SAP® MaxDB™ Expert Session

SAP® MaxDB™: Introduction into MaxDB Database Architecture
Christiane Hienger April 16, 2013

Public

SAP

SAP® MaxDB™ – Expert Session

Introduction into SAP ® MaxDB™ Database Architecture

Christiane Hienger
Heike Gursch
IMS MaxDB/liveCache Development Support
April 16, 2013

SAP

Agenda

Data Storage (Disk Area)
- Software Directories
- Data-, Log-Volumes

Database Kernel
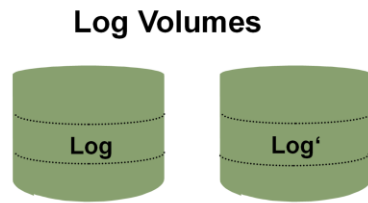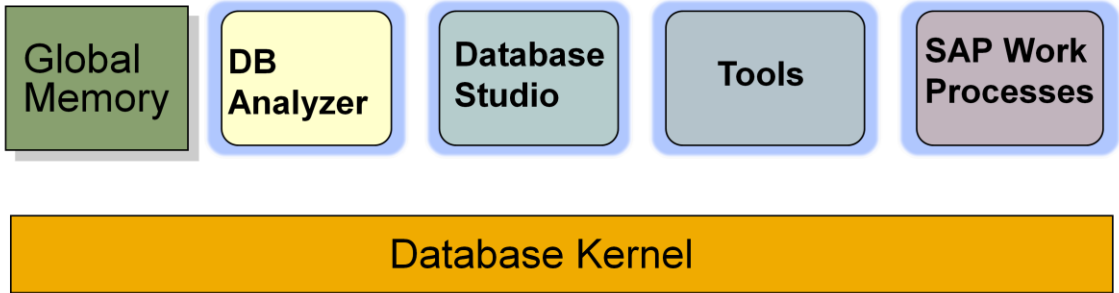- SQL Manager
- Data Access Manager

Process Layer
- Threads, User Kernel Threads,
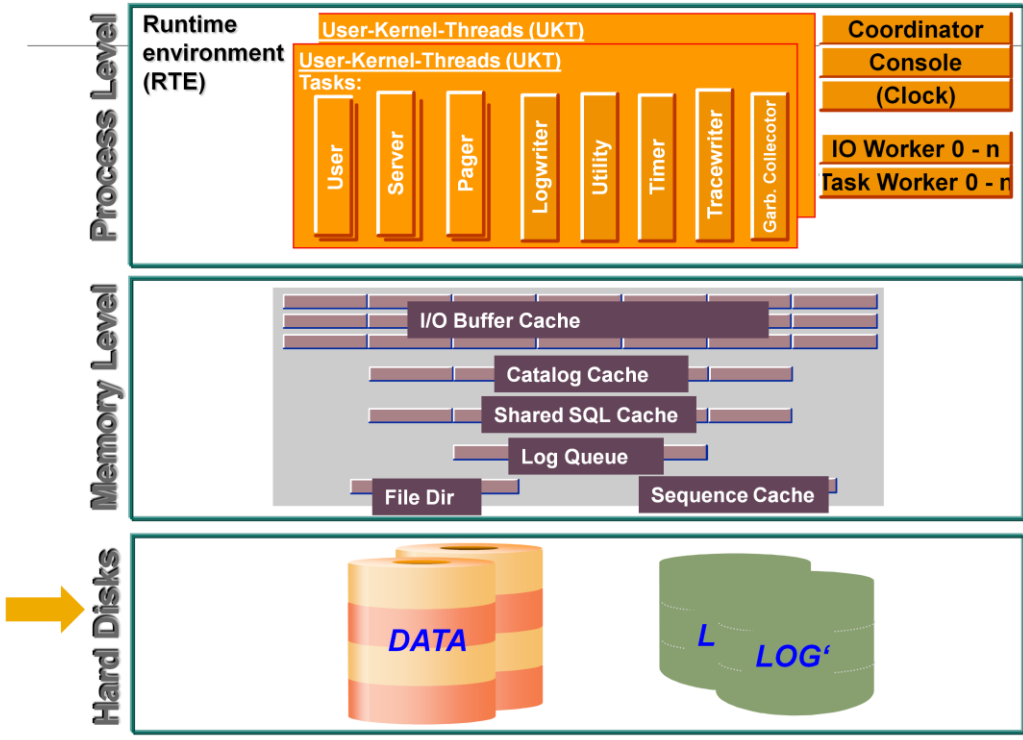- Pager, Timer; Garbage Collectors, LogWriter, TraceWriter
- Remote SQL Server, DBMServer

Memory Level
- I/O Buffer Cache, Converter Cache, Catalog Cache, File Directory, Shared SQL Cache

3

Introduction - Environment

Global Memory	DB Analyzer	Database Studio	Tools	SAP Work Processes

Database Kernel

Data Volumes

Data 1

Data 2

Data 3

Log Volumes

Log

Log'

Database Instance

Process Level

Runtime environment (RTE)

User-Kernel-Threads (UKT)
User-Kernel-Threads (UKT)
Tasks:

- User
- Server
- Pager
- Logwriter
- Utility
- Timer
- Tracewriter
- Garb. Collecotor

Coordinator
Console
(Clock)

IO Worker 0 - n
Task Worker 0 - n

Memory Level

I/O Buffer Cache
Catalog Cache
Shared SQL Cache
Log Queue
File Dir
Sequence Cache

Hard Disks
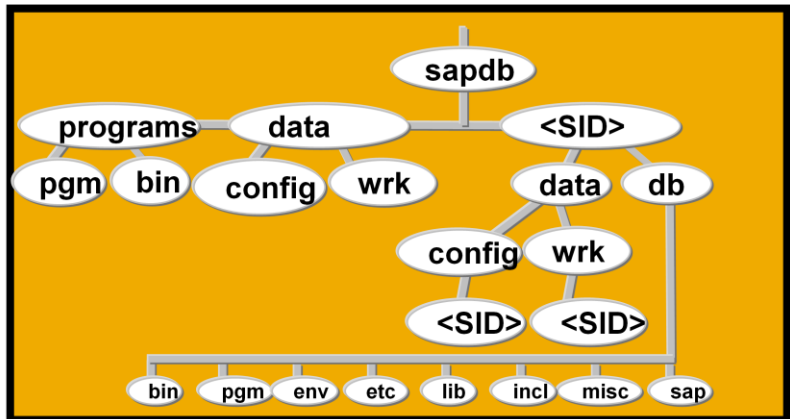
DATA

L LOG'

A database instance consists of the following three parts:

Database kernel (process level)

Caches (memory level)

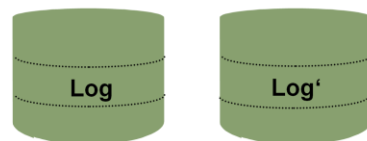Volumes (hard disk level)

Disk Areas (Software and Data)

Database Software

/sapdb

sapdb — programs — data — <SID>
programs — pgm, bin, config, wrk
data — config, wrk
<SID> — data, db
config, wrk
<SID>, <SID>
bin, pgm, env, etc, lib, incl, misc, sap

Data Volumes

Data 1
Data 2
Data 3

Log Volumes

Log
Log'

A physical disk or part of a physical disk, respectively, is denoted with the item „volume".

A database instance is installed on **three disk areas**:

- Data volumes
- Log volumes and
- Database software.

User data (tables, indexes, etc.), the SQL catalog and converter pages are stored in the **data volumes**. Data belonging to one table is evenly distributed on all data volumes by the use of an internal striping.

Changes on data are stored in form of redo log entries within the **log volumes**. This makes sure that in case of a required recovery all changes that are not contained in the recent complete data backup can be reloaded.

To guarantee secure database operation make sure to mirror the log volumes (set database parameter UseMirroredLog=YES). If the log volumes are not mirrored by the database itself, the disks have to be mirrored physically or by the operation system.

Redo log entries only contain changes of transactions, i.e. the after images. Undo log entries are administered separate from redo log entries in the data area.

