



## SAP HANA Modeling Guide

- SAP HANA Appliance Software SPS4

2012-05-09

## Copyright

© 2012 SAP AG. All rights reserved. SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company. Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company. Crossgate, m@gic EDDY, B2B 360°, B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary. These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

2012-05-09

# Contents

<b>Chapter 1</b>	<b>Concepts.....</b>	<b>5</b>
1.1	Content.....	5
1.2	Attributes and Measures.....	8
1.3	Hierarchies.....	9
1.4	Users and Roles.....	10
<b>Chapter 2</b>	<b>Overview of Modeling Data.....</b>	<b>13</b>
<b>Chapter 3</b>	<b>Adding a System.....</b>	<b>15</b>
<b>Chapter 4</b>	<b>Importing Metadata.....</b>	<b>17</b>
4.1	Configuring Data Services for Metadata Import.....	17
4.2	Creating Schemas.....	18
4.3	Importing Table Definitions.....	18
4.4	Adding Connections.....	20
<b>Chapter 5</b>	<b>Loading Data into Table Definitions.....</b>	<b>21</b>
5.1	Suspending Data Load.....	21
5.2	Resuming Data Load.....	22
5.3	Uploading Data from Flat Files.....	23
<b>Chapter 6</b>	<b>Copying Contents Delivered by SAP.....</b>	<b>25</b>
<b>Chapter 7</b>	<b>Mapping the Authoring Schema to Physical Schema at the Target .....</b>	<b>27</b>
<b>Chapter 8</b>	<b>Setting Preferences for Modeler.....</b>	<b>29</b>
<b>Chapter 9</b>	<b>Creating Content Objects.....</b>	<b>33</b>
9.1	Creating a Package.....	35
9.2	Creating an Attribute View.....	36

9.3	Creating an Analytic View.....	39
9.4	Creating a Calculation View .....	42
9.4.1	Managing Attribute Mappings.....	47
9.4.2	Constant Column.....	47
9.5	Creating an Analytic Privilege.....	48
9.6	Creating a Procedure.....	50
9.7	Using Currency and Unit of Measure.....	51
9.8	Assigning Variables.....	53
9.9	Searching Related Tables.....	54
9.10	Using System Generated Joins.....	55
9.11	Activating Objects.....	55
9.12	Creating an Input Parameter.....	56
9.13	Creating a Hierarchy.....	58
9.14	Maintaining Search Attributes.....	59
9.15	Managing Data Foundation Layout.....	60
9.16	SQL Functions on Content Assist.....	60
9.17	Functions used in Expressions.....	62
<b>Chapter 10</b>	<b>Processing Content Objects.....</b>	<b>71</b>
10.1	Previewing Data of Content Objects.....	71
10.2	Validating Models .....	71
10.3	Comparing Versions of Content Objects.....	72
10.4	Viewing Version History of Content Objects.....	72
10.5	Switching Ownership of Inactive Objects.....	72
10.6	Generating Object Documentation.....	73
10.7	Enabling Multilanguage Support for Objects.....	74
10.8	Checking Model References.....	75
10.9	Viewing the Job Log.....	75
<b>Chapter 11</b>	<b>Exporting Objects.....</b>	<b>77</b>
11.1	Creating a Delivery Unit.....	77
11.2	Exporting Objects using Delivery Unit.....	78
11.3	Exporting Objects using Developer Mode.....	78
11.4	Exporting Objects using SAP Support Mode.....	79
<b>Chapter 12</b>	<b>Importing Objects.....</b>	<b>81</b>
12.1	Importing Objects using Delivery Unit.....	81
12.2	Importing Objects using Developer Mode.....	81

# Concepts

## 1.1 Content

Information views are created to model various slices of the data stored in an SAP HANA database.

Information views use various combinations of content data (that is, non-metadata) to model a business use case. Content data can be classified as follows:

Attribute - Represents the descriptive data like customer ID, city, country, and so on.

Measure - Represents the quantifiable data such as revenue, quantity sold, counters, and so on.

Information views are often used for analytical use cases such as operational data mart scenarios or multidimensional reporting on revenue, profitability, and so on.

