define view, synonyms, and temporary tables.



User Groups

[Database users \[Page 28\]](#) can be grouped into user groups.

A user group can either be assigned to the [database user class \[Page 29\]](#) [RESOURCE \[Page 30\]](#) or to the database user class [STANDARD \[Page 30\]](#). Database users can be defined as members of a user group.

All database objects defined by members of a certain user group can be identified by the user group name. The owner of objects such as these is the user group and not the individual user that defined the objects. If a member of a user group creates objects, each member of that group can work with these objects as if they were the object owners.

Privileges can only be granted or removed from the user group as a whole and not from individual members of the group.



The Role Concept

The SAP DB database system supports different roles. A [role \[See SAP DB Library\]](#) is a grouping of [privileges \[See SAP DB Library\]](#), which can be assigned to [database users \[Page 28\]](#), [user groups \[Page 30\]](#), or other roles.

Procedure

1. A role is created using the [CREATE ROLE statement \[See SAP DB Library\]](#). This role is initially empty. Only database users belonging to database user class [DBA \[Page 29\]](#) are able to create roles. The new role [name \[See SAP DB Library\]](#) cannot be the same as the name of any other role, a user, or a user group.
2. Privileges are assigned to a role using the [GRANT statement \[See SAP DB Library\]](#). Privileges can be revoked from a role using the [REVOKE statement \[See SAP DB Library\]](#).
3. A role can be assigned to database users, user groups, or other roles using the GRANT statement and specification of the role name.
4. You use the [ALTER USER- \[See SAP DB Library\]](#) and [ALTER USERGROUP-statement \[See SAP DB Library\]](#) to define which of the roles that were assigned to a user or user group is to be used when a [database session \[Page 132\]](#) is opened.
5. During a database session, you can use the [SET statement \[See SAP DB Library\]](#) to activate other roles assigned to the user or user group
If a role is activated during a session, the current user then has all the privileges that are assigned to a role.
If a password has been assigned to a role, users assigned to that role can only activate it by entering the password in the SET statement.

Result

That a role is available and the properties of that role are all registered in the [catalog \[Page 131\]](#) in the form of metadata. A user that creates a role becomes its owner.

The roles assigned to the user or user group as a result of the ALTER USER and ALTER-USERGROUP statements are activated as soon as a database session is opened.



Roles are not active during execution of [data definition commands \[See SAP DB Library\]](#).

See also:

Reference Manual: SAP DB 7.4, Section [Role Name \[See SAP DB Library\]](#)



User Data as Options

User data and, in some cases, other data can be entered as options when calling the SAP DB tools, the C/C++ Precompiler, or the application programs.

Syntax

```
<dbmcli | loadercli | dbmgui | sqlsto | cpc <file_name>> [<options>]
```

```
...
```

```
<options>
```

| SAP DB Component | |
|----------------------|---|
| Database Manager CLI | DBMCLI options [Page 63] |
| SAP DB Loader | Options (LOADERCLI) [Page 66] |
| Database Manager GUI | DBMGUI options [Page 62] |
| SQL Studio | SQL Studio options [Page 67] |

C/C++ Precompiler

[C/C++ Precompiler options \[See SAP DB Library\]](#)

If you enter all the required user data (in most cases, the options `-u`, `-d`, `-n`) and these entries are complete and correct, you are logged on to the chosen SAP DB tool or C/C++ Precompiler is called.

For each tool, the user data required for logon to a database instance must be entered. Options are not necessarily required for the Database Manager GUI and SQL Studio, but this data can also be entered in a logon screen. For other tools, there are [required options \[Page 32\]](#), which must always be entered.

The SAP DB tool Database Manager CLI and the C/C++ Precompiler support the XUSER concept ([User Data Using XUSER \[Page 33\]](#)).



Required Options

If you do not enter all the required [user data as options \[Page 31\]](#), or if you were unable to log on to the [database instance \[Page 13\]](#) with that user data, the SAP DB tools will react as follows:

- [Database Manager GUI \[Page 61\]](#), [SQL Studio \[Page 67\]](#)
Entry of options is not required. These tools are started even if options are not entered, or if options are incorrect or incomplete. Database Manager GUI and SQL Studio always display a logon screen. In the logon screen, enter the user data required to log on to the database instance.
- [Database Manager CLI \[Page 62\]](#)
Only some of the options required to log on to the database instance were entered, or they were incorrect. The Database Manager CLI starts and you are connected to the DBM Server. However, you are not logged on to the database instance. In this case, the Database Manager CLI can only be used to call DBM commands that are not related to the database (for example, `help`, `dbm_version`).
You need to connect to the database instance to use all the functions. Enter the option `-d <database_name>`, if the database instance is on the local computer, and enter the option `-n <server_node>` as well if the database instance is on a remote computer.
Further user data (such as user name and password) can be transferred as options (for example `-u`) or, if required, as DBM commands (for example `user_logon`).



You should always start the Database Manager CLI by entering the options `-d`, `-u` and, if required, `-n` when calling the tool ([Options \(DBMCLI\) \[Page 63\]](#)).

- [LOADERCLI \[Page 65\]](#)
The minimum required option `-d <database_name>` for logging on to the database instance and the required option `-b <command_file>` for specifying the command file were not entered at all, were incomplete, or incorrect. The loader is not started, but displays a list of all possible options.
Log on to the SAP DB Loader again by entering the required options.
Further user data (such as user name and password) can be transferred as options (for example `-u`) or, if required, as commands (for example `use_user`).



When you call the tool, you should always start the SAP DB Loader by entering the required options `-d` and `-b` ([Options \(LOADERCLI\) \[Page 66\]](#)).



User Data and XUSER

Using the tool XUSER, you can predefine and save user data ([using XUSER \[Page 33\]](#)). You can access your [XUSER data \[Page 35\]](#) when you call the [Database Manager CLI \[Page 62\]](#) or a precompiler program or when you start an SAP system. In this case, the user would use the data stored under a certain user key (`user_key`).

- Windows 2000/Windows XP: You can manage individual user data in this way if, for example, more than one database user is using the same Windows PC and each user is logging on under a different name.
- UNIX: You can manage individual user data in this way if, for example, more than one database user is using the same HOME directory and each user is logging on under a different name.

Procedure

1. The database administrator ([DBA operator \[Page 29\]](#)) sets up a [database user \[Page 28\]](#) using the relevant CREATE USER statement.
2. The database administrator informs the operating system user what this database user's data is.
3. The operating system user saves the database user's data using XUSER.

Result

When the Database Manager CLI or the C/C++ Precompiler is now called using the option `-u <user_key>`, the system uses the data that was defined in `user_key` using XUSER. Enter the user key exactly as it was defined in XUSER, that is, your entries are case-sensitive.



Applications such as [SAP DB Loader \[Page 65\]](#) and [Database Manager GUI \[Page 61\]](#) and applications that use the ODBC interface (for example, [SQL Studio \[Page 67\]](#)) have no way of accessing XUSER data.



Using XUSER

User data for setting up a [database session \[Page 132\]](#) can be predefined and saved by entering a user key (`user_key`) using the tool XUSER. When the [Database Manager CLI \[Page 62\]](#) or the C/C++ Precompiler is called or an SAP system is started, the system can access the required [XUSER data \[Page 35\]](#) by entering the relevant user key).

Use

- You can use the [XUSER options \[Page 34\]](#) to generate the XUSER data (options for the required data combined with the option `set`).
- You can use the option `-b` to generate the XUSER data ([Generating the XUSER Data in the Background \[Page 34\]](#)).
- If you do not generate the XUSER data, you can use the data in the user key DEFAULT.
- XUSER data is called using option `-u` ([User Data Using XUSER \[Page 33\]](#)).
- You can use the XUSER options to delete the XUSER data (options for the required data combined with the option `clear`).



XUSER Options

When you [use the XUSER tool \[Page 33\]](#), you can use the XUSER options to help you set your XUSER data. The user key data that you set with the options overrides any data already specified in the user key.

You can use the following options:

| Option | Explanation (see also XUSER Data [Page 35]) |
|---------------------------|---|
| -h | Help texts |
| -v | XUSER version |
| -A ASCII8 | XUSER data is interpreted as ASCII8 |
| -b <file_name> | Generates XUSER data in the background [Page 34] |
| -U <user_key> | User key |
| -u <user_name>,<password> | User name and password |
| -d <database_name> | Name of the database instance [Page 132] |
| -n <database_server> | Host server name |
| -S <SQL_mode> | SQL mode [Page 138] |
| -t <timeout> | Timeout value [Page 138] |
| -I <isolation_level> | Isolation level [Page 112] |
| list | xuser list: List of data for all user keys xuser -U <user_key> list: Data for specified user key |
| clear | xuser clear: Deletes data for all user keys xuser -U <user_key> clear: Deletes data for specified user key |
| set | xuser <options> set: Data set for user key DEFAULT <options> ::= -u -d -n -S -t -I xuser -U <user_key> <options> set: Data set for specified user key |



The options `list`, `clear` and `set` must be always be the last options specified in the list.



Generating XUSER Data in the Background

If you use the [XUSER tool \[Page 33\]](#), you can generate the [XUSER data \[Page 35\]](#) in the background by using a file specified with the option `-b` when you call XUSER.

Syntax

```
xuser -b <file_name>
```

You are free to define your own file name. The file is made up of groups of 10 lines:

The option `-b <file_name>` always generates new XUSER data. Any existing XUSER data is overwritten.

Structure of the File <file_name>:

The entries in the file <file_name> begin in the first column. There are no field descriptions and they contain the following data in the specified order:

```
<user_key>
<user_name>
<password>
<database_name>
<database_server>
<SQL_mode>
                empty line
<timeout>
<isolation_level>
                empty line
```

This data is explained in more detail in the section [XUSER Data \[Page 35\]](#). If you do not want to enter any optional data, make sure the line is left blank. The empty lines in the file must be left empty, since they are meant to contain other XUSER data that is no longer relevant for SAP DB Version 7.4.



```
DEFAULT
meyer
secret
TEST1
server1
INTERNAL
                empty line
-1
-1
                empty line
```

This example contains all data required for SAP DB Version 7.4.



```
HOME
meyer
"top_secret"
TEST2
server1
                empty line
                empty line
90
1
                empty line
```

This example contains no data on the SQL mode.



XUSER Data

You can save XUSER data when you [use the XUSER \[Page 33\]](#) tool.

XUSER can manage up to 32 combinations of XUSER data for each operating system user.

- Windows 2000/Windows XP: XUSER data is stored in the database registry. You are not permitted to make manual changes in the registry because these may result in inconsistencies in the operating system.

- UNIX: XUSER data is stored in the file `.XUSER.62` (also referred to as the XUSER file) in the user's HOME directory.

Procedure

You can generate the XUSER data as followed:

- Use [XUSER options \[Page 34\]](#)
- [Generate XUSER data in the background \[Page 34\]](#)

You can enter the following XUSER data with SAP DB Version 7.4:

| XUSER Data | Explanation |
|-----------------|---|
| user_key | Name of the user key used to address this combination of XUSER data (character string with a maximum of 18 characters) XUSER data is delivered under the user key name DEFAULT in the program XUSER. This name cannot be changed. If you enter additional key names, these are case-sensitive. If you do not specify a user key, then the remaining data refers to the DEFAULT user key. |
| user_name | User name (character string with a maximum of 18 characters) If the user name contains small letters or special characters, it must appear within double quotation marks. Otherwise small letters will be converted to capitals. |
| password | Password of user (character string with a maximum of 18 characters) If the password contains small letters or special characters, it must appear within double quotation marks. Otherwise small letters will be converted to capitals. |
| database_name | Name of the database instance [Page 132] that you want to work with (character string with a maximum of 18 characters) If you do not specify a database, the system uses the name in the environment variable SERVERDB. This entry is case-sensitive. |
| database_server | Name of the server on which the database is running (character string with a maximum of 64 characters) If you do not specify a name, the local server is selected. This entry is case-sensitive. |
| SQL_mode | SQL mode [Page 138] (character string with a maximum of eight characters) If you do not specify a mode, the INTERNAL SQL mode is selected. Possible values: INTERNAL, SAPDB, ANSI, DB2, ORACLE, SAPR3 This parameter affects the precompiler programs. |
| timeout | Timeout value [Page 138] in seconds If you do not specify a value, the value -1 is chosen. This means that the default value of the database instance is used. Possible values: -1, <number>, 0 If you set the timeout value to 0, then no timeout value is used. |
| isolation_level | Isolation level [Page 112] (for application programs and precompilers only) If you do not specify a value, the value -1 is chosen. This means that the default value of the database instance is used. Possible values: -1, 0, 1, 2, 3, 10, 15, 20, 30 |



Security Concepts

- In production database systems, choose the following [log settings \[Page 37\]](#): mirror the [log area \[Page 135\]](#), the log area can be overwritten only after it has been backed up, and [redo log management \[Page 47\]](#) is always activated
- You can guarantee a [high level of security against the failure \[Page 40\]](#) of the SAP DB database system by using fault-tolerant hardware and software.
- To enable you to implement your chosen data [backup strategy \[Page 40\]](#), SAP DB offers tools for performing [data backups \[Page 42\]](#) and [log backups \[Page 43\]](#).
- The [restartability \[Page 45\]](#) of the SAP DB database system enables some errors to be corrected automatically when the database system is started.



Log Settings

The following log settings are possible:

- You can mirror the log area.
If you cannot use hardware-base mirroring for the [log area \[Page 135\]](#), we recommend that you use the software-based mirroring setting.
To make this setting, proceed as described in [Log Mode \[Page 38\]](#).
- You can decide to do without [log backups \[Page 43\]](#) and use a non-standard log backup strategy by setting the overwrite mode for the log area accordingly.
To make this setting, proceed as described in [Overwrite Mode \[Page 38\]](#).
- In very special situations, you can deactivate [redo log management \[Page 47\]](#) temporarily.
To do this, proceed as described in [Activating or Deactivating Redo Log Management \[Page 39\]](#).

Combination of Log Settings

The combination of log settings you choose depends on your security requirements.

In production systems, you should aim for the highest possible security standards. In this case, make the following log settings:

- Mirror log area (hardware-based)
- Standard log backup operation (with log backups)
- Redo log management always activated
- Perform data and log backups

If you require a lower level of security for your system, you can decide not to mirror the log area.

Use different log backup operations only if all you require in the event of a database crash is for the system to be restored to the state of an existing data backup .

Deactivating redo log management places the security of your system at serious risk, and should be considered only under exceptional circumstances.



Log Mode

One possible [log setting \[Page 37\]](#) is the log mode. Use this setting to control whether the [log area \[Page 135\]](#) is mirrored or not.

Single Log Area

By default, the database instance uses only one log area.

Even if the database instance only uses one log area, you can still guarantee a high level of security for your database system, since RAID-5 or RAID-1 hardware-based mirroring is usually implemented. RAID systems guarantee a good level of security for production use of the database.

Mirroring the Log Area

If you do not have the option of using hardware-based mirroring, you can use database methods to mirror the log area by setting the log mode accordingly.

When the log area is mirrored, the database instance uses the two log areas as follows:

- Data is written to both log areas in parallel.
- However, during a [restart \[Page 136\]](#) or log backup, the log entries are only read from one log area – the primary log entry.

When you use [standard log backups \[Page 38\]](#), neither of the two log areas can be overwritten until the log entries of the primary log area have been backed up.

If access problems occur in the primary or secondary log area, the [log volume \[Page 22\]](#) in question is marked as `BAD`, and the database instance is switched to the [operational state \[Page 136\]](#) `OFFLINE`.

The defective log volume of a log area can be recovered in the operational state `ADMIN`. In this process, the complete contents of the relevant log volume are copied from the other log area to the defective log volume.

This ensures that the database instance does not have to be recovered after a disk error that led to the loss of a log volume, as the content of the log volume was mirrored.

Procedure

Set the log mode, and configure the extra log volumes you require. Use the Database Manager for this.

Database Manager GUI: SAP DB 7.4, Section [Changing the Log Mode \[See SAP DB Library\]](#)

Database Manager CLI: SAP DB 7.4, Section [Changing Volume Parameters \[See SAP DB Library\]](#)



Overwrite Mode for the Log Area

One possible [log setting \[Page 37\]](#) that you can make is to specify the overwrite mode for the log area. This overwrite mode controls whether the [log area \[Page 135\]](#) can be overwritten or not without backing up the log entries.

Normal Log Backups

By default, the log area can be overwritten only if the appropriate log entries have been backed up, which is why you must make [log backups \[Page 43\]](#) at regular intervals (see [Backup Strategy \[Page 40\]](#)).

The log entries of the log area and the log backups are then available if you need to restore the database. As long as they are complete, you can use the data backup, log backups and the log entries in the log area to recover the database up to a chosen point in time.

If you change this default, and then reactivate it, you must then make a [complete data backup \[Page 42\]](#).

Overwriting the Log Area Without a Log Backup

The log area is overwritten cyclically, and you do not need to back up the log entries.

The log backup history is interrupted. Only the log entries of the log area are available if you need to recover the database. This usually means that you cannot recover your database instance up to the last COMMIT before the crash, or to any other time; instead, you can recover it only to the state it had at the time of the last data backup.

You can only recover the database to a time of your choice if all the log entries written after the data backup exist in the log area (which means that the log area has not been overwritten since the data backup).

Procedure

Use the Database Manager to set the overwrite mode to your requirements.



Activating or Deactivating Redo Log Management

One possible [log setting \[Page 37\]](#) that you can make is activating or deactivating [redo log management \[Page 47\]](#).



Deactivate redo log management only rarely and in special cases, and then only for a fixed period of time.

Activating Redo Log Management

Redo log management is activated by default.

If you deactivate redo log management, and then reactivate it, you must then make a [complete data backup \[Page 42\]](#).

Deactivating Redo Log Management

When you deactivate redo log management, the [Log Writer \[Page 48\]](#) no longer writes any [redo log entries \[Page 46\]](#) to the [log area \[Page 135\]](#). Disabling redo log management means that the log backup history is interrupted, so if a recovery is necessary, no log entries are available for this period. You should minimize this security risk, that is to say, you should keep the period in which there is no active redo log management as short as possible.

This setting is particularly useful, for example, when you are initializing a database instance, and the conditions are exceptionally critical for performance.

Procedure

Use the Database Manager to activate or deactivate redo log management.



Availability

Using the appropriate hardware, operating system, or database features can improve the downtime security of a [database instance \[Page 13\]](#).

- [Log volumes \[Page 22\]](#)

In a production system, the [log area \[Page 135\]](#) should always be mirrored for security. Use hardware-based methods to mirror the log area. If this is not possible, you can set the [log mode \[Page 38\]](#) of the SAP DB database system to use software-based methods for the mirroring.

Operate the log volumes in a RAID-1 configuration, since the database instance writes the log entries sequentially.

In a production system, always stick to [standard log backups \[Page 38\]](#), and never deactivate [redo log management \[Page 47\]](#), even if you use RAID systems for the data volumes.

- [Data volumes \[Page 21\]](#)

If you wish to ensure a high standard of availability, we recommend using RAID-5 or RAID-1 configurations for the [data area \[Page 131\]](#). A disk crash and change will then not affect the running of the database, if the RAID system is able to carry out a recovery.

Every volume category should be stored on a different disk.

When using fault-tolerant hardware, it is best to only use the same type of hardware when you want to extend the capacity. For example, RAID-5 systems should only be extended using RAID-5 systems and mirroring disks with mirroring disks.

UNIX: In a production system, data volumes and log volumes should be used in conjunction with raw devices. Raw devices achieve better performance, and are more secure in the event of a system failure.



Backup Strategy

One key element of the [security concept \[Page 37\]](#) for your database system is the regular backing up of your data. You should therefore perform the following [backups \[Page 41\]](#) at regular intervals: [data backups \[Page 42\]](#) and [log backups \[Page 43\]](#).

You can use [external backup tools \[Page 45\]](#) to reduce the time needed for backup.

Data Backup

Note that, in the case of a recovery, a more up-to-date data backup means that fewer log entries need to be reproduced. Therefore, perform data backups as often as possible.

- Perform a [complete data backup \[Page 42\]](#) on each day of production.
- If you cannot or do not want to perform a data backup every day, you should at least perform an [incremental data backup \[Page 42\]](#) on each day of production.



If complete data backup is active, incremental data backup cannot be started.

You can perform data backups in parallel to reduce the time required for the backups.

Log Backups

In production systems, choose the following [log settings \[Page 37\]](#): mirror the [log area \[Page 135\]](#), the log area can be overwritten only after it has been backed up, and [redo log management \[Page 47\]](#) is always activated. Perform log backups at regular intervals.

- [Automatic log backup \[Page 43\]](#)

The automatic log backup (autosave log mechanism) should always be active. Complete and incremental data backups are also possible while the automatic log backup is active.

- [Interactive log backup \[Page 44\]](#)

You should save the log entries every productive day.

If automatic log backup is not active, you must regularly check that there is enough storage space available in the [log area \[Page 135\]](#). If necessary, back up the log area immediately by starting an interactive log backup. If new log entries cannot be written to the log area because there is not enough storage space, the database stops.

- Archiving of [version files \[Page 140\]](#)

You should regularly archive the version files written during a log backup to a medium of your choice. Once the files have been archived successfully, you can delete the version files written during the log backup, and thereby ensure that there is enough space available in the directory for version files.

If you use an [external backup tool \[Page 45\]](#), you can automate the archiving of version files. To do this, follow the instructions in [Archiving Version Files of the Log Area \[See SAP DB Library\]](#).



Tapes containing data or log backups should not be directly overwritten with the next backup. If you retain, say, the last four backup generations, it may be possible to use an older backup if a media failure occurs.



Backup

The following backups should be carried out for the SAP DB database system at regular intervals:

- [Data backup \[Page 42\]](#)
- [Log backup \[Page 43\]](#)

All backups can be carried out on an [individual backup medium \[Page 138\]](#), and data backups can also be carried out on a [group of parallel backup media \[Page 134\]](#).

The naming convention for a [backup medium \[Page 129\]](#) depends on whether you use an [external backup medium \[Page 45\]](#).

- [Name of standard backup medium \[Page 135\]](#)
- [Name of external backup medium \[Page 135\]](#)

See also:

[Backup Strategy \[Page 40\]](#)



Data Backup

A data backup saves the contents of the [data volumes \[Page 21\]](#), and the database parameter values needed for recoveries.

For the SAP DB database system, we differentiate between the following types of data backup:

- [Complete data backup \[Page 42\]](#)
- [Incremental data backup \[Page 42\]](#)

You can perform complete or incremental backups in the [operational state \[Page 136\]](#) ONLINE or ADMIN.

Data backups are always consistent in themselves. To recover the database, you need, alongside the data backup, the appropriate [log backups \[Page 43\]](#) for a point-of-failure recovery or point-in-time recovery. The time you choose must be after the completion of the data backup.

You can perform data backups in parallel ([Parallel Backup \[Page 42\]](#)).

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Data Backups \[Page 43\]](#)



Complete Data Backup

In a complete [data backup \[Page 42\]](#), all the pages of the [data volumes \[Page 21\]](#) are backed up to the [backup medium \[Page 129\]](#) you specified.

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Data Backups \[Page 43\]](#)



Incremental Data Backup

In an incremental [data backup \[Page 42\]](#), all the pages of the [data volumes \[Page 21\]](#) changed since the last [complete data backup \[Page 42\]](#) are backed up to the [backup medium \[Page 129\]](#) you specified.

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Data Backups \[Page 43\]](#)



Parallel Backup

Parallel processing is possible for a [data backup \[Page 42\]](#). In this case, a [group of parallel backup media \[Page 134\]](#) must be defined.

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Data Backups \[Page 43\]](#)



Saving Data Backups

For information on how to save [data backups \[Page 42\]](#), refer to the following documentation:

- *Database Manager GUI: SAP DB 7.4*, Section [Backup Processes \[See SAP DB Library\]](#)
- *Database Manager CLI: SAP DB 7.4*, Section [Backing Up and Recovering Database Instances \[See SAP DB Library\]](#)
- *Database Management in CCMS: SAP DB OLTP* → *DBA Planning Calendar*



Log Backup

In a log backup, the contents of the [log area \[Page 135\]](#) are copied to [version files \[Page 140\]](#) or backed up using an [external backup tool \[Page 45\]](#).

A version file is generated for each log segment of the log area.

The space originally taken up by the backed up log area becomes free again after the log backup.

Log Backup Types

The following log backup options are supported for the database system SAP DB:

- [Automatic log backup \[Page 43\]](#)
- [Interactive log backup \[Page 44\]](#)

Need for log backup

- In the case of a **recovery**, a [data backup \[Page 42\]](#) alone is not enough to restore the current state of the [database instance \[Page 13\]](#) up to a certain point in time (for example, the time just before the disk error occurred). To restore the current state, both the data backup and the log entries that were written after the data backup must be imported into the database system.
- If new log entries cannot be written because there is **not enough storage space in the log area**, the database stops.

For these reasons, it is necessary to back up the log entries at regular intervals.

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Log Backups \[Page 44\]](#)



Automatic Log Backup

To guarantee the security and availability of the data in your database system, we recommend automatic [log backups \[Page 43\]](#).

If automatic log backup is activated, a log segment of the [log area \[Page 135\]](#) is saved as soon as it has been filled. This log segment is then released again. This mechanism ensures that overflows in the log area are practically impossible.

We particularly recommend activating automatic log backups for all [database instances \[Page 13\]](#) where many changes are made to the data, leading to a high volume of logs. In this way, constant monitoring of the usage level of the log area is not necessary.



As long as automatic log backup is active, you cannot start an [interactive log backup \[Page 44\]](#). However, you can perform a [data backup \[Page 42\]](#) at the same time.

You can only perform automatic log backups in the [operational state \[Page 136\]](#) ONLINE.

In the case of an automatic log backup, you can only use [version files \[Page 140\]](#) as the backup medium

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Log Backups \[Page 44\]](#)



Interactive Log Backup

You can use the interactive [log backup \[Page 43\]](#) to back up all the pages of the [log area \[Page 135\]](#) that have been written since the last log backup.

If you back up the log entries in [version files \[Page 140\]](#), a version file is also generated for the last log segment, which may not be full.

Prerequisites

A [full data backup \[Page 42\]](#) of the current database instance has been created.



You can perform interactive log backups in [operating mode \[Page 136\]](#) ONLINE or ADMIN.

You cannot perform any other backup while the interactive log backup is running.

See also:

[Backup Strategy \[Page 40\]](#)

[Saving Log Backups \[Page 44\]](#)



Saving Log Backups

For information on how to save [log backups \[Page 43\]](#), refer to the following documentation:

- *Database Manager GUI: SAP DB 7.4*, Section [Backup Procedure \[See SAP DB Library\]](#)
- *Database Manager CLI: SAP DB 7.4*, Section [Backing Up and Recovering Database Instances \[See SAP DB Library\]](#)
- *Database Management in CCMS: SAP DB OLTP* → *DBA Planning Calendar*



External Backup Tool

[Backups \[Page 41\]](#) can also be carried out using external backup tools on external backup media.

A detailed description of the use of external backup tools can be found in the following documentation: [External Backup Tools: SAP DB \[See SAP DB Library\]](#).

See also:

[External backup medium \[Page 134\]](#)

[Name of external backup medium \[Page 135\]](#)

[External backup ID \[Page 133\]](#)



Restartability

If the database fails (for example, as the result of a power failure), and the [volumes \[Page 21\]](#) were fully functioning, the database system carries out a [restart \[Page 136\]](#). This means that the effects that completed transactions had on the [data volumes \[Page 21\]](#) are reproduced (rolled forward), and the effects that uncompleted transactions would have had are cancelled out (rolled back).

If a data volume fails (physical disk error), the last, complete [data backup \[Page 42\]](#) must be imported after the problem has been rectified. However, the last consistent database state can only be recovered if all the required [redo log entries \[Page 46\]](#) can be imported from the [log area \[Page 135\]](#) or the [log backup \[Page 43\]](#) (if the information in the log area is insufficient).



Log Concept

One of the main internal functions of the SAP DB database system is logging.