Database software delivered by SAP comprises executable programs, source texts and utilities allowing to generate database processes and to work with the database instance. The software is installed within the file system in a defined directory with sub directories. Log and diagnosis files generated during database operation are also stored here.
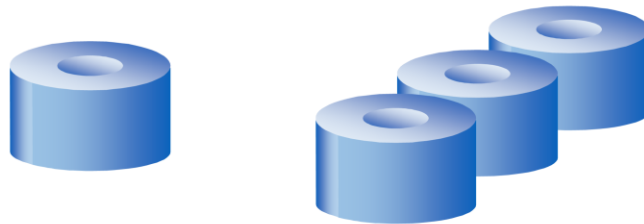
Database Kernel

Buffer Interface

SQL Manager (AK)

Buffer Interface

Data Access Manager
communication base layer (KB)

Procedural Interface

Data Access Manager
base data layer (BD)

The MaxDB kernel consists of three levels:

SQL statements sent to the database system are received by the highest level, the SQL Manager (previously known as the application communication layer).

They are then sent in compressed form to the Data Access Manager. The Data Access Manager is divided into the communication base layer (KB) and the base data (BD) layer.

The necessary individual queries are determined on the communication base layer.

The required data is then procured on the lower-level base data layer.

The data records are then sent through the layers to the querying application.

In recent years, the KB and BD layers have increasingly merged.

Since version 7.4 there has no longer been a sharp division between the two layers.

SQL Manager

SELECT name1, name2, tel1
FROM kna1
WHERE name1 = 'GARDENER'
      or name1 = 'GARDENERS'

SQL Manager

Data Access Manager

Data Access Manager

Access strategy:
1) Perform Range Scan
   on Index with ID 3422
   located on node p15714
2) Select Rows by Primary Key
   on table with ID 3421
   located on node p15709

Catalog entries for table kna1

Table kna1: ID 3421; Node p15709

Column	Type	Length
mandt	NUMCHAR	3
kunnr	NUMCHAR	8
.		
name1	CHAR	20
.		
tel1	CHAR	20
.		

Index kna1__1: ID 3422; Node p15714

Column	Type	Length
Name1	CHAR	20
Name2	CHAR	20
Prim.Key	NUMCHAR	13

Let's have a look at a relatively simple SELECT statement:

The SQL Manager receives an SQL statement that was sent as an SQL packet and first checks the syntax for correctness.

It converts the incoming SQL packets into stack code with various entries, each with a length of 8 bytes.

Catalog management, that is, the determination of information about the existing tables and their columns, also takes place on this layer.

Finally, the Optimizer in the SQL Manager decides if the data should be accessed via a secondary index.

In our simple example, we assume that the Optimizer has opted for an access strategy via the secondary index kna1_1 .

Data Access Manager (KB Layer)

SELECT name1, name2, tel1
FROM kna1
WHERE name1 = 'GARDENER'
         or name1 = 'GARDENERS'

SQL Manager

Data Access Manager

Select statement
No log Information required

Locking
No locks held on table with ID 3422

9

Transaction management and lock management are executed on the communication base layer of the Data Access Manager.

Transaction management manages information about active transactions in the database as well as the changes they make to data.

Lock management provides memory lists containing all objects of an instance that are locked at a certain point in time as well the transactions that correspond to them.

In the case of complex SQL statements (such as joins), access to individual tables and indexes is coordinated.

In our simple example, the KB layer will first access the index table kna1_1 and then, using the determined index entries, directly access the appropriate rows of table kna1.

Data Access Manager (BD layer)

```
select NAME1, NAME2, TEL1
from KNA1
where NAME1 = 'GARDENER' or NAME1 = 'GARDENERS'
```

SQL Manager

```
GARDENER...CHANCEY...00112345678
GARDENER...VINCE.....00187654321
GARDENERS..ZARAH.....00187654321
```

...Manager

Data Access Manager

The base data layer of the Data Access Manager executes accesses to data that is either still located in the data volumes or is already in the cache.

The BD layer returns the results records to the KB layer.

Database Instance

Process Level

Runtime environment (RTE)

User-Kernel-Threads (UKT)
User-Kernel-Threads (UKT)
Tasks:

- User
- Server
- Pager
- Logwriter
- Utility
- Timer
- Tracewriter
- Garb. Collecotor

Coordinator
Console
(Clock)

IO Worker 0 - n
Task Worker 0 - n

Memory Level

- I/O Buffer Cache
- Catalog Cache
- Shared SQL Cache
- Log Queue
- File Dir
- Sequence Cache

Hard Disks

DATA

L LOG'

Process Structure

Coordinator
Requestor
Console
Timer

IO Worker 0 - **n**

Task Worker 0 - **n**

User-Kernel-Thread (UKT) Tasks

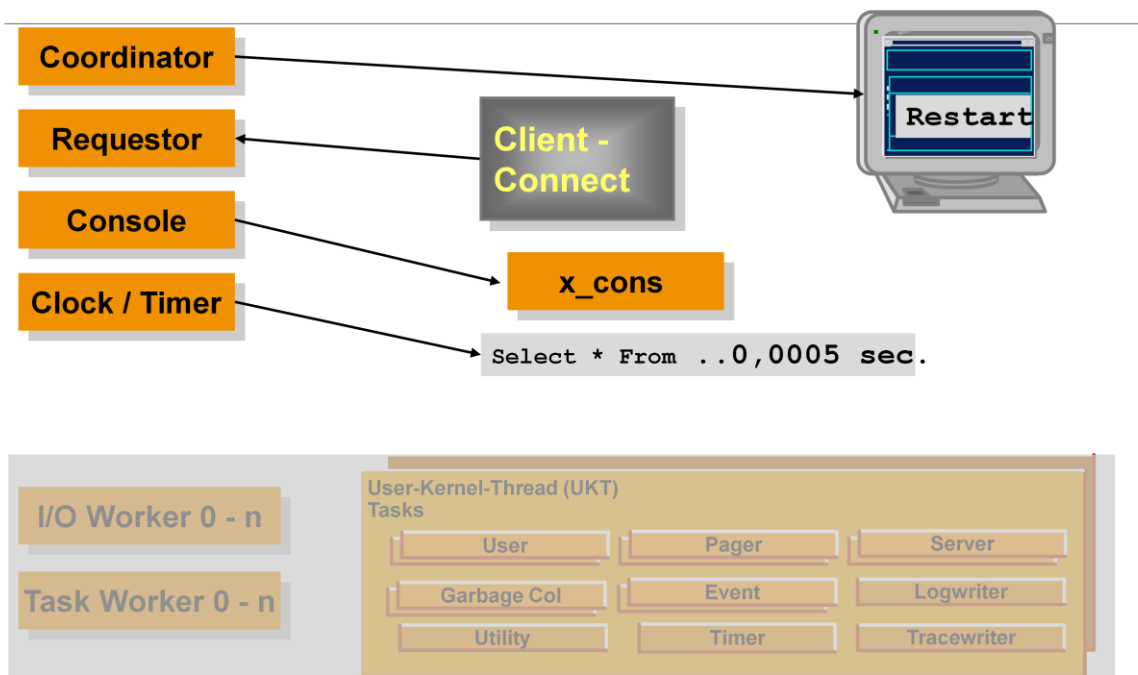| User	Pager	Server
Garbage Col	Event	Logwriter
Utility	Timer	Tracewriter

A user kernel thread forms a subset of all tasks (internal tasking).

The database kernel runs as one process divided into threads. Threads can be active in parallel on several processors within the operating system. Threads perform various tasks.

User kernel threads (UKT) consist of several tasks that perform various tasks. This tasking enables more efficient coordination of tasks than operating system tasking that employs individual threads.

The runtime environment (RTE) defines the structure of the process and the user kernel thread.

