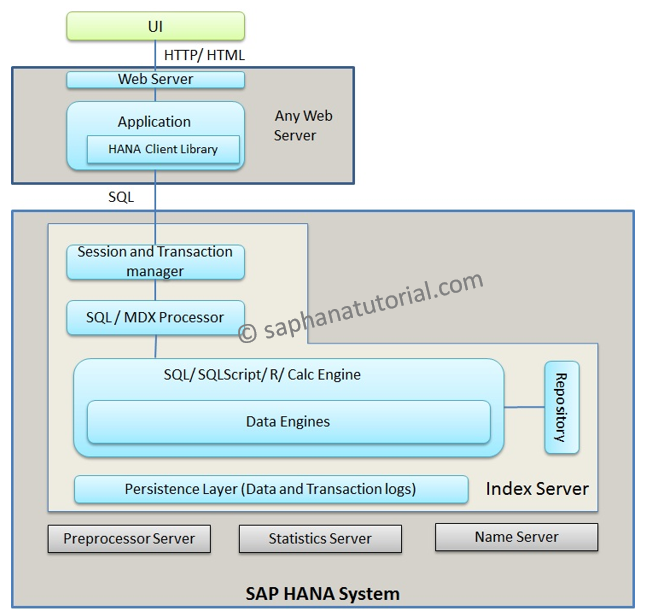
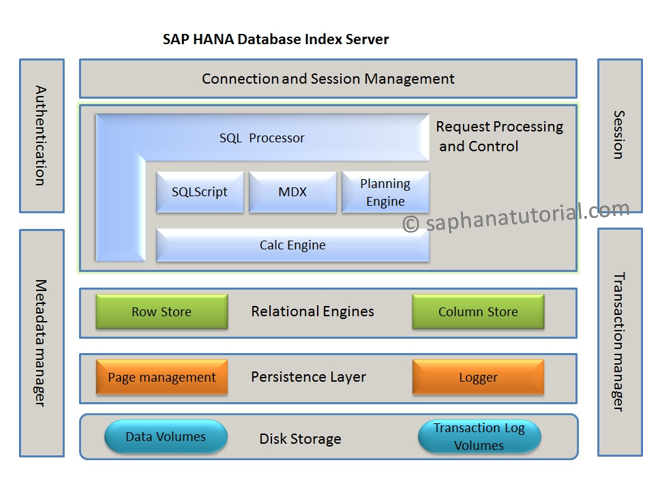
**SAP HANA Architecture Overview:**

The SAP HANA database is developed in C++ and runs on SUSE Linux Enterpise Server. SAP HANA database consists of multiple servers and the most important component is the Index Server. SAP HANA database consists of Index Server, Name Server, Statistics Server, Preprocessor Server and XS Engine.   
  
  
**Index Server:**

* + Index server is the main SAP HANA database component
  + It contains the actual data stores and the engines for processing the data.
  + The index server processes incoming SQL or MDX statements in the context of authenticated sessions and transactions.

**Persistence Layer:**   
The database persistence layer is responsible for durability and atomicity of transactions. It ensures that the database can be restored to the most recent committed state after a restart and that transactions are either completely executed or completely undone.   
**Preprocessor Server:**   
The index server uses the preprocessor server for analyzing text data and extracting the information on which the text search capabilities are based.   
**Name Server:**   
The name server owns the information about the topology of SAP HANA system. In a distributed system, the name server knows where the components are running and which data is located on which server.   
**Statistic Server:**   
The statistics server collects information about status, performance and resource consumption from the other servers in the system.. The statistics server also provides a history of measurement data for further analysis.   
**Session and Transaction Manager:**   
The Transaction manager coordinates database transactions, and keeps track of running and closed transactions. When a transaction is committed or rolled back, the transaction manager informs the involved storage engines about this event so they can execute necessary actions.   
**XS Engine:**   
XS Engine is an optional component. Using XS Engine clients can connect to SAP HANA database to fetch data via HTTP.

**The heart of SAP HANA – Index Server**

The SAP HANA Index Server contains the majority of the magic behind SAP HANA.  
  
  
**Connection and Session Management**

* This component is responsible for creating and managing sessions and connections for the database clients.
* Once a session is established, clients can communicate with the SAP HANA database using SQL statements.
* For each session a set of parameters are maintained like, auto-commit, current transaction isolation level etc.
* Users are Authenticated either by the SAP HANA database itself (login with user and password) or authentication can be delegated to an external authentication providers such as an LDAP directory.

**The Authorization Manager**

* This component is invoked by other SAP HANA database components to check whether the user has the required privileges to execute the requested operations.
* SAP HANA allows granting of privileges to users or roles. A privilege grants the right to perform a specified operation (such as create, update, select, execute, and so on) on a specified object (for example a table, view, SQLScript function, and so on).
* The SAP HANA database supports Analytic Privileges that represent filters or hierarchy drilldown limitations for analytic queries. Analytic privileges grant access to values with a certain combination of dimension attributes. This is used to restrict access to a cube with some values of the dimensional attributes.

**Request Processing and Execution Control:**

* The client requests are analyzed and executed by the set of components summarized as Request Processing and Execution Control. The Request Parser analyses the client request and dispatches it to the responsible component. The Execution Layer acts as the controller that invokes the different engines and routes intermediate results to the next execution step.
* **SQL Processor:**
  + Incoming SQL requests are received by the SQL Processor. Data manipulation statements are executed by the SQL Processor itself.
  + Other types of requests are delegated to other components. Data definition statements are dispatched to the Metadata Manager, transaction control statements are forwarded to the Transaction Manager, planning commands are routed to the Planning Engine and procedure calls are forwarded to the stored procedure processor.