In normal database operation ([operational state \[Page 136\]](#) ONLINE), the modifying SQL statements of every transaction must be logged, meaning that [log entries \[Page 45\]](#) ([redo \[Page 46\]](#) and [undo log entries \[Page 46\]](#)) are written. These log entries are required so that the database system can ensure transaction consistency and so that individual SQL statements can be reversed if required. In addition, the undo log entries are needed to enable consistent reading when no locks are set.

The task of [online logging \[Page 46\]](#) is to store the log entries so that they are constantly available for normal database operation.

In the case of a [restart or restore \[Page 51\]](#), the required log entries must be made available and processed in the correct time-based sequence.



Log Entry

The [log concept \[Page 45\]](#) includes the writing of log entries. The SAP DB database system differentiates between the following log entries:

- [Redo log entry \[Page 46\]](#)

- [Undo log entry \[Page 46\]](#)



Redo Log Entry

Redoing (rolling forward) a [transaction \[Page 139\]](#) means that the modifications of a successful transaction are repeated, that is to say, the database is set to the consistent state that it had after the transaction had ended. The redo measure means that additional, redundant data management is required.

For each transaction, the values of the changed objects are therefore recorded. These logged values are described as redo [log entries \[Page 45\]](#) (or after-image entries).

Redo log entries do not need to be saved permanently in the [log area \[Page 135\]](#) until the [COMMIT \[Page 131\]](#) is executed for the transaction ([Redo Log Management \[Page 47\]](#)).

The redo log entries are also required for the redo when the database system is [restarted or recovered \[Page 51\]](#).



Undo Log Entry

Undoing (rolling back) a [transaction \[Page 139\]](#) means that the transaction is reset, that is to say, the database is set to the consistent state that it had before the transaction was started. The undo measure means that additional, redundant data management is required.

Therefore, for each transaction, the original values of the data objects that are to be changed by the transaction, that is, the values that were available before the transaction was started, are stored. These logged values are described as undo [log entries \[Page 45\]](#) (or before-image entries).

Undo log entries are permanently stored in the [data area \[Page 131\]](#) before an SQL statement is executed ([Undo Log Management \[Page 49\]](#)). Each transaction can access its undo log entries without any locks being set.

In addition, the undo log entries are needed for the undo if the database is [restarted or recovered \[Page 51\]](#) and are used for [History Management \[Page 50\]](#).



Online Logging

The [log concept \[Page 45\]](#) for the SAP DB database system includes online logging, which is required for normal database operation.

- The [redo log entries \[Page 46\]](#) of the transactions are managed: [Redo Log Management \[Page 47\]](#)
- The [undo log entries \[Page 46\]](#) of the transactions are managed: [Undo Log Management \[Page 49\]](#)
- For the database instance type [liveCache \[Page 23\]](#), [History Management \[Page 50\]](#) is carried out for the undo log entries.



Redo Log Management

During [online logging \[Page 46\]](#), the [redo log entries \[Page 46\]](#) are written and managed.

- Active components: [transactions \[Page 139\]](#) in [user tasks \[Page 15\]](#), [log writer \[Page 48\]](#)
- Storage units: [log queue \[Page 47\]](#), [log area \[Page 135\]](#)

Prerequisites

[Log Settings \[Page 37\]](#)

Use

Redo log management plays a part in the following actions:

- During execution of a modifying transaction, the redo log entries are included in the log queue. The log writer writes the log pages from the log queue to the log area.
- During a [log backup \[Page 43\]](#), information is obtained on whether entries from the log area are to be backed up, and if so, how many.
- For display on the monitor, information is obtained on waiting periods, and the type of access to the entries in the log queue.
- In the case of a [restart or recovery \[Page 51\]](#), the required redo log entries are read from the log area.
- The database parameters and other internal information that controls logging are stored in the log area and updated at regular intervals.



Log Queue

The area of the main memory required for [redo log management \[Page 47\]](#) is called the log queue. The size of a log queue (in [log pages \[Page 48\]](#)) is determined by the database parameter [LOG_IO_QUEUE \[Page 82\]](#).

A [transaction \[Page 139\]](#) uses a log queue to obtain a main memory area for a [redo log entry \[Page 46\]](#). The transaction writes the redo log entry to the log pages of the log queues.

Writing of the log pages to the [log area \[Page 135\]](#) is carried out by the [log writer \[Page 48\]](#).

Process Flow

1. The [user task \[Page 15\]](#) of the transaction reserves main memory space for a redo log entry in the log queue.
2. The transaction writes the redo log entry to the reserved area of the log queue. The time at which the redo log entry is written to the log queue is assigned to the relevant log page (log queue sequence number).
3. The transaction releases the reserved area of the log queue for processing by the log writer, and provides information on whether it wants to wait for the log page from the log queue to be written to the log area. This behavior is always required for [COMMIT \[Page 131\]](#) and [ROLLBACK \[Page 137\]](#) operations.
If a transaction does wait for the redo log entry to be written, the log writer notifies the transaction once the relevant page has been written from the log queue to the log area,

and informs the transaction of the log sequence number that was assigned when the log page was written to the log area.



Log Page

The main storage units required for [redo log management \[Page 47\]](#) are the log pages. The size of the log pages is 8 KB. A fixed number of log pages defined in the database parameter [LOG_IO_QUEUE \[Page 82\]](#) forms the [log queue \[Page 47\]](#).

- A [transaction \[Page 139\]](#) writes [redo log entries \[Page 46\]](#) to the log pages of the log queues.
Each log page is assigned a log queue sequence number, which specifies the time at which the redo log entries are written to the log queue.
- If a log page is to be written to the [log area \[Page 135\]](#), the [log writer \[Page 48\]](#) is notified. The log writer fetches the log pages in question from the log queue, and writes them to the log area.
When each log page is written from the log queue to the log area, the log writer assigns it a log sequence number and a time stamp. These log sequence numbers can then be used to determine the unique sequence of the log pages in the log area. The time stamp is needed for a point-in-time recovery.

A log page that was written by the log writer, but was not full, remains in the log queue and can be filled with further redo log entries.



Log Writer

One active [task \[Page 138\]](#) of [redo log management \[Page 47\]](#) is the log writer.

The log writer is part of a [user kernel thread \(UKT\) \[Page 14\]](#). When the database system is started, it is initialized using permanently stored, internal configuration information. This configuration information is written to the log area at regular intervals, in particular at a [savepoint \[Page 137\]](#).

- A [log queue \[Page 47\]](#) is assigned to the log writer.
- The log writer writes the [log pages \[Page 48\]](#) that are full, or have to be written as a result of a [COMMIT \[Page 131\]](#) or [ROLLBACK \[Page 137\]](#), from the log queue to the [log area \[Page 135\]](#). The log pages are numbered (log sequence number), so that it is possible to check that all log pages were written, and to ensure the correct working sequence in the case of a [restart \[Page 136\]](#) or recovery. The log writer then notifies the transactions that were waiting for their [redo log entries \[Page 46\]](#) to be written.
- Log pages of the log queue that were not full when a write operation was performed remain in the log queue and continue to be filled, and are written to the log area in a subsequent write operation. The log writer is configured so that it always writes one and the same log page to the same physical place.
- The log writer regularly checks the state of the log area.
If the log area is full, the log writer locks the log queue so that all transactions that want to enter [redo log entries \[Page 46\]](#) into the log queue are stopped.
When the [automatic log backup \[Page 43\]](#) is active, the log writer ensures that the redo log entries from the log area are backed up automatically.
When a certain number of log entries have been written, the administrative information

is copied to the log area, and savepoints are requested, if required. In the case of a restart, this reduces the restart time.



Log Area

The volume storage area needed for [redo log management \[Page 47\]](#) is called the log area. A log area can extend across several [log volumes \[Page 22\]](#).

The log area is managed by the [log writer \[Page 48\]](#). The log writer fills the log area with [log pages \[Page 48\]](#) from the [log queue \[Page 47\]](#).



For security, the log area should always be mirrored. If possible, you should mirror the log area on a hardware basis.

If hardware-based mirroring is not possible, you can use an appropriate [log mode \[Page 38\]](#) setting.

In a production database system, make sure that redo log management is always activated, and that the log area cannot be overwritten until it has been backed up ([Log Settings \[Page 37\]](#)).

A full log causes the database to go down. Therefore, you must always carry out [log backups \[Page 43\]](#) in good time. For this purpose, make sure you activate [automatic log backup \[Page 43\]](#).

In log backups, the log area is not saved as a whole. Instead, it is saved in backup units, called log segments. The size of a log segment is defined by the parameter [LOG_SEGMENT_SIZE \[Page 82\]](#).



Undo Log Management

During [online logging \[Page 46\]](#), the [undo log entries \[Page 46\]](#) are managed. The [modifying transactions \[Page 139\]](#) store the undo log entries in [undo log files \[Page 49\]](#).

Use

Undo log management plays a part in the following actions:

- The database can create a transaction-consistent database state that reflects the state of the database system at the time of the last [savepoint \[Page 137\]](#). The information in the undo log files is sufficient for this.
- In a [restart or recovery \[Page 51\]](#), the required undo log entries are determined.
- For the database instance type [liveCache \[Page 23\]](#), [History Management \[Page 50\]](#) is carried out for the undo log entries.



Undo Log File

During [undo log management \[Page 49\]](#), every [modifying transactions \[Page 139\]](#) creates its own undo log file, in which the [undo log entries \[Page 46\]](#) are written. Every undo log entry is assigned an undo log sequence number starting with 0.

Undo log files are internal database storage structures, which are stored in the [data area \[Page 131\]](#).

Storage of the undo log files in the data area means that the information in the data area can be used to create a transaction-consistent database state that reflects the state of the database system at the time of the last [savepoint \[Page 137\]](#).

Delete Undo Log Files

- [SAP DB OLTP \[Page 23\]](#): At the end of a transaction, the transaction itself deletes the undo log files.
- [liveCache \[Page 23\]](#): At the end of the transaction, the undo log files are transferred to [history management \[Page 50\]](#), because they are needed for unlocked, consistent reading. The undo log files are deleted at a later point in time by the [garbage collector \[Page 51\]](#).

In exceptional circumstances, storage of the undo log files in the data area can lead to a full data area. In this case, increase the size of the data area.



History Management

History management, as a task during [online logging \[Page 46\]](#), is currently only used for the database instance [liveCache \[Page 23\]](#).

During history management, all information that is needed by the [garbage collectors \[Page 51\]](#) is made available, so that the deleted objects can be completely removed, that is to say, so that the [undo log files \[Page 49\]](#) can be deleted. In particular, information from the [history files \[Page 50\]](#) is written to and managed in the [history list \[Page 50\]](#).



History File

History files are used during [history management \[Page 50\]](#). A history file is an internal database file, which contains the access and statistical information about the [undo log files \[Page 49\]](#) of [transactions \[Page 139\]](#) that have ended.

When a database instance is created, the initial history files are stored in the [data area \[Page 131\]](#). The number of history files is defined by the parameter [MAXUSERTASKS \[Page 84\]](#). Once a transaction has ended, it writes an entry to its history file. This entry identifies the undo log file that is assigned to the transaction. With each [savepoint \[Page 137\]](#), the history file is saved to the data area.

The [history list \[Page 50\]](#) is used to assign the history files to the [garbage collectors \[Page 51\]](#).



History List

The main storage area required for [history management \[Page 50\]](#) is the history list.

The history list is an internal database list of all [history files \[Page 50\]](#). The history list is used to assign a suitable history file to the [garbage collectors \[Page 51\]](#) for processing.

The history list is written to the [data area \[Page 131\]](#) on each [savepoint \[Page 137\]](#). In the case of a [restart \[Page 136\]](#), the history list is rebuilt on the basis of the data stored in the data area.



Garbage Collector

The garbage collectors are active components during [history management \[Page 50\]](#).

Garbage collectors are part of a [user kernel thread \(UKT\) \[Page 14\]](#). When the database system is started, they are initialized using permanently stored, internal configuration information.

During history management, the [history list \[Page 50\]](#) is used to determine the [history file \[Page 50\]](#) that is to be processed by the garbage collector. In the [undo log files \[Page 49\]](#) that were determined using the history files, the system searches for delete operations for objects. If it finds a delete operation, the garbage collector deletes the object.



Restart or Recovery

The [log concept \[Page 45\]](#) for the SAP DB database system includes the import of the required [log entries \[Page 45\]](#), if the database instance is [restarted \[Page 136\]](#) or recovered.

The [redo log manager \[Page 51\]](#) ensures that the required log entries are made available and processed in the correct, time-based sequence.

- For each [transaction \[Page 139\]](#), the available [redo log entries \[Page 46\]](#) are stored by the [log reader \[Page 52\]](#) in a [redo log file \[Page 52\]](#).
- Transactions for which a redo is actually required are stored by the log reader in a [redo list \[Page 52\]](#).
- The [redo tasks \[Page 53\]](#) evaluate the redo list, and import the required redo and [undo log entries \[Page 46\]](#) for each transaction they find.

See also:

[Example: Restart \[Page 54\]](#)



Redo Log Manager

The redo log manager is the component that manages all the resources required for actions during a [restart or recovery \[Page 51\]](#).

Actions

- The redo log manager starts the [log reader \[Page 52\]](#) and several [redo tasks \[Page 53\]](#).
- The redo log manager can cancel any of the tasks it starts. At the end of an action, the redo log manager collects all error messages, and may then terminate the [restart \[Page 136\]](#) or recovery, if required.
- The redo log manager generates the [redo list \[Page 52\]](#)



Log Reader

The log reader is an active component during a [restart or recovery \[Page 51\]](#). The log reader is a [server task \[Page 15\]](#).

Actions

From the data that was stored at the time of the last [savepoint \[Page 137\]](#), the log reader determines the [transaction file \[Page 139\]](#), and uses this transaction file to re-generate the [transaction list \[Page 139\]](#).

The log reader determines the required information from [redo log management \[Page 47\]](#) or from the backup manager, so that all the [redo log entries \[Page 46\]](#) written since the last savepoint can be made available.

- If [transactions \[Page 139\]](#) in the transaction list are marked as ended at the time of the savepoint, the log reader enters these transactions in the [redo list \[Page 52\]](#) immediately.
- The log reader creates a [redo log file \[Page 52\]](#) for each further transaction found in the redo log entries.
- **COMMIT transactions**
If a [COMMIT \[Page 131\]](#) is found for the whole transaction, the log reader enters the transaction in the redo list.
- **ROLLBACK transactions**
If a [ROLLBACK \[Page 137\]](#) is found for the whole transaction, the log reader enters the transaction in the redo list.
- **ROLLBACK in subtransactions**
If a ROLLBACK is found within a transaction, the log reader removes the corresponding redo log entries from the redo log file. This means that these log entries are not processed unnecessarily.
- **Uncompleted transactions**
If uncompleted transactions are found after all redo log entries are read, the log reader flags these transactions with a ROLLBACK and enters them in the redo list. These transactions are then treated as ROLLBACK transactions.

For information on the further processing of the transactions, see [Redo Tasks \[Page 53\]](#).

See also:

[Example: Restart \[Page 54\]](#)



Redo Log File

When a [restart or recovery \[Page 51\]](#) is carried out, a redo log file is created for each [transaction \[Page 139\]](#) found in the [redo log entries \[Page 46\]](#). Redo log files are internal database storage structures, which are stored in the [data area \[Page 131\]](#).

The [log reader \[Page 52\]](#) copies exactly those redo log entries that belong to a transaction to each redo log file.



Redo List

The redo list is an internal database list of [transactions \[Page 139\]](#), which is in the main memory. The redo list is created by the [redo log manager \[Page 51\]](#).

If, during a [restart or recovery \[Page 51\]](#), the [log reader \[Page 52\]](#) finds a [COMMIT \[Page 131\]](#) in a [redo log file \[Page 52\]](#) for the whole transaction, this transaction is entered in the redo list.

The [redo tasks \[Page 53\]](#) use the redo list to determine the transactions that need to be repeated, and the sequence in which they are to be executed. The sequence for executing the transactions, defined in the redo list, is important to ensure that the same data is not changed simultaneously.



Redo Task

Redo tasks are active components during a [restart or recovery \[Page 51\]](#). Redo tasks are [server tasks \[Page 15\]](#).

Actions

COMMIT transactions

1. In the [redo list \[Page 52\]](#), the redo tasks look for [transactions \[Page 139\]](#) that were completed with a [COMMIT \[Page 131\]](#), and are therefore ready to be re-imported. In the redo list, the transactions are sorted by the log sequence numbers of their COMMIT.
2. The redo task starts by processing the transaction with the lowest COMMIT log sequence number. The [redo log files \[Page 52\]](#) that belong to this transaction are processed in sequence. If a [redo log entry \[Page 46\]](#) of this redo log file has a higher log sequence number than the lowest COMMIT log sequence number, it cannot be processed. In this case, the redo task waits for the COMMIT of the other transaction. Only then can the redo task continue processing the redo log file. This mechanism ensures consistent data.
3. Once a COMMIT transaction has been processed in full, the redo task deletes the redo log file and the [undo log file \[Page 49\]](#) of the transaction. The transaction is removed from the redo list.

ROLLBACK transactions

1. In the redo list, the redo tasks look for transactions that were completed with a [ROLLBACK \[Page 137\]](#).
2. The redo tasks delete the relevant redo log files. This prevents the transaction being redone unnecessarily.
3. If the ROLLBACK transaction started before the savepoint, the redo task evaluates the undo log file of the transaction, in order to set the transaction to the state before the savepoint. Once the ROLLBACK transaction has been fully processed, the redo task deletes the undo log file of the transaction.

If the ROLLBACK transaction started after the [savepoint \[Page 137\]](#), the redo task does not process the transaction further.

See also:

[Example: Restart \[Page 54\]](#)

Savepoint on Restart

[Savepoints \[Page 137\]](#) are also written during a [restart \[Page 136\]](#) (redo of a redo).

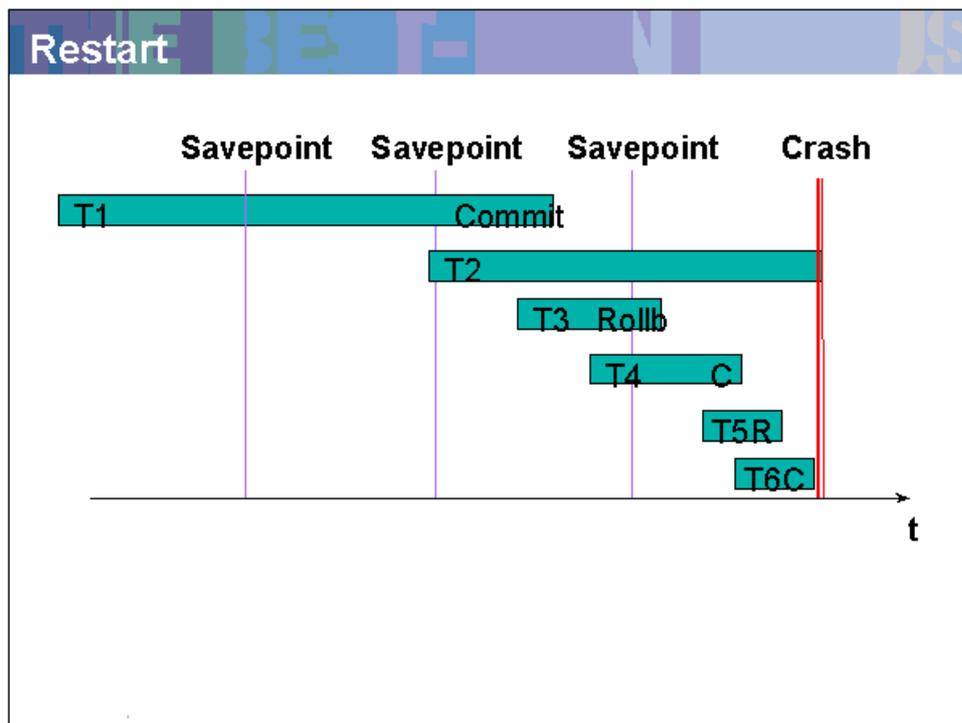
In this case, the positions of the [log reader \[Page 52\]](#) and the state of all open redo transactions are saved to the [data area \[Page 131\]](#), and reused for the next restart.

In this way, a terminated restart can be started again.

Example: Restart

The database crashed.

For the subsequent [restart \[Page 136\]](#), the database instance is recovered starting from the time of the last [savepoint \[Page 137\]](#).



No recovery measures are required for transactions T1 and T5. The changes for transaction T1 were recorded in full by the last savepoint. The changes for T5 were rolled back before the crash, and are not yet stored in the data area.

For transactions T4 and T6, the [log reader \[Page 52\]](#) creates [redo log files \[Page 52\]](#) and relevant entries in the [redo list \[Page 52\]](#). The [redo tasks \[Page 53\]](#) import the required redo log entries for these transactions.

No recovery measures are required for transactions T2 and T3. However, the modifications made before the savepoint must be reversed. The redo tasks therefore evaluate the relevant [undo log files \[Page 49\]](#).



Database Tools

The SAP DB database system offers a series of tools for working with the database instances.

- [Architecture of the SAP DB Tools \[Page 55\]](#)
- The following SAP DB tools are described in more detail:
 - [Database Manager \[Page 61\]](#)
 - [SAP DB Loader \[Page 65\]](#)
 - [Query Tools \[Page 66\]](#)



Architecture of the SAP DB Tools

The SAP DB tools [Database Manager \[Page 61\]](#) and [SAP DB Loader \[Page 65\]](#) each consist of a server part and a client part. The server part is responsible for the functions, and the user uses the client to access the tool.

The SAP DB [query tools \[Page 66\]](#) use ODBC to access the [database instance \[Page 13\]](#).

Client/Server for the SAP DB Tools

| | Client | Server |
|------------------|---|---|
| Database Manager | DBMGUI, DBMCLI Web DBM Script interface (for example to Perl or Python) available | DBM Server [Page 60] |
| SAP DB Loader | LOADERCLI Script interface (for example to Perl or Python) available | Loader Server [Page 60] . |
| Query Tools | SQL Studio Web SQL Studio | Database instance |

Prerequisites

| Client | Windows | UNIX |
|------------------------------|--|---|
| DBMCLI LOADERCLI | The X Server [Page 60] must be active as a service on the database server. | The X server must be running as a background process on the database server. |
| DBMGUI | The X server must be active as a service on the database server. | The X server must be running as a background process on the database server. The Database Manager GUI is installed on a Windows server. The database instance on the UNIX server is administered remotely. |
| Web DBM Web SQL Studio | The Web services must be installed together with the Web Server [Page 61] on one computer. | The Web services must be installed together with the Web Server on one computer. |

Architecture

- [Architecture of the Database Manager \[Page 56\]](#)
- [Architecture of the SAP DB Loader \[Page 57\]](#)

- [Architecture of the Query Tools \[Page 58\]](#)
- [Architecture of the SAP DB Web Tools \[Page 59\]](#)



Architecture of the Database Manager

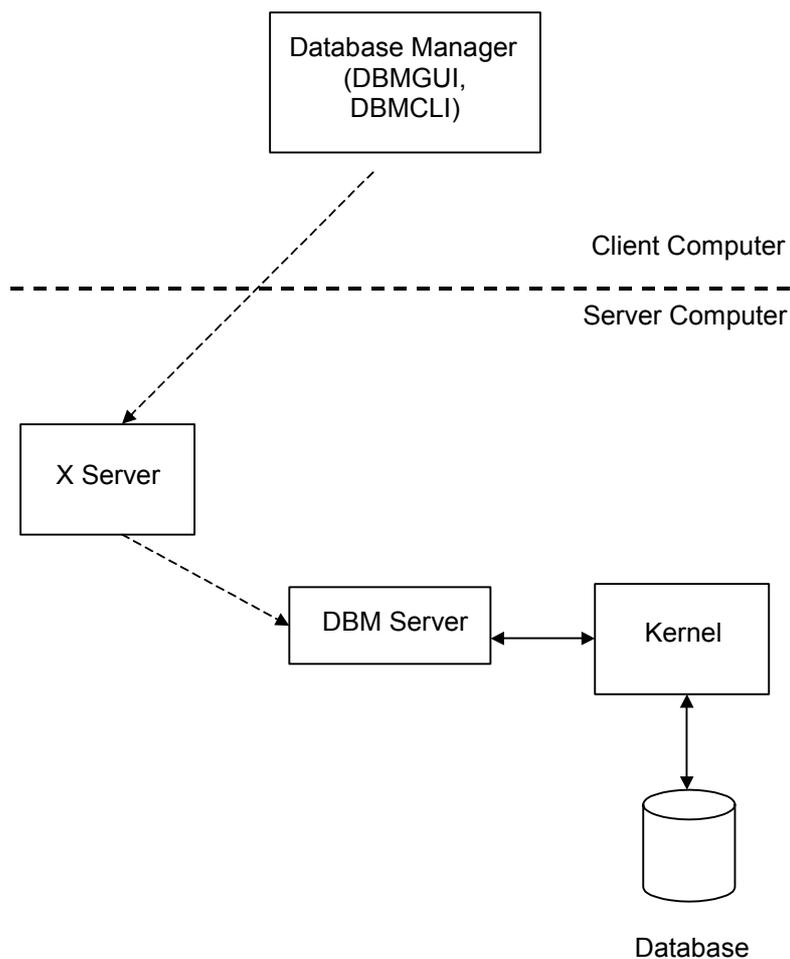
The [Database Manager \[Page 61\]](#) has a client/server architecture.

- Clients: Database Manager GUI, Database Manager CLI, WEB DBM, clients through Perl, Python, and JAVA interfaces
- Server: [DBM Server \[Page 60\]](#)

The client of the Database Manager (for example, Database Manager GUI) is remote-enabled, which means that the client and DBM server can be installed on different computers. The DBM server must always be installed on the computer that the corresponding database instance is installed on.



The Database Manager client is on one computer, and the DBM server and the database instance are on another computer.



Explanation

At the request of the client (Database Manager GUI or Database Manager CLI), the [X server \[Page 60\]](#) starts the DBM server. Once a connection has been successfully established, an additional X server instance is started, which is needed for transporting the data packages across the network.

See also:

[Architecture of the SAP DB Web Tools \[Page 59\]](#)

[Architecture of the SAP DB Tools \[Page 55\]](#)



Architecture of the SAP DB Loader

The [SAP DB Loader \[Page 65\]](#) has a client/server architecture.

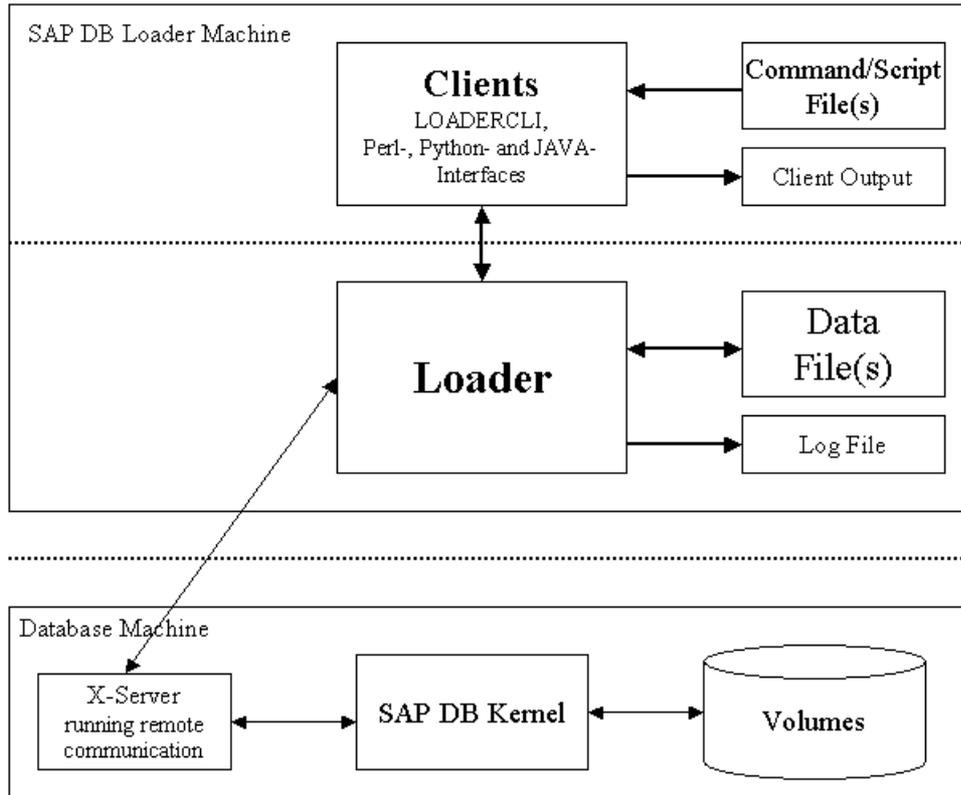
- Clients: SAP DB Loader CLI (LOADERCLI), clients through Perl, Python and JAVA interfaces
- Server: [Loader \[Page 60\]](#).

