# AP HANA - In-Memory Computing Engine

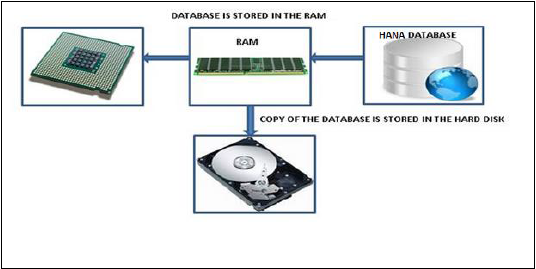
An In-Memory database means all the data from source system is stored in a RAM memory. In a conventional Database system, all data is stored in hard disk. SAP HANA In-Memory Database wastes no time in loading the data from hard disk to RAM. It provides faster access of data to multicore CPUs for information processing and analysis.

**Features of In-Memory Database**

The main features of SAP HANA in-memory database are −

* SAP HANA is Hybrid In-memory database.
* It combines row based, column based and Object Oriented base technology.
* It uses parallel processing with multicore CPU Architecture.
* Conventional Database reads memory data in 5 milliseconds. SAP HANA In-Memory database reads data in 5 nanoseconds.

It means, memory reads in HANA database are 1 million times faster than a conventional database hard disk memory reads.



Analysts want to see current data immediately in real time and do not want to wait for data until it is loaded to SAP BW system. SAP HANA In-Memory processing allows loading of real time data with use of various data provisioning techniques.

**Advantages of In-Memory Database**

* HANA database takes advantage of in-memory processing to deliver the fastest data-retrieval speeds, which is enticing to companies struggling with high-scale online transactions or timely forecasting and planning.
* Disk-based storage is still the enterprise standard and price of RAM has been declining steadily, so memory-intensive architectures will eventually replace slow, mechanical spinning disks and will lower the cost of data storage.
* In-Memory Column-based storage provides data compression up to 11 times, thus, reducing the storage space of huge data.
* This speed advantages offered by RAM storage system are further enhanced by the use of multi-core CPUs, multiple CPUs per node and multiple nodes per server in a distributed environment.

# SAP HANA - Studio

SAP HANA studio is an Eclipse-based tool. SAP HANA studio is both, the central development environment and the main administration tool for HANA system. Additional features are −

* It is a client tool, which can be used to access local or remote HANA system.
* It provides an environment for HANA Administration, HANA Information Modeling and Data Provisioning in HANA database.

SAP HANA Studio can be used on following platforms −

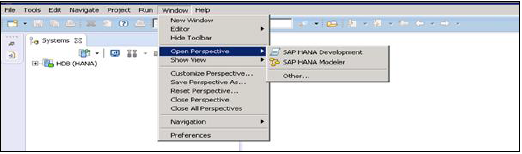
* Microsoft Windows 32 and 64 bit versions of: Windows XP, Windows Vista, Windows 7
* SUSE Linux Enterprise Server SLES11: x86 64 bit
* Mac OS, HANA studio client is not available

Depending on HANA Studio installation, not all features may be available. At the time of Studio installation, specify the features you want to install as per the role. To work on most recent version of HANA studio, Software Life Cycle Manager can be used for client update.

**SAP HANA Studio Perspectives / Features**

SAP HANA Studio provides perspectives to work on the following HANA features. You can choose Perspective in HANA Studio from the following option −

**HANA Studio → Window → Open Perspective → Other**



**Sap Hana Studio Administration**

Toolset for various administration tasks, excluding transportable design-time repository objects. General troubleshooting tools like tracing, the catalog browser and SQL Console are also included.

**SAP HANA Studio Database Development**

It provides Toolset for content development. It addresses, in particular, the DataMarts and ABAP on SAP HANA scenarios, which do not include SAP HANA native application development (XS).

**SAP HANA Studio Application Development**

SAP HANA system contains a small Web server, which can be used to host small applications. It provides Toolset for developing SAP HANA native applications like application code written in Java and HTML.

# SAP HANA - Studio Administration View

To Perform HANA Database Administration and monitoring features, SAP HANA Administration Console Perspective can be used.

Administrator Editor can be accessed in several ways −

* **From System View Toolbar** − Choose Open Administration default button
* **In System View** − Double Click on HANA System or Open Perspective



## HANA Studio: Administrator Editor

In Administration View: HANA studio provides multiple tabs to check configuration and health of the HANA system. Overview Tab tells General Information like, Operational Status, start time of first and last started service, version, build date and time, Platform, hardware manufacturer, etc.

## Adding a HANA System to Studio

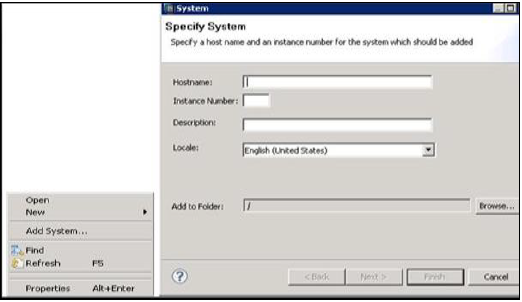
Single or multiple systems can be added to HANA studio for administration and information modeling purpose. To add new HANA system, host name, instance number and database user name and password is required.