Threads I

Coordinator

Requestor

Console

Clock / Timer

Client - Connect

Restart

x_cons

`Select * From ..0,0005 sec.`

I/O Worker 0 - n	User-Kernel-Thread (UKT) Tasks		
	User	Pager	Server
Task Worker 0 - n	Garbage Col	Event	Logwriter
	Utility	Timer	Tracewriter

Public   13

When the runtime environment is started, that is, when the database instance is started in the Admin state, first the **coordinator thread** is generated. This thread is of particular importance.
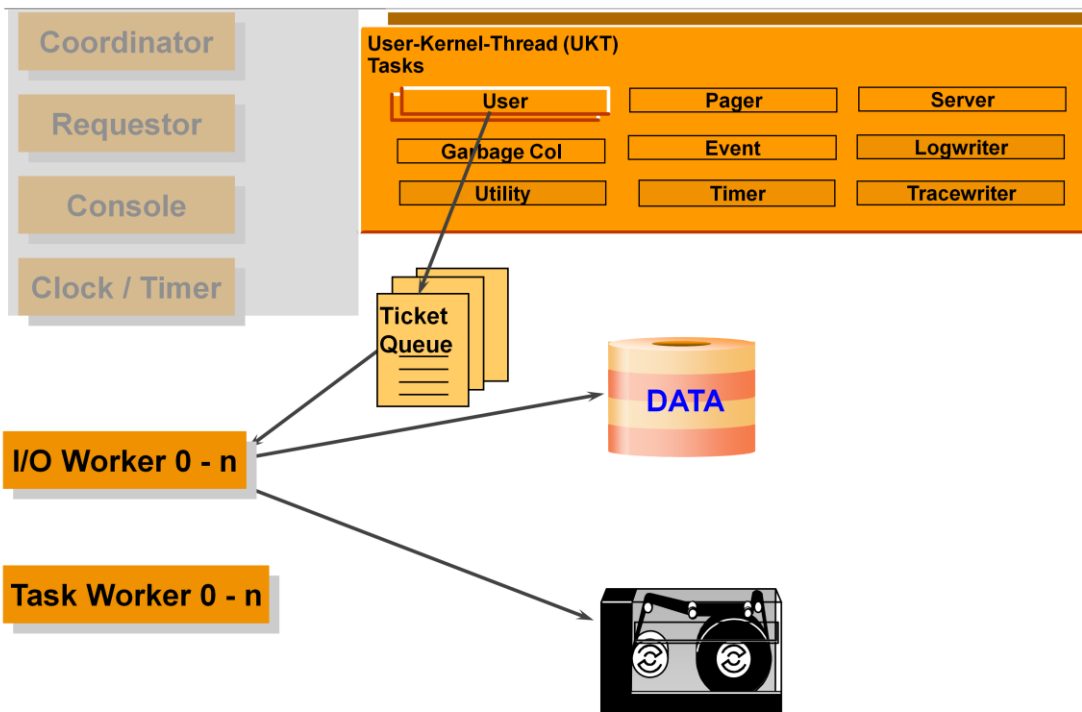
- • When started, the coordinator thread uses database parameters to determine the configuration of the memory and process of the database instance. For this reason, changes to database parameters take effect only after you have restarted the database instance.

- • The coordinator thread also coordinates the start procedures of the other threads and monitors them while the database instance is in operation.

- • If operating errors occur, the coordinator thread can stop other threads.

The **requestor thread** receives logon requests from the user processes to the database. The logon is assigned to a task within the user kernel thread.

The **console thread** collects information about the internal states of the database kernel when x_cons is being used.

The **clock thread** and the **timer thread** calculate internal times, for example to determine how much time was required to execute an SQL statement.

Threads II

	User-Kernel-Thread (UKT) Tasks		
	User	Pager	Server
	Garbage Col	Event	Logwriter
	Utility	Timer	Tracewriter

Coordinator

Requestor

Console

Clock / Timer

Ticket Queue

DATA

I/O Worker 0 - n

Task Worker 0 - n

I/O threads are responsible for processing the write and read requests to and from data and log volumes that are requested by the corresponding tasks. MaxDB supports asynchronous I/O requests.
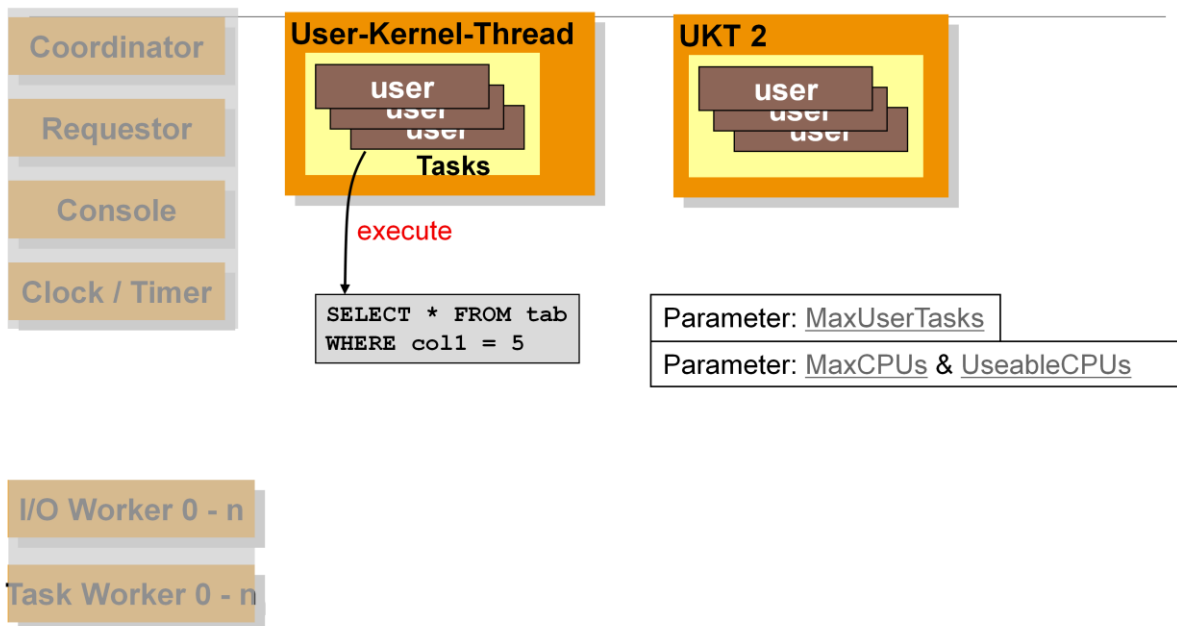
Until version 7.6 the number of **I/O threads** is primarily dependent on the number of volumes in the database instance. As a rule, two **I/O threads** are activated for each data and log volume and one for the writing of the database trace

As of version 7.7 **I/Oworker threads** are taken from a pool and activated on request; I/O is done asynchronously. After finishing the I/O requests the workers are returned to the pool.

As of verison 7.8 the user tasks utilize the asynchronous I/O for scans. The user tasks send several parallel I/O orders to the I/O systems. They don't wait until the I/O system has read every single block from disk.

The Task Worker Threads are not used for I/O requests. User tasks use task Worker threads to execute orders asynchronously e.g. In hot stand by environemnt to send the log position to the stand by node.

User Kernel Threads and Tasks

User-Kernel-Thread

user
user
user

Tasks

Coordinator

Requestor

Console

Clock / Timer

execute

```
SELECT * FROM tab
WHERE col1 = 5
```

UKT 2

user
user
user

Parameter: MaxUserTasks

Parameter: MaxCPUs & UseableCPUs

I/O Worker 0 - n

Task Worker 0 - n

Each user session is assigned exactly one **user task** at logon.
The maximum number of available user tasks is determined by the database parameter **MaxUserTasks**. This parameter also restricts the number of user sessions that can be logged on to the database system simultaneously.
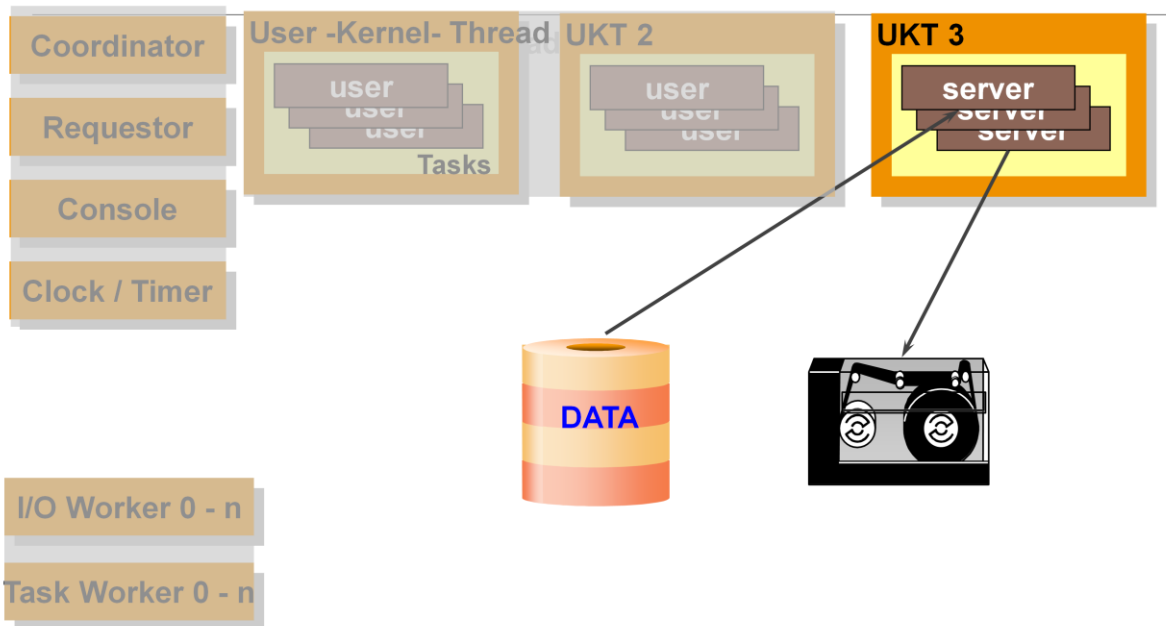The database parameter **MaxTaskStackSize** determines the memory usage of the user tasks.