The SAP DB Loader is also remote-enabled, which means that the SAP DB Loader and database instance can be installed on different computers.

The client of the SAP DB Loader (for example, SAP DB Loader CLI) is remote-enabled, which means that the client and Loader can be installed on different computers. It is important that the [media \[See SAP DB Library\]](#) (for example files) of the SAP DB Loader must always be on the computer that the Loader is installed on.



In practice, the following configuration is frequently used:
All SAP DB Loader components are on one computer, and the database instance is on another computer.



Explanation

1. When requested by the client (for example, SAP DB Loader CLI), the Loader is started directly, **without** the [X server \[Page 60\]](#) being called.
2. Communication between the Loader and the database kernel is established via the X server.

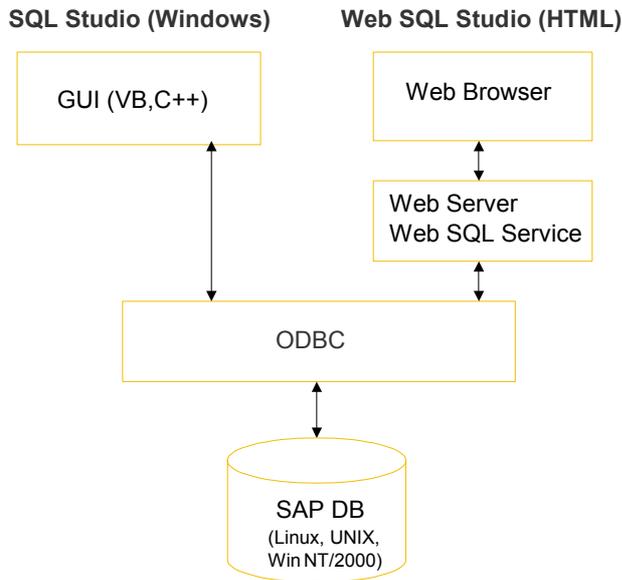
See also:

[Architecture of the SAP DB Tools \[Page 55\]](#)



Architecture of the Query Tools

- [SQL Studio \[Page 67\]](#) (GUI for Microsoft Windows): When requested by the client, a connection is set up to the [database instance \[Page 13\]](#) through the ODBC interface. After this, communication can take place between the client and the database instance.
- [Web SQL Studio \[Page 67\]](#): Calling the Web SQL Studio in the Web browser sets up a connection to the client (Web SQL service integrated into the [Web server \[Page 61\]](#)). The communication between this client and the database instance takes place through the ODBC interface (see also [Architecture of the SAP DB Web Tools \[Page 59\]](#)).

**See also:**

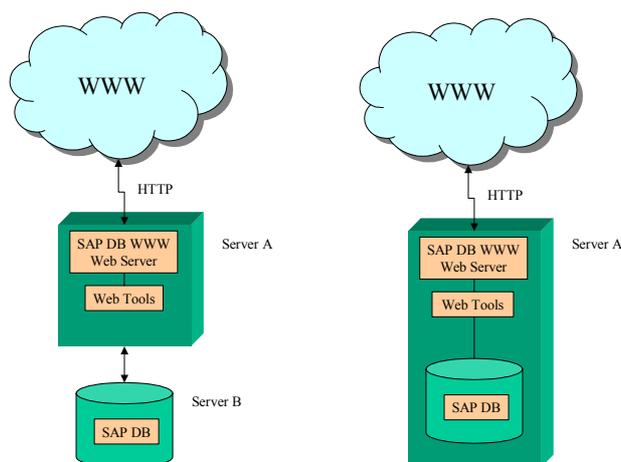
[Architecture of the SAP DB Tools \[Page 55\]](#)



Architecture of the SAP DB Web Tools

The SAP DB Web tools are implemented as a Web service and can be used with the SAP DB Web Server as well as with the well-known Apache [Web server \[Page 61\]](#).

The Web services are installed together with the Web server on one computer. The database system can be on a different computer or on the same computer as the Web server.



A connection to the client (Web DBM or Web SQL service integrated into the Web Server) is created by calling the SAP DB Web tool ([Web DBM \[Page 64\]](#) or [Web SQL Studio \[Page 67\]](#)) in the Web Browser. The communication between the client and the [database instance \[Page](#)

13] takes place through the [DBM Server \[Page 60\]](#) for the Web DBM tool, and through the ODBC interface for the Web SQL Studio tool ([Architecture of the Query Tools \[Page 58\]](#)).

See also:

[Architecture of the SAP DB Tools \[Page 55\]](#)

X Server

The X server (Remote SQL Server) is the communication instance of the SAP DB software when the individual components are located on different computers.

- Database applications and servers of the SAP DB tools (for example, Loader) address the X server when they want to establish a connection with a database instance installed on another computer.
- Clients of the SAP DB tools (for example, DBMCLI, LOADERCLI) address the X server when they want to establish a connection with a SAP DB tool server (for example, DBM Server, Loader) installed on a different computer.

Each time a connection is successfully established, an additional X server instance is started (Windows: Thread, UNIX: Process). This X server instance is needed to transport the data packages across the network. When the connection ends, the X server instance is closed.

See also:

[Architecture of the Database Manager \[Page 56\]](#)

[Architecture of the SAP DB Loader \[Page 57\]](#)

DBM Server

The Database Manager Server (DBM Server) is the server part of the [Database Manager \[Page 61\]](#). It is installed by the server installation on the database server.

The DBM Server creates the connection to the [database instance \[Page 13\]](#) and can access its environment using operating system resources.

Client applications, such as the [Database Manager GUI \[Page 61\]](#), the [Database Manager CLI \[Page 62\]](#), or the Web DBM program, which is integrated into the [Web Server \[Page 61\]](#), create a connection to the DBM Server and exchange data with the DBM Server using a Request-Response mechanism.

See also:

[Architecture of the Database Manager \[Page 56\]](#)

Loader Server

The [SAP DB Loader \[Page 65\]](#) has its own loader server (loader) The Loader must be installed on the computer that the [media \[See SAP DB Library\]](#) are located on.

The Loader creates the connection to the [database instance \[Page 13\]](#) and can access its environment using operating system resources.

The Loader can communicate remotely with the database instance and with the client.

See also:

[Architecture of the SAP DB Loader \[Page 57\]](#)

[SAP DB Loader: SAP DB 7.4 \[See SAP DB Library\]](#)

Web Server

The Web Server is the client part of the SAP DB Web tools [Web DBM \[Page 64\]](#) and [Web SQL Studio \[Page 67\]](#). The server for the Web DBM tool is the [DBM Server \[Page 60\]](#), the server for the Web SQL Studio tool is the [database instance \[Page 13\]](#).

Possible Web servers are the SAP DB Web Server or the well-known Web server Apache. The SAP DB Web server is installed during the installation of the SAP DB Web tools. Apache must be installed separately or configured. For more information, see the [Installation Guide Web Tools: SAP DB 7.4 \[See SAP DB Library\]](#).

Web DBM and Web SQL Studio are operating system-independent. These tools are installed once, and can then be called from any browser. This enables both occasional and frequent users to make use of the Database Manager and SQL Studio functions quickly and easily in the network.

See also:

[Architecture of the SAP DB Web Tools \[Page 59\]](#)

Database Manager

The **Database Manager** is a [database tool \[Page 55\]](#) for managing SAP DB database.

The tasks of the Database Manager comprise creating, controlling, monitoring, backing up, and occasionally recovering [database instances \[Page 13\]](#) on the local computer or on remote computers.

Architecture

The Database Manager consists of a server part and a client part. You have the same functions in the Database Manager regardless of which client you use.

Server/Client for the Database Manager

| Server | Client |
|--------------------------------------|---|
| DBM Server [Page 60] | Database Manager GUI [Page 61] Database Manager CLI [Page 62] Web DBM [Page 64] A script interface exists. |

See also:

[Architecture of the Database Manager \[Page 56\]](#)

[Architecture of the SAP DB Web Tools \[Page 59\]](#)

Database Manager GUI

The [Database Manager \[Page 61\]](#) has a user-friendly graphical user interface, the Database Manager GUI (DBMGUI). If you want to use the Database Manager to monitor several SAP DB database instances, which may be on different computers, you should use the Database Manager GUI.

The Database Manager GUI can only be used on hosts with Windows operating systems. If you want to use the functionality of the Database Manager on other operating system platforms, you must use the [Database Manager CLI \[Page 62\]](#) or [Web DBM \[Page 64\]](#).

Call

You have the following options for calling the Database Manager GUI:

- Choose *Start* → *Programs* → *SAP DB* → *Database Manager*.
- You can start the Database Manager GUI from the command line. In this case, you can specify [options \[Page 62\]](#) for the Database Manager program.

See also:

Database Manager GUI: SAP DB 7.4, section [Calling the Database Manager GUI \[See SAP DB Library\]](#)



Options (DBMGUI)

When you call the [Database Manager GUI \[Page 61\]](#) at command line level, you can specify options (*Database Manager GUI: SAP DB 7.4* → [Starting the DBMGUI \[See SAP DB Library\]](#)). If you do not enter any options, you can make the entries needed for logging on to the database instance on the logon screen.



You want to specify the following [user data as options \[Page 31\]](#): the [DBM operator \[Page 26\]](#) and the name of the [database instance \[Page 13\]](#)

```
dbmgui -u dbmmann,secret -d MK1
```

The Database Manager is called and a database session is created for the DBM operator **dbmmann**, password **secret** with the registered database instance **MK1**.



Database Manager CLI

The [Database Manager \[Page 61\]](#) has a command-line oriented client, the Database Manager CLI (DBMCLI). The Database Manager CLI is operating-system-independent.

You can use the Database Manager CLI to perform all Database Manager actions. You can also use the Database Manager CLI to schedule these actions in the background. You should use this option to automate Database Manager actions that have to be performed regularly.

If you do not have a host with a Windows operating system, you can only use the clients Database Manager CLI and [Web DBM \[Page 64\]](#) to manage database instances.

Call

```
dbmcli [<options>] [<command>]
```

You can transfer [options \[Page 63\]](#) and a maximum of one [DBM command \[Page 64\]](#) in a command line to the Database Manager CLI.

For a link to be established with a database instance on the local computer, you must enter at least the name of the database instance (option `-d <database_name>`) when calling the Database Manager CLI. If the required database instance is on a remote server, you must also enter this server name (option `-n <database_server>`).

You can open an interactive Database Manager CLI session if you do not enter any DBM commands other than the required options. You can then enter the required DBM commands interactively.

You can write the required DBM commands to a separate file `<file_name>`. In this case, when you call the Database Manager CLI, enter option `-i <file_name>` in addition to the required options.

See also:

Database Manager CLI: SAP DB 7.4, section [Functions of the Database Manager CLI \[See SAP DB Library\]](#)

Options (DBMCLI)

A series of options can be specified for the [Database Manager CLI \[Page 62\]](#) (*Database Manager CLI: SAP DB 7.4*, Section [Options \[See SAP DB Library\]](#)).

To connect to the database instance on the local server, you must enter at least option `-d <database_name>`. If you do not enter a [DBM command \[Page 64\]](#) in the command line when calling the Database Manager, an interactive Database Manager CLI session is opened.



You want to specify the following [user data as options \[Page 31\]](#): [DBM operator \[Page 26\]](#) and the name of the [database instance \[Page 13\]](#) on the local server:

```
dbmcli -u dbmmann,secret -d MK1
```

The Database Manager is called and a database session is created for the DBM operator `dbmmann`, password `secret` with the registered database instance `MK1`. You can then enter the required DBM commands interactively.



You want to specify the name of the database instance on the local server as an option. You want to start the database instance.

```
dbmcli -d MK1
user_logon dbmmann,secret
db_online
exit
```

The Database Manager is called and an interactive database session is created with registered database instance `MK1`. Details about the DBM operator `dbmmann`, password `secret` are made in the interactive DBM command `user_logon`. Database instance `MK1` is started with DBM command `db_online`.



You want to specify the name of the database instance on the local server as an option. You want to start the database instance. You want this action to be performed in the background.

Create file `startMK1` with the following contents:

```
user_logon dbmmann,secret
db_online
exit
```

Execute the command in the background

```
dbmcli -d MK1 -i startMK1
```

The Database Manager is called and an interactive database session is created with registered database instance **MK1**. File `startMK1` is processed.

DBM Commands

Commands can be transferred to the [Database Manager CLI \[Page 62\]](#) (*Database Manager CLI: SAP DB 7.4*, Section [DBM Commands \[See SAP DB Library\]](#)).

Syntax

`<command_name> [<parameters>]`

A DBM command is always made up of the command name and optional parameters affecting its execution.



```
dbmcli -u dbmmann,secret -d MK1 param_getfull cache_size
```

The Database Manager is called and a database session is created for the DBM operator `dbmmann`, password `secret` with the registered database instance **MK1**. The DBM command `param_getfull` is used to request all data for database parameter [CACHE_SIZE \[Page 79\]](#).

Web DBM

The [Database Manager \[Page 61\]](#) has a web-based client, the Web DBM. Install the Web DBM once in the network. The Web DBM can then be called from all browsers.

If you do not have a Windows operating system, you can only use the clients [Database Manager CLI \[Page 62\]](#) and Web DBM to monitor database instances.

Web DBM provides you with basically the same functionality as the [Database Manager GUI \[Page 61\]](#). Operation depends on the Web-based application.

See also:

[Architecture of the SAP DB Web Tools \[Page 59\]](#)

Differences between Web DBM and Database Manager GUI

With the Database Manager GUI, you can monitor several database instances simultaneously. With the Web DBM, you can only monitor one database instance at a time. When you log on, you enter the database instance that you want to monitor. If you want to switch to another database instance, you must log off, and log on again entering the new database instance.

It is possible to open several browsers simultaneously. In this way, you can monitor a different database instance in each browser with the Web DBM.

In contrast to the Database Manager GUI, a timeout may occur when working with the Web DBM. This results in the connection to the database instance being broken. If this happens, you must log on again.

Structure of the Web DBM

All administrative functions are listed in the window of the Web DBM. The functions are divided up into the same groups as in the Database Manager GUI. The colored symbols (green dots and red squares) indicate whether an administrative function can be used in the

database instance in its present state. There is no menu bar or toolbar of the kind found in the Database Manager GUI.

Instead of a list of database instances registered in the Database Manager GUI, the Web DBM displays the detailed status of the database instance. The most important values displayed in the status have a link that takes you to the relevant function. The functions for starting and stopping the database instance are at the bottom of the status display.

The work area is located below the status display. The subsequent web pages corresponding to the selected administrative function are displayed here.

The structure of these web page is similar to the corresponding screens in the Database Manager GUI. The bottom of these pages often contains a toolbar with the available administrative functions. If a new administrative function is selected in the Web DBM, and in a new page displayed in the work area as a result of this, the previous display and status is lost. To return to the previous display, you must reselect the relevant administrative function.

The header line contains a logoff link that enables you to end the current connection with a database instance.



SAP DB Loader

The SAP DB Loader is a [database tool \[Page 55\]](#) for loading and unloading data. In addition to processing [Loader commands \[Page 66\]](#), the SAP DB Loader can also execute all [SQL statements \[See SAP DB Library\]](#) (see [SAP DB Loader: SAP DB 7.4 \[See SAP DB Library\]](#)).

Architecture

The SAP DB Loader consists of a server part and a client part.

| Server | Client |
|----------------------------------|---|
| Loader [Page 60] | SAP DB Loader CLI A script interface (for example to Perl and Python) is available |

If you want to react to SAP DB Loader return codes, you must use the script interface.

See also: [Architecture of the SAP DB Loader \[Page 57\]](#)

Call with SAP DB Loader CLI

```
loadercli [<options>] -b <command_file>
```

When you [call the SAP DB Loader with the CLI \[See SAP DB Library\]](#), you can specify [options \[Page 66\]](#), commands, and SQL statements. The commands and SQL statements must be stored in a [command file \[See SAP DB Library\]](#) (<command_file>), which you specify using option `-b <command_file>`.

For a link to be established with a database instance on the local computer, you must enter at least the name of the database instance (option `-d <database_name>`) when calling the Loader.

When creating the connection between the Loader and the [database instance \[Page 13\]](#), the Loader first uses the specified options. The commands and SQL statements of the command file are then processed in the specified sequence.

See also: [Calling with Loader Perl Script \[See SAP DB Library\]](#), [Calling with Loader Python Script \[See SAP DB Library\]](#)

Options (LOADERCLI)

When you call the [SAP DB Loader \[Page 65\]](#) with the SAP DB Loader CLI (LOADERCLI), you can specify a number of options for the SAP DB Loader. For a list of all possible options, see *SAP DB Loader: SAP DB 7.4*, Section [Options \[See SAP DB Library\]](#).

To connect to the database instance on the local server, you must enter at least option `-d <database_name>`. In addition, you must specify the name of a [command file \[See SAP DB Library\]](#) in option `-b <command_file>`.



You want to specify the following [user data as options \[Page 31\]](#): [database user \[Page 28\]](#) and the name of the [database instance \[Page 13\]](#)

```
loadercli -u samplename,secret -d TST -b command.dat
```

The Loader is called and a database session is created for the database user `samplename`, password `secret`, with the database instance `TST`. All further commands statements can be found in command file `command.dat`.

Loader Commands

A series of commands and [SQL statements \[See SAP DB Library\]](#) can be transferred to the [SAP DB Loader \[Page 65\]](#). For a list of all possible commands, see *SAP DB Loader: SAP DB 7.4*, Section [Commands \[See SAP DB Library\]](#).

If you use the SAP DB Loader CLI (LOADERCLI) to call the Loader, these must be transferred in a [command file \[See SAP DB Library\]](#) `<command_file>` ([Option -b \[See SAP DB Library\]](#)).

Query Tools

The SAP DB query tools are [database tools \[Page 55\]](#) that enable easy access to [application data \[Page 128\]](#) and the [database catalog \[Page 131\]](#) of an SAP DB database instance.

You can use the query tools to create, execute, and manage any number of SQL statements. The SQL Studio for Microsoft Windows also gives you the option of editing data records in a screen, and of creating database queries with visual support.

Architecture

The client part of the query tools creates a connection to the [database instance \[Page 13\]](#) through the ODBC interface.

| Server | Client |
|-------------------|--|
| Database instance | SQL Studio [Page 67] Web SQL Studio [Page 67] |

See also:

[Architecture of the Query Tools \[Page 58\]](#)

[Architecture of the SAP DB Web Tools \[Page 59\]](#)



SQL Studio

The SQL Studio is one of the [query tools \[Page 66\]](#). The SQL Studio has an easy-to-use graphical user interface.

The SQL Studio as described here can only be used on Microsoft Windows operating systems. You can use the [Web SQL Studio \[Page 67\]](#) for other operating system platforms.

Prerequisites

Check if the database instance is started.

Call

You have the following options for calling the graphical user interface of the SQL Studio:

- Choose *Start* → *Programs* → *SAP DB* → *SQL Studio*. Log on to a database instance.
- You can start SQL Studio from the command line. In this case, you can specify [options \[Page 67\]](#).

See also:

SQL Studio: SAP DB 7.4, section [Starting SQL Studio \[See SAP DB Library\]](#)



Options (SQL Studio)

You can start [SQL Studio \[Page 67\]](#) from the command line. To do this, execute the program SQLSTO, with options, if necessary (SQL Studio: SAP DB 7.4 → [Starting SQL Studio \[See SAP DB Library\]](#)).

Procedure

1. Change to the directory in which the program `sqlsto.exe` is stored.
2. Enter the following command:
`sqlsto [<options>]`



You want to specify the following [user data as options \[Page 31\]](#): [database operator \[Page 28\]](#) and the name of the [database instance \[Page 13\]](#) on the local server:

```
sqlsto -u samplename,secret -d MK1
```

SQL Studio is called and a database session is created for the database user `samplename`, password `secret`, with the database instance `MK1`.



Web SQL Studio

The Web SQL Studio is one of the [query tools \[Page 66\]](#). The Web SQL Studio is a Web-based client that enables easy access to [application data \[Page 128\]](#) and the [database catalog \[Page 131\]](#) of an SAP DB database instance. You can use the Web SQL Studio to create, execute, and manage SQL statements.

Install the Web SQL Studio once in the network. It can then be called from all browsers.

Microsoft Windows operating systems can also use the query tool [SQL Studio \[Page 67\]](#) with additional functions.

See also:

[Architecture of the SAP DB Web Tools \[Page 59\]](#)

Prerequisites

Check whether the database instance has been started.

Call

Enter the following address in the browser:

`http://<web_server>:<port>/websql`

Enter the name of the database computer, the name of the database instance, the user name, and the user password.

See also [Web SQL Studio: SAP DB 7.4 \[See SAP DB Library\]](#)



Directory Structure of the Database for SAP Systems

How the directories required for the SAP DB database system are distributed to the available hard disks has a significant impact on the security and performance of your database system. A good distribution of files on the hard disk must meet the following requirements:

- There must be enough free space to allow the database to expand
- The data must be stored securely
- The hardware must meet the performance requirements

The optimum design of the directory structure depends upon the specified hardware configurations. There is no single solution or definition for the distribution of the files of an SAP DB [database instance \[Page 13\]](#).

When attempting to find the optimum data distribution for your database environment in a production system, bear in mind the information in the following sections:

[Availability \[Page 40\]](#)

[Performance Requirements \[Page 73\]](#)

[Example Configuration \[Page 73\]](#)

[Various Database Systems \[Page 73\]](#)

[SAP DB Directories \[Page 68\]](#)



SAP DB Directories

The following table contains the distribution of the SAP DB directories on the hard disk for a [database instance](#).

SAP DB Directories

| Directory Name (Variables) | Description |
|--|-------------------------------------|
| <independent_data_path> | Directory that, among other things, |

| | |
|---|--|
| | contains the following subdirectories with instance data : <ul style="list-style-type: none"> • Configuration directory • Run directory |
| <independent_data_path>/config | Configuration directory that, among other things, contains configuration files [Page 77] for database parameters, database tools, and files for user authorization |
| <independent_data_path>/wrk/<database_name> | Run directory of the database instance <database_name> that, among other things, contains the log files [Page 75] |
| <independent_program_path> | Directory with the following content: <ul style="list-style-type: none"> • Programs Independent of the Database Software Version • Libraries for the Client Run-Time Environment |
| <dependent_path> | Directory that contains the programs that are dependent on the database software version |
| /sapdb/<database_name>/data | Directory for data volumes |
| /sapdb/<database_name>/log | Directory for volumes <ul style="list-style-type: none"> • Log volumes • Mirrored log volumes |
| /sapdb/<database_name>/save | Directory for backup files <ul style="list-style-type: none"> • Log backups • Data backups |

See also:

[Example: SAP DB Directory Structure](#)

[Display SAP DB Directories](#)

[Define SAP DB Directories](#)



Instance Data

Instance data of the [database instance \[Page 13\]](#) in the following [SAP DB directories \[Page 68\]](#) in the directory <independent_data_path>:

- [Run directories \[Page 137\]](#): <independent_data_path>/wrk/<database_name>
- Configuration directory: <independent_data_path>/config

[Variables \[Page 140\]](#)



(UNIX)

/sapdb/data/wrk/LVC (Run directory for the LVC instance)

/sapdb/data/wrk/SDB (Run directory for the SDB instance)

/sapdb/data/wrk/P01 (Run directory for the P01 instance)
/sapdb/data/config (Configuration directory for all instances)



Programs that Are Independent of the Database Software Version

Programs that are independent of the database software version are only installed once for each computer, since they are needed for the database services that exist for each computer.

[SAP DB Directories \[See SAP DB Library\]](#)

[Variables \[Page 140\]](#)

The programs that are independent of the database software version are stored in the **IndepProgPath** directory: <independent_program_path>.

In this directory and its subdirectories, you will always find the programs for the most recently installed version of the database software.



/sapdb/programs/bin/x_server
(version-independent program X server)

/sapdb/programs/pgm/dbmcli
(version-independent program Database Manager CLI)



Libraries for the Client Run-time Environment

The libraries required for the client run-time environment are installed once on each computer, but must be available in different versions.

[SAP DB Directories \[Page 68\]](#)

[Variables \[Page 140\]](#)

The libraries are stored in a subdirectory of the **IndepProgPath** directory:
<independent_program_path>/runtime/<version>



/sapdb/programs/runtime/7240/lib



Programs that Are Dependent on the Database Software Version

The programs that are dependent on the database software version are installed once per [database instance \[Page 13\]](#). This means that the database software version of a database instance is independent of the database software versions of other database instances running on the same machine.

[SAP DB Directories \[Page 68\]](#)

[Variables \[Page 140\]](#)

The programs that are dependent on the database software version are situated in the directory <dependent_path>.



- (UNIX)
- /sapdb/LVC/db (InstRoot directory for the *liveCache* instance LVC)
- /sapdb/SDB/db (InstRoot directory for the Content Server instance SDB)
- /sapdb/P01/db (InstRoot directory for the database instance P01)

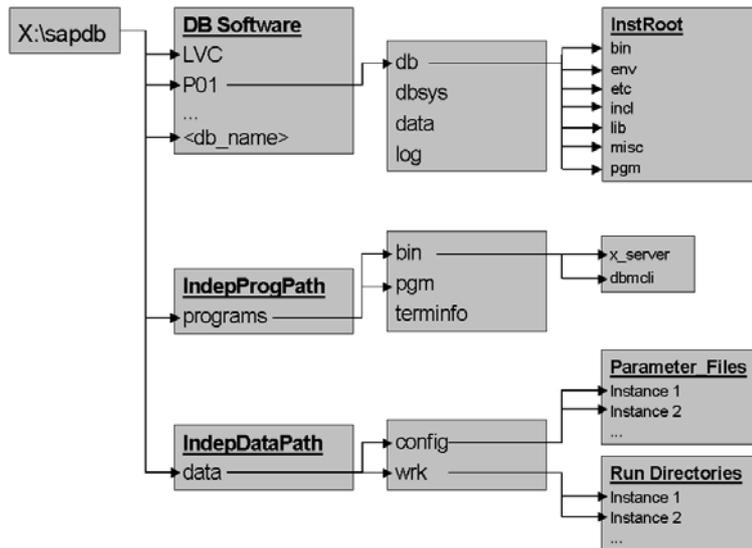
Client Tools

The [database tools \[Page 55\]](#) supported by SAP DB can be installed onto the database server or onto a computer of your choice.

The GUI clients are only installed once per computer. The settings for each user can be stored user-specifically.

You can use a [SAP DB directory \[Page 68\]](#) of your choice for the installation of the database tools.