* Port 3615 should be open to connect to Database
* Port 31015 Instance No 10
* Port 30015 Instance No 00
* SSh port should also be opened

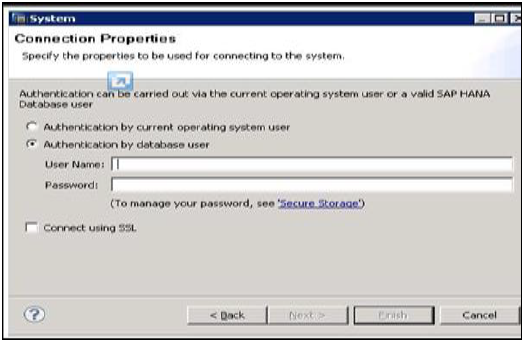
## Adding a System to Hana Studio

To add a system to HANA studio, follow the given steps.

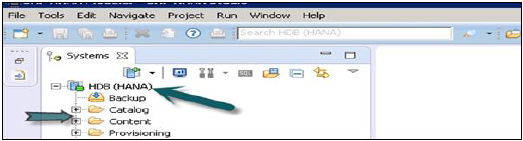
Right Click in Navigator space and click on Add System. Enter HANA system details, i.e. Host name & Instance number and click next.



Enter Database user name and password to connect to SAP HANA database. Click on Next and then Finish.



Once you click on Finish, HANA system will be added to System View for administration and modeling purpose. Each HANA system has two main sub-nodes, Catalog and Content.



## Catalog and Content

### Catalog

It contains all available Schemas i.e. all data structures, tables and data, Column views, Procedures that can be used in Content tab.

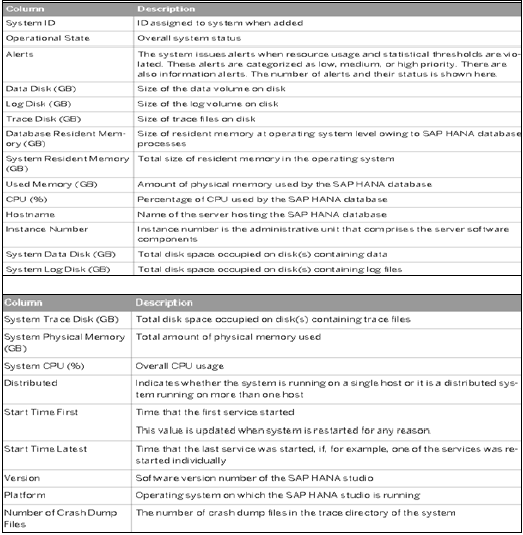
### Content

The Content tab contains design time repository, which holds all information of data models created with the HANA Modeler. These models are organized in Packages. The content node provides different views on same physical data.

# SAP HANA - System Monitor

System Monitor in HANA studio provides an overview of all your HANA system at a glance. From System Monitor, you can drill down into details of an individual system in Administration Editor. It tells about Data Disk, Log disk, Trace Disk, Alerts on resource usage with priority.

The following Information is available in System Monitor −



# SAP HANA - Information Modeler

AP HANA Information Modeler; also known as HANA Data Modeler is heart of HANA System. It enables to create modeling views at the top of database tables and implement business logic to create a meaningful report for analysis.

## Features of Information Modeler

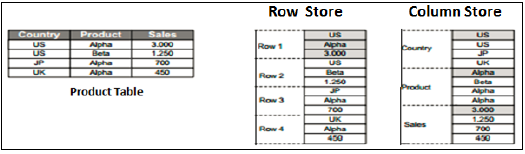
* Provides multiple views of transactional data stored in physical tables of HANA database for analysis and business logic purpose.
* Informational modeler only works for column based storage tables.
* Information Modeling Views are consumed by Java or HTML based applications or SAP tools like SAP Lumira or Analysis Office for reporting purpose.
* Also possible to use third party tools like MS Excel to connect to HANA and create reports.
* SAP HANA Modeling Views exploit real power of SAP HANA.

There are three types of Information Views, defined as −

* Attribute View
* Analytic View
* Calculation View

## Row vs Column Store

SAP HANA Modeler Views can only be created on the top of Column based tables. Storing data in Column tables is not a new thing. Earlier it was assumed that storing data in Columnar based structure takes more memory size and not performance Optimized.



With evolution of SAP HANA, HANA used column based data storage in Information views and presented the real benefits of columnar tables over Row based tables.

### Column Store

In a Column store table, Data is stored vertically. So, similar data types come together as shown in the example above. It provides faster memory read and write operations with help of In-Memory Computing Engine.

In a conventional database, data is stored in Row based structure i.e. horizontally. SAP HANA stores data in both row and Column based structure. This provides Performance optimization, flexibility and data compression in HANA database.

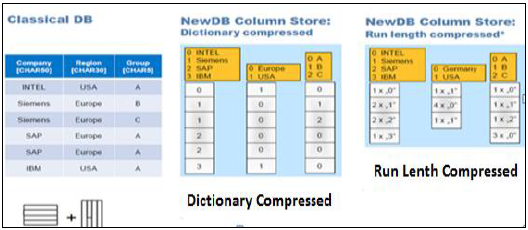
Storing Data in Columnar based table has following benefits −

* Data Compression
* Faster read and write access to tables as compared to conventional Row based storage
* Flexibility & parallel processing
* Perform Aggregations and Calculations at higher speed

There are various methods and algorithms how data can be stored in Column based structure- Dictionary Compressed, Run Length Compressed and many more.