There are three types of information views: attribute view, analytic view, and calculation view. All three types of information views are non-materialized views. This creates agility through the rapid deployment of changes.

### **Attribute View**

An attribute view is used to model an entity based on the relationships between attribute data contained in multiple source tables.

For example, customer ID is the attribute data that describes measures (that is, who purchased a product). However, customer ID has much more depth to it when joined with other attribute data that further describes the customer (customer address, customer relationship, customer status, customer hierarchy, and so on).

You create an attribute view to locate the attribute data and to define the relationships between the various tables to model how customer attribute data, for example, will be used to address business needs.

You can model the following elements within an attribute view:

- Simple attributes
- Calculated attributes
- Hierarchies

#### **Note:**

For more information about the attributes and hierarchies mentioned above, see sections 1.2 and 1.3.

You can choose to further fine-tune the behavior of the attributes of an attribute view by setting the properties as follows:

- Filters to restrict values that are selected when using the attribute view.
- Attributes can be defined as *Hidden* so that they can be used in processes but are not visible to end users.
- Attributes can be defined as key attributes and used when joining multiple tables.
- The *Drill Down Enabled* property can be used to indicate if an attribute is available for further drill down when consumed.

Attribute views can later be joined to tables that contain measures within the definition of an analytic view or a calculation view to create virtual star schema on the SAP HANA data.

### **Analytic View**

An analytic view is used to model data that includes measures.

For example, an operational data mart representing sales order history would include measures for quantity, price, and so on.

The data foundation of an analytic view can contain multiple tables. However, measures that are selected for inclusion in an analytic view must originate from only one of these tables (for business requirements that include measure sourced from multiple source tables, see calculation view ).

Analytic views can be simply a combination of tables that contain both attribute data and measure data. For example, a report requiring the following:

```
Customer_ID Order_Number Product_ID Quantity_Ordered Quantity_Shipped
```

Optionally, attribute views can also be included in the analytic view definition. In this way, you can achieve additional depth of attribute data can be achieved. The analytic view inherits the definitions of any attribute views that are included in the definition. For example:

```
Customer_ID/Customer_Name Order_Number  
Product_ID/Product_Name/Product_Hierarchy Quantity_Ordered Quantity_Shipped
```

You can model the following elements within an analytic view:

- Simple attributes
- Calculated attributes
- Private attributes
- Simple measures
- Calculated measures
- Restricted measures

#### **Note:**

For more information about the attributes and measures mentioned above, see section 1.2.

- Variables
- Input parameters

#### **Note:**

For more information about the variables and input parameters mentioned above, see sections 9.8 and 9.12.

You can choose to further fine-tune the behavior of the attributes and measures of an analytic view by setting the properties as follows:

- Filters to restrict values that are selected when using the analytic view.
- Attributes can be defined as *Hidden* so that they are able to be used in processes but are not viewable to end users.
- Attributes can be defined as key attribute and used when joining multiple tables.
- The *Drill Down Enabled* property can be used to indicate if an attribute is available for further drill down when consumed.
- Aggregation type on measures
- *Currency* and *Unit of Measure* parameters

### **Calculation View**

A calculation view is used to define more advanced slices on the data in SAP HANA database. Calculation views can be simple and mirror the functionality found in both attribute views and analytic views. However, they are typically used when the business use case requires advanced logic that is not covered in the previous types of information views.

For example, calculation views can have layers of calculation logic, can include measures sourced from multiple source tables, can include advanced SQL logic, and so on. The data foundation of the calculation view can include any combination of tables, column views, attribute views and analytic views. You can create joins, unions, projections, and aggregation levels on the sources.

You can model the following elements within a calculation view:

- Simple attributes
- Calculated attributes
- Private attributes
- Simple measures
- Calculated measures
- Restricted measures
- Counters
- Hierarchies (created outside of the attribute view)

**Note:**

For more information about the the attributes, measures, counters, and hierarchies mentioned above, see sections 1.2 and 1.3.

- Variables
- Input parameters

**Note:**

For more information about the variables and input parameters mentioned above, see sections 9.8 and 9.12.

Calculation views can include measures and be used for multi-dimensional reporting or can contain no measures and used for list-type of reporting. Calculation views can either be created using a graphical editor or using a SQL editor. These various options provide maximum flexibility for the most complex and comprehensive business requirements.