Example: SAP DB Directory Structure



Example for the structure of the [SAP DB directory \[Page 68\]](#)

Displaying SAP DB Directories

You can use the following Database Manager CLI commands to display the paths for the [SAP DB directories \[Page 68\]](#):

| Directory Name (Variables [See SAP DB Library]) | Database Manager CLI Command |
|---|------------------------------|
|---|------------------------------|

| | |
|----------------------------|---|
| <independent_data_path> | <code>dbm_getpath indepdatapath</code> |
| <independent_program_path> | <code>dbm_getpath indepprogrpath</code> |
| <dependent_path> | <code>db_enum</code> |



Define SAP DB Directories

You can use the following Database Manager CLI commands to define the paths for the [SAP DB directories \[Page 68\]](#):

| Directory Name (Variables [Page 140]) | Database Manager CLI Command |
|---|--|
| <independent_data_path> | <code>dbm_setpath indepdatapath <independent_data_path></code> |
| <independent_program_path> | <code>dbm_setpath indepprogrpath <independent_program_path></code> |
| <dependent_path> | <code>inst_reg <dependent_path></code> |



Directory Structure of the Database System for Open Source

How the directories required for the SAP DB database system are distributed to the available hard disks has a significant impact on the security and performance of your database system. A good distribution of files on the hard disk must meet the following requirements:

- There must be enough free space to allow the database to expand
- The data must be stored securely
- The hardware must meet the performance requirements

The optimum design of the directory structure depends upon the specified hardware configurations. There is no single solution or definition for the distribution of the files of an SAP DB [database instance \[Page 13\]](#).

When attempting to find the optimum data distribution for your database environment in a production system, bear in mind the information in the following sections:

[Availability \[Page 40\]](#)

[Performance Requirements \[Page 73\]](#)

[Example Configuration \[Page 73\]](#)

[Various Database Systems \[Page 73\]](#)

[SAP DB Directories \[Page 74\]](#)

Performance Requirements

For performance reasons, each of the different types of [volume \[Page 21\]](#) should be stored on a different disk. Therefore, create [data volumes \[Page 21\]](#) and [log volumes \[Page 22\]](#) on different disks. Because all changes to the database instance are logged in the [log areas \[Page 135\]](#), it is the log volumes for a [database instance \[Page 13\]](#) that see the most write activity.

Even when you use RAID-5 systems, configure the database instance with multiple data volumes. Performance will be better with many data volumes than with a single one, because some parallel mechanisms used by the database system depend on the number of configured data volumes.

For performance reasons, the log volumes must not be created on RAID-5 systems but only on dedicated disks or RAID-1 systems.

If swap or paging areas and log entries are kept on the same disk, performance will be negatively affected.

UNIX: Raw devices should be used for the data volumes and log volumes, because accessing to data in raw devices is generally quicker than accessing data in files.

Example Configuration



The [data volumes \[Page 21\]](#) and [log volumes \[Page 22\]](#) are on different disks. The log volumes are mirrored.

Using Multiple Database Systems

For performance reasons, the SAP DB database system should not be installed on the same computer along with other database systems.

IF you want to install multiple SAP DB [database instances \[Page 13\]](#), then install each database instance on a separate computer. This allows you to deal with any performance problems more effectively.



SAP DB Directories

The following table contains the distribution of the data on a hard disk for an [SAP DB database instance \[Page 13\]](#).

SAP DB Directories

| Directory Name (Variables [Page 140]) | Description |
|---|--|
| <independent_data_path> | Directory that, among other things, contains the following subdirectories: <ul style="list-style-type: none"> • Configuration directory • Run directories [Page 137] |
| <independent_data_path>/config | Configuration directory that, among other things, contains configuration files [Page 77] for database parameters, database tools, and files for user authorization |
| <independent_data_path>/wrk/<database_name> | Run directory of the database instance <database_name> that, among other things, contains the log files [Page 75] |
| <independent_program_path> | Programs that are independent of the database software version are only installed once for each computer, since they are needed for the database services that exist once for each computer. The libraries required for the client run-time environment are installed once on each computer, but must be available in different versions. |
| <dependent_path> | Directory that contains the server software that is dependent on the database version |

The client tools supported by SAP DB can be installed on the database host or on a computer of your choice.

The GUI clients are only installed once per computer. The settings for each user can be stored user-specifically.

The directory for the client tools is freely-definable during the installation.

See also:

[Display SAP DB Directories \[Page 75\]](#)

[Define SAP DB Directories \[Page 75\]](#)



Displaying SAP DB Directories

You can use the following Database Manager CLI commands to display the paths for the [SAP DB directories \[Page 74\]](#):

| Directory Name (Variables [Page 140]) | Database Manager CLI Command |
|---|---|
| <independent_data_path> | <code>dbm_getpath indepdatapath</code> |
| <independent_program_path> | <code>dbm_getpath indepprogrpath</code> |
| <dependent_path> | <code>db_enum</code> |



Define SAP DB Directories

You can use the following Database Manager CLI commands to define the paths for the [SAP DB directories \[Page 74\]](#):

| Directory Name (Variables [Page 140]) | Database Manager CLI Command |
|---|--|
| <independent_data_path> | <code>dbm_setpath indepdatapath <independent_data_path></code> |
| <independent_program_path> | <code>dbm_setpath indepprogrpath <independent_program_path></code> |
| <dependent_path> | <code>inst_reg <dependent_path></code> |



Database Files

The system creates a range of files for each [database instance \[Page 13\]](#). These files are stored in various locations in the directory structure, depending on their file types.

There are two types of file:

- [Log files \[Page 75\]](#)
- [Configuration files \[Page 77\]](#)

See also:

[Directory Structure of the Database System for SAP Systems \[Page 68\]](#)

[Directory Structure of the Database System for Open Source Systems \[Page 72\]](#)



Log Files

For each [database instance \[Page 13\]](#), the system creates a range of log files, depending on the actions that have been performed. These files are then stored in the [run directory \[Page 137\]](#) of the database instance and its subdirectories. If the run directory is not defined, the files are stored in the directory <independent_data_path>\wrk.

Log Files of the Database Instance

| File Name (File ID, Class [Page 77]) | File Content |
|---|--|
| knldiag (KNLDIAG, protocol) | Current kernel log; this log is created when the database instance is started with the size specified in KERNELDIAGSIZE [Page 81] . Messages about starts in the operational state [Page 136] ADMIN are not overwritten cyclically. Messages from the ONLINE state are overwritten cyclically. |
| knldiag.old (KNLDIAGOLD, protocol) | Last kernel log Each time the database instance is started, the file knldiag is renamed as knldiag.old. |
| knldiag.err (KNLDIAGERR, protocol) | Kernel error messages This file is not overwritten. New error messages are added to the end of the file. |
| knldiag.evt (KNLEVT, protocol) | Log of database events |
| knltrace (KNLTRC, dump) | Log file for the database trace [Page 133] This file can also contain information on the causes of crashes. This file is initialized each time the database instance is started. |
| <database_name>.prt (KNLTRCPRT, protocol) | Test version of database trace |
| knldump (KNLDUMP, dump) | Backup of the main memory information after a database crash |
| rtedump (RTEDUMP, dump) | Memory dump from runtime environment of database instance |
| <database_name>.pah (DBMPAHI, protocol) | History of database parameters |

Log Files of Database Tools

| | |
|----------------------|--|
| dbm.* dbahist.prt | Database Manager Log Files [See SAP DB Library] |
| loader.prt | Log file [See SAP DB Library] of SAP DB Loader (always saved to the directory <independent_data_path>\wrk) |

Communication Log Files

| | |
|-------------|---|
| xserver.prt | X-Server [Page 60] log file |
|-------------|---|

Installation Log File

| | |
|--|---|
| SAPDB<profile>_install-<date>-<time>.log | Log file [See SAP DB Library] of the standard installation of the database software (program SDBINST) or of the update of existing database software (program SDBUPD) |
|--|---|

liveCache Log Files (for SAP Systems Only)

| | |
|------------|--|
| lcinit.log | Start or initialization of liveCache [Page 23] with CCMS functions from the SAP system |
| lcinit.his | History of all liveCache actions Information from the log lcinit.log is added to this file. |
| lc<token> | Log of liveCache activities |

**Classes of Log Files**

[Log files \[Page 75\]](#) can have the following classes:

| | |
|----------|--|
| protocol | General log files |
| backup | Log files for backups and restores |
| lvc | Special log files for the database instance type liveCache |
| dump | Memory dumps |

**Configuration Files**

The system creates separate configuration files for each [database instance \[Page 13\]](#).

- Configuration file with [database parameters \[Page 77\]](#)
The database parameter configuration file <database_name> is located in the configuration directory: <independent_data_path>\config\<database_name>. The history of the database instance parameters is part of the [log files \[Page 75\]](#).
- Configuration files of the database tools
[Configuration Files of the Database Manager \[See SAP DB Library\]](#)

See also:

[SAP DB Directories \[Page 74\]](#)

**Database Parameters**

To initialize the database parameters, use is generally made of the default configuration which was stored when the database software was installed.

The configuration generated by the system will always be runnable. If necessary, you can still adjust the database parameters originally set to suit any specific requirements you may have.

Alternatively you can use the configuration of a [database instance \[Page 13\]](#) already present on the computer or the configuration from a [data backup \[Page 42\]](#). Even after this you can still adjust the database parameters to suit your own requirements.

The database parameters are divided into [general database parameters \[Page 78\]](#), [special database parameters \[Page 78\]](#) and [support database parameters \[Page 79\]](#).

Display or change database parameters

You can use the [Database Manager \[Page 61\]](#) to display or change database parameters.

See:

Database Manager GUI: SAP DB 7.4, Section [Displaying and Changing Current Database Parameters \[See SAP DB Library\]](#)

Database Manager CLI: SAP DB 7.4, Section [Configuring Database Instances \[See SAP DB Library\]](#)



General Database Parameters

[Database parameters \[Page 77\]](#) that determine the general database features

| | | |
|--|--|---|
| CACHE_SIZE [Page 79] | INSTANCE_TYPE [Page 80] | KERNELVERSION [Page 82] |
| LOG_SEGMENT_SIZE [Page 82] | MAXBACKUPDEVS [Page 83] | MAXCPU [Page 83] |
| MAXDATAVOLUMES [Page 83] | MAXLOCKS [Page 83] | MAXLOGVOLUMES [Page 84] |
| MAXUSERTASKS [Page 84] | RESTART_SHUTDOWN [Page 85] | RUNDIRECTORY [Page 85] |



Special Database Parameters (Extended)

[Database parameters \[Page 77\]](#) that determine the special database features

| | | |
|--|--|---|
| BACKUP_BLOCK_CNT [Page 79] | CAT_CACHE_SUPPLY [Page 79] | DATE_TIME_FORMAT [Page 79] |
| DEADLOCK_DETECTION [Page 79] | DEFAULT_CODE [Page 79] | DEVNO_BIT_COUNT [Page 80] |
| JOIN_MAXTAB_LEVEL4 [Page 81] | JOIN_MAXTAB_LEVEL4 [Page 80] | JOIN_SEARCH_LEVEL [Page 81] |
| KERNELDIAGSIZE [Page 81] | LOG_BACKUP_TO_PIPE [Page 82] | LOG_IO_QUEUE [Page 82] |
| LRU_FOR_SCAN [Page 82] | MAXRGN_REQUEST [Page 84] | MAXSERVERTASKS [Page 84] |
| MP_RGN_LOOP [Page 84] | OPTIM_MAX_MERGE [Page 84] | REQUEST_TIMEOUT [Page 85] |
| SEQUENCE_CACHE [Page 85] | SESSION_TIMEOUT [Page 86] | UTILITY_PROT_SIZE [Page 86] |
| _DATA_CACHE_RGNS [Page 86] | _EVENT_ALIVE_CYCLE [Page 86] | _MAXEVENTS [Page 86] |
| _MAX_MESSAGE_FILES [Page 86] | _ROW_RGNS [Page 87] | _TAB_RGNS [Page 87] |
| _TRANS_RGNS [Page 87] | _TREE_RGNS [Page 87] | _UNICODE [Page 87] |



Support Database Parameters

There are certain [database parameters \[Page 77\]](#) that should not, or do not have to be modified for normal database operation.

The setting of these database parameters requires a detailed knowledge of the database system, and should therefore only be carried out by the [SAP DB Support \[Page 144\]](#) team.



BACKUP_BLOCK_CNT

The [special database parameter \[Page 78\]](#) BACKUP_BLOCK_CNT denotes the block size in pages when data is backed up or restored.



CACHE_SIZE

The [general database parameter \[Page 78\]](#) CACHE_SIZE denotes the size of the [I/O buffer cache \[Page 19\]](#) in [pages \[Page 136\]](#).



CAT_CACHE_SUPPLY

The [special database parameter \[Page 78\]](#) CAT_CACHE_SUPPLY denotes the memory size of the [catalog cache \[Page 19\]](#) in pages for all [user tasks \[Page 15\]](#).



DATE_TIME_FORMAT

The [special database parameter \[Page 78\]](#) DATE_TIME_FORMAT denotes the default format for displaying dates and times.



DEADLOCK_DETECTION

The [special database parameter \[Page 78\]](#) DEADLOCK_DETECTION denotes the maximum search level for deadlock detection.

Deadlocks that have not been detected by the deadlock detection up to the given search level of the specified value are only resolved by the [REQUEST_TIMEOUT \[Page 85\]](#).

If `DEADLOCK_DETECTION = 0`, deadlock detection is disabled.



DEFAULT_CODE

If no other code attribute was specified by appropriate SQL statements, the [special database parameter \[Page 78\]](#) DEFAULT_CODE denotes which [code attribute \[See SAP DB Library\]](#) was used to create columns of data type [CHAR\[ACTER\] \[See SAP DB Library\]](#), [VARCHAR \[See SAP DB Library\]](#) and [LONG\[VARCHAR\] \[See SAP DB Library\]](#) in the database instance.



DEVNO_BIT_COUNT

The [special database parameter \[Page 78\]](#) DEVNO_BIT_COUNT denotes the number of bits in the converter block address that is reserved for the logical device number of a data volume.

Default value: 8



Size of a page: 8 KB

DEVNO_BIT_COUNT is set to 8

SAP DB can therefore manage 256 data volumes. Each data volume can have a maximum size of 128 GB.



The value that was set for DEVNO_BIT_COUNT when the database instance was installed should not be subsequently changed. If you change this database parameter during database operation, you must then recover the database instance to update the numbering of the data volumes.

Possible values for DEVNO_BIT_COUNT: $6 \leq \text{DEVNO_BIT_COUNT} \leq 12$

The higher the value of the database parameter DEVNO_BIT_COUNT, the more data volumes can be managed. However, a high number of data volumes means the capacity of the individual data volumes is reduced.



INSTANCE_TYPE

The [general database parameter \[Page 78\]](#) INSTANCE_TYPE denotes the [database instance type \[Page 22\]](#).

- OLTP: [SAP DB OLTP \[Page 23\]](#)
- LVC: [liveCache \[Page 23\]](#)
- BW: [SAP DB OLAP \[Page 23\]](#)
- CS: [SAP DB Document Server \[Page 23\]](#)
- EMERGE: [SAP DB E-Catalog \[Page 24\]](#)



JOIN_MAXTAB_LEVEL9

The [special database parameter \[Page 78\]](#) JOIN_MAXTAB_LEVEL9 denotes the maximum number of tables allowed in a join when selecting the join sequence algorithm.

The default value is 4.

See also:

[JOIN_SEARCH_LEVEL \[Page 81\]](#)



JOIN_MAXTAB_LEVEL4

The [special database parameter \[Page 78\]](#) JOIN_MAXTAB_LEVEL4 denotes the maximum number of tables allowed in a join when selecting the join sequence algorithm.

The default value is 16.

See also:

[JOIN_SEARCH_LEVEL \[Page 81\]](#)



JOIN_SEARCH_LEVEL

The [special database parameter \[Page 78\]](#) JOIN_SEARCH_LEVEL determines the algorithm for the join sequence search. The level specified here determines how many resources and how much time the join sequence search takes.

- 9 (Join sequence search level 9): All possible join sequences are calculated.
- 4 (join sequence level 4): Various join sequences are calculated, depending on the query structure (transformer algorithm).
- 1 (join sequence search level 1): The simplest algorithm is used for the join sequence search (greedy algorithm).
- 0 (classified join sequence search level 0, level 0 is the default setting): The algorithm that is used for the join sequence search depends on the number of tables that were selected in join.

Joins with n number of tables, where $n \leq \text{JOIN_MAXTAB_LEVEL9 [Page 80]}$: Level 9 of the join sequence search is used.

Joins with n number of tables, where

$\text{JOIN_MAXTAB_LEVEL9} < n \leq \text{JOIN_MAXTAB_LEVEL4 [Page 81]}$: Level 4 of the join sequence search is used.

Joins with n number of tables, where $\text{JOIN_MAXTAB_LEVEL4} < n$: Level 1 of the join sequence search is used.



JOIN_MAXTAB_LEVEL4 is set to 16.

JOIN_MAXTAB_LEVEL9 is set to 4.

JOIN_SEARCH_LEVEL is set to 0.

5 tables are used for a join.

Join sequence search level 4 (transformer algorithm) is used for this join

because the database parameter

$\text{JOIN_MAXTAB_LEVEL9} < 5 \leq \text{JOIN_MAXTAB_LEVEL4}$ is most appropriate.



KERNELDIAGSIZE

The [special database parameter \[Page 78\]](#) KERNELDIAGSIZE denotes the size of log file of the [kernel \[Page 134\]](#) in KB.



KERNELVERSION

The [general database parameter \[Page 78\]](#) KERNELVERSION denotes which database software version is being used.



LOG_BACKUP_TO_PIPE

The [special database parameter \[Page 78\]](#) LOG_BACKUP_TO_PIPE determines whether or not a [log backup \[Page 43\]](#) is permitted (YES) or not permitted (NO) in pipes.



LOG_IO_QUEUE

The [special database parameter \[Page 78\]](#) LOG_IO_QUEUE denotes the size of the [log queue \[Page 47\]](#) in [pages \[Page 48\]](#).



LOG_SEGMENT_SIZE

The [general database parameter \[Page 78\]](#) LOG_SEGMENT_SIZE denotes the size of a log segment in the [log area \[Page 135\]](#) in pages.

For [interactive backups \[Page 44\]](#), the log segment size determines the size of the segments that the [redo log entries \[Page 46\]](#) are saved to. It also determines the intervals at which log areas are [backed up automatically \[Page 43\]](#).

The size of the log segment depends on the size of the individual [log volumes \[Page 22\]](#) and must not be greater than the sum of all log volumes.

| User-defined value | Log segment sizes used by the system |
|---|---|
| No definition of log segment size (LOG_SEGMENT_SIZE = 0) | A third of the total log area. |
| The log segment size in pages is smaller than or the same as 50% of the total log area. | The defined value for LOG_SEGMENT_SIZE. |
| The log segment > 50% of the total log area. | 50% of the log area. |



LRU_FOR_SCAN

The [special database parameter \[Page 78\]](#) LRU_FOR_SCAN governs where data pages that have been scanned into the [data cache \[Page 20\]](#) are placed in the LRU (last recently used) list.

- **no** (default setting): The data pages scanned into the data cache are placed at the end of the LRU list. This means the data pages will be removed from the data cache promptly if other pages are read into the data cache, and new entries are made in the LRU list.
- **yes**: The data pages read into the data cache are handled according to the LRU algorithm, which means that they are not removed immediately from the data cache when new pages are read.



MAXARCHIVELOGS

See [MAXLOGVOLUMES \[Page 84\]](#)



MAXBACKUPDEVS

The [general database parameter \[Page 78\]](#) MAXBACKUPDEVS denotes the maximum number of files or tape devices that can be used in parallel.

You can speed up the process of backing up and restoring [data volumes \[Page 21\]](#) by using more than one file or tape device in parallel.



MAXCPU

The [general database parameter \[Page 78\]](#) MAXCPU denotes the maximum number of CPUs allowed on the [database instance \[Page 13\]](#). MAXCPU defines over how many CPUs the [user tasks \[Page 15\]](#) generating the main load are distributed.

When defining MAXCPU, however, remember that operating system resources must also be available for other processes.

This means you can define and also restrict the number of CPUs the database instance uses for user tasks on multiprocessor computers ([Multiprocessor Configuration \[Page 25\]](#)).

If you are using a single processor computer, you should choose a value of 1 for the MAXCPU.



MAXDATADEVSPACES

See [MAXDATAVOLUMES \[Page 83\]](#)



MAXDATAVOLUMES

The [general database parameter \[Page 78\]](#) MAXDATAVOLUMES (depending on the version, may also be called MAXDATADEVSPACES) denotes the maximum number of [data volumes \[Page 21\]](#).



MAXLOCKS

The [general database parameter \[Page 78\]](#) MAXLOCKS determines the maximum number of entries in the lock list in which the [locks \[Page 109\]](#) (line or table locks) of all users and their lock requests are stored.

See also:

[Lock behavior \[Page 109\]](#)



MAXLOGVOLUMES

The [general database parameter \[Page 78\]](#) MAXDATAVOLUMES (depending on the version, may also be called MAXDATADEVSPACES) denotes the maximum number of [data volumes \[Page 22\]](#).



MAXRGN_REQUEST

The [special database parameter \[Page 78\]](#) MAXRGN_REQUEST denotes the maximum number of times a [task \[Page 138\]](#) can attempt to access a critical section. If this number is exceeded, the task lets another task access the CPU as long as it belongs to the same [user kernel thread \[Page 14\]](#).



MAXSERVERTASKS

The [special database parameter \[Page 78\]](#) MAXSERVERTASKS denotes the maximum number of [server tasks \[Page 15\]](#) that are allowed when processing tasks.



MAXUSERTASKS

The [general database parameter \[Page 78\]](#) MAXUSERTASKS denotes the maximum number of users that can work on a database at any one time ([user tasks \[Page 15\]](#), [database sessions \[Page 132\]](#)).

Overconfiguration exceeding the actual requirement leads to increased demands on address space, especially shared memory.



Once the number of configured database sessions is reached, no other users can connect to the database instance concerned. The number of active database sessions is therefore a critical parameter of database operation and must be monitored.



MP_RGN_LOOP

The [special database parameter \[Page 78\]](#) MP_RGN_LOOP denotes the maximum number of times a [task \[Page 138\]](#) may attempt to access a critical section that has been locked by another task. If this number is exceeded, the status of the task changes to “Waiting”.



OPTIM_MAX_MERGE

The [special database parameter \[Page 78\]](#) OPTIM_MAX_MERGE denotes how the optimizing algorithm for merging index lists changes.

If the number of pages of an index that need to be merged exceeds the value specified in `OPTIM_MAX_MERGE`, this index will not be used for an index merging strategy.



REQUEST_TIMEOUT

The [special database parameter \[Page 78\]](#) `REQUEST_TIMEOUT` denotes the maximum amount of time (in seconds) you should have to wait for a [lock \[Page 109\]](#) to be released.

This database parameter limits the amount of time you have to wait until a lock is lifted by other users for all [database sessions \[Page 132\]](#).

If a lock request cannot be satisfied within the time defined here, a message is returned to the waiting database session. Any changes made previously in the transactions are undone.



RESTART_SHUTDOWN

The [general database parameter \[Page 78\]](#) `RESTART_SHUTDOWN` determines whether [database instance \[Page 13\]](#) modes should be changed automatically or not.

The database parameter only takes effect if you are using a Windows operating system (NT, 2000, XP).

- **Manual:** All changes to the mode of the database instance must be initiated by the [DBA \[Page 29\]](#).
- **Auto:** The following changes are made to the database instance automatically: The Windows Service Manager switches the database instance straight to [operating mode \[Page 136\]](#) `ONLINE` when it starts. A database instance in `ONLINE` mode will be shut down automatically if the operating system shuts down.



Since the shutdown of an operating system is subject to a time limit, it may interrupt the automatic shutdown of a database instance, particularly if you are working with large database instances where a high volume of data is being changed.

If you restart the database after the shutdown was interrupted, you must import log entries to restore the database instance. We therefore recommend you set the database parameter `RESTART_SHUTDOWN` to `manual`.



RUNDIRECTORY

The [general database parameter \[Page 78\]](#) `RUNDIRECTORY` determines which directory is used as the [run directory \[Page 137\]](#) on the [database instance \[Page 13\]](#).



SEQUENCE_CACHE

The [special database parameter \[Page 78\]](#) `SEQUENCE_CACHE` determines how big the sequence cache is in pages.



SESSION_TIMEOUT

The [special database parameter \[Page 78\]](#) SESSION_TIMEOUT specifies the [timeout value \[Page 138\]](#), the time (in seconds) that a [database session \[Page 132\]](#) can remain inactive before being timed out.

If no SQL statement is issued within the specified time, the database system terminates the database session concerned (ROLLBACK WORK RELEASE statement).



UTILITY_PROT_SIZE

The [special database parameter \[Page 78\]](#) UTILITY_PROT_SIZE denotes the size of the log file (dbm.utl) that was written in the [run directory \[Page 137\]](#) of the [database instance \[Page 13\]](#).



_DATA_CACHE_RGNS

The [special database parameter \[Page 78\]](#) _DATA_CACHE_RGNS determines how many critical regions can be worked with simultaneously in the [data cache \[Page 20\]](#).



_EVENT_ALIVE_CYCLE

The [special database parameter \[Page 78\]](#) _EVENT_ALIVE_CYCLE denotes the number of seconds an event cycle takes.



_MAXEVENTS

The [special database parameter \[Page 78\]](#) _MAXEVENTS denotes the maximum number of events stored by the [kernel \[Page 134\]](#) in the cache for processing by the [Database Manager \[Page 61\]](#).



_MAX_MESSAGE_FILES

The [special database parameter \[Page 78\]](#) _MAX_MESSAGE_FILES denotes the maximum number of trace files that can be opened at one time.

 **_ROW_RGNS**

The [special database parameter \[Page 78\]](#) `_ROW_RGNS` denotes the number of critical regions in which you can check lock collisions in rows.

 **_TAB_RGNS**

The [special database parameter \[Page 78\]](#) `_TAB_RGNS` denotes the number of critical regions in which you can check lock collisions in tables.

 **_TRANS_RGNS**

The [special database parameter \[Page 78\]](#) `_TRANS_RGNS` denotes the number of critical regions in which you can check lock collisions in [transactions \[Page 139\]](#) simultaneously.

 **_TREE_RGNS**

The [special database parameter \[Page 78\]](#) `_TREE_RGNS` denotes the number of critical regions in which you can check lock collisions in [B* trees \[Page 98\]](#) simultaneously.

 **_UNICODE**

The [special database parameter \[Page 78\]](#) `_UNICODE` determines whether user data and metadata for database objects is saved in [UNICODE \[Page 88\]](#).

This database parameter cannot be changed once the SAP DB database system is installed.

See also:

[Installing a UNICODE-Enabled Database \[Page 88\]](#)

 **SAP DB as UNICODE Database**

SAP DB can be used as a UNICODE database.

- [UNICODE \[Page 88\]](#)
- [Installing a UNICODE-Enabled Database \[Page 88\]](#)
- [UNICODE and SQL \[Page 90\]](#)
- [UNICODE in Programming Languages \[Page 93\]](#)

 **UNICODE**

Data types such as CHAR ASCII and CHAR EBCDIC are mainly suited to English and central European languages. With other character sets, a code attribute is usually used for these data types. This code attribute uses a different presentation code to ASCII and EBCDIC, even for internal storage in the database system. This causes problems if you want to access these database systems using a different character set, or if you want to exchange data between database systems with different character sets.

You can avoid these problems by using internal character coding in accordance with UNICODE. Internally, the UNICODE data is stored in UTF-16/UCS-2 format. In UTF-16/UCS-2 format, all characters are two bytes long.

SAP DB is able to display various presentation codes in UNICODE format (UNICODE code in line with ISO 10646, page 1).

Metadata in UNICODE

The names of the database objects (such as table or column names) can be stored internally in UNICODE and can therefore then be displayed in the required presentation code in the database tools.

Application data in UNICODE

SAP DB supports the code attribute UNICODE for the data types **CHAR[ACTER]**, **VARCHAR** and **LONG[VARCHAR]**.

See also:

[Installing a UNICODE-Enabled Database \[Page 88\]](#)

Reference Manual: SAP DB 7.4, [Code attribute \[See SAP DB Library\]](#)

**Installing a UNICODE-Enabled Database**

To make storage of [metadata \[Page 131\]](#) and [application data \[Page 128\]](#) in [UNICODE \[Page 88\]](#), proceed as follows:

Metadata in UNICODE

When installing the database instance, set the database parameter [_UNICODE \[Page 87\]](#) to YES.