The general database parameter **MaxCPUs** specifies the number of user kernel threads among which the user tasks are distributed. Other tasks and global threads use very little CPU time. The parameter **MaxCPUs** allows you to specify how many processors the database should use in parallel.

The parameter **UseableCPUs** allows an online adjustment of the number of used user kernel thread. This makes dynamic configuration changes according the available CPUs in the system possible.

As of version 7.4.03, user tasks can switch from one UKT to another if the previously-responsible UTK is overburdened. This results in better scaling for multiprocessor servers (SMP). To use this function, set the parameter **LoadBalancingCheckInterval** to a value greater than 0.

As of version 7.8 Load Balancing is released for MaxDB and liveCache instances and used by default. The scheduler immediately moves the task to an idle user kernel thread if the current thread is overloaded.

15

Server Tasks

Coordinator	User -Kernel- Thread UKT 2	UKT 3
Requestor	user / user / user	server / server / server
Console	Tasks	
Clock / Timer		

DATA

I/O Worker 0 - n

Task Worker 0 - n

Server tasks are primarily used to back up data. Some server tasks read from the data volumes; others write to the backup medium.
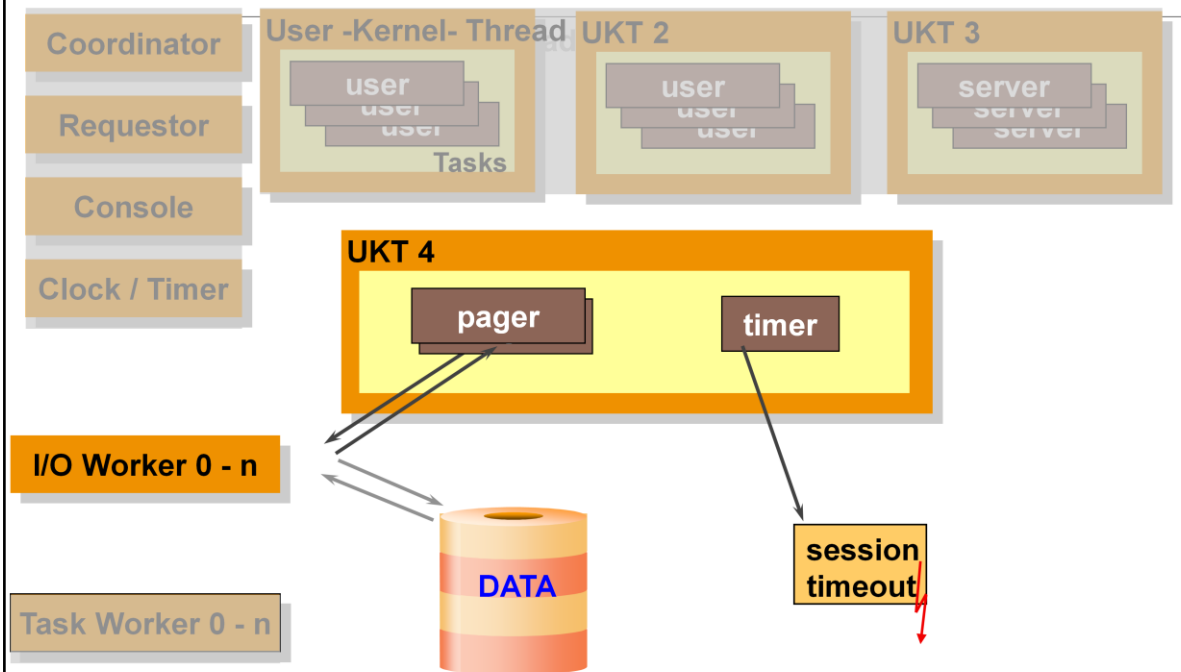
The CREATE INDEX statement instructs the server tasks to read the table data in parallel from the data volumes.

The system automatically calculates the number of server tasks needed in the configuration of the database instance from the number of data volumes.

As of version 7.6 the server tasks in certain cases are named reflecting their assigned task.

More information: Note 1672994 Server Task Configuration

Pager and Timer Tasks

Coordinator	User -Kernel- Thread UKT 2	UKT 3

Coordinator

Requestor

Console

Clock / Timer

User -Kernel- Thread UKT 2

user
user
user

Tasks

UKT 2

user
user
user

UKT 3

server
server
server

UKT 4

pager   timer

I/O Worker 0 - n

DATA

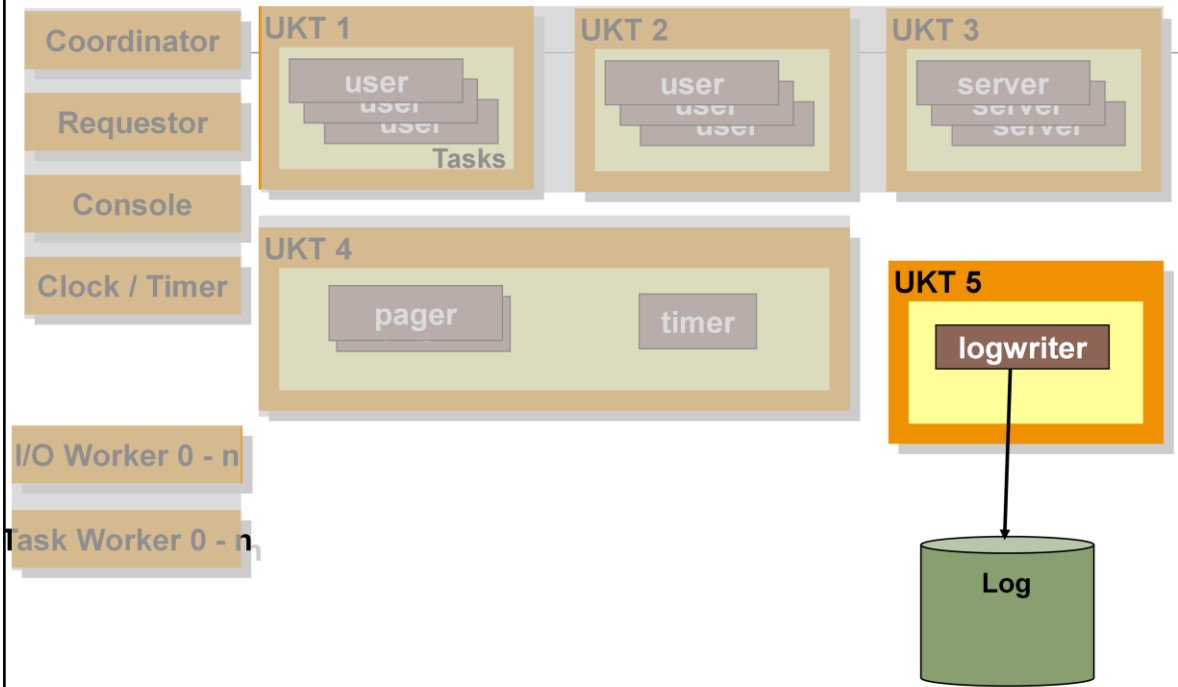session
timeout

Task Worker 0 - n

Pager tasks are responsible for writing data from the data cache to the data volumes. They become active when a savepoint is being executed.
The system calculates the number of pagers. It depends primarily on the data cache size and the number of data volumes.