**SQLScript:**

* The SAP HANA database has its own scripting language named SQLScript that is designed to enable optimizations and parallelization. SQLScript is a collection of extensions to SQL.
* SQLScript is based on side effect free functions that operate on tables using SQL queries for set processing. The motivation for SQLScript is to offload data-intensive application logic into the database.

**Multidimensional Expressions (MDX):**

* MDX is a language for querying and manipulating the multidimensional data stored in OLAP cubes.
* Incoming MDX requests are processed by the MDX engine and also forwarded to the Calc Engine.

**Planning Engine:**

* Planning Engine allows financial planning applications to execute basic planning operations in the database layer. One such basic operation is to create a new version of a data set as a copy of an existing one while applying filters and transformations. For example: planning data for a new year is created as a copy of the data from the previous year.
* Another example for a planning operation is the disaggregation operation that distributes target values from higher to lower aggregation levels based on a distribution function.

**Calc engine:**

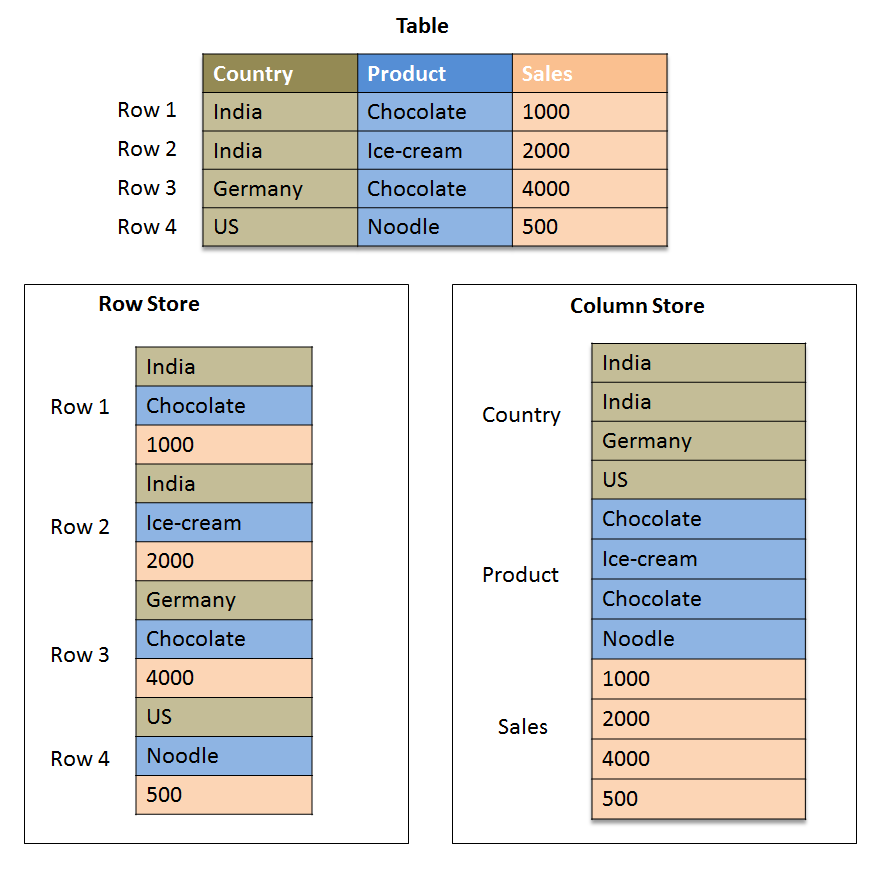
* The SAP HANA database features such as SQLScript and Planning operations are implemented using a common infrastructure called the Calc engine.
* The SQLScript, MDX, Planning Model and Domain-Specific models are converted into Calculation Models. The Calc Engine creates Logical Execution Plan for Calculation Models. The Calculation Engine will break up a model, for example some SQL Script, into operations that can be processed in parallel.

**Transaction Manager:**   
In HANA database, each SQL statement is processed in the context of a transaction. New sessions are implicitly assigned to a new transaction. The Transaction Manager coordinates database transactions, controls transactional isolation and keeps track of running and closed transactions. When a transaction is committed or rolled back, the transaction manager informs the involved engines about this event so they can execute necessary actions.   
The transaction manager also cooperates with the persistence layer to achieve atomic and durable transactions.   
**Metadata Manager:**

* Metadata can be accessed via the Metadata Manager component. In the SAP HANA database, metadata comprises a variety of objects, such as definitions of relational tables, columns, views, indexes and procedures.
* Metadata of all these types is stored in one common database catalog for all stores. The database catalog is stored in tables in the Row Store. The features of the SAP HANA database such as transaction support and multi-version concurrency control, are also used for metadata management.

In the center of the figure you see the different data Stores of the SAP HANA database. A store is a sub-system of the SAP HANA database which includes in-memory storage, as well as the components that manages that storage.   
  
**The Row Store:**   
The Row Store is the SAP HANA database row-based in-memory relational data engine.   
**The Column Store:**   
The Column Store stores tables column-wise. It originates from the TREX (SAP NetWeaver Search and Classification) product.   
Want to know more about Row Data and Column Data Storage?   
Check Column Data Storage vs Row Data Storage: How Different are they Really?   
  