Application data in UNICODE

1. When installing the database instance, set the database parameter `_UNICODE` to YES.
2. Enter the [code attribute \[See SAP DB Library\]](#) UNICODE.



Please note that SAP DB UNICODE data is stored internally in UTF-16/UCS-2 format. As a result, double the space is required to store the UNICODE data in the database instance.

Procedure

[Setting Database Parameter `_UNICODE` \[Page 89\]](#)

[Setting Code Attribute UNICODE \[Page 89\]](#)



Setting the Database Parameter `_UNICODE`

In order to [install a UNICODE-enabled database \[Page 88\]](#), the database parameter `_UNICODE` [\[Page 87\]](#) must be set to `YES`.



Note that you cannot change the database parameter `_UNICODE` once it has been set.

You can use the Database Manager to set the database parameter `_UNICODE` when you install the database instance.

Database Manager GUI

1. Start the Database Wizard.
2. Perform the first five installation steps.
3. In step 6 (*Parameters*), choose *Extended*.
4. Set the `_UNICODE` parameter to `YES`.
5. Continue with the installation.

See also: *Database Manager GUI: SAP DB 7.4*, section [Creating or Initializing a New Database Instance \[See SAP DB Library\]](#)

Database Manager CLI

A script with configuration information for the database instance contains, among other things, lines for definition of the database parameters. You must insert the following line at this point:

```
param_put _UNICODE YES
```

See also: *Database Manager CLI: SAP DB 7.4*, [Changing the Value of a Database Parameter \[See SAP DB Library\]](#)



Setting Code Attribute `UNICODE`

To be able to store the [application data \[Page 128\]](#) in [UNICODE \[Page 88\]](#), the database must be UNICODE-enabled and you must set the [code attribute \[See SAP DB Library\]](#) `UNICODE` for the required application data ([Installing a UNICODE-Enabled Database \[Page 88\]](#)). You can set the code attribute in the following ways:

- Database parameter [DEFAULT_CODE \[Page 79\]](#) is **not** set to value `UNICODE`. In the column definition, make sure you enter **code attribute `UNICODE`**. Application data is then only stored in `UNICODE` for these column values. If you do not enter a code attribute, the code entered in the parameter `DEFAULT_CODE` (that is to say, not `UNICODE`) is used for these column values.
- Database parameter `DEFAULT_CODE` is set to value `UNICODE`. In the column definition, enter code attribute `UNICODE`. Application data is stored in `UNICODE` for these column values. If you do not enter a code attribute, the application data is stored in `UNICODE` for these column values, because the parameter `DEFAULT_CODE` is set to `UNICODE`.



Database parameter `DEFAULT_CODE` is set to value `UNICODE`.

| Column definition | Result |
|-------------------|--------|
|-------------------|--------|

| | |
|-------------------------|---|
| CHAR (n) UNICODE | UNICODE column |
| CHAR (n) | UNICODE column |
| CHAR (n) ASCII | ASCII column |
| CHAR (n) BYTE | Code neutral, i.e. the column values are not converted by the database system |

For information on setting database parameters, see the following documentation:

- *Database Manager GUI: SAP DB 7.4*, Section [Displaying and Changing Current Database Parameters \[See SAP DB Library\]](#)
- *Database Manager CLI: SAP DB 7.4*, Section [Changing the Value of a Database Parameter \[See SAP DB Library\]](#)



UNICODE and SQL

UNICODE can be used for metadata, application data and in SQL statements if [installation of a UNICODE-enabled database \[Page 88\]](#) has been carried out.

UNICODE for Metadata

If the database is UNICODE-enabled, all columns in the system tables that can be used to request the metadata have a data type with the [code attribute \[See SAP DB Library\]](#) UNICODE.

UNICODE for Application Data

To make application data UNICODE-enabled, you must [set the UNICODE code attribute \[Page 89\]](#) in a UNICODE-enabled database for the required application data. The [UNICODE \[Page 88\]](#) code attribute can be used for the data types CHAR[ACTER] (n), VARCHAR (n) and LONG[VARCHAR]:

- CHAR[ACTER] (n) UNICODE
- VARCHAR (n) UNICODE
- LONG[VARCHAR] UNICODE

[Example 1 \[Page 91\]](#) illustrates the definition of Java class `TableDef`. Java class `TableDef` can be used to display the results of various column definitions.

Displaying the Column Definition of a Table

```
java TableDef <jdbcurl> <table_name>
```

Creating a temporary table using the determined column definitions, and displaying these column definitions

```
<command> ::= java TableDef <jdbcurl> <table_name>  
<column_definition>
```

```
<jdbcurl> ::=  
jdbc:sapdb:<database_name>?user=<user_name>&password=<password>
```



```
java TableDef jdbc:sapdb:TST?user=TEST&password=TEST DUMMY  
a varchar (20)
```

```
TABLE: DUMMY
A: VARCHARASCII (20)
```

UNICODE in SQL Statements

SQL statements can contain both UNICODE literals and UNICODE identifiers. The prerequisite for implementing these SQL statements is a UNICODE-enabled client (C/C++-Precompiler, JDBC, ODBC, SQL Studio or Web SQL Studio).

The prerequisite for using UNICODE in the SQL Studio and Web SQL Studio is that a UNICODE-enabled ODBC has been installed. SQL Studio and Web SQL Studio are used on the operating system Windows 2000. This operating system supports UNICODE.



Example 1

The Java class `TableDef` can be used to display the results of various column definitions (see [UNICODE and SQL \[Page 90\]](#), section *UNICODE for application data*).

TableDef Definition

```
import java.sql.*;
/**
 *
 */
public class TableDef
{
    private Connection connection;
    /**
     * creates a new TableDef
     */
    public
    TableDef (
        String url)
        throws SQLException, ClassNotFoundException
    {
        // load class
        Class.forName ("com.sap.dbtech.jdbc.DriverSapDB");
        // create connection
        this.connection = DriverManager.getConnection(url,
            new java.util.Properties ());
        this.connection.setAutoCommit (false);
    }
    /**
     *
     */
    protected void
    createTable (
        String tableName,
        String createCommand)
        throws SQLException
    {
        String fullCommand = "CREATE TABLE " + tableName + " ("
            + createCommand + " )";
        Statement stmt = this.connection.createStatement ();
        stmt.execute (fullCommand);
    }
    /**
     *
     */
    protected void
    showTableDef (
        String tableName)
        throws SQLException
    {
        System.out.println ("Table: " + tableName); // #print
    }
}
```

```

        DatabaseMetaData metaData = this.connection.getMetaData ();
        ResultSet tableColumns = metaData.getColumns(null,
            metaData.getUserName (), tableName, null);
        while (tableColumns.next ()) {
            String columnName = tableColumns.getString
("COLUMN_NAME");
            String typeName = tableColumns.getString ("TYPE_NAME");
            int colSize = tableColumns.getInt ("COLUMN_SIZE");
            System.out.println (" " + columnName
                + ": " + typeName + " (" + colSize + ")"); // #print
        }
    }
    /**
     *
     */
    protected void
    close ()
    {
        try {
            this.connection.rollback ();
            this.connection.close ();
        }
        catch (SQLException sqlExc) {
            // ignore
        }
    }
    /**
     *
     */
    static protected String
    join (
        String [] args,
        int startIndex)
    {
        if (startIndex >= args.length) {
            return null;
        }
        StringBuffer result = new StringBuffer ();
        for (int i = startIndex; i < args.length; ++i) {
            result.append(args [i]);
            result.append (' ');
        }
        return result.toString();
    }
    /**
     * used when called form the command line
     */
    public static void main (String [] args)
    throws ClassNotFoundException
    {
        String url = args [0];
        String tableName = args [1];
        String createCommand = join (args, 2);
        TableDef tableDef = null;

        try {
            tableDef = new TableDef (url);
            if (createCommand != null) {
                tableDef.createTable (tableName, createCommand);
            }
            tableDef.showTableDef (tableName);
        }
        catch (SQLException sqlExc) {
            System.out.println (sqlExc);
        }
        finally {
            if (tableDef != null) {
                tableDef.close ();
            }
        }
    }

```

```

    }
}
}

```

UNICODE in Programming Languages

JDBC, ODBC, the C/C++ Precompiler and Python support [UNICODE \[Page 88\]](#).

JDBC

Since Java works with UNICODE strings, it can read and write UNICODE columns.

If you also want to use UNICODE in SQL statements, you must set the `unicode` CONNECT-property to `true`. SQL statements are then transferred to the database instance in UTF-16/UCS-2 format. If the transfer package for the SQL statements is not large enough, you can increase its size using database parameter `_PACKET_SIZE`.

ODBC

UNICODE is supported in the ODBC driver.

Depending on your operating system, you must take account of the following factors:

| Operating System | |
|------------------|--|
| Windows 2000 | The ODBC driver only exports the UNICODE and/or Wide functions of the ODBC-API. ANSI functions are mapped to the relevant Wide functions by the <i>Windows Driver Manager</i> . This means that applications can use both the ANSI and the UNICODE functions of the ODBC-API. |
| UNIX/Linux | The use of the ODBC driver is currently not possible on platforms for which the standard UNICODE type <code>WCHAR_T</code> is defined with four bytes. The database and ODBC driver process UNICODE internally as values that are two bytes long. Both the ANSI and UNICODE variants of the ODBC-API are defined in the driver. Applications that do not require the functionality of a <i>Driver Manager</i> can be statically linked with the ODBC driver. |

C/C++ Precompiler

During CONNECT, the C/C++ Precompiler checks whether the database is UNICODE-enabled ([Database parameter _UNICODE \[Page 89\]](#) = YES).

See also: [Example 2 \[Page 94\]](#) (`HelloUnicodeDB.cpc`) and [Working with UNICODE Data \[See SAP DB Library\]](#)

Python

Python 2.*: UNICODE columns can be read and written. UNICODE character strings cannot be used as SQL statements.

Python 1.5.2: UNICODE-type output values are converted to ASCII. If this conversion fails, an error is reported.



Example 2

HelloUnicodeDB.cpc is a Precompiler program that uses UNICODE host variables (see [UNICODE in Programming Languages \[Page 93\]](#), Section *C/C++ Precompiler*).

HelloUnicodeDB.cpc Definition

```

/*****
**

module      : HelloUnicodeDB.cpc

-----
--

responsible : MarcoP

special area: demonstration program precompiler
description : demonstration program for unicode features of SAPDB
precompiler

The following steps are needed to execute the demo program:
1. customize the connect data in the embedded SQL program source
(line 43 - 46)

2. precompile/compile the embedded SQL program HelloUnicodeDB.cpc
   cpc [-u <userid>,<password> -d <database_name> -n <server_node>]
HelloUnicodeDB

3. linking the embedded SQL program HelloUnicodeDB.cpc
   cpclnk HelloUnicodeDB

last changed: 2000-04-30  17:17
see also    :

-----
--

copyright:   Copyright by SAP AG, 2001

*****/

/*=====
*
* INCLUDES                                     *
*=====
*/
#include <stdio.h>
#include <string.h>

/*=====
*
* DEFINES                                     *
*=====
*/

#define KNLIIDNTFR 63      /* max. number of character for a unicode
kernel identifier*/

/* connect data */

```

```

#define USERID      "TEST"          /* username */
#define PASSWD      "TEST"          /* password */
#define SERVERNODE  "localhost"     /* computer name */
#define SERVERDB    "TST"           /* name of database*/

/*=====
*
*   MACROS                                                    *
*=====
*/

/*=====
*
*   LOCAL CLASSES, STRUCTURES, TYPES, UNIONS ...           *
*=====
*/

/*=====
*
*   STATIC/INLINE FUNCTIONS (PROTOTYPES)                   *
*=====
*/

/* print UCS2 string as 7-bit Ascii, replace non-ascii characters
with '?'*/
static void printAs7bitAscii(SQLUCS2 *aUCS2Str, int length)
{
    int i;
    for (i = 0; i < length; i++) {
        if ( aUCS2Str[i] == 0x0000
            || aUCS2Str[i] == 0x0020 )
            return;
        if ( aUCS2Str[i] < 0x007f
            && aUCS2Str[i] > 0x0000)
            printf("%c",aUCS2Str[i]);
        else
            printf("?");
    }
}

/* format sql error for output */
static char *FormatSQLError(sqlcatype *sqlca)
{
    static char buffer[512];
#ifdef sql_oracle

    sprintf(buffer, "(%d):%.s", sqlca->sqlcode,
            sqlca->sqlerrml, sqlca->sqlerrmc);
#else
    sprintf(buffer, "(%d):%.s", sqlca->sqlcode,
            sqlca->sqlerrm.sqlerrml, sqlca->sqlerrm.sqlerrmc);
#endif
    return(buffer);
}

/*=====
*
*   METHODS                                                    *
*=====
*/

int main(int argc, char **argv)

```

```

{
EXEC SQL BEGIN DECLARE SECTION;
char *user      = USERID;
char *pwd       = PASSWD;
char *servernode = SERVERNODE;
char *serverdb  = SERVERDB;
/* "SELECT TABLENAME FROM DOMAIN.TABLES " encoded in UCS2 */
SQLUCS2 sqlstmt[36] = {0x0053, 0x0045, 0x004C, 0x0045, 0x0043,
0x0054, 0x0020, 0x0054, 0x0041, 0x0042,
                        0x004C, 0x0045, 0x004E, 0x0041, 0x004D,
0x0045, 0x0020, 0x0046, 0x0052, 0x004F,
                        0x004D, 0x0020, 0x0044, 0x004F, 0x004D,
0x0041, 0x0049, 0x004E, 0x002E, 0x0054,
                        0x0041, 0x0042, 0x004C, 0x0045, 0x0053,
0x0000};
SQLUCS2 resultstring[64];
EXEC SQL END DECLARE SECTION;

/* set connect properties */
EXEC SQL SET SERVERDB :serverdb ON :servernode;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* connect to database */
EXEC SQL CONNECT :user IDENTIFIED BY :pwd;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* parse a unicode sql command and give it a statement name */
EXEC SQL PREPARE stmt1 FROM :sqlstmt;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* declare a cursor for a prepared statement name */
EXEC SQL DECLARE curs1 CURSOR FOR stmt1;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* open the cursor "curs1" */
EXEC SQL OPEN curs1;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* loop over resultset */
while (sqlca.sqlcode != 100)
{
    /* fetch a single row into a character hostvariable encoded in
UCS2 */
EXEC SQL FETCH curs1 INTO :resultstring;
if (sqlca.sqlcode != 0 && sqlca.sqlcode != 100) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}
else {
    printAs7bitAscii(resultstring, KNLIDNTFR);
    printf("\n");
}
}
}

```

```

}

/* close the cursor */
EXEC SQL CLOSE curs1;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}

/* commit and release the session */
EXEC SQL COMMIT WORK RELEASE;
if (sqlca.sqlcode != 0 ) {
    printf("\n%s\n", FormatSQLError(&sqlca));
}
}

/*=====
*
*   END OF CODE
*=====
*/

```



Data Management Using B* Trees

The SAP DB data management architecture ensures efficient data storage on disks and fast data access. The SAP DB database system performs automatic load balancing, which makes reorganizations unnecessary. The [data areas \[Page 131\]](#) can be extended online.

Structures and algorithms are available that are required for realizing an efficient I/O strategy, and that make reorganizations unnecessary. The data management algorithms have the following characteristics:

- Sort data on SELECT ([Table Access \(SELECT\) using B* Trees \[Page 103\]](#))
- Sort data on INSERT ([Table Access \(INSERT\) using B* Trees \[Page 105\]](#))
- UPDATE data in place ([Table Access \(UPDATE\) using B* Trees \[Page 107\]](#))
- DELETE data in place ([Table Access \(DELETE\) using B* Trees \[Page 106\]](#))

For this data management, the database system uses the following logical storage structures:

- Primary tables (with [primary keys \[Page 98\]](#) and [secondary keys \[Page 98\]](#))
- Secondary key tables (data of the secondary keys)
- [B* Tree \[Page 98\]](#)
There are build B* trees for the following tables ([B* Trees for Tables \[Page 101\]](#)):
primary tables and secondary key tables
For tables with LONG columns (data type LONG also known as binary large objects (BLOBs)) there is a special data processing using B* trees for tables with LONG columns.



Concepts

[Primary Key \[Page 98\]](#)

[Secondary Key \[Page 98\]](#)

[B* Tree \[Page 98\]](#)

[Table ID \[Page 100\]](#)



Primary Key

Each SAP DB table has a primary key. The primary key is either defined by the [database user \[Page 28\]](#) or generated internally. A user-defined primary key can consist of multiple columns.



Secondary Key

Secondary keys can be defined for each table to optimize the data access via SQL statements. They can refer to any column combination and they help to prevent sequential scans over the table. Like the [primary key \[Page 98\]](#), the secondary key can consist of multiple columns.

A secondary key is often called an index (not to be confused with B* index).



B* Tree

As far as is possible, all SAP DB data is stored in structures called B* trees. B* trees give much more efficient access to table rows than other access methods (such as sequential searches).

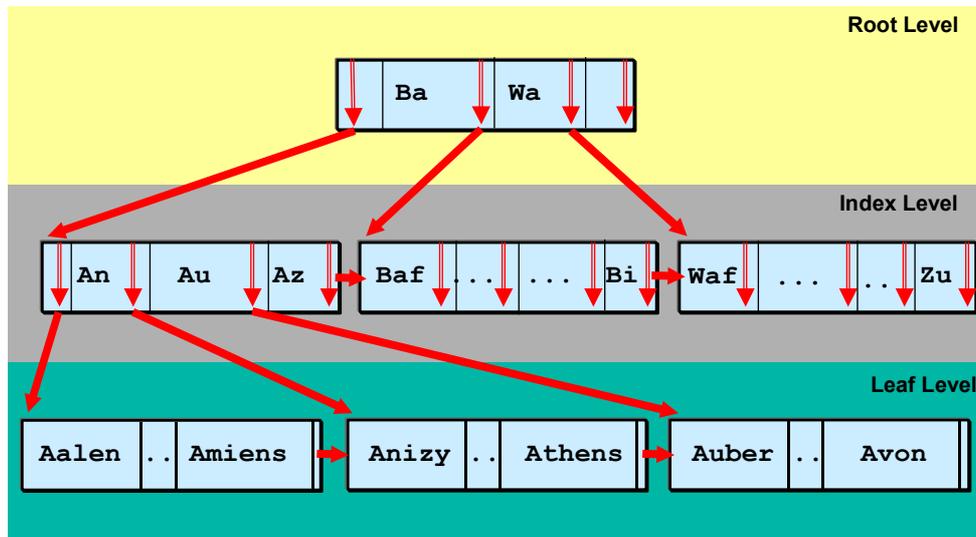
A B* tree spans several levels from the **root level** at the top, to the **index levels**, through to the lowest **leaf level** (one root level, n index levels (n>=1), one leaf level).

The storage units are the [pages \[Page 136\]](#).

In SAP DB Version 7.4, all subsequent comparisons in a B* tree use ASCII code (each character is assigned a numerical value between 0 and 255). The comparisons are performed for all values on a character-by-character basis.

The sort criterion for building a B* tree is the [primary key \[Page 98\]](#).

B* Tree



See also:

[Root/Index Page \[Page 99\]](#)

[Leaf Page \[Page 100\]](#)

[Table Access \[Page 100\]](#)

[B* Trees for Tables \[Page 101\]](#)



Root/Index Page

Entries in root pages and index pages in a [B* tree \[Page 98\]](#) contain two sections.

- First section: This section contains an initial segment of the key fields of a table row. This section is called the **separator**. SAP DB uses only that part of the [primary key \[Page 98\]](#) that is needed to differentiate the following entries. This minimizes the memory needed by these pages.
The first separator in the leftmost page of each level (including the root level) contains only the pointer to the leftmost page of the next level down.
The average length of the separators depends on the structure and the selectivity of the primary key. If a very large number of key fields have to be evaluated to enable the entries (records) to be distinguished from each other, but the characteristics in the front key fields are often the same, then this results in longer separators.
- Second section: This section contains the **logical address** of a page at a lower index level or at the leaf level. The number of entries in this section depends on the length of the separators.



Leaf Page

Leaf pages in a [B* tree \[Page 98\]](#) contain the content of table rows, that is the [application data \[Page 128\]](#). The number of entries in a leaf page depends on the total length of the table rows.

The leaf pages in the B* tree are sorted from left to right in ascending order. In other words, each leaf page contains only the table rows that fit its sort area.

The table rows are located in the first section (**data section**) of the leaf page. The rows in the data section are not sorted by default. The last section of the leaf page contains a **position list** with addresses that point to the corresponding table rows within the leaf list. The position list is always sorted in ascending order.

The storage space required to store the entire table depends on the length of the separators and on the total length of table rows.



Table Access

The logical term **table** denotes primary data, including data in columns of type LONG and data in secondary key structures.

- As specified by its schema definition, a table has exactly one [B* tree \[Page 98\]](#) for the primary data, and exactly one B* tree for each [secondary key \[Page 98\]](#).
- If the table definition contains columns of data type LONG, an additional B* tree is created to accommodate all data of this type that does not exceed a specified length (data of type *short* LONG). Data items of type LONG that exceed this specific length are each stored in separate B* trees (data of type *long* LONG).

All searches and changes in the table are executed in the computer's main memory via SQL statements (SELECT, INSERT, UPDATE, DELETE,...), which means that they are carried out very quickly. The B* tree structure changes itself for all INSERT, UPDATE and DELETE statements whenever there is not enough space in the target page for the new information, or whenever the fill level of a page falls below the predefined fill level.

See also:

[B* Trees for Tables \[Page 101\]](#)

[Table Access Using a B* Tree \[Page 103\]](#)



Table ID

The [database user \[Page 28\]](#) specifies a table by entering its name. The table is accessed internally using a table ID of a fixed length.

The relationship between the table name and this table ID is stored in the [database catalog \[Page 131\]](#).

In addition, SAP DB has a special management structure (file directory), in which the [root pages \[Page 99\]](#) of [B* trees \[Page 98\]](#) are assigned to table IDs.

The Table IDs are stored in the file directory together with a type flag and other information. The type flag specifies whether the table contains primary data, [secondary key \[Page 98\]](#), or LONG data. The type flag specification enables one table ID to be used for all the B* trees required for the logical structures of the same table.

B* Trees for Tables

B* trees [\[Page 98\]](#) are set up for the following kinds of table:

- Tables with primary data
- Tables with [secondary keys \[Page 98\]](#)
- Tables with LONG columns

See also: [Table Access \[Page 100\]](#)

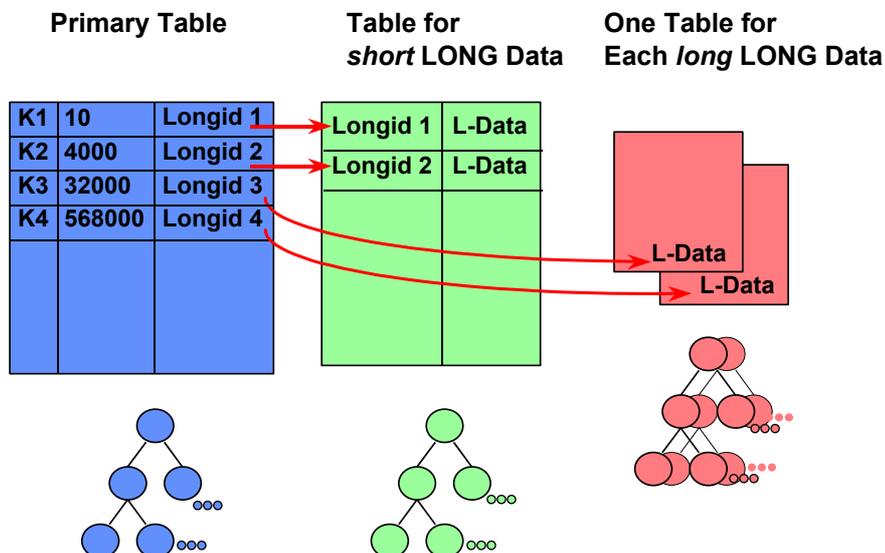
You can find some examples for setting up B* trees here:

- [B* Trees for Tables with LONG Columns \[Page 101\]](#)
- [B* Trees for Tables with Secondary Keys \[Page 102\]](#)
- [B* Trees for Tables with LONG Columns and Secondary Keys \[Page 102\]](#)

B* Trees for Table with LONG Columns

An example of [B* trees for tables \[Page 101\]](#) is a [B* tree \[Page 98\]](#) structure in which the table has primary data and data of data type LONG.

B* Trees for Table with LONG Columns



The primary table contains a column of data type LONG. The numerical column defines the length of the LONG field for each line. There is one B* tree for the primary data, another for data of type *short* LONG and multiple B* trees for other data of type *long* LONG.

The primary table contains a fixed-length reference to the contents of the LONG field in each row. This reference points uniquely to the storage location of the content of the LONG field, either as a key to a B* tree (*short* LONG) or as the [table ID \[Page 100\]](#) of one of the multiple B* trees (*long* LONG).



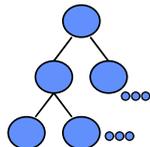
B* Trees for Tables with Secondary Key

An example of [B* trees for tables \[Page 101\]](#) is a [B* tree \[Page 98\]](#) structure in which the table has primary data and a [secondary key \[Page 98\]](#).

B* Trees for Tables with Secondary Key

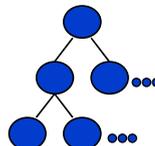
Primary Table

| | | | |
|----|----|----|--|
| K1 | 10 | 10 | |
| K2 | 20 | 10 | |
| K3 | 10 | 10 | |
| K4 | 20 | 40 | |
| K5 | 10 | 10 | |
| K6 | 20 | 40 | |
| K7 | 40 | 30 | |
| | | | |



Secondary Key Table

| | | | | | |
|------|----|----|----|--|--|
| 1010 | K1 | K3 | K5 | | |
| 2010 | K2 | | | | |
| 2040 | K4 | K6 | | | |
| 4030 | K7 | | | | |



The table has a two-column multiple secondary key. There is one B* tree for the primary data and a second B* tree for the data of the secondary table.



B* Trees for Tables with LONG Columns and Secondary Key

An example of [B* trees for tables \[Page 101\]](#) is a [B* tree \[Page 98\]](#) structure in which the table has primary data, a [secondary key \[Page 98\]](#), and data of data type LONG.

B* Trees for Tables with LONG Columns and Secondary Key

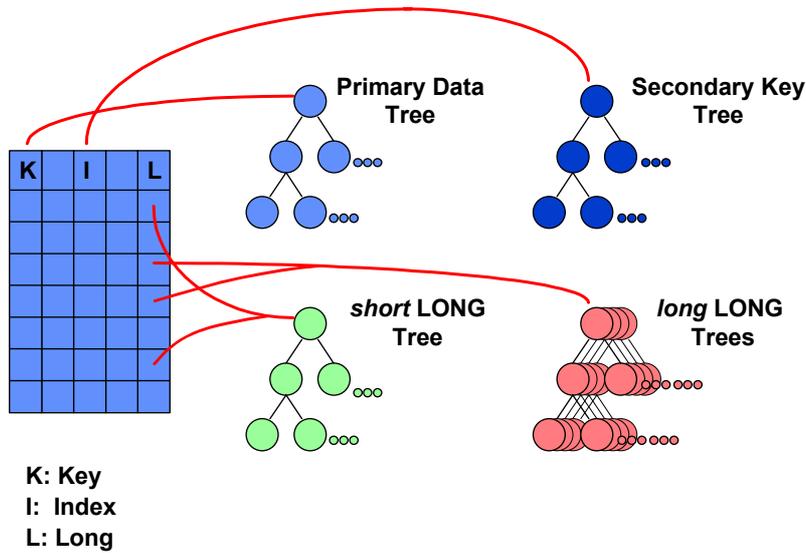


Table Access Using B* Tree

For all SELECT and DML (data manipulation language) statements, the database system uses the same search algorithm to determine the [leaf page \[Page 100\]](#) in which a table entry can be found or has to be changed.