In Dictionary Compressed, cells are stored in form of numbers in tables and numeral cells are always performance optimized as compared to characters.

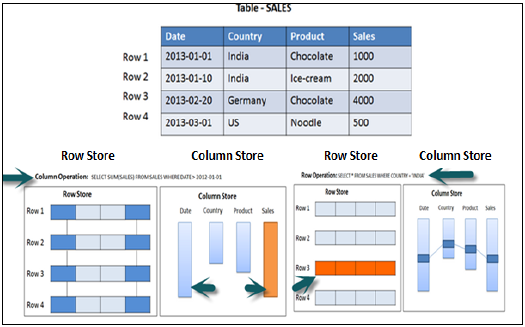
In Run length compressed, it saves the multiplier with cell value in numerical format and multiplier shows repetitive value in table.



## Functional Difference - Row vs Column Store

It is always advisable to use Column based storage, if SQL statement has to perform aggregate functions and calculations. Column based tables always perform better when running aggregate functions like Sum, Count, Max, Min.

Row based storage is preferred when output has to return complete row. The example given below makes it easy to understand.



In the above example, while running an Aggregate function (Sum) in sales column with Where clause, it will only use Date and Sales column while running SQL query so if it is column based storage table then it will be performance optimized, faster as data is required only from two columns.

While running a simple Select query, full row has to be printed in output so it is advisable to store table as Row based in this scenario.

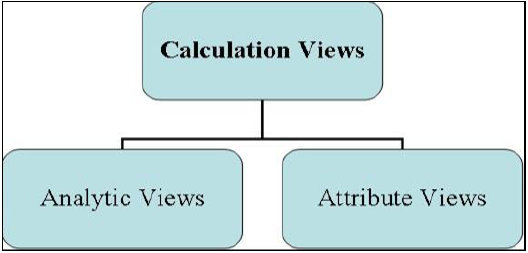
## Information Modeling Views

### Attribute View

Attributes are non-measurable elements in a database table. They represent master data and similar to characteristics of BW. Attribute Views are dimensions in a database or are used to join dimensions or other attribute views in modeling.

Important features are −

* Attribute views are used in Analytic and Calculation views.
* Attribute view represent master data.
* Used to filter size of dimension tables in Analytic and Calculation View.



### Analytic View

Analytic Views use power of SAP HANA to perform calculations and aggregation functions on the tables in database. It has at least one fact table that has measures and primary keys of dimension tables and surrounded by dimension tables contain master data.

Important features are −

* Analytic views are designed to perform Star schema queries.
* Analytic views contain at least one fact table and multiple dimension tables with master data and perform calculations and aggregations
* They are similar to Info Cubes and Info objects in SAP BW.
* Analytic views can be created on top of Attribute views and Fact tables and performs calculations like number of unit sold, total price, etc.

### Calculation Views

Calculation Views are used on top of Analytic and Attribute views to perform complex calculations, which are not possible with Analytic Views. Calculation view is a combination of base column tables, Attribute views and Analytic views to provide business logic.

Important features are −

* Calculation Views are defined either graphical using HANA Modeling feature or scripted in the SQL.
* It is created to perform complex calculations, which are not possible with other views- Attribute and Analytic views of SAP HANA modeler.
* One or more Attribute views and Analytic views are consumed with help of inbuilt functions like Projects, Union, Join, Rank in a Calculation View.

# SAP HANA - Modeling

SAP HANA Modeler option is used to create Information views on the top of schemas → tables in HANA database. These views are consumed by JAVA/HTML based applications or SAP Applications like SAP Lumira, Office Analysis or third party software like MS Excel for reporting purpose to meet business logic and to perform analysis and extract information.

HANA Modeling is done on the top of tables available in Catalog tab under Schema in HANA studio and all views are saved under Content table under Package.

You can create new Package under Content tab in HANA studio using right click on Content and New.

All Modeling Views created inside one package comes under the same package in HANA studio and categorized according to View Type.

Each View has different structure for Dimension and Fact tables. Dim tables are defined with master data and Fact table has Primary Key for dimension tables and measures like Number of Unit sold, Average delay time, Total Price, etc.

## Fact and Dimension Table

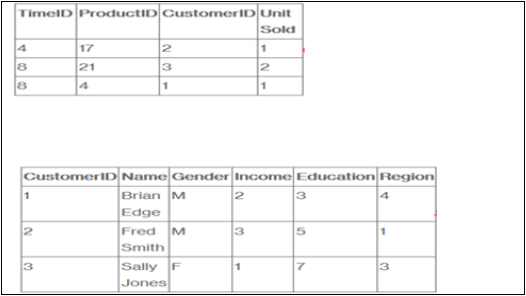
Fact Table contains Primary Keys for Dimension table and measures. They are joined with Dimension tables in HANA Views to meet business logic.

**Example of Measures** − Number of unit sold, Total Price, Average Delay time, etc.

Dimension Table contains master data and is joined with one or more fact tables to make some business logic. Dimension tables are used to create schemas with fact tables and can be normalized.