**Persistence Layer:**   
The Persistence Layer is responsible for durability and atomicity of transactions. This layer ensures that the database is restored to the most recent committed state after a restart and that transactions are either completely executed or completely undone. To achieve this goal in an efficient way, the Persistence Layer uses a combination of write-ahead logs, shadow paging and savepoints.   
  
The Persistence Layer offers interfaces for writing and reading persisted data. It also contains the Logger component that manages the transaction log. Transaction log entries are written explicitly by using a log interface or implicitly when using the virtual file abstraction.

Overview of Row Data Storage and Column Data Storage

Relational databases typically use row-based data storage. However Column-based storage is more suitable for many business applications. SAP HANA supports both row-based and column-based storage, and is particularly optimized for column-based storage.   
  
As shown in the figure below, a database table is conceptually a two-dimensional structure composed of cells arranged in rows and columns.   
  
Because computer memory is structured linearly, there are two options for the sequences of cell values stored in contiguous memory locations:   
  
**Row Storage** - It stores table records in a sequence of rows.   
**Column Storage** - It stores table records in a sequence of columns i.e. the entries of a column is stored in contiguous memory locations.   
  
Traditional databases store data simply in rows. The HANA in-memory database stores data in both rows and columns. It is this combination of both storage approaches that produces the speed, flexibility and performance of the HANA database.

Advantages of column-based tables:

**Faster Data Access:**

Only affected columns have to be read during the selection process of a query. Any of the columns can serve as an index.

**Better Compression:**

Columnar data storage allows highly efficient compression because the majority of the columns contain only few distinct values (compared to number of rows).

**Better parallel Processing:**

In a column store, data is already vertically partitioned. This means that operations on different columns can easily be processed in parallel. If multiple columns need to be searched or aggregated, each of these operations can be assigned to a different processor core

Advantages and disadvantages of row-based tables:

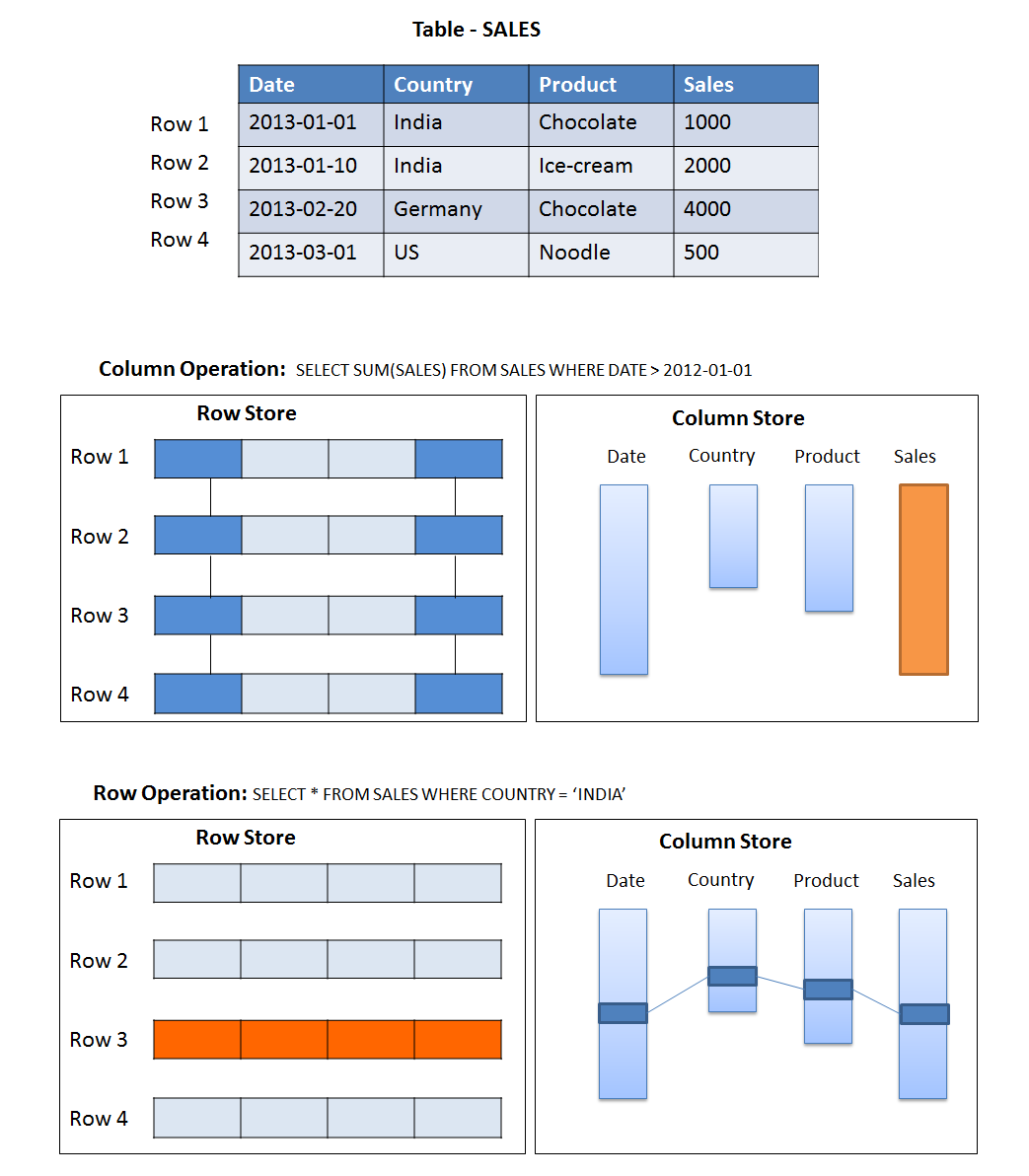
**Row based tables have advantages in the following circumstances:**

* + The application needs to only process a single record at one time (many selects and/or updates of single records).
  + The application typically needs to access a complete record (or row).
  + Neither aggregations nor fast searching are required.
  + The table has a small number of rows (e. g. configuration tables, system tables).

