The Index Server of the SAP HANA database is a core component that orchestrates

the database's operations.

The Connection and Session Management which creates and manages sessions and

connections for the database clients such as SAP Business Objects Reporting tools or

applications.

The Transaction Manager coordinates transactions, controls transactional isolation,

and keeps track of running and closed transactions.

The client requests are analyzed and executed by a set of specialized engines and

processors that handle Request Processing and Execution Control. Once a session

is established, the database client typically uses SQL statements to communicate

with this module. For analytical applications, the multidimensional query language

MDX is also supported.

Incoming SQL requests are received by the SQL Processor. This component executes

the Data Manipulation Language (DML) statements, such as INSERT, SELECT,

UPDATE or DELETE. Other types of requests are delegated to other components.

For example, Data Definition Language (DDL) statements, such as the definition

of relational tables, columns, views, indexes and procedures, are dispatched to the

Metadata Manager.

Planning commands are routed to the Planning Engine that allows financial planning

applications to execute basic planning operations in the database layer.

The SAP HANA database offers programming capabilities to execute

application-specific calculations inside the database system. The SAP HANA database

has its own programming languages. SQLScript is used to write database stored

procedures. Procedure calls are forwarded to the Stored Procedure processor.

lncoming MDX requests are processed by the MDX engine and also forwarded to the

Calculation Engine, which is a common infrastructure that also supports SQL Script,

MD and Planning operations.

The Persistence Layer component manages the communication between the Index

Server and the File System that store the Data volume and Transaction Log volume.

High-Availability enables the failover of a node within one distributed SAP HANA

appliance. Failover uses a cold standby node and gets triggered automatically.

Landscape Up to 3 master name-servers can be defined. During startup one server gets

elected as active master. The active master assigns a volume to each starting index

server or no volume in case of standby servers.

Master name-server failure

In case of a master name-server failure, another of the remaining name-servers will

become active master.

Index-server failure

The master name-server detects an index-server failure and executes the failover.

During the failover the master name-server assigns the volume of the failed

index-server to the standby server.

Persistence Layer

Data and log volumes are used as follows:

• On a regular basis, data pages and before images (undo log pages) are written in

the data volumes. This process is called a Savepoint.

• Between two savepoints, after images (redo log pages) are written in the log

volumes. This is done each time a transaction is committed.

The savepoint process relies to a concept called Shadow Memory.

Shadow paging is used to undo changes that were persisted since the last savepoint.

With the shadow page concept, physical disk pages written by the last savepoint

are not overwritten until the next savepoint is successfully completed. Instead, new

physical pages are used to persist changed logical pages. Until the next savepoint is

complete, two physical pages may exist for one logical page:

• The shadow page, which still contains the version of the last savepoint

• The current physical page which contains the changes written to disk after the

last savepoint

System Restart Procedure

After a restart, the system is restored from the savepoint versions of the data pages.

This ·way, all data changes written since the last savepoint are not restored. After the

savepoint is restored, the log is replayed to restore the most recent committed state.

The system restart includes the following actions:

• Restore data

Reload the last savepoint

Seach the undo log for uncommitted transactions saved with last savepoint

(stored on the data volume) and roll them back

Search the redo log for committed transactions since last savepoint (stored

on the log volume) and re-execute them

• Load all the tables of the row store into memory

• Load the tables of the column store that are marked for preload into memory

Note: Only tables marked for preload are loaded into in memory during

startup.

Tables marked for loading on demand ·will only be loaded into memory

at first access.