**Example of Dimension Table** − Customer, Product, etc.

Suppose a company sells products to customers. Every sale is a fact that happens within the company and the fact table is used to record these facts.



For example, row 3 in the fact table records the fact that customer 1 (Brian) bought one item on day 4. And, in a complete example, we would also have a product table and a time table so that we know what she bought and exactly when.

The fact table lists events that happen in our company (or at least the events that we want to analyze- No of Unit Sold, Margin, and Sales Revenue). The Dimension tables list the factors (Customer, Time, and Product) by which we want to analyze the data.

# SAP HANA - Schema in Data Warehouse

Schemas are logical description of tables in Data Warehouse. Schemas are created by joining multiple fact and Dimension tables to meet some business logic.

Database uses relational model to store data. However, Data Warehouse use Schemas that join dimensions and fact tables to meet business logic. There are three types of Schemas used in a Data Warehouse −

* Star Schema
* Snowflakes Schema
* Galaxy Schema

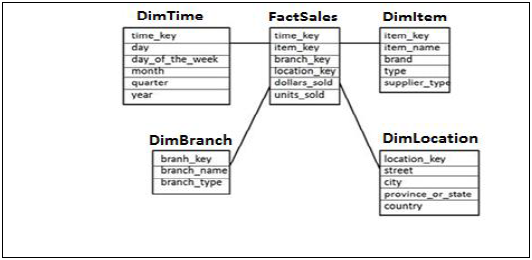
**Star Schema**

In Star Schema, Each Dimension is joined to one single Fact table. Each Dimension is represented by only one dimension and is not further normalized.

Dimension Table contains set of attribute that are used to analyze the data.

**Example** − In example given below, we have a Fact table FactSales that has Primary keys for all the Dim tables and measures units\_sold and dollars\_ sold to do analysis.

We have four Dimension tables − DimTime, DimItem, DimBranch, DimLocation



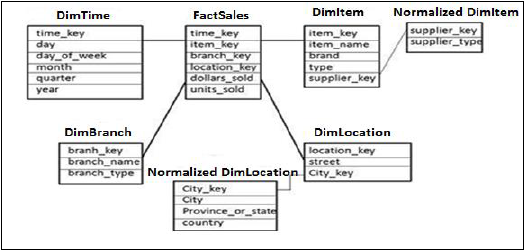
Each Dimension table is connected to Fact table as Fact table has Primary Key for each Dimension Tables that is used to join two tables.

Facts/Measures in Fact Table are used for analysis purpose along with attribute in Dimension tables.

**Snowflakes Schema**

In Snowflakes schema, some of Dimension tables are further, normalized and Dim tables are connected to single Fact Table. Normalization is used to organize attributes and tables of database to minimize the data redundancy.

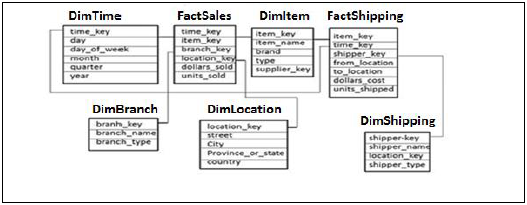
Normalization involves breaking a table into less redundant smaller tables without losing any information and smaller tables are joined to Dimension table.



In the above example, DimItem and DimLocation Dimension tables are normalized without losing any information. This is called Snowflakes schema where dimension tables are further normalized to smaller tables.

**Galaxy Schema**

In Galaxy Schema, there are multiple Fact tables and Dimension tables. Each Fact table stores primary keys of few Dimension tables and measures/facts to do analysis.



In the above example, there are two Fact tables FactSales, FactShipping and multiple Dimension tables joined to Fact tables. Each Fact table contains Primary Key for joined Dim tables and measures/Facts to perform analysis.

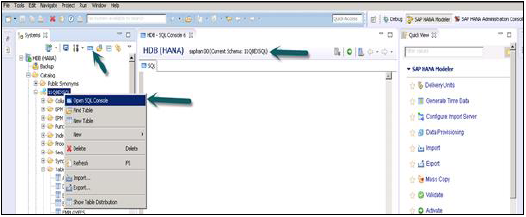
# SAP HANA - Tables

Tables in HANA database can be accessed from HANA Studio in Catalogue tab under Schemas. New tables can be created using the two methods given below −

* Using SQL editor
* Using GUI option

## SQL Editor in HANA Studio

SQL Console can be opened by selecting Schema name, in which, new table has to be created using System View SQL Editor option or by Right click on Schema name as shown below −



Once SQL Editor is opened, Schema name can be confirmed from the name written on the top of SQL Editor. New table can be created using SQL Create Table statement −

Create column Table Test1 (

ID INTEGER,

NAME VARCHAR(10),

PRIMARY KEY (ID)

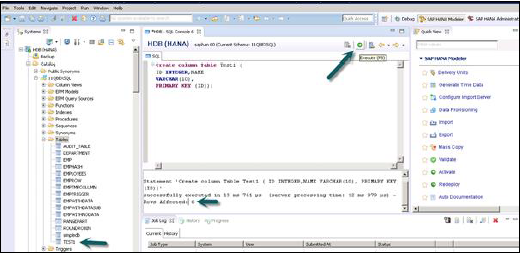
);

In this SQL statement, we have created a Column table “Test1”, defined data types of table and Primary Key.