**Row based tables have dis-advantages in the following circumstances:**

* + In case of analytic applications where aggregation are used and fast search and processing is required. In row based tables all data in a row has to be read even though the requirement may be to access data from a few columns.

Which type of tables should be preferred - Row-based or Column-based?

In case of analytic applications where aggregations are used and fast search and processing is required row-based storage are not good. In row based tables all data in a row has to be read even though the requirement may be to access data from a few columns. Hence these queries on huge amounts of data take a lot of time.   
  
In columnar tables, this information is stored physically next to each other, significantly increasing the speed of certain data queries.   
  
The following example shows the different usage of column and row storage, and positions them relative to row and column queries. Column storage is most useful for OLAP queries (queries using any SQL aggregate functions) because these queries get just a few attributes from every data entry. But for traditional OLTP queries (queries not using any SQL aggregate functions), it is more advantageous to store all attributes side-by-side in row tables. HANA combines the benefits of both row- and column-storage tables.   
  
**Conclusion:**

To enable fast on-the-fly aggregations, ad-hoc reporting, and to benefit from compression mechanisms it is recommended that transaction data is stored in a column-based table.   
  
The SAP HANA data-base allows joining row-based tables with column-based tables. However, it is more efficient to join tables that are located in the same row or column store. For example, master data that is frequently joined with transaction data should also be stored in column-based tables.

# HANA Memory Usage

So far we learnt about SAP HANA Architecture and Column Data Storage Vs Row Data Storage   
In this article we will explain memory usage and calculation in SAP HANA.

## Introduction:

SAP HANA is a leading in-memory database and data management platform, specifically developed to take full advantage of the capabilities provided by modern hardware to increase application performance. By keeping all relevant data in main memory (RAM), data processing operations are significantly accelerated.   
  
**"SAP HANA has become the fastest growing product in SAP's history."**   
  
A fundamental SAP HANA resource is memory. Understanding how the SAP HANA system requests, uses and manages this resource is crucial to the understanding of SAP HANA. SAP HANA provides a variety of memory usage indicators, to allow monitoring, tracking and alerting.   
  
This article explores the key concepts of SAP HANA memory utilization, and shows how to understand the various memory indicators.

**Memory Concepts:**

As an in-memory database, it is critical for SAP HANA to handle and track its memory consumption carefully and efficiently. For this purpose, the SAP HANA database pre-allocates and manages its own memory pool and provides a variety of memory usage indicators to allow monitoring.   
  
SAP HANA tracks memory from the perspective of the host. The most important concepts are as follows:   
  
**Physical memory:**

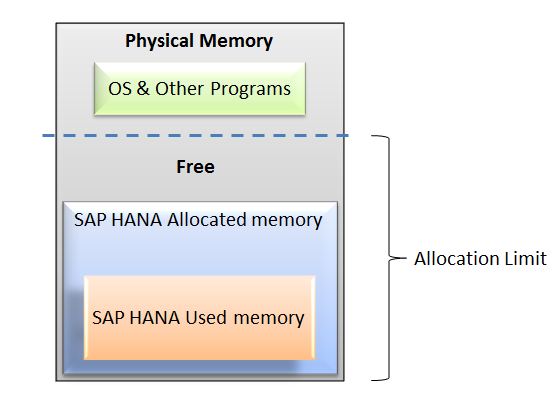
The amount of (system) physical memory available on the host.

**SAP HANA Allocated memory**

The memory pool reserved by SAP HANA from the operating system.

**SAP HANA Used memory**

The amount of memory from this pool that is actually used by the SAP HANA database.



**Determining Physical Memory Size:**

Physical memory (DRAM) is the basis for all memory discussions. On most SAP HANA hosts, it ranges from 256 gigabytes to 2 terabytes. It is used to run the Linux operating system, SAP HANA, and all other programs that run on the host. The following table lists the various ways of determining the amount of physical memory:   
  
You can use the M\_HOST\_RESOURCE\_UTILIZATION view to explore the amount of Physical Memory as follows:   
  
**Determine Available Physical Memory:**  
Execute the SQL query:

select round((USED\_PHYSICAL\_MEMORY + FREE\_PHYSICAL\_MEMORY) /1024/1024/1024, 2)   
as "Physical Memory GB"   
from PUBLIC.M\_HOST\_RESOURCE\_UTILIZATION;

Execute the Linux command:

cat /proc/meminfo | grep MemTotal

**Determine Free Physical Memory:**  
Execute the SQL query:

select round(FREE\_PHYSICAL\_MEMORY/1024/1024/1024, 2)   
as "Free Physical GB"   
from PUBLIC.M\_HOST\_RESOURCE\_UTILIZATION;

Execute the Linux command:

awk 'BEGIN {sum = 0};   
/^(MemFree|Buffers|Cached):/ {sum = sum + $2}; END {print sum}' /proc/meminfo

**SAP HANA Allocated Memory Pool:**

The SAP HANA database (across its different processes) reserves a pool of memory before actual use.   
  