## 1.2 Attributes and Measures

### Attributes

Attributes are individual non-measurable analytical elements.

- **Simple Attributes**

Simple attributes are individual non-measurable analytical elements that are derived from the data foundation.

For example, PRODUCT\_ID and PRODUCT\_NAME are attributes of a PRODUCT subject area.

- **Calculated Attributes**

Calculated attributes are derived from one or more existing attributes or constants.

For example, deriving the full name of a customer (first and last name), assigning a constant value to an attribute that can be used for arithmetic calculations.

- **Private Attributes**

Private attributes used in an analytic view allow you to customize the behavior of an attribute for only that view.

For example, if an analytic view or a calculation view include an attribute view, it inherits the behavior of the attributes from the attribute view (set the parameter once and it is replicated in all views consuming it).

By contrast, if you create an analytic view for one specific use case in which you want a particular attribute to behave differently than it does in the attribute view to which it belongs, you can define it as a private attribute.

### Measures

Measures are simple measurable analytical elements. Measures are derived from analytic and calculation views.

- **Simple Measures**

A simple measure is a measurable analytical element that is derived from the data foundation.

For example, PROFIT.

- **Calculated Measure**

Calculated measures are defined based on a combination of data from OLAP cubes, arithmetic operators, constants, and functions.

For example, calculated measures can be used to calculate the total sales of a product across five regions, or to assign a constant value to a measure for a calculation.

- **Restricted Measure**

Restricted measures are used to filter the value based on the user-defined rules for the attribute values.

- **Counters**

Counters add a new measure to the calculation view definition to count the recurrence of an attribute. For example, to count how many times Product appears.

**Note:**

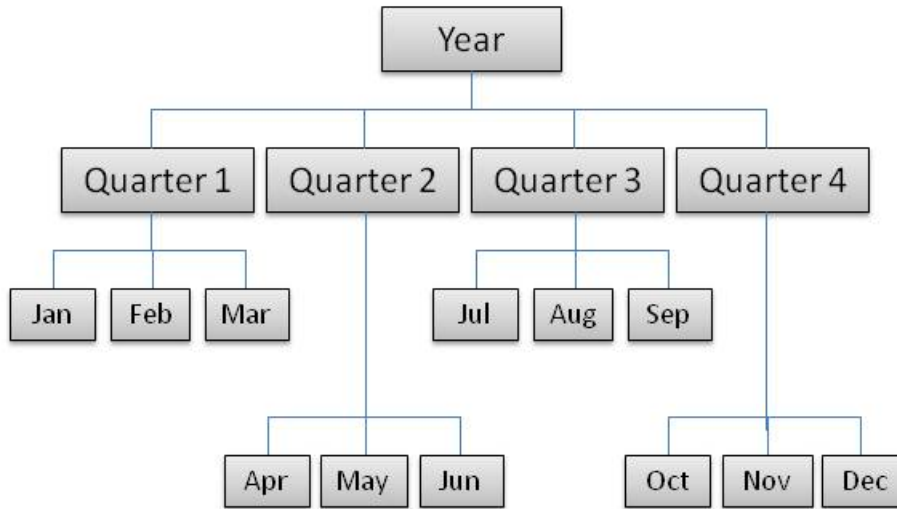
You can choose to hide the attributes and measures that are not required for client consumption.

For example, for a complex calculation that is derived from a series of computations, you can hide the levels of computations that are not required for reporting purposes.

## 1.3 Hierarchies

Hierarchies are used to structure and define the relationship between attributes of attribute views and calculation views that are used for business analysis. Exposed models that consist of attributes in hierarchies simplify the generation of reports.

For example, consider the TIME attribute view with YEAR, QUARTER, and MONTH attributes. You can use these YEAR, QUARTER, and MONTH attributes to define a hierarchy for the TIME attribute view as follows:



The following types of hierarchies are supported:

- **Level Hierarchy**

Level Hierarchies are hierarchies that are rigid in nature, where the root and the child nodes can be accessed only in the defined order. For example, organizational structures, and so on.