Once you write Create table SQL query, click on Execute option on top of SQL editor right side. Once the statement is executed, we will get a confirmation message as shown in snapshot given below −

Statement 'Create column Table Test1 (ID INTEGER,NAME VARCHAR(10), PRIMARY KEY (ID))'

successfully executed in 13 ms 761 μs (server processing time: 12 ms 979 μs) − Rows Affected: 0



Execution statement also tells about the time taken to execute the statement. Once statement is successfully executed, right click on Table tab under Schema name in System View and refresh. New Table will be reflected in the list of tables under Schema name.

Insert statement is used to enter the data in the Table using SQL editor.

Insert into TEST1 Values (1,'ABCD')

Insert into TEST1 Values (2,'EFGH');

Click on Execute.

You can right click on Table name and use Open Data Definition to see data type of the table. Open Data Preview/Open Content to see table contents.

## Creating Table using GUI Option

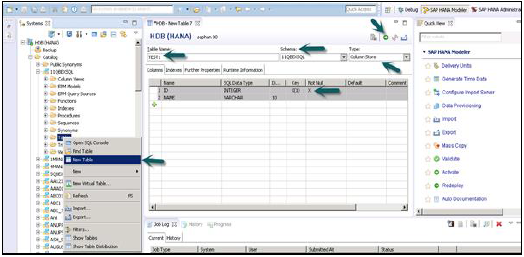
Another way to create a table in HANA database is by using GUI option in HANA Studio.

Right Click on Table tab under Schema → Select ‘New Table’ option as shown in snapshot given below.

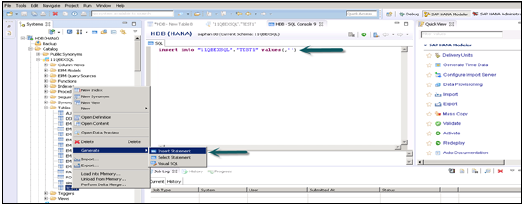
Once you click on New Table → It will open a window to enter the Table name, Choose Schema name from drop down, Define Table type from drop down list: Column Store or Row Store.

Define data type as shown below. Columns can be added by clicking on + sign, Primary Key can be chosen by clicking on cell under Primary key in front of Column name, Not Null will be active by default.

Once columns are added, click on Execute.



Once you Execute (F8), Right Click on Table Tab → Refresh. New Table will be reflected in the list of tables under chosen Schema. Below Insert Option can be used to insert data in table. Select statement to see content of table.



### Inserting Data in a table using GUI in HANA studio

You can right click on Table name and use Open Data Definition to see data type of the table. Open Data Preview/Open Content to see table contents.

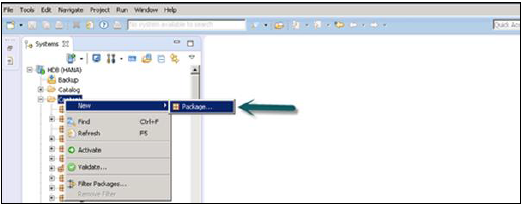
To use tables from one Schema to create views we should provide access on the Schema to the default user who runs all the Views in HANA Modeling. This can be done by going to SQL editor and running this query −

**GRANT SELECT ON SCHEMA "<SCHEMA\_NAME>" TO \_SYS\_REPO WITH GRANT OPTION**

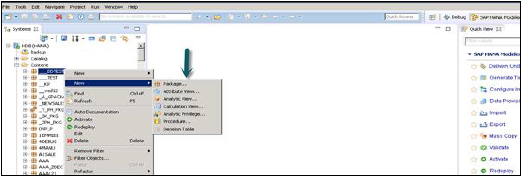
# SAP HANA - Packages

SAP HANA Packages are shown under Content tab in HANA studio. All HANA modeling is saved inside Packages.

You can create a new Package by Right Click on Content Tab → New → Package



You can also create a Sub Package under a Package by right clicking on the Package name. When we right click on the Package we get 7 Options: We can create HANA Views Attribute Views, Analytical Views, and Calculation Views under a Package.



You can also create Decision Table, Define Analytic Privilege and create Procedures in a Package.

When you right click on Package and click on New, you can also create sub packages in a Package. You have to enter Package Name, Description while creating a Package.

# SAP HANA - SQL Overview

SQL stands for Structured Query Language.

It is a standardized language for communicating with a database. SQL is used to retrieve the data, store or manipulate the data in the database.

SQL statements perform the following functions −

* Data definition and manipulation
* System management
* Session management
* Transaction management
* Schema definition and manipulation

The set of SQL extensions, which allow developers to push data into database, is called **SQL scripts**.

## Data Manipulation Language (DML)

DML statements are used for managing data within schema objects. Some examples −

* **SELECT** − retrieve data from the database
* **INSERT** − insert data into a table
* **UPDATE** − updates existing data within a table

## Data Definition Language (DDL)

DDL statements are used to define the database structure or schema. Some examples −

* **CREATE** − to create objects in the database
* **ALTER** − alters the structure of the database
* **DROP** − delete objects from the database

## Data Control Language (DCL)

Some examples of DCL statements are −

* **GRANT** − gives user's access privileges to database
* **REVOKE** − withdraw access privileges given with the GRANT command

## Why do We Need SQL?

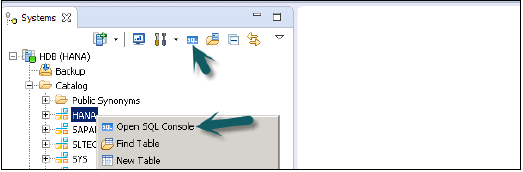
When we create Information Views in SAP HANA Modeler, we are creating it on top of some OLTP applications. All these in back end run on SQL. Database understands only this language.