Here you find some examples for table accessing using B* trees:

- [Table Access \(SELECT\) Using B* Tree \[Page 103\]](#)
- [Table Access \(INSERT\) Using B* Tree \[Page 105\]](#)
- [Table Access \(DELETE\) Using B* Tree \[Page 106\]](#)
- [Table Access \(UPDATE\) Using B* Tree \[Page 107\]](#)

See also:

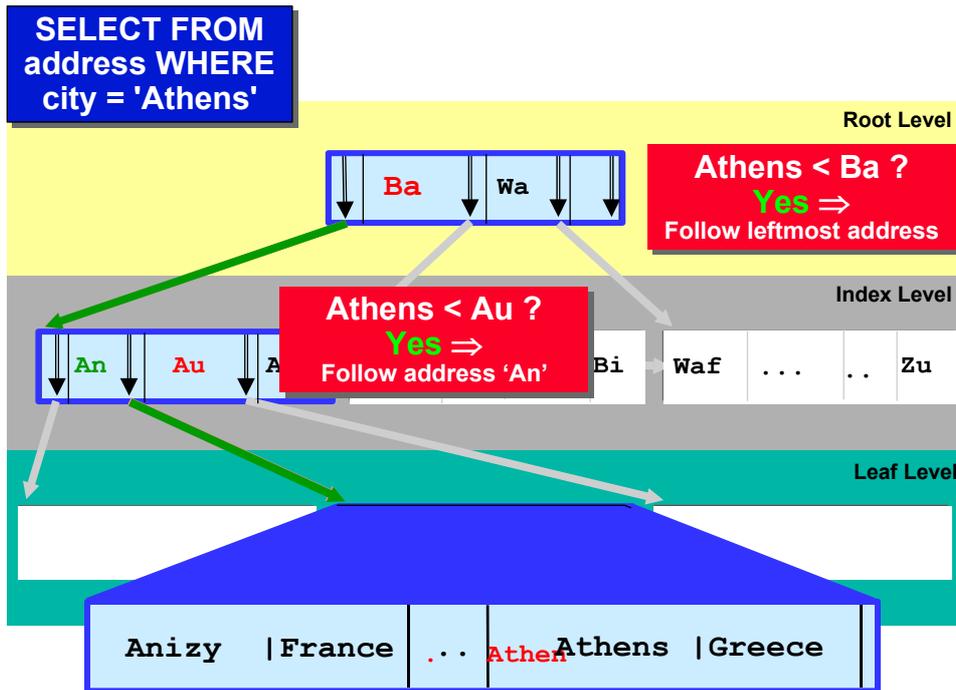
[Table Access \[Page 100\]](#)



Table Access (SELECT) Using B* Tree

[Table Access Using B* Tree \[Page 103\]](#) is illustrated using the example of a SELECT statement.

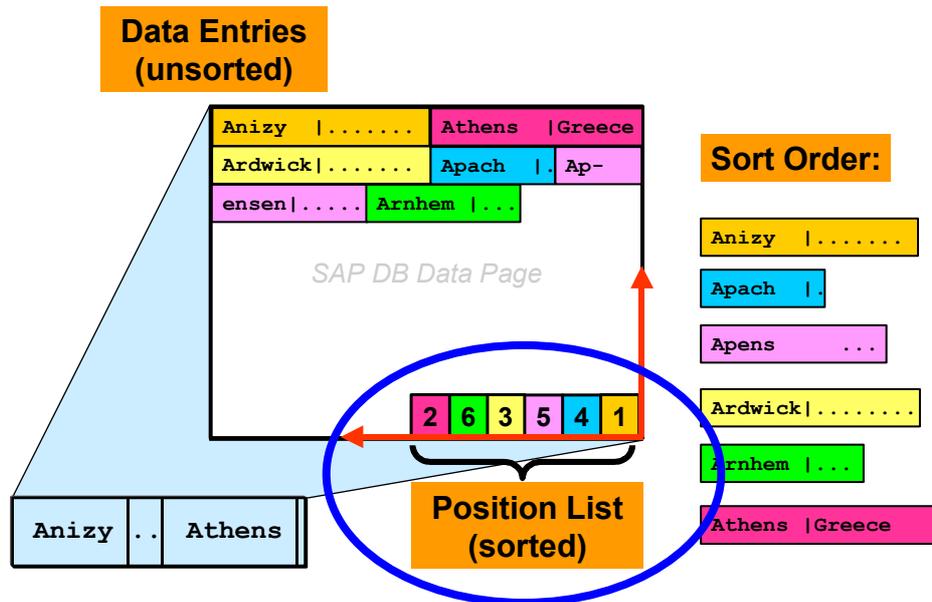
Table Access (SELECT) using B* Tree (1)



The *address* table is defined via the *city* [primary key \[Page 98\]](#) column. In the *address* table, the system is to search for an entry with value *Athens* for the *city* primary key field.

1. The search starts at the *root level* of the [B* tree \[Page 98\]](#). The database system compares the value *Athens* with the value of the first entry in the [root page \[Page 99\]](#), *Ba*.
As the value *Athens* is lower than *Ba*, the relevant address information is evaluated. This points to an [index page \[Page 99\]](#).
2. The search continues at the *index level*. The value *Athens* is greater than the value of the first entry in the data page, *An*. The next value in the page is evaluated. As the value of *Athens* is smaller than the value of *Au*, the corresponding address information is evaluated. This points to a [leaf page \[Page 100\]](#).

Table Access (SELECT) using B* Tree (2)



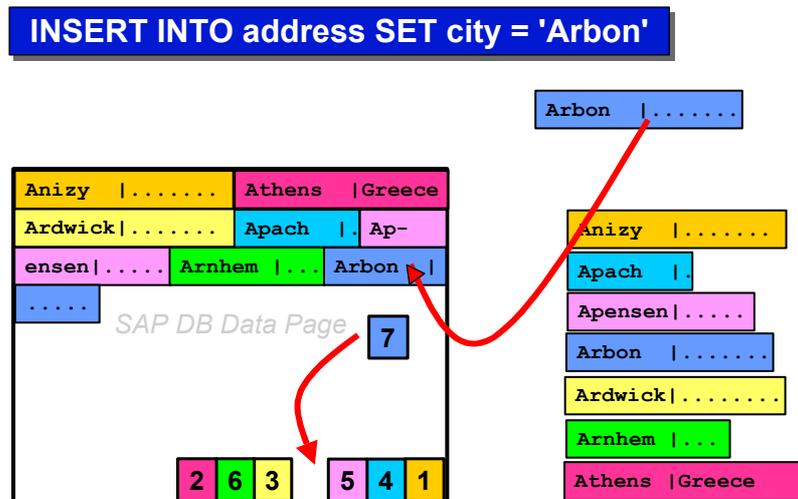
3. The search continues at the *leaf level*. In the position list of the leaf page a binary search algorithm is used. The corresponding address information for the table entry with the key value *Athens* is evaluated.
4. The database system scans the data section in the leaf page until it has found the corresponding table entries. The search is then complete.



Table Access (INSERT) Using B* Tree

[Table Access Using B* Tree \[Page 103\]](#) is illustrated using the example of a INSERT statement.

Table Access (INSERT) Using B* Tree



The *address* table is defined via the *city* [primary key \[Page 98\]](#) column. In the *address* table, insert an entry with the value *Arbon* for the *city* primary key field.

If there is enough space in the [leaf page \[Page 100\]](#) of the [B* tree \[Page 98\]](#) for the new entry, SAP DB inserts the entry at the end of the data section and updates the position list. The address of the new entry is written to the correct position in the position list. In the example above, this is position 4. Position 4 points to the new table entry number 7.

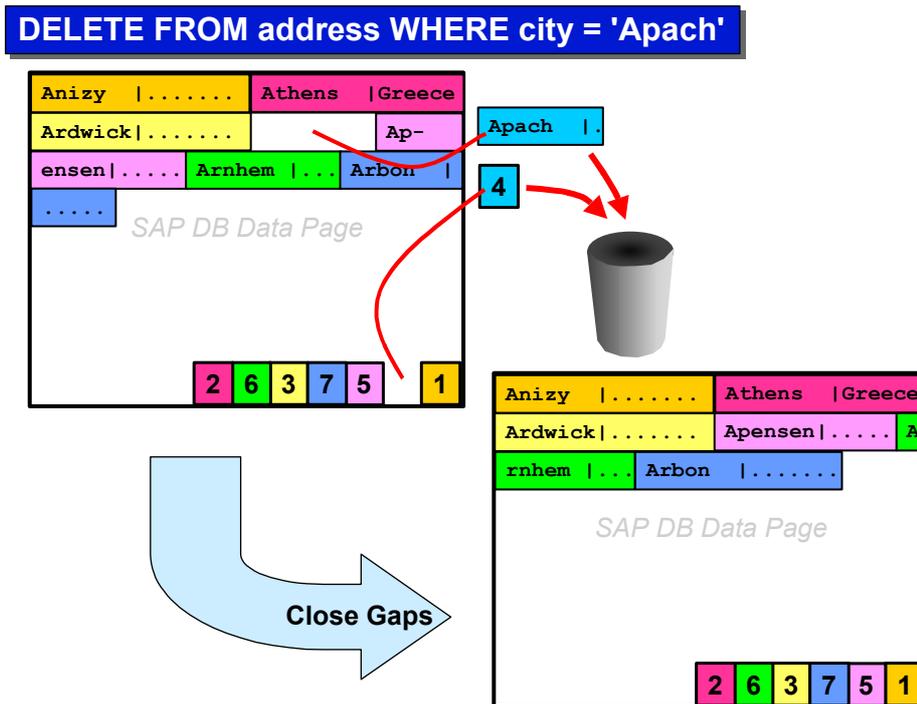
The position list and data section converge when data is inserted in a data page.



Table Access (DELETE) Using B* Tree

[Table Access Using B* Tree \[Page 103\]](#) is illustrated using the example of a DELETE statement.

Table Access (DELETE) Using B* Tree



The *address* table is defined via the *city* [primary key \[Page 98\]](#) column. In the *address* table, delete an entry with the value *Apach* for the *city* primary key field.

If an entry is deleted from the table, gaps appear in the data section and in the position list of the [leaf page \[Page 100\]](#) in the [B* tree \[Page 98\]](#). The database system closes these gaps and updates the position list to save storage space.

Table Access (UPDATE) Using B* Tree

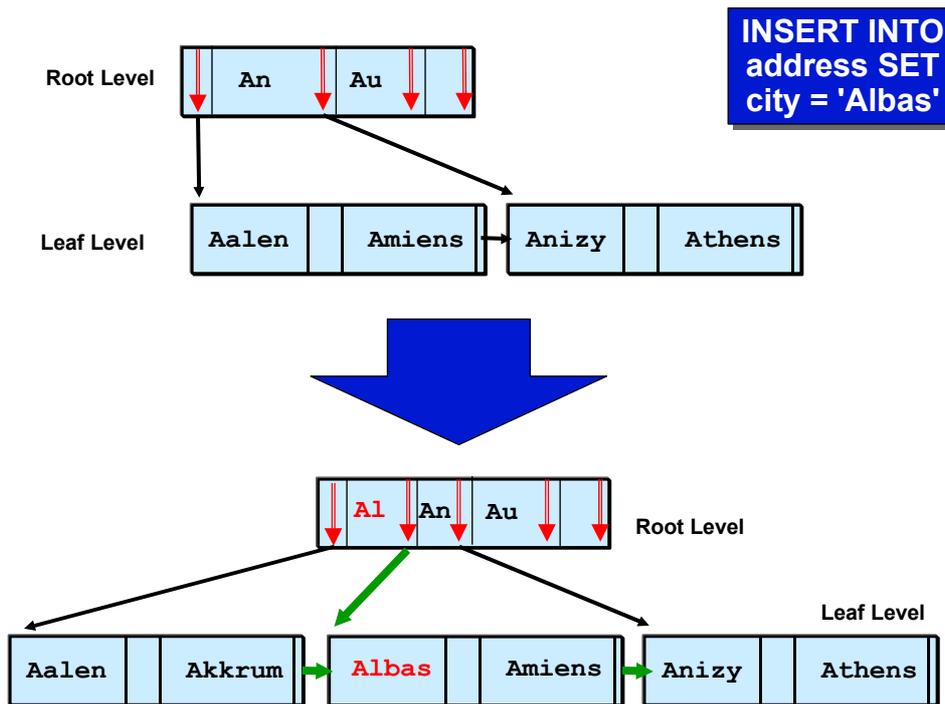
[Table Access Using B* Tree \[Page 103\]](#) is illustrated using the example of an UPDATE statement.

- If an UPDATE is performed and the [primary key \[Page 98\]](#) remains unchanged, only the contents of the table row are changed.
- If the data length changes, the positions of subsequent entries and their addresses in the position list of the [leaf page \[Page 100\]](#) in the [B* tree \[Page 98\]](#) are changed.
- If the key field changes, the UPDATE is executed as a DELETE ([Table Access \(DELETE\) Using B* Tree \[Page 106\]](#)) with a subsequent INSERT ([Table Access \(INSERT\) Using B* Tree \[Page 105\]](#)).

Changes in the B* Tree Structure

The [B* tree \[Page 98\]](#) structure changes in all cases of INSERT, UPDATE, and DELETE when there is either not enough space in the target page or the fill level falls below the predefined fill level.

Changes in the B* Tree Structure



The *address* table is defined via the *city* [primary key \[Page 98\]](#) column. In the *address* table, insert an entry with the value *Albas* for the *city* primary key field. The new table row is too large for the corresponding [leaf page \[Page 100\]](#) in the [B* tree \[Page 98\]](#).

- A new leaf page is created. The new row of the table is incorporated in this leaf page. In other words, the beginning of the sort area of the new leaf page is specified by this row of the table. The sort sequence of the individual leaf pages must be restored. To do so, it may be necessary to copy the table rows that belong in the new sort area from the leaf page with the next smallest sort area to the new leaf page, and then delete them from the previous leaf page. This procedure is called *page split*.
- Pointers to consecutive pages are updated whenever required.
- Address and separator information on the new page is also inserted in the corresponding [root or index page \[Page 99\]](#) one level higher. If the index page is too small to accommodate this information, a new index page is inserted at this index level as well. If the root page at the uppermost level is too small to accommodate a new entry, a new index level is created.

A chain of page splits can result in the B* tree being rebalanced.

See also:

[Non-Uniform Distribution of Data Pages \[Page 108\]](#)



Non-Uniform Distributions of Data Pages

[Changes in the B* tree structure \[Page 107\]](#) can lead to a non-uniform distribution of the data pages. If certain branches of the [B* tree \[Page 98\]](#) have more pages than others, data page distribution is not uniform. This can affect performance, because more accesses are generally required to find data.

Non-uniform distributions are detected by SAP DB when INSERT, UPDATE, or DELETE statements are performed, and the tree is rebalanced when the current operation is carried out. This means that the tree is constantly maintained for optimum operation. During this

procedure, page entries are moved to new locations and page pointers are redirected. As a result, data pages are used more efficiently.

Uniform distribution of data prevents individual data regions from overflowing. The only restriction on the size of tables is the storage space available in the database system.



Lock Behavior

The SAP DB database system is a complete [transactional \[Page 139\]](#) implementation that allows concurrent transactions to access the same database objects. To ensure synchronization and isolation of individual transactions, lock mechanisms are required.

Lock Behavior

The basic idea behind a lock mechanism is that the transaction locks as many of the database objects it is using for as long as is required to ensure the transaction is executed correctly.

Locking an object means that other transactions are not able to use it in certain ways.

The more locks that are set, and the longer these stay in place, the less concurrency is possible in database operation. For this reason, locks should only be set if they are absolutely necessary, and locks that have been set should be released as soon as possible.

- Lock mechanisms differ according to the type of [lock \[Page 109\]](#) selected.
- Locks can be requested and released implicitly and explicitly ([Requesting and Releasing a Lock \[Page 111\]](#)).
- Locks can be set at row, table, and database catalog level.
If too many row locks (shared or exclusive locks) are requested by a transaction for a table, the database system attempts to lock the table. If this causes collisions with other locks, the database system continues to request row locks. This means that table locks are obtained without waiting periods. The limit beyond which the system attempts to transform row locks into table locks depends on the database parameter [MAXLOCKS \[Page 83\]](#).



Lock

The [lock behavior \[Page 109\]](#) depends among other things on the type of lock. Tables and rows are locked in a [transaction \[Page 139\]](#) with a defined lock mode ([Requesting and Releasing a Lock \[Page 111\]](#)).

The database system differentiates between the following locks:

- [Shared lock \[Page 110\]](#)
- [Exclusive lock \[Page 111\]](#)
- [Optimistic lock \[Page 111\]](#)
- Special locks for the metadata of the [database catalog \[Page 131\]](#), which must, however, always be explicitly assigned.

The following table provides an overview of the possible parallel shared locks (S locks) and exclusive locks (E locks).

| Can another transaction | A transaction has a | | | | | |
|---|----------------------|--------|--------------------|--------|--------------------------------------|--------|
| | E lock On a table | S lock | E lock On a row | S lock | E lock On the database catalog | S lock |
| Request an E lock for the table? | No | No | No | No | No | Yes |
| Request an S lock for the table? | No | Yes | No | Yes | No | Yes |
| Request an E lock for any line of the table? | No | No | --- | --- | No | Yes |
| Request an E lock for the locked row? | --- | --- | No | No | --- | --- |
| Request an E lock for another row? | --- | --- | Yes | Yes | --- | --- |
| Request an S lock for any row of the table? | No | Yes | --- | --- | No | Yes |
| Request an S lock for the locked row? | --- | --- | No | Yes | --- | --- |
| Request an S lock for another row? | --- | --- | Yes | Yes | --- | --- |
| Change the definition of the table in the database catalog? | No | No | No | No | No | No |
| Change the definition of the table in the database catalog? | Yes | Yes | Yes | Yes | No | Yes |

A lock collision exists in the cases which are marked with "No"; i.e., after having requested a lock within a transaction, the user must wait for the lock to be released until one of the above situations or one of the situations that are marked with "Yes" in the matrix occurs.

- If no lock has been assigned to a transaction for a data object, then a shared or exclusive lock can be requested within any transaction and the lock is immediately assigned to the transaction.
- If a shared lock has been assigned to a transaction T for a data object, and if no lock has been assigned to any concurrent transaction for this data object, then the transaction T can request an exclusive lock for this data object and the lock is immediately assigned to this transaction.
- If an exclusive lock has been assigned to a transaction for a data object, then a shared can, but need not, be requested for this transaction.



Shared Lock

One type of [lock \[Page 109\]](#) is the shared lock.

Once a shared lock is assigned to a [transaction \[Page 139\]](#) for a particular data object, concurrent transactions can access the object but not modify it. Other transactions can set a shared lock, but not an [exclusive lock \[Page 111\]](#) for this object.

Shared locks refer to a row or a table.

See also:

[Requesting and Releasing a Lock \[Page 111\]](#)



Exclusive Lock

One type of [lock \[Page 109\]](#) is the exclusive lock.

Once an exclusive lock is assigned to a [transaction \[Page 139\]](#) for a particular data object, other transactions cannot modify this object. Transactions that check for the existence of exclusive locks, or that want to set exclusive or [shared locks \[Page 110\]](#), conflict with the existing exclusive lock of another transaction. They do not have access to the locked object.

Exclusive locks refer to a row or a table.

See also:

[Requesting and Releasing a Lock \[Page 111\]](#)



Optimistic Lock

One type of [lock \[Page 109\]](#) is the optimistic lock at row level. It only makes sense to use an optimistic lock if one of the [isolation levels \[Page 112\]](#) [0 \[Page 113\]](#), [1 or 10 \[Page 113\]](#) or [15 \[Page 114\]](#) was assigned. An optimistic row lock must be explicitly requested by specifying a [LOCK statement \[See SAP DB Library\]](#).

An update operation performed by the current user on a row that the user set an optimistic lock on is only actually carried out if this row has not been changed in the meantime by a concurrent [transaction \[Page 139\]](#). If this row has been changed in the meantime by a concurrent transaction, the update operation of the current user is rejected.

The optimistic lock is released in both cases.

- If the update operation was **successful**, an [exclusive lock \[Page 111\]](#) is set for this row.
- If the update operation was **not successful**, it should be repeated after reading the row again with or without optimistic lock. In isolation level 0, an explicit lock must be specified for the new read operation. In this way, it can be ensured that the update is done to the current state and that no modifications made in the meantime are lost.

The request of an optimistic lock only collides with an exclusive lock. Concurrent transactions do not collide with an optimistic lock.



Requesting and Releasing a Lock

To control [lock behavior \[Page 109\]](#), you can request and release [locks \[Page 109\]](#).

Depending on the lock mode selected, locks can be requested and released implicitly or explicitly.

Request Locks

- Request locks **implicitly**
The lock mode can be determined by specifying an [isolation level \[Page 112\]](#) in the [CONNECT statement \[See SAP DB Library\]](#).
In this case, the database system requests locks implicitly during processing of an SQL statement in accordance with the specified isolation level. These locks are then assigned to the [transaction \[Page 139\]](#) that contains this SQL statement. All SQL statements that result in a change (INSERT/UPDATE/DELETE), continuously request an [exclusive lock \[Page 111\]](#) implicitly.
- Request locks **explicitly**
The lock mode can be defined by explicitly specifying locks in the [LOCK statement \[See SAP DB Library\]](#). These locks are then assigned to the transaction that contains this LOCK statement.
Individual rows of a table can be locked by specifying a [LOCK option \[See SAP DB Library\]](#) in an SQL statement. The LOCK option can also be used to change the isolation level for the specified SQL statement, and thereby the locks for the complete table.
Explicit lock requests are possible with every isolation level.

If the database system has to wait too long for locks to be released when setting explicit or implicit locks, it issues a return code to this effect. The user can then respond to this return code, e.g., by terminating the transaction. In this case, the database system does not execute an implicit ROLLBACK WORK.

Release Locks

The locks assigned to a transaction are usually released at the end of the transaction, making the respective database objects accessible again to other transactions.

The locks assigned to a transaction by the LOCK statement are usually released when the transaction ends provided that the [COMMIT statements \[See SAP DB Library\]](#) or the [ROLLBACK statements \[See SAP DB Library\]](#) that end a transaction do not contain a LOCK statement.

- The length of time for which an implicit [shared lock \[Page 110\]](#) is maintained also depends on the isolation level. Exclusive locks set implicitly cannot be released within a transaction.
- Exclusive locks for rows that have not yet been modified and shared locks on rows can be released by the [UNLOCK statement \[See SAP DB Library\]](#) before the end of the transaction.
- Exclusive locks on updated rows, exclusive locks that were requested by an SQL statement that leads to a change (INSERT/UPDATE/DELETE), and table blocks cannot be released within a transaction, only after the transaction has ended.



Isolation Level

The lock mode plays an important part in the [lock behavior \[Page 109\]](#). The lock mode is determined by specifying an isolation level.

By defining an isolation level, you specify whether and how [shared locks \[Page 110\]](#) and [exclusive locks \[Page 111\]](#) are implicitly requested or released ([Requesting and Releasing a Lock \[Page 111\]](#)).

The selected isolation level affects both the degree of concurrency and the guaranteed consistency.

- A high degree of concurrency is characterized by a state in which a maximum number of concurrent [transactions \[Page 139\]](#) can process a database without long waiting periods for locks to be released.
- As far as consistency is concerned, [phenomena \[Page 115\]](#) can arise through concurrent access to the same database at different isolation levels.

The lower the value of the isolation level, the higher the degree of concurrency and the lower the guaranteed consistency. This means that a compromise between concurrency and consistency that best suits the requirements of the application at hand must always be found.

You can use the [CONNECT statement \[See SAP DB Library\]](#) to help define the isolation level you require. The following isolation levels exist:

[Isolation Level 0 \[Page 113\]](#) (uncommitted)

[Isolation Level 1 or 10 \[Page 113\]](#) (committed)

[Isolation Level 15 \[Page 114\]](#)

[Isolation Level 2 or 20 \[Page 114\]](#) (repeatable)

[Isolation Level 3 or 30 \[Page 114\]](#) (serializable)



Isolation Level 0

If [isolation level \[Page 112\]](#) 0 (uncommitted) is specified, rows can be read without [shared locks \[Page 110\]](#) being requested, that is to say, shared locks are not implicitly requested. For this reason, there is no guarantee that a given row will still be in the same state when it is read again within the same [transaction \[Page 139\]](#) as when it was accessed earlier, since it may have been modified in the meantime by a concurrent transaction.

Furthermore, there is no guarantee that the state of a row that was read has already been recorded in the database using COMMIT WORK.

When rows are inserted, updated or deleted, implicit [exclusive locks \[Page 111\]](#) are assigned to the transaction for the rows concerned. These cannot be released until the end of the transaction.

Isolation level 0 is suitable for applications that use their own methods to enable concurrent access to objects, so guaranteeing concurrent access throughout the database system.



Isolation Level 1 or 10

If [isolation level \[Page 112\]](#) 1 or 10 (committed) is specified, a [shared lock \[Page 110\]](#) is assigned to the [transaction \[Page 139\]](#) for a read row R1 of a table. When the next row R2 in the same table is read, the lock on R1 is released and a shared lock is assigned to the transaction for the row R2.

For data retrieval using a [QUERY statement \[See SAP DB Library\]](#), the database system ensures that, at the time each row is read, no [exclusive lock \[Page 111\]](#) has been assigned to other transactions for the given row. It is impossible to predict, however, whether a QUERY statement causes a shared lock for a row of the specified table or not and for which row this may occur.

When rows are inserted, updated or deleted, implicit exclusive locks are assigned to the transaction for the rows concerned. These cannot be released until the end of the transaction.



Isolation Level 15

The following rules apply if [isolation level \[Page 112\] 15](#) is specified:

For all SQL statements, the behavior described for [isolation level 1 or 10 \[Page 113\]](#) also applies for isolation level 15: The only difference is that, with isolation level 15, [shared locks \[Page 110\]](#) are requested for all the tables addressed by the SQL statement before processing starts. When the SQL statement generates a result table that is not physically stored, these locks are not released until the end of the transaction or until the result table is closed. Otherwise, they are released immediately after the SQL statement has been processed.

When rows are inserted, updated or deleted, implicit [exclusive locks \[Page 111\]](#) are assigned to the [transaction \[Page 139\]](#) for the rows concerned. These cannot be released until the end of the transaction.



Isolation Level 2 or 20

If [isolation level \[Page 112\] 2 or 20 \(repeatable\)](#) is specified, [shared locks \[Page 110\]](#) are requested for all the tables addressed by an SQL statement [data query \[See SAP DB Library\]](#) before processing starts.

When an SQL statement generates a result table that is not physically stored, these locks are not released until the end of the transaction or until the result table is closed. Otherwise, they are released immediately after the SQL statement has been processed. This table shared lock is not assigned to the [transaction \[Page 139\]](#) in the case of SQL statements in which just one table row is processed that is determined by [key specifications \[See SAP DB Library\]](#) or by `CURRENT OF <result_table_name>`.

In addition, an implicit shared lock is assigned to the transaction for each row read while an SQL statement is being processed. These locks can only be released with the [UNLOCK statement \[See SAP DB Library\]](#) or by ending the transaction.

When rows are inserted, updated or deleted, [exclusive locks \[Page 111\]](#) are assigned implicitly to the transaction for the rows concerned. These cannot be released until the end of the transaction. However, no locks are set for the entire table.



Isolation Level 3 or 30

If [isolation level \[Page 112\] 3 or 30 \(serializable\)](#) is specified, a table [shared lock \[Page 110\]](#) is implicitly assigned to the [transaction \[Page 139\]](#) for every table addressed by an SQL statement.

These shared locks cannot be released until the end of the transaction. This table shared lock is not assigned to the transaction in the case of SQL statements in which just one table row is processed that is determined by [key specifications \[See SAP DB Library\]](#) or by `CURRENT OF <result_table_name>`.