- **Parent/Child Hierarchy**

Value hierarchies are hierarchies that are very similar to BOM (parent and child) and Employee Master (Employee and Manager). The hierarchy can be explored based on a selected parent, and there are cases where the child can be a parent. This hierarchy is derived based on the value.

## 1.4 Users and Roles

Users and roles are assigned to create and modify content in the modeler perspective. For more information, refer to <http://help.sap.com/hana> -> SAP HANA Administration and Security-> see section *SAP HANA Users, Roles, and Authorizations*.

There are two types of users with roles specific to modeling activities:

- **SAP\_TEMPLATE\_MODELING**

This is used as a template role that can be used to create users to work on content.

- **SAP\_TEMPLATE\_CONTENT\_ADMIN**

This is used as a template role for users who are responsible for managing repository content at a higher level, and for managing teams who develop and test the content.

Users with this role are able to:

- Maintain delivery units
- Export and import content
- Create, update, and delete active native and imported packages and objects in these packages. They can also grant all these privileges to other users.

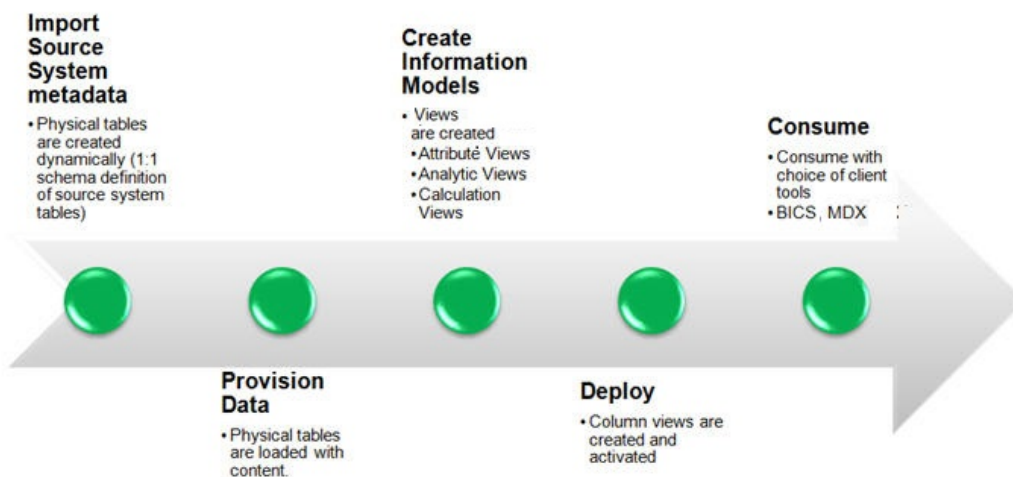


## Overview of Modeling Data

SAP HANA modeler is a graphical data modeling tool which allows you to design analytical models and later analytical privileges, that governs the access to those models.

The figure below shows the process flow for modeling within SAP HANA.

### Modeling Process Workflow



#### Prerequisites

- You have installed all SAP HANA components that are necessary to enable data replication. For information about installing SAP HANA, see [SAP HANA Installation Guide](#).
- The modeler perspective is part of the SAP HANA studio and will be installed together with the SAP HANA studio, as described in [SAP HANA Database - Studio Installation and Update Guide](#).
- You have configured all SAP HANA components that are necessary to enable data replication. For more information see the *Configuring Data Replication* section in [SAP HANA Master Guide](#).

#### Procedure

The tasks that you can perform in the modeler are grouped as follows:

##### 1. Importing the Metadata

You need to import the table definitions in the modeler perspective. For more information, see Importing Metadata.

## **2. Provisioning Data**

You need to load data into the table definitions in the modeler perspective. More information: [Loading Data into Table Definitions](#).

### **Note:**

You can use the features offered by the modeler only if, the modeler version is compatible with the appropriate server version.

## **3. Creating Content Objects**

You need to create content objects in the modeler perspective. More information: [Creating Content Objects](#).

## **4. Processing Content Objects**

For more information about the tasks that you can perform after creation of content objects, see [Processing Content Objects](#).

## Adding a System

You use this procedure to add a system in which you want to create your content objects for analytical purposes.