To do a testing if our report will meet the business requirement we have to run SQL statement in database if Output is according to the requirement.

HANA Calculation views can be created in two ways - Graphical or using SQL script. When we create more complex Calculation views, then we might have to use direct SQL scripts.

### How to open SQL console in HANA Studio?

Select the HANA system and click on SQL console option in system view. You can also open SQL console by right click on Catalog tab or any on any Schema name.



SAP HANA can act both as Relational as well as OLAP database. When we use BW on HANA, then we create cubes in BW and HANA, which act as relational database and always produce a SQL Statement. However, when we directly access HANA views using OLAP connection, then it will act as OLAP database and MDX will be generated.

# SAP HANA - Data Types

You can create row or Column store tables in SAP HANA using create table option. A table can be created by executing a data definition create table statement or using graphical option in HANA studio.

When you create a table, you also need to define attributes inside it.

**SQL statement to create a table in HANA Studio SQL Console** −

Create column Table TEST (

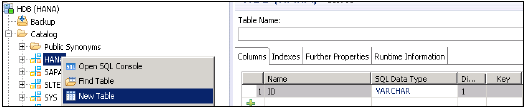
ID INTEGER,

NAME VARCHAR(10),

PRIMARY KEY (ID)

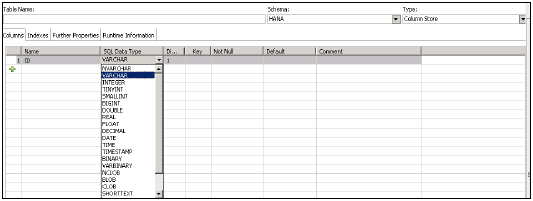
);

**Creating a table in HANA studio using GUI option** −



When you create a table, you need to define the names of columns and SQL data types. The Dimension field tells the length of value and the Key option to define it as primary key.

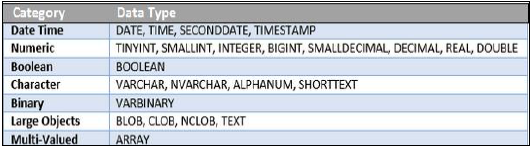
SAP HANA supports the following data types in a table −



SAP HANA supports 7 categories of SQL data types and it depends on the type of data you have to store in a column.

* Numeric
* Character/ String
* Boolean
* Date Time
* Binary
* Large Objects
* Multi-Valued

The following table gives the list of data types in each category −



**Date Time**

These data types are used to store date and time in a table in HANA database.

* **DATE** − data type consists of year, month and day information to represent a date value in a column. Default format for a Date data type is YYYY-MM-DD.
* **TIME** − data type consists of hours, minutes, and seconds value in a table in HANA database. Default format for Time data type is HH: MI: SS.
* **SECOND DATE** − data type consists of year, month, day, hour, minute, second value in a table in HANA database. Default format for SECONDDATE data type is YYYY-MM-DD HH:MM:SS.
* **TIMESTAMP** − data type consists of date and time information in a table in HANA database. Default format for TIMESTAMP data type is YYYY-MM-DD HH:MM:SS:FFn, where FFn represents fraction of second.

**Numeric**

* **TinyINT** − stores 8 bit unsigned integer. Min value: 0 and max value: 255
* **SMALLINT** − stores 16 bit signed integer. Min value: -32,768 and max value: 32,767
* **Integer** − stores 32 bit signed integer. Min value: -2,147,483,648 and max value: 2,147,483,648
* **BIGINT** − stores 64 bit signed integer. Min value: -9,223,372,036,854,775,808 and max value: 9,223,372,036,854,775,808
* **SMALL** − Decimal and Decimal: Min value: -10^38 +1 and max value: 10^38 -1
* **REAL** − Min Value:-3.40E + 38 and max value: 3.40E + 38
* **DOUBLE** − stores 64 bit floating point number. Min value: -1.7976931348623157E308 and max value: 1.7976931348623157E308

**Boolean**

Boolean data types stores Boolean value, which are TRUE, FALSE

**Character**

* **Varchar** − maximum of 8000 characters.
* **Nvarchar** − maximum length of 4000 characters
* **ALPHANUM** − stores alphanumeric characters. Value for an integer is between 1 to 127.
* **SHORTTEXT** − stores variable length character string which supports text search features and string search features.

**Binary**

Binary types are used to store bytes of binary data.

**VARBINARY** − stores binary data in bytes. Max integer length is between 1 and 5000.

**Large Objects**

LARGEOBJECTS are used to store a large amount of data such as text documents and images.

* **NCLOB** − stores large UNICODE character object.
* **BLOB** − stores large amount of Binary data.
* **CLOB** − stores large amount of ASCII character data.
* **TEXT** − it enables text search features. This data type can be defined for only column tables and not for row store tables.
* **BINTEXT** − supports text search features but it is possible to insert binary data.