When rows are inserted, updated or deleted, [exclusive locks \[Page 111\]](#) are assigned implicitly to the transaction for the rows concerned. These cannot be released until the end of the transaction.



Phenomena

During processing of concurrent [transactions \[Page 139\]](#), unclarified situations, or “phenomena” can arise. When you define the [lock behavior \[Page 109\]](#) in your database system, you should take account on the ways of avoiding these phenomena.

The following phenomena can occur if the same data is accessed concurrently:

- [Dirty Read \[Page 115\]](#)
- [Non-Repeatable Read \[Page 115\]](#)
- [Phantom \[Page 115\]](#)

The lock mode is defined via the [isolation level \[Page 112\]](#) ([Requesting and Releasing a Lock \[Page 111\]](#)). Depending on the isolation level you determine, you can prevent these phenomena from occurring.

The following table provides an overview of which phenomena can occur in the individual isolation levels:

| | Isolation Level 0 | Isolation Level 1 | Isolation Level 2 | Isolation Level 3 |
|---------------------|-------------------|-------------------|-------------------|-------------------|
| Dirty Read | + | - | - | - |
| Non-Repeatable Read | + | + | - | - |
| Phantom | + | + | + | - |



Dirty Read

One of the [phenomena \[Page 115\]](#) that can occur during processing of concurrent [transactions \[Page 139\]](#) is the dirty read phenomenon.

A row is modified in the course of a transaction T1, and a transaction T2 reads this row before T1 has been concluded with the COMMIT statement. T1 then executes the ROLLBACK statement, i.e. T2 has read a row that never actually existed.



Non-Repeatable Read

One of the [phenomena \[Page 115\]](#) that can occur during processing of concurrent [transactions \[Page 139\]](#) is the non-repeatable read phenomenon.

A transaction T1 reads a row. A transaction T2 then modifies or deletes this row, concluding with the COMMIT statement. If T1 subsequently reads the row again, it either receives the modified row or a message indicating that the row no longer exists.



Phantom

One of the [phenomena \[Page 115\]](#) that can occur during processing of concurrent [transactions \[Page 139\]](#) is the phantom phenomenon.

A transaction T1 executes an SQL statement S that reads a set M of rows that fulfill a [search condition \[See SAP DB Library\]](#). A transaction T2 then uses the INSERT statement or the

UPDATE statement to create at least one additional row that satisfies the search condition. If S is subsequently re-executed within T1, the set of read rows will differ from M.



Creating a Homogeneous System Copy

Use

You want to create a copy of your database instance, for example, for test or development purposes, or to set up a [standby database \[Page 118\]](#).

Prerequisites

- The database instance that you want to copy has the [database instance type \[Page 22\]](#) OLTP, document server, OLAP, or E Catalog, and the processor architectures of the source and target hosts are [compatible \[Page 117\]](#).
- The database instance that you want to copy has the database instance type [liveCache \[Page 23\]](#). Both the processor architectures and the operating systems of the source and target hosts are identical, and you are using the same [DB procedures \[See SAP DB Library\]](#) on both hosts.

Procedure

Use the [Database Manager \[Page 61\]](#) for the following steps:

1. Make a [complete data backup \[Page 42\]](#) of the original instance.
2. Initialize an existing target instance, or install a new target instance for *recovery*. The target instance must have the same database instance type as the original instance, and the same software version 7.x.



The target instance must not switch to the [operational state \[Page 136\]](#) ONLINE before you import the data backup from the original instance (step 5). If the target instance has already been ONLINE, reinitialize it.

3. Configure the [data volumes \[Page 21\]](#) of the target instance so that they have enough space for the import of the complete data backup of the original instance (10% larger than the complete data backup of the original instance, for example).
4. In the target instance, create a [backup medium \[Page 129\]](#) for importing data backups. The medium you use must be compatible with the medium that you used to create the complete data backup of the original instance.
5. Import the complete data backup of the original instance into the target instance.



If using the Database Manager GUI, proceed as described under [Restoring Without a Backup History \[See SAP DB Library\]](#).

6. Start the target instance.
7. Update the system tables in the target instance.
8. In the target instance, restore any indexes that are no longer up-to-date (BAD INDEXES).

Example

[Homogeneous System Copy with the Database Manager CLI \[Page 117\]](#)

Operating System Compatibility for Homogeneous System Copies

If you want to create a homogeneous system copy of your SAP DB database instance, the processor architectures of the source and target hosts must be compatible. The following compatibility matrix is valid for the database instance types OLTP, Document Server, OLAP, and E-Catalog.

| | Win32 | Win64 | Linux 32 | Linux 64 | Tru64 | AIX | HP-UX | Solaris |
|----------|-------|-------|----------|----------|-------|-----|-------|---------|
| Win32 | ✗ | ✗ | ✗ | ✗ | ✗ | | | |
| Win64 | ✗ | ✗ | ✗ | ✗ | ✗ | | | |
| Linux 32 | ✗ | ✗ | ✗ | ✗ | ✗ | | | |
| Linux 64 | ✗ | ✗ | ✗ | ✗ | ✗ | | | |
| Tru64 | ✗ | ✗ | ✗ | ✗ | ✗ | | | |
| AIX | | | | | | ✗ | ✗ | ✗ |
| HP-UX | | | | | | ✗ | ✗ | ✗ |
| Solaris | | | | | | ✗ | ✗ | ✗ |



The media that you use for creating and importing the data backup must also be compatible.

Homogeneous System Copy with the Database Manager CLI

Creating a backup medium for complete data backups in the original instance:

```
dbmcli -d TST -u dbm,dbm medium_put databasecopy /usr/sapdb/dbcopy FILE DATA
0 0 NO
```

Creating a complete data backup of the original instance:

```
dbmcli -d TST -u dbm,dbm -uUTL -c backup_start databasecopy
```

Installing the target instance:

```
dbmcli -s -R <dependent_path> db_create TST1 dbm,dbm
```

Copying and modifying database parameters from the original instance:

```
dbmcli -d TST1 -u dbm,dbm -uSRV -c recover_config databasecopy
dbmcli -d TST1 -u dbm,dbm
> param_startsession
> param_put RUNDIRECTORY <newrundirectory>
> param_put DIAG_HISTORY_PATH <newdiaghistoricalpath>
> param_checkall
> param_commitsession
```

Configuring data volumes of the target instance:

```
dbmcli -d TST1 -u dbm,dbm param_addvolume DATA <newdatavolume> <datatype>
<datasize>
dbmcli -d TST1 -u dbm,dbm param_addvolume LOG <newlogvolume> <logtype>
<logsize>
```

Creating the backup medium in the target instance:

```
dbmcli -d TST1 -u dbm,dbm medium_put databasecopy /usr/sapdb/dbcopy FILE
DATA 0 0 NO
```

Switching the target instance to the operational state ADMIN:

```
dbmcli -d TST1 -u dbm,dbm db_admin
```

Initializing the target instance:

```
dbmcli -d TST1 -u dbm,dbm util_execute INIT CONFIG
```

Importing the data backup of the original instance into the target instance:

```
dbmcli -d TST1 -u dbm,dbm -uUTL -c recover_start databasecopy
```

Switching the target instance to the operational state ONLINE:

```
dbmcli -d TST1 -u dbm,dbm db_online
```

Loading the system tables; assigning passwords for SYSDBA and DOMAIN users:

```
dbmcli -d TST1 -u dbm,dbm load_systab -u <sysdba>,<password> -ud <password2>
```



Standby Databases with SAP DB

SAP DB gives you additional downtime security by supporting the setup of [standby databases \[Page 118\]](#).

A standby instance is a copy of the active database instance (original instance). To keep the contents of your database up-to-date, the [log backups \[Page 43\]](#) of the original instance are imported at regular intervals. The standby instance always has the [operational state \[Page 136\]](#) ADMIN.

If you experience problems with the original instance, you [start operating \[Page 119\]](#) the standby instance immediately, and carry on working without a lengthy period of downtime.

Depending on the configuration, you can also restore the standby instance to a specified state in the past.

See also: [Example \[Page 122\]](#)



Setting Up a Standby Instance

This section describes how you set up a [standby database with SAP DB \[Page 118\]](#).

Procedure

Use the [Database Manager \[Page 61\]](#) for the following steps:

1. Create the standby instance by making a copy of your original instance. Proceed as described in [Creating a Homogeneous System Copy \[Page 116\]](#) (points 1-5). However, do **not** start the copy in the [operational state \[Page 136\]](#) ONLINE; instead, keep it in the operational state ADMIN.



Set up the original instance and standby instance on different hosts.

2. In the standby instance, create a [backup medium \[Page 129\]](#) for importing log backups.
3. Make log backups in your active original instance at regular intervals.
4. Import these log backups into your standby instance with the time delay of your choice.



In the Database Manager GUI, import the backups as described under [Restoring Without a Backup History \[See SAP DB Library\]](#).

In the Database Manager CLI, use the [restore commands \[See SAP DB Library\]](#) `recover_start` and `recover_replace`.

Notes

Keep the following points in mind:

- The [data volumes \[Page 21\]](#) and [log volumes \[Page 22\]](#) of the standby instance must be configured with enough space to import the log backups of the original instance.
- The two instances cannot access the same files. This means that their [run directories \[Page 137\]](#) must be physically separate.
- You can use pipes to transport the data backup. You can transport log backups with ftp, or copy them with an exported file system (such as NFS or a shared file system).



You can start the standby instance in the operational state `ONLINE`, if you want it to replace the original instance as the active instance (see [Starting the Standby Instance as an Active Instance \[Page 119\]](#)). If the standby instance has been in the state `ONLINE`, but you still want to use it as a standby instance, then you must initialize the instance, and import a complete data backup of the original instance again.



Starting the Standby Instance as an Active Instance

If you experience problems with your original instance, then you can stop it, and switch the [operational state \[Page 136\]](#) of the standby instance to `ONLINE`. The standby instance can then take over the role of the active instance immediately.

Loss of Data

Remember that in this procedure, the standby instance does not contain any data changes that have not yet been imported in [log backups \[Page 43\]](#) from the original instance.



If you want to start the standby instance **without losing any data**, or you want to start it in a **state from the past**, you have the following options, depending on the configuration and state of your original instance:

- [Importing Log Backups up to a Specific Time \[Page 120\]](#)
- [Importing Another Manual Log Backup \[Page 121\]](#)
- [Copying the Log Volumes of the Original Instance \[Page 121\]](#)

Prerequisites

- You have set up a [standby instance \[Page 118\]](#).

- The standby instance has not been in the operational state `ONLINE` since it was last initialized.

Procedure

Use the [Database Manager \[Page 61\]](#) for the following steps:

1. Make sure that the original instance is in the operational state `OFFLINE`.
2. Start the standby instance in the operational state `ONLINE`.
3. In this standby instance, restore any indexes that are no longer up-to-date (`BAD INDEXES`).



In the Database Manager GUI, proceed as described in [Restoring the Indexes After a Database Restore \[See SAP DB Library\]](#).

In the Database Manager CLI, use the command `sql_recreateindex` to [recreate a damaged index \[See SAP DB Library\]](#).

4. Make a [complete data backup \[Page 42\]](#) of the standby instance.

Result

You can now operate the former standby instance as an active database instance. The original instance can become the standby instance. To do this, initialize the original instance, and proceed as described in [Setting Up a Standby Instance \[Page 118\]](#).



Importing Log Backups up to a Specific Time

You can start the standby instance in a specific state from the past. You may need to do this if, for example, you lose data during an operation in the original instance, and want to restore the state of the data before the error.

Prerequisites

- You have set up a [standby instance \[Page 118\]](#).
- The shadow instance has not been in the operational state `ONLINE` since it was last initialized.
- You have imported log backups from the original instance only up to a point in time **before** your chosen restore time.

Procedure

If the log backups are imported automatically from the original instance into the standby instance, then cancel the imports (`recover_cancel`), and import manually any log backups that are missing, up until your chosen time. Proceed as follows:

- When you import the log backup for the relevant point in time in the Database Manager GUI, choose the option *Restore until a specific time* (see also [Restoring Without a Backup History \[See SAP DB Library\]](#)).
- When you import the log backup for the relevant point in time in the Database Manager CLI, use the command `recover_start` with the option `UNTIL <date> <time>` (see also [Commands for Restoring \[See SAP DB Library\]](#)).

Result

The log entries are imported into the standby instance up to the chosen time. Any changes you have made after this time in the original instance are ignored by the standby instance.

You can now [start operating the standby instance as an active instance \[Page 119\]](#).



Importing Another Manual Log Backup

If it is possible to create another manual log backup in the original instance, then you can start the standby instance without losing data, which means that it is in the same state as the original instance.

Prerequisites

- You have set up a [standby instance \[Page 118\]](#).
- The shadow instance has not been in the operational state `ONLINE` since it was last initialized.
- The [log volumes \[Page 22\]](#) of the original instance are not damaged.
- You can create a manual log backup in the original instance in the operational state `ADMIN`.

Procedure

1. Create a manual log backup in the original instance.
2. Import this log backup into the standby instance.

Result

You can now [start operating the standby instance as an active instance \[Page 119\]](#).



Copying the Log Volumes of the Original Instance

If the [log volumes \[Page 22\]](#) of the original instance are not damaged, and you can use operating system tools to copy their content to the log volumes of the standby instance, then you can start the standby instance without losing data or in a state from the past.

Prerequisites

- You have set up a [standby instance \[Page 118\]](#).
- The shadow instance has not been in the operational state `ONLINE` since it was last initialized.
- The log volumes of the original instance are not damaged.
- The number of log volumes and their size are the same in both instances.
- The original instance is in the operational state `OFFLINE`.

Procedure

1. If the log backups are imported automatically from the original instance into the standby instance, then cancel the imports (`recover_cancel`).

2. Use operating system tools to copy the log volumes from the original instance to the standby instance.
3. Import any missing log backups from the original instance into the standby instance.

Result

You can now [start operating the standby instance as an active instance \[Page 119\]](#).

You can also start the standby instance in a state from a time chosen in the past.



To do this in the Database Manager GUI, choose *Instance* → *Restart Until*.

To do this in the Database Manager CLI, use the command `db_online -u`.



Example: Standby Database

This example of an input script shows you how to use the Database Manager CLI to [set up a standby instance \[Page 118\]](#) and [start operating it as an active instance \[Page 119\]](#).



For more detailed information on working with the Database Manager CLI administration tool, and the meaning of the commands, see [Database Manager CLI: SAP DB 7.4 \[See SAP DB Library\]](#).

This example is valid for database instances with a database version lower than 7.4.04.

Input Script

Initializing an existing standby instance:

```
db_offline
db_admin
util_connect
util_execute init config
util_release
```



The command `init_config` deletes all data in the database instance permanently.

Creating a backup medium in the standby instance for importing a complete data backup from the original instance:

```
medium_put <medium_name_data> <path_data> <medium_type> DATA
```

Importing the data backup:

```
util_connect
recover_start <medium_name_data>
util_release
```

Creating a backup medium in the standby instance for importing the log backups from the original instance:

```
medium_put <medium_name_log> <path_log>\save.log FILE LOG
```

The log backups are numbered automatically and saved in the file `save.log <nnn>` in the path `<path_log>`.

Importing the log backups during a utility session:

```
util_connect
```

```
recover_start <medium_name_log> LOG 001
recover_replace <medium_name_log> <path_log>\save.log 002
recover_replace <medium_name_log> <path_log>\save.log 003
...
```

The return code is -8020 (another medium is needed).

Starting to operate the standby instance as an active instance while terminating the utility session:

```
recover_ignore
```

In this case, the standby instance does not include the changes that have not yet been imported with the log backups from the original instance.



You can also terminate the import of the log backups without starting the standby instance in the operational mode `ONLINE` at the same time. To do this, use the command `recover_cancel`.

As of Version 7.4.03, this command switches the state of a database instance to `OFFLINE`. Before you can specify other commands, you must first use the command `db_admin` to switch the operational state of the standby instance to `ADMIN`.



SAP DB Version 7.4

Development of the independent SAP DB software was begun in 1997 on the basis of version 6.1. Since then, SAP AG has significantly further developed and improved the SAP DB technology. The most important parts of the original software package have been completely redesigned and integrated into the SAP DB software. In this way, SAP DB meets the [requirements \[Page 123\]](#) for a modern database system.

- [SAP DB Improvements Since 1997 \[Page 123\]](#)
- [SAP DB Tools \[Page 124\]](#)
- [Technical Specification of SAP DB Version 7.4 \[Page 124\]](#)
- [New Developments in SAP DB Version 7.4 \[Page 126\]](#)



Requirements for a Database System

- High performance
- High availability
- Easy to operate
- Low cost of ownership



SAP DB Improvements Since 1997

- Optimized checkpoint handling
- Parallel creation of indexes
- Parallel backup and restore

- Optimized locking and logging
- Better handling of “log full” and “database full” errors
- Interactive support for the restore process
- Reduced converter cache size (factor 10)
- Improved monitoring and diagnostics
- Unicode support (UTF-16)
- Support for the Linux operating system
- 64 bit UNIX versions as operating system platform
- New database tools



SAP DB Tools

Database Management

You can use the **Database Manager** to manage databases. You can use the following clients:

- Database Manager GUI (DBMGUI) on Windows-based operating systems
- Database Manager CLI (DBMCLI)
- Web DBM

You can make additional clients available using a script interface to Perl or Python.

Entering SQL Statements

You can use the **query tools** to enter SQL statements, such as data requests. You can use the following clients:

- SQL Studio on Microsoft Windows-based operating systems
- Web SQL Studio

Unloading and Loading Data

You can use the **SAP DB Loader** to unload and load data. You can use the following client:

- SAP DB Loader CLI (LOADERCLI)

You can make additional clients available using a script interface to Perl or Python.



Technical Specification of SAP DB Version 7.4

Restrictions

| | Maximum Value |
|-------------------------------------|--|
| Size of data area | 32 TB (with 8 KB page size) |
| Number of data volumes per database | Default value: 256 You can use the database parameter DEVNO_BIT_COUNT to set a value between 64 and 4096 when you install the |

| | |
|---|---|
| | database system |
| Size of a data volume | Default value: 128 GB Depends on the number of data volumes |
| Size of the log area. | 32 TB |
| SQL statement length | >= 16 KB (Default value 64 KB, defined by a database parameter) |
| SQL character string length | Defined by a database parameter |
| Identifier length | 32 characters |
| Numeric precision | 38 places |
| Number of tables | unlimited |
| Number of columns per table (with KEY) | 1024 |
| Number of columns per table (without KEY) | 1023 |
| Number of primary key columns per table | 512 |
| Number of columns in an index | 16 |
| Number of foreign key columns per table | 16 |
| Number of the columns in an ORDER or GROUP clause | 128 |
| Number of columns in a SELECT statement | 1023 |
| Number of columns in an INSERT statement | 1024 |
| Number of columns in a results table | 1023 |
| Number of join tables in a SELECT statement | 64 |
| Number of triggers per Basis table | 3 |
| Number of indexes per table | 255 |
| Number of referential CONSTRAINT definitions (foreign key dependencies) per table | unlimited |
| Number of references per table | unlimited |
| Number of rows per table | limited by database size |
| Internal length of a table row | 8088 bytes |
| Total of internal lengths of all key columns | 1024 bytes |
| Total of internal lengths of all columns belonging to an index | 1024 bytes |
| Internal length of a LONG column | 2 GB |
| Length of the columns in an ORDER or GROUP clause | 1020 bytes |
| Nested trigger levels | unlimited |
| Nested subqueries | 127 |
| Number of join conditions in a WHERE clause of a SELECT statement | 128 |
| Number of correlated columns in an SQL statement | 64 |
| Number of correlated columns in an SQL | 16 |

| | |
|--|------|
| statement | |
| Number of parameters in an SQL statement | 2000 |



New Developments in SAP DB Version 7.4

New I/O Management

- No system volume
The system volume no longer exists.
The converter pages are distributed among all [data volumes \[Page 21\]](#).
- No converter cache
The converter pages and data pages are kept in a shared [I/O buffer cache \[Page 19\]](#).
You now only require the database parameter [CACHE_SIZE \[Page 79\]](#) for the configuration of the size of the I/O buffer cache.
- Parallel converter queries (multi-region converter)
The [converter \[Page 19\]](#) can now be accessed in multiple regions at the same time.

Database Can Be Reduced

- The size of the database is independent of the largest page number used for a data page. In some circumstances, gaps in the sequence of the page numbers can mean that the database is significantly smaller than suggested by the highest number.
- The number of data volumes at the time of the database restart can be different from the number of data volumes at the time of the backup.

New Log Management

- Before and after images split
Before images are stored in transaction-specific [undo log files \[Page 49\]](#) in the [data area \[Page 131\]](#).
After images are stored in the [log area \[Page 135\]](#).
- Parallel save of before images
In parallel transactions, the before images are saved in parallel in transaction-specific undo log files.

New Data Backup

- Checkpoints are obsolete.
- Each [data backup \[Page 42\]](#) is consistent internally. This means that you can start the database instance after importing the data backup without needing to import log entries. The state of the data then matches the state when the data backup was started. You can use the log entries in the log area or in [log backups \[Page 43\]](#) to recover the state of the database instance to a chosen time after the data backup.
- [Incremental data backups \[Page 42\]](#) contain all changes since the last complete data backup
This simplifies the recovery process. When you recover the database instance, you need only to import the last complete data backup and the **last** incremental data backup, as well as the log entries required for your chosen point-in-time.



Terms

| | | |
|--|---|--|
| After image [Page 46] | Application data [Page 128] | Automatic log backup [Page 43] |
| Availability [Page 40] | | |
| Backup [Page 41] | Backup history [Page 129] | Backup ID [Page 129] |
| Backup medium [Page 129] | Before image [Page 46] | B* tree [Page 98] |
| Cache [Page 19] | Catalog [Page 131] | Catalog cache [Page 19] |
| COMMIT [Page 131] | Complete data backup [Page 42] | Configuration files [Page 77] |
| Console [Page 132] | Converter [Page 19] | |
| Data area [Page 131] | Data backup [Page 42] | Data cache [Page 20] |
| Database administrator [Page 29] | Database catalog [Page 131] | Database consistency [Page 130] |
| Database console [Page 132] | Database instance [Page 13] | Database instance type [Page 22] |
| Database Manager [Page 61] | Database Manager CLI [Page 62] | Database Manager GUI [Page 61] |
| Database Manager operator (DBM operator) [Page 26] | Database name [Page 132] | Database parameters [Page 77] |
| Database session [Page 132] | Database structure check [Page 130] | Database tools [Page 55] |
| Database trace [Page 133] | Database user [Page 28] | Data volume [Page 21] |
| Data Writer [Page 14] | DBA [Page 29] | DBMCLI [Page 62] |
| DBMGUI [Page 61] | DBM operator [Page 133] | DBM Server [Page 60] |
| Dev thread [Page 18] | Dirty read [Page 115] | DOMAIN [Page 30] |
| Exclusive lock [Page 111] | External backup ID [Page 133] | External backup medium [Page 134] |
| External backup tool [Page 45] | | |
| Garbage Collector [Page 51] | Group of parallel backup media [Page 134] | |
| History file [Page 50] | History list [Page 50] | Homogeneous system copy [Page 117] |
| Incremental data backup [Page 42] | Instance type [Page 134] | Interactive log backup [Page 44] |
| I/O buffer cache [Page 19] | Isolation level [Page 112] | |
| Kernel [Page 134] | | |
| Language support (Mapchar sets) [Page 134] | liveCache [Page 23] | Loader Server [Page 60] |
| Lock [Page 109] | Log area [Page 135] | Log backup [Page 43] |
| Log entry [Page 45] | Log files [Page 75] | Logging [Page 46] |
| Log mode [Page 38] | Log queue [Page 47] | Log settings [Page 37] |

| | | |
|---|---|---|
| Log volume [Page 22] | Log Writer [Page 48] | |
| Multiprocessor configuration [Page 25] | | |
| Name of external backup medium [Page 135] | Name of standard backup medium [Page 135] | Normal log backups [Page 38] |
| Online logging [Page 46] | Operational state [Page 136] | Optimistic lock [Page 111] |
| Page [Page 136] | Page pool [Page 136] | Pager [Page 14] |
| Parallel backup [Page 42] | | |
| Query tools [Page 66] | | |
| Redo list [Page 52] | Redo log entry [Page 46] | Redo log file [Page 52] |
| Redo log management [Page 47] | RESOURCE [Page 30] | Restart [Page 136] |
| Role concept [Page 30] | ROLLBACK [Page 137] | Run directory [Page 137] |
| SAP DB Document Server [Page 23] | SAP DB E-Catalog [Page 24] | SAP DB Loader [Page 65] |
| SAP DB OLAP [Page 23] | SAP DB OLTP [Page 23] | SAP DB user classes [Page 26] |
| Savepoint [Page 137] | Server task [Page 15] | Session [Page 132] |
| Shared lock [Page 110] | Single backup medium [Page 138] | SQL lock [Page 109] |
| SQL mode [Page 138] | SQL Studio [Page 67] | Standby database [Page 118] |
| SYSDBA [Page 29] | | |
| Task [Page 138] | Thread [Page 13] | Timeout value [Page 138] |
| Transaction [Page 139] | Transaction file [Page 139] | Transaction list [Page 139] |
| Undo log entry [Page 46] | Undo log file [Page 49] | Undo log management [Page 49] |
| UNICODE [Page 88] | User [Page 28] | User groups [Page 30] |
| User task [Page 15] | Utility task [Page 16] | |
| Version file [Page 140] | Version notation [Page 144] | |
| Volume [Page 21] | | |
| Web DBM [Page 64] | Web Server [Page 61] | Web SQL Studio [Page 67] |
| X Server [Page 60] | | |



Application Data

The application data of a [database instance \[Page 13\]](#) are all the rows of all base tables and all the index entries created for the base tables.



Backup History

Information on all actions that have been carried out that relate to saving and recovering the [database instance \[Page 13\]](#) are registered in chronological order in the backup history.

The system writes a backup history to log file `dbm.knl`. This file is in the database instance's [run directory \[Page 137\]](#).



Backup ID

The database system automatically gives each backup a backup ID to allow them to be identified. This ID uniquely identifies the backups done since the creation of the database instance. If external backup tools are used for the backup, the backup is given an [external backup ID \[Page 133\]](#).

The backup ID consists of the **type of backup** (complete or incremental [data backup \[Page 42\]](#) or [log backup \[Page 43\]](#)) and a **sequential number**.

Full and incremental data backups are numbered sequentially together. Log backups are numbered in a separate sequence.

After the backup, the backup ID is written to the log `dbm.knl`. You can display all the contents of this file in the [backup history \[Page 129\]](#).

In a restore, the ID is shown first and must be confirmed before the operation can start.



If the backup medium is a tape or a cassette, you should write the backup ID on the sticker of the tape or cassette at the end of the backup.

Formal Description of the Backup IDs

```
<backup_id> ::= <save_type>_<sequence_no>
```

```
<save_type> ::= DAT | PAG | LOG
```

```
<sequence_no> ::= <number>
```



Backup Medium

A backup medium is assigned to each backup you perform. Backup media include files, tapes, autoloaders, and pipes.

Backup media can be defined before or during the backup process and they are defined separately for each individual [database instance \[Page 13\]](#). You can define [single backup media \[Page 138\]](#) or combine media to form a group of [parallel backup media \[Page 134\]](#).

Recommended Backup Media

- The recommended backup media for [data backups \[Page 42\]](#) are tape and autoloader. Data may also be backed up to pipes or files.
- [Log backups \[Page 43\]](#) are always made to files ([version files \[Page 140\]](#)).

For security reasons, archive your files or versions files.

You should only use pipes if you are working with an [external backup tool \[Page 45\]](#).



In general, only archive files to a separate medium that can be stored in another location, where it is protected from fire and similar hazards.

You must follow the manufacturer's recommendations on how frequently backup media should be used.

Naming

When names are assigned, we differentiate between [standard backup media \[Page 135\]](#) and [backup media for external backup tools \[Page 135\]](#).