This pool of allocated memory is pre-allocated from the operating system over time, up to a predefined global allocation limit, and is then efficiently used as needed by the SAP HANA database code. More memory is allocated to the pool as used memory grows. If used memory nears the global allocation limit, the SAP HANA database may run out of memory if it cannot free memory. The default allocation limit is 90% of available physical memory, but this value is configurable.   
  
To find the global allocation limit of the database, run below SQL query:

select HOST, round(ALLOCATION\_LIMIT/1024/1024/1024, 2) as "Allocation Limit GB"   
from PUBLIC.M\_HOST\_RESOURCE\_UTILIZATION

**Effective Allocation Limit:**   
In addition to the global allocation limit, each process running on the host has an allocation limit, the process allocation limit. Given that all processes cannot collectively consume more memory than the global allocation limit, each process also has what is called an effective allocation limit. The effective allocation limit of a process specifies how much physical memory a process can in reality consume given the current memory consumption of other processes.   
  
**Example:**  
A single-host system has 100 GB physical memory. Both the global allocation limit and the individual process allocation limits are 90% (default values). This means the following:

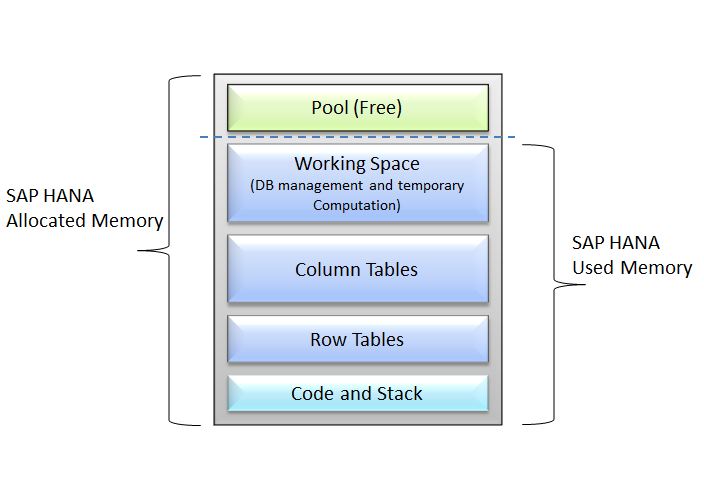
* + Collectively, all processes of the HANA database can use a maximum of 90 GB.
  + Individually, each process can use a maximum of 90 GB.

If 2 processes are running and the current memory pool of process 1 is 50 GB, then the effective allocation limit of process 2 is 40 GB. This is because process 1 is already using 50 GB and together they cannot exceed the global allocation limit of 90 GB.

**SAP HANA Used Memory:**

Used memory serves several purposes:

* + Program code and stack
  + Working space and data tables (heap and shared memory)

**The program code area** contains the SAP HANA database itself while it is running. Different parts of SAP HANA can share the same program code.   
  
**The stack** is needed to do actual computations.   
  
**The heap and shared memory** are the most important part of used memory. It is used for working space, temporary data and for storing all data tables.   
  
  
You can use the M\_SERVICE\_MEMORY view to explore the amount of SAP HANA Used Memory as follows:  
  
**Total Memory Used:**

SELECT round(sum(TOTAL\_MEMORY\_USED\_SIZE/1024/1024)) AS "Total Used MB"   
FROM SYS.M\_SERVICE\_MEMORY;

**Code and Stack Size:**

SELECT round(sum(CODE\_SIZE+STACK\_SIZE)/1024/1024) AS "Code+stack MB"   
FROM SYS.M\_SERVICE\_MEMORY;

**Total Memory Consumption of All Columnar Tables:**

SELECT round(sum(MEMORY\_SIZE\_IN\_TOTAL)/1024/1024) AS "Column Tables MB"   
FROM M\_CS\_TABLES;

**Total Memory Consumption of All Row Tables**

SELECT round(sum(USED\_FIXED\_PART\_SIZE +   
USED\_VARIABLE\_PART\_SIZE)/1024/1024) AS "Row Tables MB"   
FROM M\_RS\_TABLES;

**Total Memory Consumption of All Columnar Tables by Schema:**

SELECT SCHEMA\_NAME AS "Schema",   
round(sum(MEMORY\_SIZE\_IN\_TOTAL) /1024/1024) AS "MB"   
FROM M\_CS\_TABLES GROUP BY SCHEMA\_NAME ORDER BY "MB" DESC;

**Memory Consumption of Columnar Tables:**

The SAP HANA database loads columnar tables into memory column by column only upon use. This is sometimes called "lazy loading". This means that columns that are never used are not loaded, which avoids memory waste.   
  
When the SAP HANA database runs out of allocated memory, it may also unload rarely used columns to free up some memory. Therefore, if it is important to precisely measure the total, or "worst case", amount of memory used for a particular table, it is best to ensure that the table is fully loaded first by executing the following SQL statement:

LOAD table\_name ALL.

To examine the memory consumption of columnar tables, you can use the M\_CS\_TABLES and M\_CS\_COLUMNS views.   
  