**Multivalued**

Multivalued data types are used to store collection of values with same data type.

**Array**

Arrays store collections of value with the same data type. They can also contain null values.

# SAP HANA - SQL Operators

An operator is a special character used primarily in SQL statement's with WHERE clause to perform operation, such as comparisons and arithmetic operations. They are used to pass conditions in a SQL query.

Operator types given below can be used in SQL statements in HANA −

* Arithmetic Operators
* Comparison/Relational Operators
* Logical Operators
* Set Operators

**Arithmetic Operators**

Arithmetic operators are used to perform simple calculation functions like addition, subtraction, multiplication, division and percentage.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition − Adds values on either side of the operator |
| - | Subtraction − Subtracts right hand operand from left hand operand |
| \* | Multiplication − Multiplies values on either side of the operator |
| / | Division − Divides left hand operand by right hand operand |
| % | Modulus − Divides left hand operand by right hand operand and returns remainder |

**Comparison Operators**

Comparison operators are used to compare the values in SQL statement.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Checks if the values of two operands are equal or not, if yes then condition becomes true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. |
| <> | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. |
| !< | Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true. |
| !> | Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true. |

**Logical operators**

Logical operators are used to pass multiple conditions in SQL statement or are used to manipulate the results of conditions.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| ALL | The ALL Operator is used to compare a value to all values in another value set. |
| AND | The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. |
| ANY | The ANY operator is used to compare a value to any applicable value in the list according to the condition. |
| BETWEEN | The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. |
| EXISTS | The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria. |
| IN | The IN operator is used to compare a value to a list of literal values that have been specified. |
| LIKE | The LIKE operator is used to compare a value to similar values using wildcard operators. |
| NOT | The NOT operator reverses the meaning of the logical operator with which it is used. Eg − NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator**. |
| OR | The OR operator is used to compare multiple conditions in an SQL statement's WHERE clause. |
| IS NULL | The NULL operator is used to compare a value with a NULL value. |
| UNIQUE | The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates). |

**Set Operators**

Set operators are used to combine results of two queries into a single result. Data type should be same for both the tables.

* **UNION** − It combines the results of two or more Select statements. However it will eliminate duplicate rows.
* **UNION ALL** − This operator is similar to Union but it also shows the duplicate rows.
* **INTERSECT** − Intersect operation is used to combine the two SELECT statements, and it returns the records, which are common from both SELECT statements. In case of Intersect, the number of columns and datatype must be same in both the tables.
* **MINUS** − Minus operation combines result of two SELECT statements and return only those results, which belong to first set of result and eliminate the rows in second statement from the output of first.

# SAP HANA - SQL Functions

There are various SQL functions provided by SAP HANA database −

* Numeric Functions
* String Functions
* Fulltext Functions
* Datetime Functions
* Aggregate Functions
* Data Type Conversion Functions
* Window Functions
* Series Data Functions
* Miscellaneous Functions

**Numeric Functions**

These are inbuilt numeric functions in SQL and use in scripting. It takes numeric values or strings with numeric characters and return numeric values.

* **ABS** − It returns the absolute value of a numeric argument.

Example − SELECT ABS (-1) "abs" FROM TEST;

abs

1

ACOS, ASIN, ATAN, ATAN2 (These functions return trigonometric value of the argument)

* **BINTOHEX** − It converts a Binary value to a hexadecimal value.
* **BITAND** − It performs an AND operation on bits of passed argument.
* **BITCOUNT** − It performs the count of number of set bits in an argument.
* **BITNOT** − It performs a bitwise NOT operation on the bits of argument.
* **BITOR** − It perform an OR operation on bits of passed argument.
* **BITSET** − It is used to set bits to 1 in <target\_num> from the <start\_bit> position.
* **BITUNSET** − It is used to set bits to 0 in <target\_num> from the <start\_bit> position.
* **BITXOR** − It performs XOR operation on bits of passed argument.
* **CEIL** − It returns the first integer that is greater or equal to the passed value.
* COS, COSH, COT ((These functions return trigonometric value of the argument)
* **EXP** − It returns the result of the base of natural logarithms e raised to the power of passed value.
* **FLOOR** − It returns the largest integer not greater than the numeric argument.
* **HEXTOBIN** − It converts a hexadecimal value to a binary value.
* **LN** − It returns the natural logarithm of the argument.
* **LOG** − It returns the algorithm value of a passed positive value. Both base and log value should be positive.

Various other numeric functions can also be used − MOD, POWER, RAND, ROUND, SIGN, SIN, SINH, SQRT, TAN, TANH, UMINUS

**String Functions**

Various SQL string functions can be used in HANA with SQL scripting. Most common string functions are −

* **ASCII** − It returns integer ASCII value of passed string.
* **CHAR** − It returns the character associated with passed ASCII value.
* **CONCAT** − It is Concatenation operator and returns the combined passed strings.
* **LCASE** − It converts all character of a string to Lower case.
* **LEFT** − It returns the first characters of a passed string as per mentioned value.
* **LENGTH** − It returns the number of characters in passed string.
* **LOCATE** − It returns the position of substring within passed string.
* **LOWER** − It converts all characters in string to lowercase.
* **NCHAR** − It returns the Unicode character with passed integer value.
* **REPLACE** − It searches in passed original string for all occurrences of search string and replaces them with replace string.
* **RIGHT** − It returns the rightmost passed value characters of mentioned string.
* **UPPER** − It converts all characters in passed string to uppercase.
* **UCASE** − It is identical to UPPER function. It converts all characters in passed string to uppercase.