The **timer task** is used for handling all types of timeout situations (such as session timeouts and lock request timeouts).
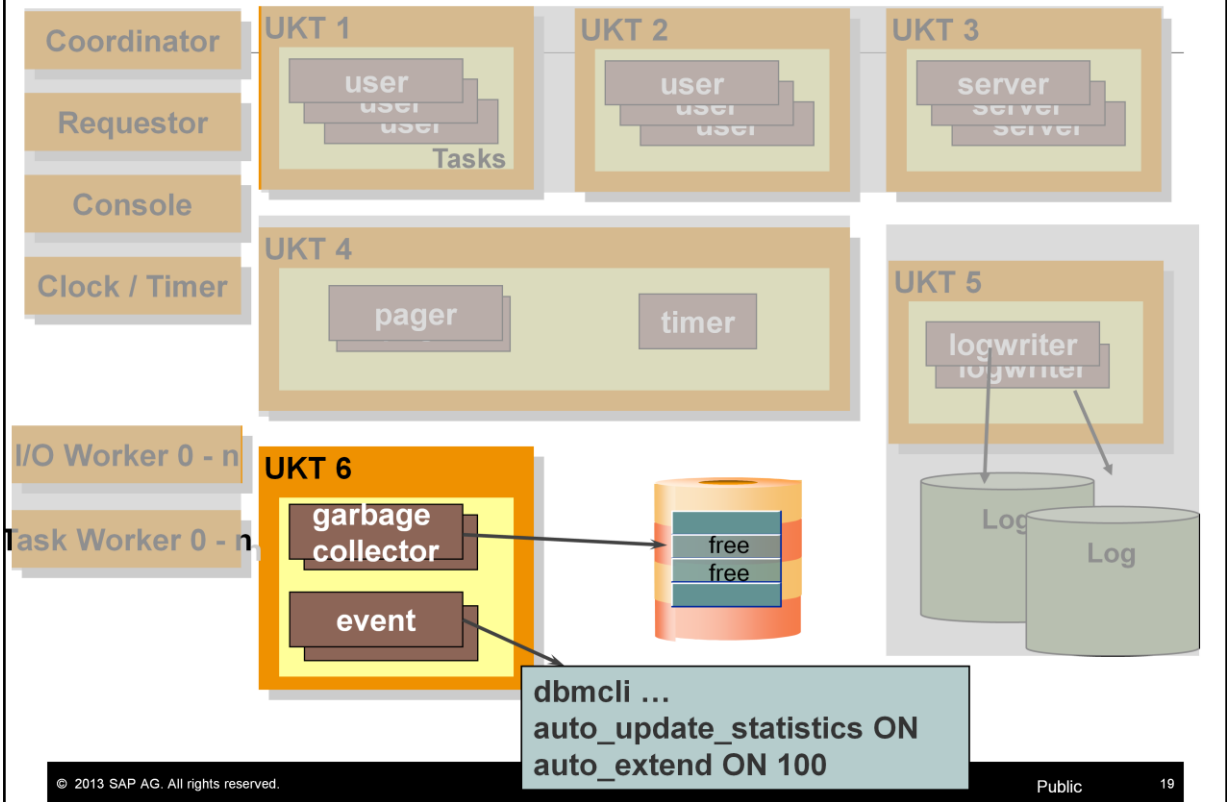
Logwriter

Coordinator	UKT 1	UKT 2	UKT 3
Requestor	user user user Tasks	user user user	server server server
Console			
Clock / Timer	UKT 4		UKT 5
	pager timer		logwriter
I/O Worker 0 - n			
Task Worker 0 - n			Log

The **Logwriter task** is responsible for writing redo log entries in the log volumes.
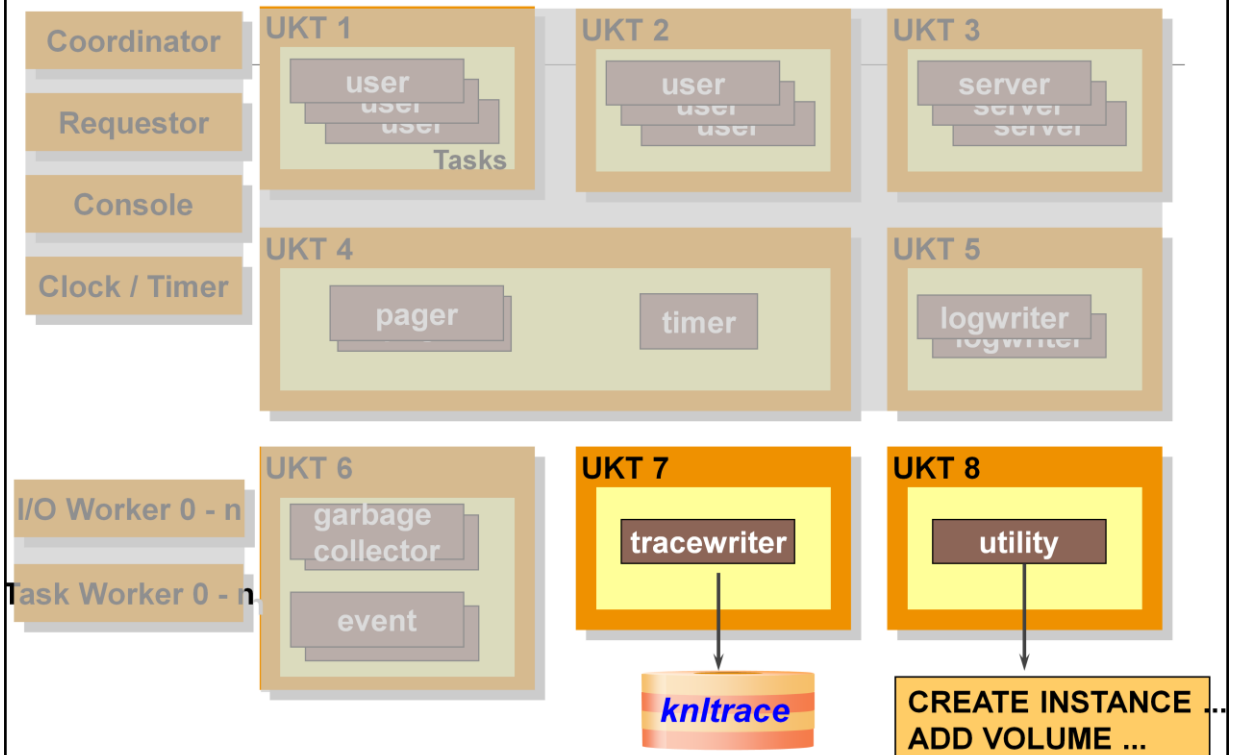
More information: Note 869267 FAQ SAP MaxDB Logging

Garbage Collectors and Event Tasks

Coordinator

Requestor

Console

Clock / Timer

I/O Worker 0 - n

Task Worker 0 - n

UKT 1
user
user
user
Tasks

UKT 2
user
user
user

UKT 3
server
server
server

UKT 4
pager
timer

UKT 5
logwriter
logwriter

UKT 6
garbage collector
event

free
free

Log
Log

dbmcli …
auto_update_statistics ON
auto_extend ON 100

Public   19

Garbage collectors release undo log files to free space management. With DROP TABLE statement, Garbage collectors delete the data in the tables asynchronously. Users do not have to wait for all data to be deleted.