The following examples show how you can use these views to examine the amount of memory consumed by a specific table. You can also see which of its columns are loaded and the compression ratio that was accomplished.   
  
**List All Columnar Tables of Schema 'SYSTEM':**

SELECT TABLE\_NAME AS "Table", round(MEMORY\_SIZE\_IN\_TOTAL/1024/1024, 2) as "MB"   
FROM M\_CS\_TABLES WHERE SCHEMA\_NAME = 'SYSTEM' ORDER BY "MB" DESC;

**Show Column Details of Table "TABLE1":**

SELECT COLUMN\_NAME AS "Column", LOADED AS "Is Loaded",   
round(UNCOMPRESSED\_SIZE/1024/1024) AS "Uncompressed MB",   
round(MEMORY\_SIZE\_IN\_MAIN/1024/1024) AS "Main MB",   
round(MEMORY\_SIZE\_IN\_DELTA/1024/1024) AS "Delta MB",   
round(MEMORY\_SIZE\_IN\_TOTAL/1024/1024) AS "Total Used MB",   
round(COMPRESSION\_RATIO\_IN\_PERCENTAGE/100, 2) AS "Compr. Ratio"   
FROM M\_CS\_Columns WHERE TABLE\_NAME = 'TABLE1;

**Note:** The M\_CS\_TABLES and M\_CS\_COLUMNS views contain a lot of additional information (such as cardinality, main-storage versus delta storage and more). For example, use the following query to obtain more information:

SELECT \* FROM M\_CS\_COLUMNS WHERE TABLE\_NAME = '"' and COLUMN\_NAME = '"'

**Memory Consumption of Row-Ordered Tables:**

Several system tables are in fact row-ordered tables. You can use the M\_RS\_TABLES view to examine the memory consumption of row-ordered tables.   
  
For instance, you can execute the following SQL query, which lists all row tables of schema "SYS" by descending size:

SELECT SCHEMA\_NAME, TABLE\_NAME, round((USED\_FIXED\_PART\_SIZE +   
USED\_VARIABLE\_PART\_SIZE)/1024/1024, 2) AS "MB Used"   
FROM M\_RS\_TABLES   
WHERE schema\_name = 'SYS' ORDER BY "MB Used" DESC, TABLE\_NAME

**Memory Consumption Configuration:**

By default, SAP HANA can pre-allocate up to 90% of the available physical memory on the host. There is normally no reason to change the value of this variable, except in the case where a license was purchased for less than the total of the physical memory. In this case, you should change the global allocation limit to remain in compliance with the license.   
  
**Example 1:**

You have a server with 512GB, but purchased an SAP HANA license for only 384 GB. Set the global\_allocation\_limit to 393216 (384 \* 1024 MB).

**Example 2:**

You have a distributed HANA system on four hosts with 512GB each, but purchased an SAP HANA license for only 768 GB. Set the global\_allocation\_limit to 196608 (192 \* 1024 MB on each host).

**Resident memory:**

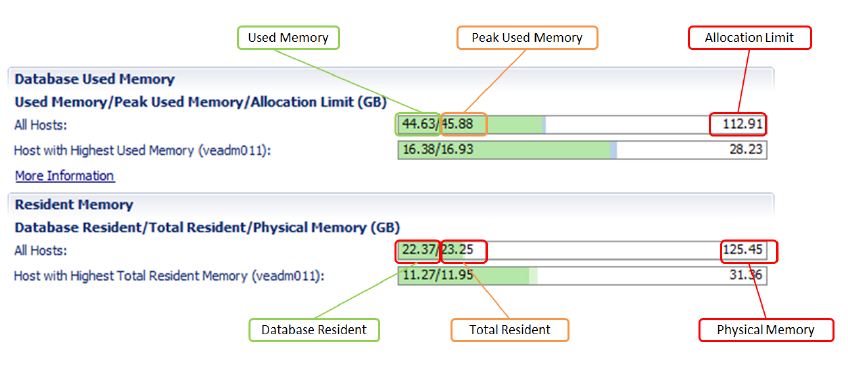
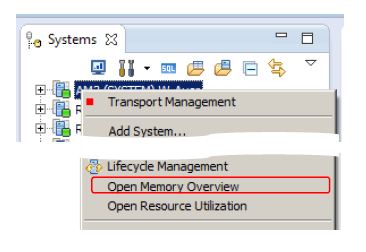
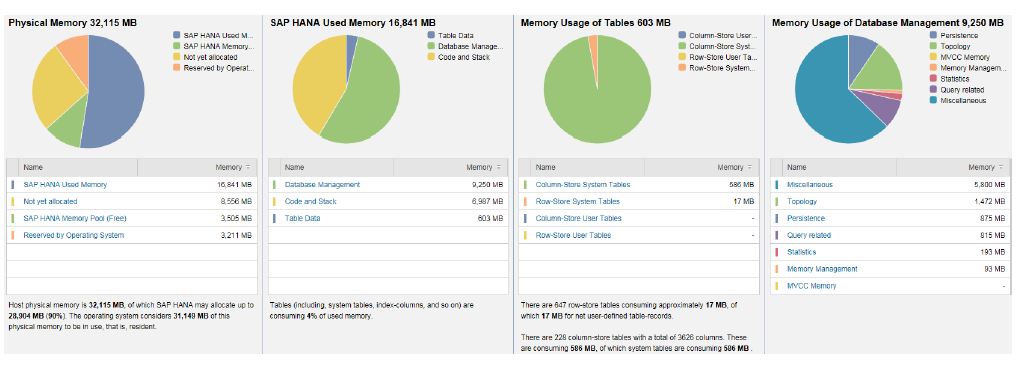
Resident memory is the physical memory actually in operational use by a process.   
  