Other string functions that can be used are − LPAD, LTRIM, RTRIM, STRTOBIN, SUBSTR\_AFTER, SUBSTR\_BEFORE, SUBSTRING, TRIM, UNICODE, RPAD, BINTOSTR

**Date Time functions**

There are various Date Time functions that can be used in HANA in SQL scripts. Most common Date Time functions are −

* **CURRENT\_DATE** − It returns the current local system date.
* **CURRENT\_TIME** − It returns the current local system time.
* **CURRENT\_TIMESTAMP** − It returns the current local system timestamp details (YYYY-MM-DD HH:MM:SS:FF).
* **CURRENT\_UTCDATE** − It returns current UTC (Greenwich Mean date) date.
* **CURRENT\_UTCTIME** − It returns current UTC (Greenwich Mean Time) time.
* CURRENT\_UTCTIMESTAMP
* **DAYOFMONTH** − It returns the integer value of day in passed date in argument.
* **HOUR** − It returns integer value of hour in passed time in argument.
* **YEAR** − It returns the year value of passed date.

Other Date Time functions are − DAYOFYEAR, DAYNAME, DAYS\_BETWEEN, EXTRACT, NANO100\_BETWEEN, NEXT\_DAY, NOW, QUARTER, SECOND, SECONDS\_BETWEEN, UTCTOLOCAL, WEEK, WEEKDAY, WORKDAYS\_BETWEEN, ISOWEEK, LAST\_DAY, LOCALTOUTC, MINUTE, MONTH, MONTHNAME, ADD\_DAYS, ADD\_MONTHS, ADD\_SECONDS, ADD\_WORKDAYS

**Data Type Conversion Functions**

These functions are used to convert one data type to other or to perform a check if conversion is possible or not.

Most common data type conversion functions used in HANA in SQL scripts −

* **CAST** − It returns the value of an expression converted to a supplied data type.
* **TO\_ALPHANUM** − It converts a passed value to an ALPHANUM data type
* **TO\_REAL** − It converts a value to a REAL data type.
* **TO\_TIME** − It converts a passed time string to the TIME data type.
* **TO\_CLOB** − It converts a value to a CLOB data type.

Other similar Data Type conversion functions are − TO\_BIGINT, TO\_BINARY, TO\_BLOB, TO\_DATE, TO\_DATS, TO\_DECIMAL, TO\_DOUBLE, TO\_FIXEDCHAR, TO\_INT, TO\_INTEGER, TO\_NCLOB, TO\_NVARCHAR, TO\_TIMESTAMP, TO\_TINYINT, TO\_VARCHAR, TO\_SECONDDATE, TO\_SMALLDECIMAL, TO\_SMALLINT

There are also various Windows and other miscellaneous functions that can be used in HANA SQL scripts.

* **Current\_Schema** − It returns a string containing the current schema name.
* **Session\_User** − It returns the user name of current session

# SAP HANA - SQL Expressions

An Expression is used to evaluate a clause to return values. There are different SQL expressions that can be used in HANA −

* Case Expressions
* Function Expressions
* Aggregate Expressions
* Subqueries in Expressions

## Case Expression

This is used to pass multiple conditions in a SQL expression. It allows the use of IF-ELSE-THEN logic without using procedures in SQL statements.

### Example

SELECT COUNT( CASE WHEN sal < 2000 THEN 1 ELSE NULL END ) count1,

COUNT( CASE WHEN sal BETWEEN 2001 AND 4000 THEN 1 ELSE NULL END ) count2,

COUNT( CASE WHEN sal > 4000 THEN 1 ELSE NULL END ) count3 FROM emp;

This statement will return count1, count2, count3 with integer value as per passed condition.

## Function Expressions

Function expressions involve SQL inbuilt functions to be used in Expressions.

## Aggregate Expressions

Aggregate functions are used to perform complex calculations like Sum, Percentage, Min, Max, Count, Mode, Median, etc. Aggregate Expression uses Aggregate functions to calculate single value from multiple values.

**Aggregate Functions** − Sum, Count, Minimum, Maximum. These are applied on measure values (facts) and It is always associated with a dimension.

Common aggregate functions include −

* Average ()
* Count ()
* Maximum ()
* Median ()
* Minimum ()
* Mode ()
* Sum ()

## Subqueries in Expressions

A subquery as an expression is a Select statement. When it is used in an expression, it returns a zero or a single value.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow −

* Subqueries must be enclosed within parentheses.
* A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
* An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.
* Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.
* The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
* A subquery cannot be immediately enclosed in a set function.
* The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

### Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows −

### Example

SELECT \* FROM CUSTOMERS

WHERE ID IN (SELECT ID

FROM CUSTOMERS

WHERE SALARY > 4500) ;

+----+----------+-----+---------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+---------+----------+

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+---------+----------+