Event tasks send messages about database events to the Database Manager (e.g. DBStudio). You can use the Event Dispatcher to define reactions. For example, you can have the database enlarge automatically when it gets full (auto_extend). You can also have the Event Dispatcher automatically update statistics when certain events occur (auto_update_statistics).
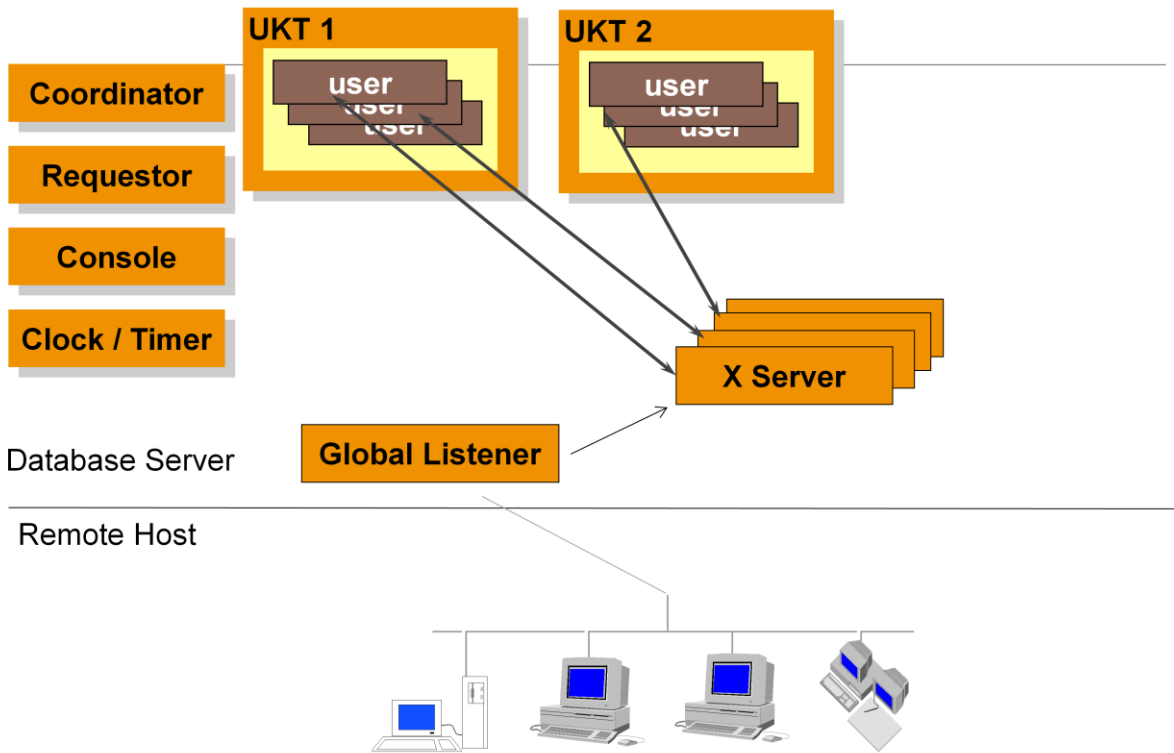
Utility–, Tracewriter Task

Coordinator

Requestor

Console

Clock / Timer

I/O Worker 0 - n

Task Worker 0 - n

UKT 1
user
user
user
Tasks

UKT 2
user
user
user

UKT 3
server
server
server

UKT 4
pager   timer

UKT 5
logwriter
logwriter

UKT 6
garbage collector
event

UKT 7
tracewriter
→ *knltrace*

UKT 8
utility
→ CREATE INSTANCE ...
ADD VOLUME ...

MaxDB offers the possibility of writing a special log, the database trace. The active tasks write the trace data to a buffer. If requested, the **trace writer task** writes the data from the buffer to the file knltrace.

The **utility task** is reserved exclusively for the administration of the database instance.
Automatic log backup can be executed in parallel with other administration tasks as it does not occupy a utility session after it is activated.

More Information about Kernel Trace: Note 837385 Database Trace (Vtrace)

Remote SQL Server

	UKT 1	UKT 2
Coordinator	user	user
Requestor		
Console		
Clock / Timer		

X Server

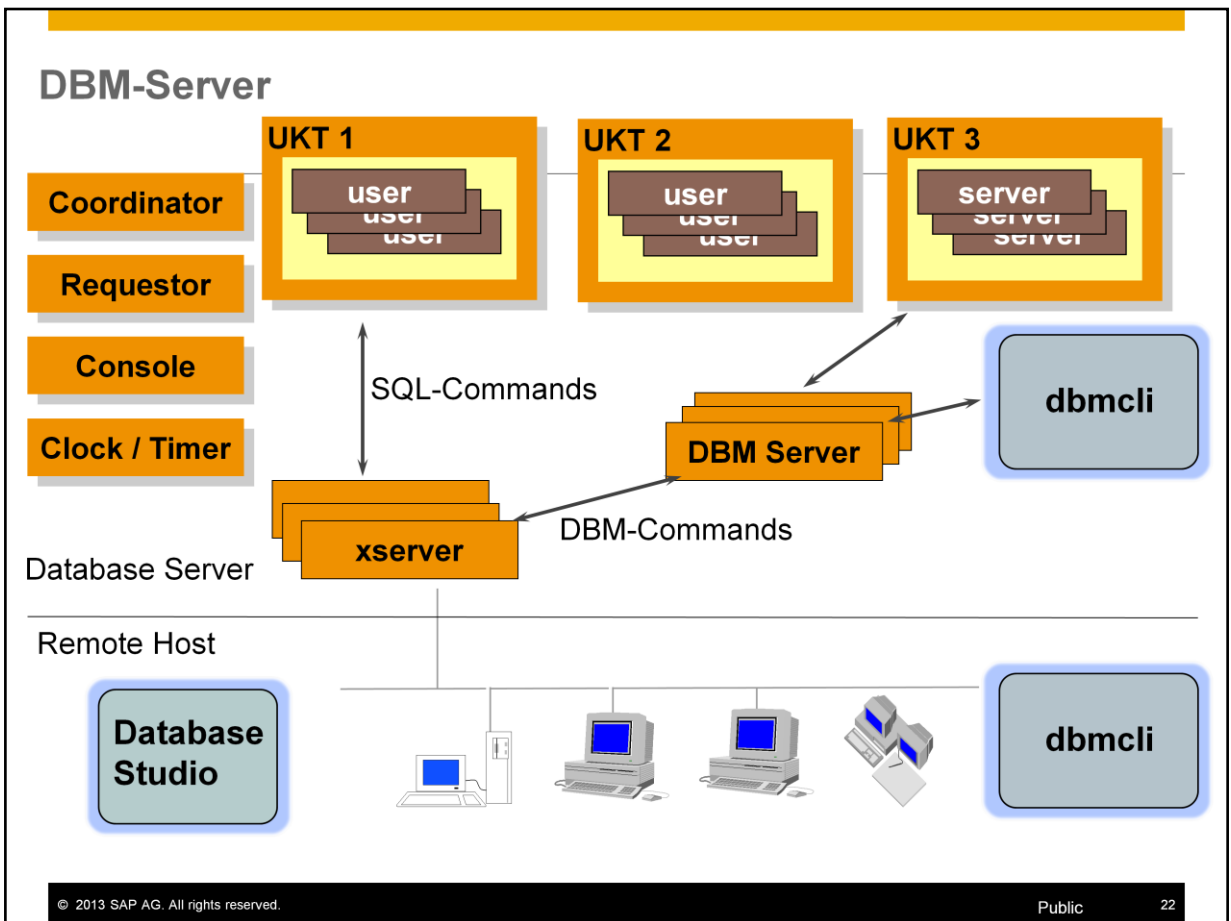Database Server — **Global Listener**

Remote Host

The **X-Server** is the communication server of the database system. It listens at a service port for connection requests from database applications and database tools. In the process list, this process is called **vserver**. A new vserver process is generated for every user process that logs on to the database remotely. The generating process serves the user; the new process waits for the next user logon. On Windows, an additional thread is started for the user logon.

On Windows, the X-Server runs as a service.

Local user sessions communicate with the database instance via a shared memory.

MaxDB 7.8 introduced the isolated software installation. Every database installed for SAP application uses it's own port number. Clients first connect to a global listener which returns the instance specific port number. The client then connects to the x_server assigned to the instance.

DBM-Server

UKT 1	UKT 2	UKT 3
user	user	server

- Coordinator
- Requestor
- Console
- Clock / Timer

SQL-Commands

DBM Server

dbmcli

Database Server

xserver

DBM-Commands

Remote Host

Database Studio

dbmcli

The DBM server is the database tool used to process administrative commands.