Over time, the operating system may "swap out" some of a process' resident memory, according to a least-recently-used algorithm, to make room for other code or data. Thus, a process' resident memory size may fluctuate independently of its virtual memory size. In a properly sized SAP HANA appliance there is enough physical memory, and thus swapping is disabled and should not be observed.   
  
To display the size of the Physical Memory and Resident part, you can use the following SQL command:

select HOST, round((USED\_PHYSICAL\_MEMORY + FREE\_PHYSICAL\_MEMORY)/1024/1024/1024, 2) as "Physical Memory GB",   
round(USED\_PHYSICAL\_MEMORY/1024/1024/1024, 2) as "Resident GB"   
from PUBLIC.M\_HOST\_RESOURCE\_UTILIZATION

**Memory Sizing:**

Memory sizing is the process of estimating, in advance, the amount of memory that will be required to run a certain workload on SAP HANA. To understand memory sizing, you will need to answer the following questions:   
  
**1. What is the size of the data tables that will be stored in SAP HANA?**   
You may be able to estimate this based on the size of your existing data, but unless you precisely know the compression ratio of the existing data and the anticipated growth factor, this estimate may only be partially meaningful.   
  
**2. What is the expected compression ratio that SAP HANA will apply to these tables?**  
The SAP HANA Column Store automatically uses a combination of various advanced compression algorithms (dictionary, LRE, sparse, and more) to best compress each table column separately. The achieved compression ratio depends on many factors, such as the nature of the data, its organization and data-types, the presence of repeated values, the number of indexes (SAP HANA requires fewer indexes), and more.   
  
**3. How much extra working memory will be required for DB operations and temporary computations?**  
The amount of extra memory will somewhat depend on the size of the tables (larger tables will create larger intermediate result-tables in operations like joins), but even more on the expected work load in terms of the number of users and the concurrency and complexity of the analytical queries (each query needs its own workspace).   
  
SAP Notes 1514966, 1637145 and 1736976 provide additional tools and information to help you size the required amount of memory, but the most accurate method is ultimately to import several representative tables into a SAP HANA system, measure the memory requirements, and extrapolate from the results.

**SAP HANA Studio:**

You can view some of the most important memory indicators on the Overview tab of the SAP HANA studio administrative perspective:   
  
For even more details, check out the new Memory Overview feature of the SAP HANA studio. To access it, right click on a system in the Systems View, and select "Open Memory Overview" in the context menu, as follows:   
  
This will open the Memory Overview, which looks as follows:   
  
  
**Note:** To view the Memory Overview, you need Monitoring privileges. E.g. use the following SQL statement (replace 'youruser' with the actual user name): call GRANT\_ACTIVATED\_ROLE('sap.hana.admin.roles::Monitoring','youruser')

**Summary:**

SAP HANA maintains many system views and memory indicators, to provide a precise way to monitor and understand the SAP HANA memory utilization. The most important of these indicators is Used Memory and the corresponding historic snapshots. In turn, it is possible to drill down into very detailed reports of memory utilization using additional system views, or by using the convenient Memory Overview from the SAP HANA studio.   
  
Since SAP HANA contains its own memory manager and memory pool, external indicators, like the host-level Resident Memory size, or the process-level virtual and resident memory sizes, can be misleading when estimating the real memory requirements of a SAP HANA deployment.

# System Generated Schemas

A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects.

Types of Schemas

There are 3 types of schemas.

* 1. User Defined Schema
  2. System Defined Schema
  3. SLT Derived Schema

**User Defined Schema:**

These are created by user (DBA or System Administrator)

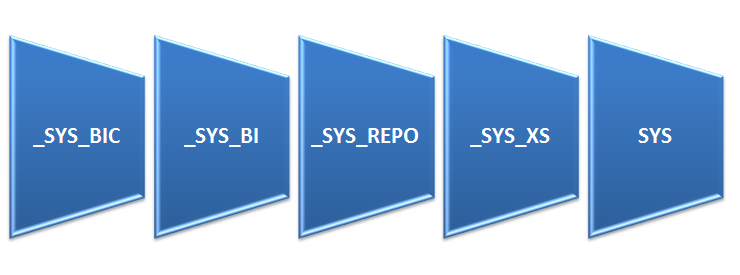
**SLT Derived Schema:**

When SLT is configured, it creates schema in HANA system. All the tables replicated into HANA system are contained in this schema

**System Defined Schema:**

These schemas are delivered with the SAP HANA database and contains HANA system information. There are system schemas like \_SYS\_BIC, \_SYS\_BI, \_SYS\_REPO, \_SYS\_STATISTICS etc.

System Generated Schemas

  
**\_SYS\_BIC:**

This schema contains all the columns views of activated objects. When the user activates the Attribute View/Analytic View/Calculation View/Analytic Privilege /Procedure, the respective run-time objects are created under \_SYS\_BIC/ Column Views.