Checking the Database Structures

You have two options for checking the database structures:

- Checking the database structures for all tables and indexes (CHECK DATA)
- Checking the database structures for selected tables (CHECK TABLE)



Checking a database structure takes a long time, which is why you must only schedule the check at times when there is a low database load (on a weekend, for example).

Check the database structure as infrequently as possible, however, at least once during a backup history. Also make sure that you remove a backup from the backup history only if a database structure check was performed after the backup was made.

Checking the Database Structures for All Tables and Indexes (CHECK DATA)

You can perform this database structure check in the [operational state \[Page 136\]](#) `ADMIN` or `ONLINE`. The [B* trees \[Page 98\]](#) of the tables and indexes are checked. In the case of the database instance type `liveCache`, the corresponding `liveCache` structures are also checked.

If a data page flagged as `BAD` is found more than once when the database instance is `ONLINE`, then the corresponding B* tree or `liveCache` structure is also flagged as `BAD`. You can clean up this situation by checking the database structure.

If you check the database structure in the operational state `ADMIN`, any pages that are no longer referenced are taken out of the [Converter \[Page 19\]](#).

Procedure

Database Manager: SAP DB 7.4, section [Checking the Database Structure \[See SAP DB Library\]](#)

Database Administration in CCMS: SAP DB OLTP, section *Scheduling a Database Structure Check*

Database Administration in CCMS: liveCache, section *Checking the Database Structures*

Checking the Database Structures for Selected Tables (CHECK TABLE)

You can make this database structure check in the operational state `ONLINE` only. You can check individual tables. If a data page flagged as `BAD` is found when the database instance is `ONLINE` (and hence a B* tree flagged as `BAD`), then you can clean up this situation by checking the database consistency.

Procedure

- Checking a table
Database Administration in CCMS: SAP DB OLTP, section Tables/Views
- Checking all tables
Database Administration in CCMS: SAP DB OLTP, section Scheduling a Database Structure Check



COMMIT

In a COMMIT, all the changes made by a [transaction \[Page 139\]](#) or a [subtransaction \[See SAP DB Library\]](#) on the [database instance \[Page 13\]](#) are recorded.

- Changes closed with a COMMIT can no longer be reversed with a [ROLLBACK \[Page 137\]](#).
- As a result of a COMMIT, a new transaction is implicitly opened.

In normal database operation, the database system performs the required COMMIT actions independently. However, COMMIT can also be explicitly requested using appropriate SQL statements.



In a [restart \[Page 136\]](#), the system checks which transactions were closed with a COMMIT. These actions are redone. Transactions not yet closed with a COMMIT are undone.



Data Area

The total of all [data volumes \[Page 21\]](#) is called the data area.

In the data area, there must always be sufficient space to hold all the data that arises during database operation.



Database Catalog

The database catalog of a [database instance \[Page 13\]](#) consists of **metadata** in which definitions of database objects such as Basis tables, view tables, synonyms, value ranges, indexes, users, and user groups are stored.

The database catalog is made up of a number of sections. One section consists of the information about the installation of the database system and the metadata with the definitions of users and user groups. This section is not assigned to any user or user group.

For each user or user group, the catalog contains a section in which the metadata of the objects generated by this user or user group is stored. This includes metadata on Basis tables, view tables, and so on.

A user can only access the metadata of another user or another user group if authorized to do so.



Database Console

The Database Console (program X_CONS) is a tool that gives you a quick overview of the operating system resources being used by the database system, the distribution of the [database sessions \[Page 132\]](#) among the operating system threads, and the status of the active database sessions.

- Windows
The [console thread \[Page 17\]](#) processes requests from the Database Console. To do this, the X_CONS program communicates with the console thread.
- UNIX
The database console receives some of the information it requires directly from the shared memory area of the database kernel, and some of the information from the communication between X_CONS and the console thread.

Database Console Commands

You have the following options for using the Database Console:

- Starting the X_CONS program from the operating system level:
`x_cons <database_name> <command> [<time>] [<count>]`
For an overview of all possible commands, use the command:
`x_cons <database_name> help`
- If you are using an SAP system, you can call the database console from the CCMS. For details on this, see the following documentation:
Database Administration in CCMS: SAP DB OLTP, section *Database Console*
Database Administration in CCMS: liveCache, section *Console*



Database Name

In principle, the name of a [database instance \[Page 13\]](#) is freely definable, but must meet the following conditions:

- The maximum length of a database name is 8 characters.
- The database name can contain the characters from the 7-bit ASCII character set.
- The database name can start with a letter or a number.
- The database system does not distinguish between uppercase and lowercase characters.



Database Session

When you create a user, a password is assigned to it. In order to work with a [database instance \[Page 13\]](#), users must enter a user name and password that the database system recognizes.

When a user enters a correct user name and password, they are granted access to the database instance. This opens a database session and automatically calls up the initial transaction.

You can only work with the database instance within a database session. Users must end each session explicitly.

If the user does not belong to a user group, the “Current User” is the user name they entered to gain access to the database instance. If, on the other hand, the user does belong to a user group, the “Current User” is name of the user group.

See also:

[User Concept \[Page 25\]](#)

Database Trace

You can activate a database trace. When you do this, the [trace writer task \[Page 15\]](#) logs all reactions of the database kernel to database statements in the log `knltrace`. This also means that the database trace can be used not only to trace errors that occur when statements are processed but also to provide a more exact classification of inconsistencies caused, for example, by hardware errors.



Only activate a database trace if you are asked to do so by the [SAP DB Support \[Page 144\]](#) team. We do not recommend that you activate the database trace function unless it is really necessary, because it seriously affects the performance of the database instance.

While the database trace is activated, run the [database instance \[Page 13\]](#) with the smallest load possible, and only perform those actions needed to reproduce the error.

Deactivate the database trace as soon as the actions needed for the analysis have been logged.

Procedure

Database Manager GUI: SAP DB 7.4, section [Activating the Database Trace Function \[See SAP DB Library\]](#)

Database Manager CLI: SAP DB 7.4, section [Database Trace Functions \[See SAP DB Library\]](#)



DBM Operator

See [Database Manager Operator \(DBM Operator\) \[Page 26\]](#)



External Backup ID

Every backup that is created on a [backup medium \[Page 129\]](#) using an external backup tool has a unique external backup ID. This ID is generated by the [DBM Server \[Page 60\]](#) or the backup tool during the backup. If a backup was created using a media group, every part of this backup has its own external backup ID.

The external backup ID is written to logs `dbm.knl` ([backup history \[Page 129\]](#)) and `dbm.ebf` during backup.

The external backup ID is used during a restore to request the desired backup. The external backup ID is displayed before the restore operation, and can be corrected if necessary. To do this, the [Database Manager \[Page 61\]](#) transfers the external backup ID to the DBM Server as a command parameter, and the DBM server forwards this to the external backup tool.



External Backup Medium

See [Name of an external backup mediums \[Page 135\]](#)



Group of Parallel Backup Media

You can carry out a parallel [backup \[Page 41\]](#) to several [backup media \[Page 129\]](#) and recover data from parallel backup media. For this purpose you define a group of parallel media that is identified by a single, grouped media name.

Parallel backup media are simultaneously written or read by the database instance. This increases data throughput - and thus the speed of backup or restore.



Instance Type

See [Database instance type \[Page 22\]](#)



Kernel

The [threads \[Page 13\]](#) of a [database instance \[Page 13\]](#) are often referred to as the kernel.



Language Support (MapChar Sets)

A Mapchar set is a character set that maps language-specific characters to the [ASCII code \[See SAP DB Library\]](#), [EBCDIC code \[See SAP DB Library\]](#), or [UNICODE \[Page 88\]](#). This allows you to convert characters. MapChar sets enable country-specific letters to be mapped to one or two non-country-specific letters (for example, representation of "ü" as "ue").

Mapchar sets are only of significance for internal system functions. For example, the Mapchar set is used by the [MAPCHAR\(x,n,i\) \[See SAP DB Library\]](#) SQL function to sort fields containing umlauts.

- When the database instance is configured, a conversion of these country-specific letters is implicitly allowed and stored under the Mapchar set name DEFAULTMAP (defined as ASCII).
- If a Mapchar set is defined as UNICODE, then it can be used for UNICODE values only.
- A Mapchar set that is not defined as UNICODE cannot be used for UNICODE values.

You can change the Mapchar set DEFAULTMAP, and also define other Mapchar sets.

You can use the Database Manager to create, display, change, or delete Mapchar sets. Proceed as described in the documentation *Database Manager GUI: SAP DB 7.4*, sections [Creating Mapchar Sets \[See SAP DB Library\]](#), [Displaying, Changing and Deleting Mapchar Sets \[See SAP DB Library\]](#).



Log Area

The volume storage area needed for [redo log management \[Page 47\]](#) is called the log area. A log area can extend across several [log volumes \[Page 22\]](#).

The log area is managed by the [log writer \[Page 48\]](#). The log writer fills the log area with [log pages \[Page 48\]](#) from the [log queue \[Page 47\]](#).



For security, the log area should always be mirrored. If possible, you should mirror the log area on a hardware basis.

If hardware-based mirroring is not possible, you can use an appropriate [log mode \[Page 38\]](#) setting.

In a production database system, make sure that redo log management is always activated, and that the log area cannot be overwritten until it has been backed up ([Log Settings \[Page 37\]](#)).

A full log causes the database to go down. Therefore, you must always carry out [log backups \[Page 43\]](#) in good time. For this purpose, make sure you activate [automatic log backup \[Page 43\]](#).

In log backups, the log area is not saved as a whole. Instead, it is saved in backup units, called log segments. The size of a log segment is defined by the parameter [LOG_SEGMENT_SIZE \[Page 82\]](#).



Name of a Standard Backup Medium

When you define a [backup medium \[Page 129\]](#), you usually choose a name that relates to the practical situation, and that specifies the path and properties of the tape device or tape, the autoloader, or the files.

In contrast, different naming conventions apply for the external backup medium if you use [external backup tools \[Page 45\]](#).

See also [Name of an External Backup Medium \[Page 135\]](#)



Name of External Backup Medium

If you are using [external backup tools \[Page 45\]](#), there are naming conventions that you should follow for the [backup media \[Page 129\]](#).

The following character strings at the beginning of a backup media name determine that external backup tools are used for backups to this medium or restores from this medium:

| Character String | Backup Tool to Be Used |
|------------------|------------------------|
| ADSM | ADSM/TSM (IBM/Tivoli) |
| BACK | Backint for Oracle |

| | |
|------------|--------------------|
| | Backint for SAP DB |
| NSR | NetWorker (Legato) |

If the [Database Manager \[Page 61\]](#) is not to address any of the backup tools listed above, use backup media names that do not begin with any of the above reserved character strings.

See:

[Name of standard backup medium \[Page 135\]](#)

Operational State

The following operational states exist for a [database instance \[Page 13\]](#):

- **ONLINE:** The database instance has been started, and users can log on.
- **ADMIN:** The database instance is only available to administrators for administrative work.
- **OFFLINE:** The database instance is not running.

Page

A page is a basic physical unit of disk input and output, that is, a page corresponds to a certain number of blocks. Storage and disk space is often measured in pages.

For the SAP DB database system, the page size is fixed at 8 KB.

Page Pool

The page pool is used to manage the pages of the [I/O buffer cache \[Page 19\]](#). The pages of the I/O buffer cache are dynamically assigned to the [Converter \[Page 19\]](#) and the [data cache \[Page 20\]](#) using the information in the page pool.

Restart

When the database system is restarted, all the available [redo log entries \[Page 46\]](#) from the [log area \[Page 135\]](#) are reproduced, and if necessary, [undo log entries \[Page 46\]](#) are evaluated in order to roll back [transactions \[Page 139\]](#).

The system starts with the status of the database system that was recorded with the last [savepoint \[Page 137\]](#).

1. Using the [transaction file \[Page 139\]](#), the [log reader \[Page 52\]](#) regenerates the [transaction list \[Page 139\]](#).
2. All the transactions that were still open at the time of the savepoint are determined using the transaction list.
3. The transactions are processed in the way described in [Restart or Recovery \[Page 51\]](#).

See also:

[Restartability \[Page 45\]](#)

[Example: Restart \[Page 54\]](#)

ROLLBACK

In a ROLLBACK, all the changes made by a [transaction \[Page 139\]](#) or a [subtransaction \[See SAP DB Library\]](#) on the [database instance \[Page 13\]](#) are reversed.

- Changes closed with a [COMMIT \[Page 131\]](#) can no longer be reversed with a ROLLBACK.
- As a result of a ROLLBACK, a new transaction is implicitly opened.

In normal database operation, the database system performs the required ROLLBACK actions independently. However, ROLLBACK can also be explicitly requested using appropriate SQL statements.



In a [restart \[Page 136\]](#), the system checks which transactions were canceled or closed with a ROLLBACK. The actions of these transactions are undone.

Run Directory

The run directory is created when a new [database instance \[Page 13\]](#) is installed by the database system (database parameter [RUNDIRECTORY \[Page 85\]](#)). The run directory is used to store log files in which all actions and errors are logged. All the relative path names relate to the run directory of the database instance.

See also:

[SAP DB Directories \[Page 74\]](#)

Savepoint

The system carries out a backup action at regular intervals for the most important database information. This backup action is known as a savepoint.

It writes all changed [pages \[Page 136\]](#) held in the [data cache \[Page 20\]](#) to the [data area \[Page 131\]](#).

- The data cache contains, among other things, the [undo log files \[Page 49\]](#). As these are also permanently backed up to the data area by a savepoint, the [database instance \[Page 13\]](#) can be restored to a transaction-consistent state in a [restart \[Page 136\]](#).
- At the time of a savepoint, the data of all open [transactions \[Page 139\]](#) is written to the [transaction file \[Page 139\]](#).

See also:

[Savepoint on Restart \[Page 54\]](#)



Single Backup Medium

You have the option of defining single [backup media \[Page 129\]](#). The database system performs its [backup \[Page 41\]](#) to this medium. Where the capacity of the backup medium is too small for the backup that has started, the system asks for a succeeding medium.



SQL Mode

The SAP DB database system is capable of executing database applications that are written in accordance with one of the following definitions:

- INTERNAL: internal database system definition
- ANSI: ANSI standard in accordance with ANSI X3.135-1992, entry SQL
- DB2: Definition DB2 Version 4
- ORACLE: Definition ORACLE7

When you log on to the database system, you can select one of the definitions listed above (see [CONNECT Statement \[See SAP DB Library\]](#)). The default SQL mode is INTERNAL.

The database system is capable of checking new applications to ascertain whether they correspond to one of the above definitions. In particular, this means that no extensions that go beyond the selected definition are regarded as correct.

Support of other SQL modes is restricted with regard to DDL statements.



[SQL Mode ORACLE: SAP DB 7.4 \[See SAP DB Library\]](#)



Task

A task is executed by the [database instance \[Page 13\]](#).

The database tasks are bundled in [user kernel threads \(UKT\) \[Page 14\]](#).

There is a range of [task states \[Page 16\]](#).



Timeout Value

The timeout value defines the maximum period of inactivity during a [database session \[Page 132\]](#). The inactivity period is the time interval between the completion of an SQL statement and the next SQL statement. The database session is terminated with a ROLLBACK WORK RELEASE when the specified maximum inactivity period is exceeded.

Timeout values are specified in seconds.

Setting the Timeout Value

You set the timeout value with a database parameter and, if necessary, SQL statements.

- The default timeout value for all users is set with the database parameter [SESSION_TIMEOUT \[Page 86\]](#).

- You can use a [CREATE USER statement \[See SAP DB Library\]](#) or an [ALTER USER statement \[See SAP DB Library\]](#) to define a user-specific timeout value.
- You can use a [CREATE USERGROUP statement \[See SAP DB Library\]](#) or an [ALTER USERGROUP statement \[See SAP DB Library\]](#) to define a timeout value for a group of users. If a user belongs to a user group that has a timeout value, then this value is also the user-specific timeout value.
- When a user connects to the database instance, the current timeout value is determined:
If the user-specific timeout value is larger than the timeout value defined with SESSION_TIMEOUT, or if there is no user-specific timeout value, then the current timeout value is the one defined in SESSION_TIMEOUT. Otherwise, the current timeout value is the user-specific timeout value.
- You can specify a timeout value when you connect to the database instance ([CONNECT statement \[See SAP DB Library\]](#)). This timeout value must be less than or equal to the current timeout value.
If you do not specify a timeout value, then the current timeout value that the system has determined is valid.



Transaction

A transaction is a series of database operations that form a unit with regard to data backup and synchronization.

Transactions are closed with [COMMIT \[Page 131\]](#) or [ROLLBACK \[Page 137\]](#).

See also:

Reference Manual: SAP DB 7.4, Section [Transaction \[See SAP DB Library\]](#)

[Lock behavior \[Page 109\]](#)

[Transaction file \[Page 139\]](#)

[Transaction list \[Page 139\]](#)



Transaction File

The transaction file is an internal database file to which the data of all open [transactions \[Page 139\]](#) is written at the time of a [savepoint \[Page 137\]](#). The transaction file is stored in the [data area \[Page 131\]](#).

During a [restart \[Page 136\]](#), the [log reader \[Page 52\]](#) uses this transaction file to regenerate the [transaction list \[Page 139\]](#).



Transaction List

The transaction list is an internal database list of all open modifying transactions. The transaction list is located in the main memory.

All modifying transactions are globally available in the database via this transaction list. The transaction list contains a transaction entry for each [transaction \[Page 139\]](#). This transaction entry contains references to any [redo \[Page 46\]](#) and [undo log entries \[Page 46\]](#) that may exist.

During a [restart \[Page 136\]](#), the [log reader \[Page 52\]](#) uses the [transaction file \[Page 139\]](#) to regenerate the transaction list.



Version File

If, during a [log backup \[Page 43\]](#), you use a [backup medium \[Page 129\]](#) that refers to a file, the content of the [log area \[Page 135\]](#) is copied to files known as “version files.” One version file is generated for each log segment.

The names of the generated version files consist of the version file name entered during definition of the backup medium, and a number that the database system assigns in sequence during backup of the log segments.

Archive the version files as described in [Backup Strategy \[Page 40\]](#).



Variables

The table below shows which variables are used.

| | |
|----------------------------|---|
| <arch> | Name of the operating system architecture in the path specifications |
| <build> | Build number of the SAP DB software |
| <database_name> | Name of the database instance [Page 13] |
| <database_server> | Name of the host on which the database instance is installed |
| <dba_user> | Name of the DBA user [Page 29] |
| <dbm_user> | Name of the DBM operator [Page 133] |
| <dependent_path> | Location of the server software that is dependent on the database version This path must be unique. Many different directories with different versions can exist alongside each other. Any programs in these version-specific directories are not executed directly by the user; instead, they are called by programs located in <independent_program_path>. For an RPM-based installation (Linux), this directory is defined as follows: /opt/sapdb/depend74 |
| <domain_user> | Name of the DOMAIN user [Page 30] |
| <group> | UNIX/Linux: Name of the group of the SAP DB software |
| <independent_data_path> | Location of the data, configuration, and run directories of SAP DB database instances and SAP DB applications For an RPM-based installation (Linux), this directory is defined as follows: /var/opt/sapdb/indep_data |
| <independent_program_path> | Location of the programs and libraries shared by the SAP DB database instances and SAP DB applications Choose a directory that has a large enough memory for any future enhancements of the client software. For an RPM-based installation (Linux), this directory is |

| | |
|---------------------|---|
| | defined as follows: /opt/sapdb/indep_prog |
| <inst_profile> | Name of the installation profile [See SAP DB Library] |
| <os> | Name of the operating system in the path specifications |
| <os_user> | Name of the operating system user |
| <owner> | UNIX/Linux: Name of the owner of the SAP DB software |
| <package_directory> | Directory where the software package is located after it has been installed |
| <package_name> | Logical name of the software package [See SAP DB Library] |
| <password> | Password |
| <path> | Path specification |
| <program_path> | Standard Windows directory for the user software (for example, C:\Program Files) |
| <server_name> | Name of the host server |
| <sysdba_user> | Name of the SYSDBA user [Page 29] |
| <user_name> | Name of the user |
| <version> | Version number in accordance with the four-character SAP DB Version Notation [Page 144] |
| <web_server> | Name of the host on which the Web server [Page 61] is installed |



SAP DB Documentation

Basic Information

| Title | Explanation |
|---|--|
| The SAP DB Database System [Page 1] | Introduction to the architecture of the SAP DB database system, user and backup concepts, database tools, directory structure, database parameters, data management and processing, documentation, terminology, and so on. |
| Reference Manual: SAP DB 7.4 [See SAP DB Library] | SQL statements and their syntax |
| SQL Mode ORACLE: SAP DB 7.4 [See SAP DB Library] | Special features of SQL statements in the SQL mode ORACLE |
| System Tables: SAP DB 7.4 [See SAP DB Library] | SAP DB system tables and their evaluation |
| Optimizer: SAP DB 7.4 [See SAP DB Library] | Functions of the SAP DB Optimizer |
| Messages: SAP DB 7.4 [See SAP DB Library] | List of all SAP DB messages |
| SAP High Availability [See SAP DB Library] | Conditions for guaranteeing the high availability of |

| | |
|--|---|
| | an SAP system For SAP customers only |
|--|---|

Tools

| Title | Explanation |
|---|--|
| Database Manager <ul style="list-style-type: none"> • Database Manager GUI: SAP DB 7.4 [See SAP DB Library] • Web DBM: SAP DB 7.4 [See SAP DB Library] • Database Manager CLI: SAP DB 7.4 [See SAP DB Library] | Creating and managing database instances <ul style="list-style-type: none"> • Graphical user interface • Web-based user interface • Command line user interface |
| External Backup Tools: SAP DB 7.4 [See SAP DB Library] | Configuration and use of ADSM/TSM, Backint for Oracle, Backint for SAP DB and NetWorker |
| Query Tools <ul style="list-style-type: none"> • SQL Studio: SAP DB 7.4 [See SAP DB Library] • Web SQL Studio: SAP DB 7.4 [See SAP DB Library] | Data queries and data processing <ul style="list-style-type: none"> • Graphical query tool • Web-based query tool |
| Loader: SAP DB 7.4 [See SAP DB Library] | Loading and unloading data |
| Database Analyzer: SAP DB [See SAP DB Library] | Analysis of the performance of database instances |

Installation Documentation

| Title | Explanation |
|---|---|
| Database Software Installation Guide: SAP DB 7.4 [See SAP DB Library] | Standard installation, update, and uninstallation of all SAP DB software |
| Web Tools Installation Guide: SAP DB 7.4 [See SAP DB Library] | Installation of SAP DB Web Tools |
| Installation Manual: SAP DB [See SAP DB Library] | Installation, update and uninstallation of installation profiles and individual SAP DB software components, updates of database instances, including software |
| <i>SAP Web Application Server Installation on UNIX: SAP DB</i> | Installation of an SAP system with a SAP DB database on UNIX Available only to SAP customers on the SAP Service Marketplace |
| <i>SAP Web Application Server Installation on Windows NT: SAP DB</i> | Installation of an SAP system with a SAP DB database on Microsoft Windows Available only to SAP customers on the SAP Service Marketplace |
| <i>SAP Web Application Server 630: Homogeneous and Heterogeneous System Copy</i> | Copy of the SAP system Available only to SAP customers on the SAP Service Marketplace |

Interfaces

| Title | Explanation |
|---|---|
| C/C++ Precompiler User Manual: SAP DB | Options for the C/C++ Precompiler, Embedded |

| | |
|--|---|
| 7.4 [See SAP DB Library] | SQL |
| ODBC Manual: SAP DB [See SAP DB Library] | Basics and special features of the SAP DB ODBC driver |

Development

| Title | Explanation |
|--|--|
| Development Environment: SAP DB [See SAP DB Library] | Usage of the development environment |
| Source Text for Tools: SAP DB [See SAP DB Library] | Creating source texts |
| Web Based Problem Tracking System: SAP DB [See SAP DB Library] | Web interface for internal SAP DB program PTS (Problem Tracking System) for documenting problem messages about SAP DB software |

Database Administration in CCMS

(Required by SAP customers only)

| Title | Explanation |
|---|--|
| Database Administration in CCMS: SAP DB OLTP [See SAP DB Library] | Using the SAP system to analyze and monitor a SAP DB database instance, scheduling actions, alert monitoring |
| Database Administration in CCMS: liveCache [See SAP DB Library] | Using the SAP system to analyze a liveCache database instance, alert monitoring |

SAP DB Documentation

- For current SAP DB documentation and other important information, see the SAP DB Homepage at <http://www.sapdb.org>:
Choose *Documentation* → *SAP DB Online Library* or *Download*.
- For SAP DB documentation for SAP systems, see the Help Portal at <http://help.sap.com>:
Choose *SAP NetWeaver* → *SAP Web Application Server* → *SAP Web Application Server <release>* → *<language>* → *SAP NetWeaver Components* → *SAP DB*.
- For liveCache documentation for SAP systems, see the Help Portal at <http://help.sap.com>:
Choose *SAP NetWeaver* → *SAP Web Application Server* → *SAP Web Application Server <release>* → *<language>* → *SAP NetWeaver Components* → *liveCache*.
- For high availability documentation for SAP systems, see the Help Portal at <http://help.sap.com>:
Choose *SAP NetWeaver* → *SAP Web Application Server* → *SAP Web Application Server <release>* → *<language>* → *SAP NetWeaver Components* → *SAP Web Application Server* → *Computing Center Management System* → *SAP High Availability*.
- For installation guides for SAP solutions, see the SAP Service Marketplace at <http://service.sap.com> (available to SAP customers only):
Choose *Installation/Upgrade Guides*.
Choose *SAP NetWeaver in Detail* → *Solution Life-Cycle Management* → *Installation* → *Installation and Upgrade Guides* → *SAP Web Application Server*.



SAP DB Software

You can obtain the SAP DB software free of charge:

- SAP DB-Homepage: <http://www.sapdb.org>
You can download the latest SAP DB version from the SAP DB homepage.
- Software CD: material number 50056472
You can order this CD in the SAP Shop: <http://www.sap.com/company/shop>
- SAP installation package
SAP customers receive the SAP installation package. This installation package also includes the SAP DB software.

You can also obtain the required installation descriptions free of charge: see [SAP DB Documentation \[Page 141\]](#)



SAP DB Version Notation

SAP DB versions are specified according to the following version notation:

`<major>.<minor>.<cl>.<build>`

| | |
|-------|---------------------------------|
| major | n – one-digit version number |
| minor | n – one-digit version number |
| cl | nn – two-digit correction level |
| build | nn – two-digit build level |



7.4.03.30

SAP DB Version 7.4, correction level 3, build number 30

In a technical context (such as the SAP Service Marketplace), you can also see the following notation:

`<major>.<minor>.<cl> Build <build>`

| | |
|-------|-------------------------------|
| build | nnn – three-digit build level |
|-------|-------------------------------|



7.4.03 Build 030

SAP DB Version 7.4, correction level 3, build number 30



SAP DB Support

There are two ways of obtaining support for your SAP DB database system:

Online Support

Online support is based on mutual help between SAP DB users. To obtain online support, become a member of the *sapdb.general@listserv.sap.com* mailing list. You can ask questions and describe problems by sending an email to this mailing list. Both the SAP DB development team and members of the Open Source community can then reply to your emails. Online support does not guarantee fixed response times. However, the SAP DB development team continually monitors the correspondence in the mailing list and ensures that every problem is responded to.

This support is free of charge.

Premium Support

You conclude a Premium Support Contract with SAP, and then have access to the complete SAP support infrastructure. For communication purposes, you establish a Customer Competence Center with a certain number of contacts.

The fees for Premium Support are EUR 60,000 per year for each Customer Competence Center. If you have already concluded a SAP DB maintenance contract, the fees for your support contract are reduced.



The SAP DB homepage contains further information about support (http://www.sapdb.org/sap_db_support.htm) and how to become a member of the mailing list (http://www.sapdb.org/sap_db_contact.htm).