The DBM server establishes the connection from the database clients (for example, Database Manager CLI (DBMCLI), Database Studio, DBMRFC) to the database kernel.
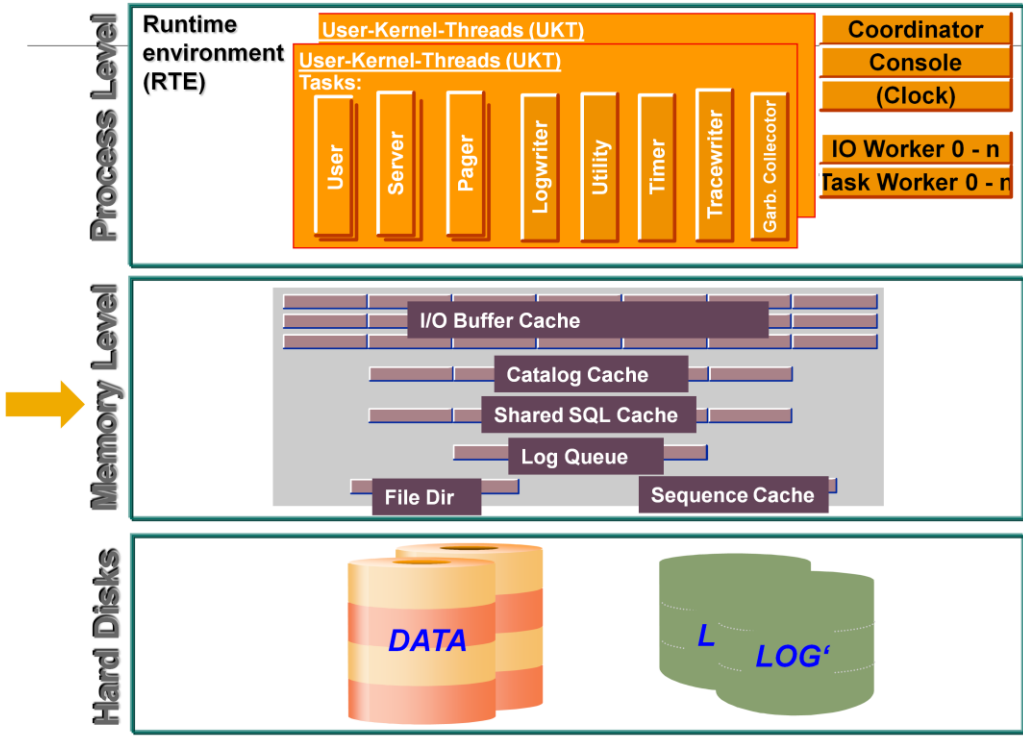
Only administrative commands that are started by the database tools Database Manager CLI, Database Studio or by SAP tools via DBMRFC (for example, DBA Planning Calendar (transaction DB13), DBA Cockpit (transaction DBACockpit)) communicate via the DBM server.

If such commands are sent remotely to the database host, a connection is first established via the remote SQL server (Versions >= 7.8 global listener or Versions < 7.8 X server), which in turn starts a DBM server.
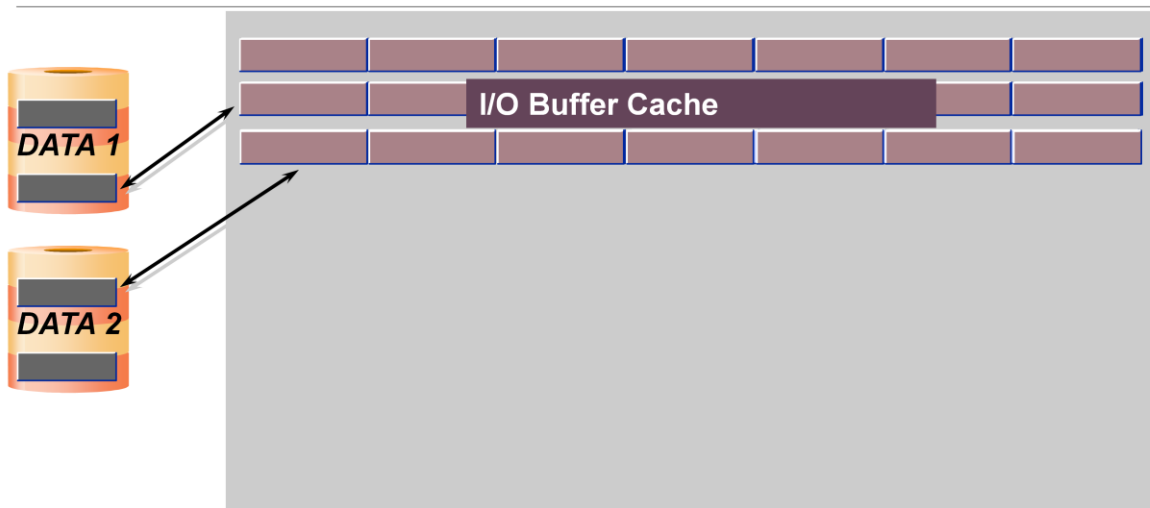
If you send an SQL statement via sql_execute/db_execute or an administration command via db_execute to the database, these will

also communicate via a DBM server session.

More Information: Note 1694323 FAQ Remote SQL Server and DBM Server

Database Instance

Process Level

Runtime environment (RTE)

User-Kernel-Threads (UKT)
User-Kernel-Threads (UKT)
Tasks:

- User
- Server
- Pager
- Logwriter
- Utility
- Timer
- Tracewriter
- Garb. Collecotor

Coordinator
Console
(Clock)
IO Worker 0 - n
Task Worker 0 - n

Memory Level

- I/O Buffer Cache
- Catalog Cache
- Shared SQL Cache
- Log Queue
- File Dir
- Sequence Cache

Hard Disks

DATA

L LOG'

Public

23

I/O Buffer Cache

Read and write operations to the volumes are buffered in order to save time-critical disk accesses. The corresponding main memory structures are called caches. They can be dimensioned according to the user profile.

The **I/O buffer cache** contains the last read- or write-accessed pages of the data volumes. It is shared by all simultaneously active users.

The hit rate, that is, the ratio of successful accesses to the total number of accesses to the **I/O buffer cache**, is a crucial measure of performance. It should be greater than 98%. Successful access means that the required data was already available in the data cache.

In addition to data pages, the I/O buffer cache also contains **converter pages**. Converter pages, like data pages, are stored in the data volumes. They store the assignment of the logical data page numbers to their physical position in the data volumes.
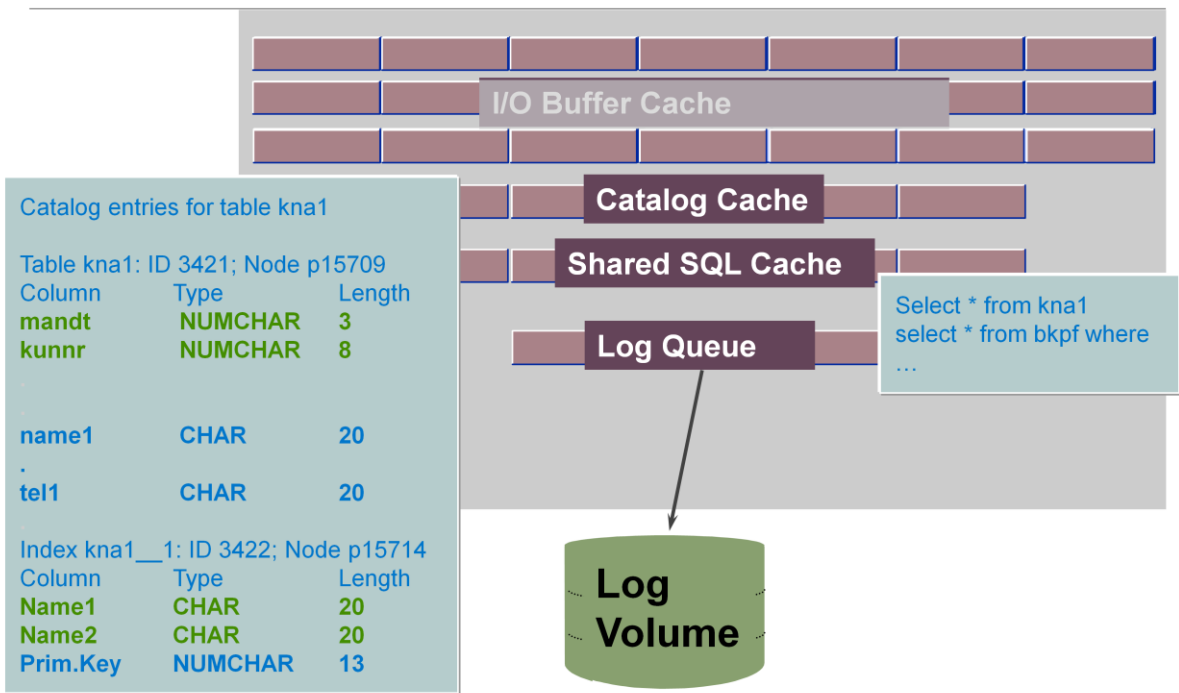
The number of converter pages is calculated automatically. It can increase with data growth. When deletions are performed, converter pages are released.

All converter pages are held in the cache. Each converter page contains 1861 entries that are managed for data pages.

The database parameter used for setting the size of the I/O buffer cache is called **CacheMemorySize**.