**\_SYS\_REPO:**

Whatever the objects are there in the system is available in repository. This schema contains the list of Activated objects, Inactive Objects, Package details and Runtime Objects information etc.   
Also \_SYS\_REPO user must have SELECT privilege with grant option on the data schama.   
Read more about "GRANT SELECT PRIVILEGE ON \_SYS\_REPO"

**\_SYS\_BI:**

This schema stores all the metadata of created column Views. It contains the tables for created Variables, Time Data (Fiscal, Gregorian), Schema Mapping and Content Mapping tables.

**\_SYS\_STATISTICS:**

This schema contains all the system configurations and parameters.

**\_SYS\_XS:**

This schema is used for SAP HANA Extended Application Services.

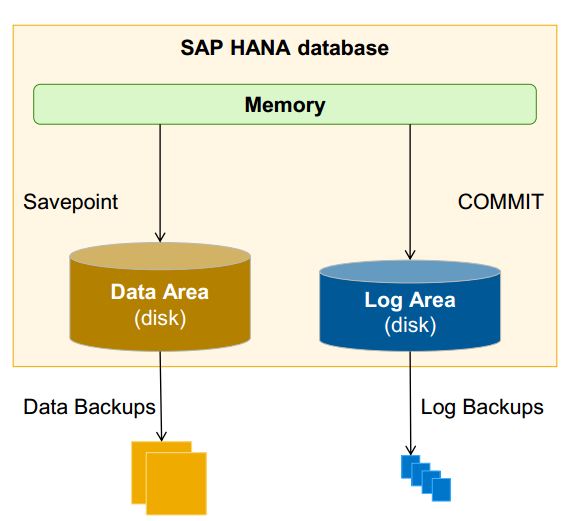
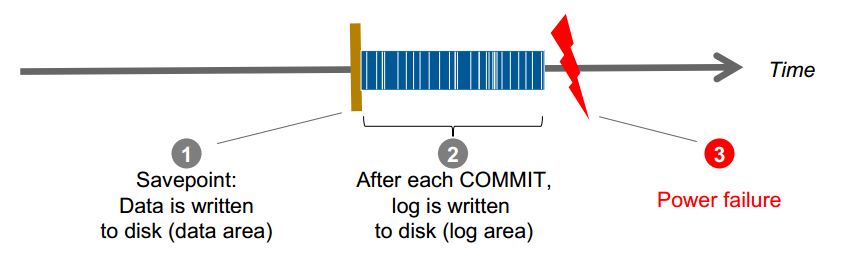
# Backup and Recovery

SAP HANA is an in-memory database. This means all the data is in RAM. As we all know that RAM is a volatile memory and all the data get lost when power goes down.   
This leads to a very obvious question:   
**What happens when power goes down in SAP HANA? Do we loose all the valuable data?**   
  
The answer is NO.  
SAP HANA is an in-memory database which means all the data resides in RAM. But **there is also a disc memory just for backup purpose**.   
  
In-memory computing is safe: The SAP HANA database holds the bulk of its data in memory for maximum performance, but still uses persistent storage (disk memory) to provide a fallback in case of failure.

Why Backup is Required?

In database technology, **atomicity, consistency, isolation, and durability (ACID)** is a set of requirements that guarantees that database transactions are processed reliably:   
A transaction has to be **atomic**. That is, if part of a transaction fails, the entire transaction has to fail and leave the database state unchanged.   
The **consistency** of a database must be preserved by the transactions that it performs.   
**Isolation** ensures that no transaction is able to interfere with another transaction.   
**Durability** means that after a transaction has been committed it will remain committed.   
  
While the first three requirements are not affected by the in-memory concept, durability is a requirement that cannot be met by storing data in main memory alone.   
Main memory is volatile storage. That is, it looses its content when it is out of electrical power. To make data persistent, it has to reside on non-volatile storage, such as hard drives, SSD, or Flash devices.

How Backup and Recovery Works in SAP HANA?

  
  
The main memory (RAM) in SAP HANA is divided into pages. When a transaction changes data, the corresponding pages are marked and written to disk storage in regular intervals.   
In addition, a database log captures all changes made by transactions. Each committed transaction generates a log entry that is written to disk storage. This ensures that all transactions are permanent.   
  
Figure below illustrates this. SAP HANA stores changed pages in savepoints, which are asynchronously written to disk storage in regular intervals (by default every 5 minutes).   
The log is written synchronously. That is, a transaction does not return before the corresponding log entry has been written to persistent storage, in order to meet the durability requirement, as described above.   
  
After a power failure, the database can be restarted like a disk-based database.   
The database pages are restored from the savepoints, and then the database logs are applied (rolled forward) to restore the changes that were not captured in the savepoints.   
This ensures that the database can be restored in memory to exactly the same state as before the power failure.   
  
  
**Data backup can be taken manually or can be scheduled.**

Few Important Concepts:

**What is Database Backup and Recovery**

Backup and Recovery is the process of copying/storing data for the specific purpose of restoring. Backing up files can protect against accidental loss of user data, database corruption, hardware failures, and even natural disasters.

**Savepoint:**

A savepoint is the point at which data is written to disk as backup. This is a point from which the Database Engine can start applying changes contained in the backup disk during recovery after an unexpected shutdown or crash.   
The database administrator determines the frequency of savepoints.

Data and Log:

**Data backups**

* + Contain the current payload of the data volumes (data and undo information)
  + Manual (SAP HANA studio, SQL commands), or scheduled (DBA Cockpit)

**Log backups**

* + Contain the content of closed log segments; the backup catalog is also written as a log backup
  + Automatic (asynchronous) whenever a log segment is full or the timeout for log backup has elapsed