In versions < 7.4, data pages and converter pages were managed in separate caches, the data cache and the converter cache.

Catalog Cache, Log Queue

In the **catalog cache**, the database system stores user-specific data and global data from the database catalog. Data that has been displaced from the catalog cache is moved to the data cache. As of version 7.6 the catalog cache is a really shared cache. The information of this cache is shared among all users. In older versions, each user task was assigned to a separate area. The total of all catalog caches can increase up to the value that has been configured with the database parameter CAT_CACHE_SUPPLY.
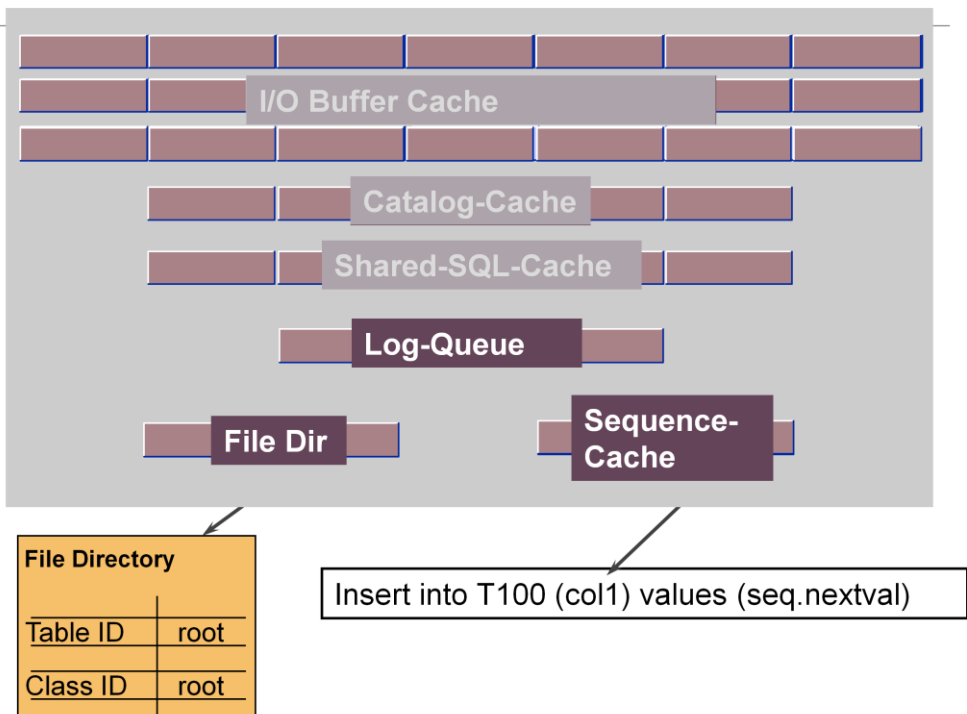
All statements to be executed will be held in the **shared SQL cache** along with their execution plans. The shared SQL cache is shared by all users, that is, a statement is only accepted once. When shared SQL is not being used, the statements of each user are kept separately in the catalog cache. The setting NO for database parameter SHAREDSQL deactivates use of this cache.

A statement, together with the statement text, is stored only once in the shared SQL cache. This allows you to see which statements are active in the database at any time. The shared SQL manager also collects monitor data. Shared SQL offers the following advantages over versions prior to 7.6:

- A statement only has to be prepared once rather for each user.

- Statements are stored only once. That saves space in the main memory.

- Storing the data in the main memory offers better monitoring possibilities.

The **log queue** allows log entries to be written to the log volumes asynchronously and increases the likelihood of group commits. In a group commit, several write transactions are completed in the log area with an I/O.
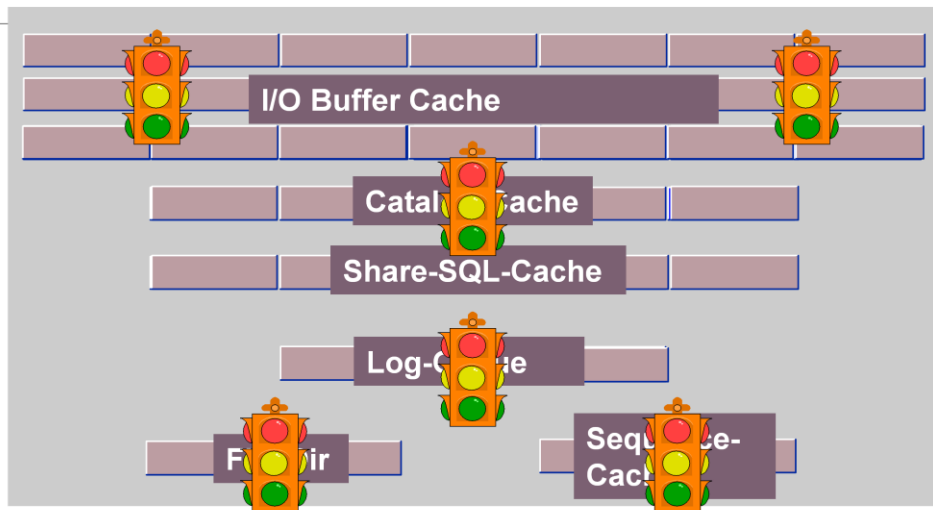
File Directory, Sequence Cache

File Directory	
Table ID	root
Class ID	root

Insert into T100 (col1) values (seq.nextval)

The **file directory** is required for the internal organization of the database instance. The assignments of the root pages of the B*-trees to the table IDs and a type flag are stored here. The type flag specifies whether the table contains primary data, secondary key or LONG data. With version 7.6 implementation of the file directory has changed. It now holds addtional figures about numbers of rows and table sizes. Thus asking for the number of rows of even extremely large tables using "SELECT COUNT(*) FROM <table>" can be executed very fast.

The **sequence cache** stores current data on number generators.

Cache Access

I/O Buffer Cache

Catalog Cache

Share-SQL-Cache

Log-Cache

F... ir

Sequence-Cache

Critical sections in memory are protected by synchronization mechanisms (Regions)

Accesses to caches can be synchronized over one or more regions. Depending on its size, the data cache is comprised of 8 to 64 segments of the same size, each of which is protected by exactly one region.

If a task or thread accesses a critical section, the region locks this section for all other tasks or processes.

Other main memory structures are also managed via synchronization mechanisms provided by the database.

Reader-writer locks are used to synchronize the shared SQL cache. Reader-writer locks are used in version 7.5 and up. In contrast to regions, reader-writer locks make it possible to distinguish between shared and exclusive locks.

More Information: Note 1681336 MaxDB Synchronization Objects

Questions

SAP® MaxDB™ Database Architecture

SAP® MaxDB™ – Expert Sessions Learning Map (1)

SAP® MaxDB ™ Features
Session 1: Low TCO with the SAP MaxDB Database
Session 6: New Features in SAP MaxDB Version 7.7
Session 8: New Features in SAP MaxDB Version 7.8

SAP® MaxDB ™ Administration
Session 2: Basic Administration with Database Studio
Session 3: CCMS Integration into the SAP System
Session 11: SAP MaxDB Backup and Recovery
Session 13: Third-Party Backup Tools

SAP® MaxDB ™ Problem Analysis
Session 5: SAP MaxDB Data Integrity
Session 14: SAP MaxDB Tracing
Session 12: Analysis of SQL Locking Situations

SAP® MaxDB ™ Installation/Upgrade
Session 7: SAP MaxDB Software Update Basics

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

Public   29

SAP® MaxDB™ – Expert Sessions Learning Map (2)

SAP® MaxDB™ Architecture	SAP® MaxDB™ Performance
Session 18: Introduction MaxDB Database Architecture	Session 4: Performance Optimization with SAP MaxDB
Session 15: SAP MaxDB No-Reorganization Principle	Session 9: SAP MaxDB Optimized for SAP BW
Session 17: SAP MaxDB Shadow Page Algorithm	Session 16: SAP MaxDB SQL Query Optimization (Part 1)
Session 12: Analysis of SQL Locking Situations	Session 16: SAP MaxDB SQL Query Optimization (Part 2)
Session 10: SAP MaxDB Logging	

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

Public   30

Thank You!
Bye, Bye – And Remember Next Session

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

Next Session: 11.06.2013
SAP® MaxDB™ Parameter - Handling

Thank you

Contact information:

Christiane Hienger
Development Expert IMS
Christiane.Hienger@sap.com