### Procedure

1. From the desktop, choose **Start > Studio**.
2. Choose **Window > Open Perspective > Other**.
3. Choose **Modeler**.
4. Choose **OK**.
5. Navigate to the **Navigator** pane.
6. From the context menu, choose **Add System**.
7. Enter the host name, instance number, and a description for the system you want to add.
8. Choose **Next**.
9. Select the required options.
10. Choose **Finish**.

### Result

If you add multiple systems, then:

- You can define a specific system as a default system by choosing **Window > Preferences > Modeler > Quick Launch**.
- You can switch to a different system by choosing **Select System** in the **Quick Launch** page.



# Importing Metadata

## 4.1 Configuring Data Services for Metadata Import

You use this procedure to enable the SAP HANA modeler for importing of metadata.

### Procedure

#### 1. Set the Passphrase

1. Log on to the Central Management Console of SAP BusinessObjects Enterprise (BOE).
2. Choose **Manage > Applications**.
3. Choose **Data Services Application**.
4. Choose **Settings**.
5. In the **Encryption Passphrase** field, enter the passphrase that you have been using for the SAP In\_Memory Computing studio.
6. Choose **Save**.
7. Restart the TOMCAT and BOE services.

#### 2. Disable Session Security

1. Log on to the Data Services Management Console.
2. Choose **Administrator**.
3. In the navigator pane, choose **Web Services**.
4. Choose the **Web Services Configuration** tab page.
5. Select the **Import\_Repo\_Object** checkbox to save the connection details.
6. Select **Disable Session Security** from the dropdown menu.
7. Choose **Apply**.

#### 3. Creating a Data Source

1. Go to **Start > Control panel > Administrative Tools**.
2. Choose **Data Sources (ODBC)**.
3. Choose the **System DSN** tab page.
4. Choose **Add**.
5. Select **HDBODBC** from the driver list.
6. Choose **Finish**.
7. Enter a name and description for the data source.
8. Enter server details.

**Note:**

The format in which you need to enter details is <host>:3<instance number>15. For example, vml2562.wdf.sap.corp:34715.

9. Enter the required database details.

**Note:**

The format in which you need to enter the database details is <SID><instance number>. For example, M4747.

10. Choose **Connect**.

## 4.2 Creating Schemas

You use this procedure to create schemas where you import table definitions.

**Procedure**

1. From the **Quick Launch** tab page, choose **SQL Editor**.
2. In the SQL editor, write the script, *create schema <schema name>*.
3. Execute the script.

**Result**

System creates the schema, and places it under the **Catalog** node.

## 4.3 Importing Table Definitions

You use this procedure to import table definitions (metadata) from a required source system. You use these table definitions for creating various content models such as attribute, analytic, and calculation views.

Based on your requirement, you use one of the following approaches:

- **Mass Import:** To import all table definitions from a source system. For example, you use this approach if this is a first import from the given source system.
- **Selective Import:** To import only selected table definitions from a source system. For example, you use this approach if there are only few table definitions added or modified in the source system after your last import.

**Prerequisites**

You have configured the SAP HANA modeler for importing metadata using Data Services infrastructure.

For more information, see [Configuring Data Services for Metadata Import](#).

**Procedure****Importing All Table Definitions**

1. In the **File** menu, choose **Import**.
2. Expand the **SAP HANA Content** node.
3. Choose **Mass Import of Metadata**.
4. Choose **Next**.
5. Select the target system where you want to import all the metadata.
6. Choose **Next**.
7. In the **Connection Details** dialog, enter the OS User and Password of the target system.
8. Select the required source system.

**Note:**

If the required system is not available in the dropdown list, you need to contact your administrator.

9. Choose **Finish**.

**Importing Selected Table Definitions**

1. From the **File** menu, choose **Import**.
2. Expand the **SAP HANA Content** node.
3. Choose **Selective Import of Metadata**.
4. Choose **Next**.
5. Select the target system where you want to import the metadata.
6. Choose **Next**.
7. Select the required source system.

**Note:**

If the required system is not available in the dropdown list, you need to add the new source system. For more information, [Adding Connections](#).

8. In the **Type of Objects to Import** field, select the required type.
9. Choose **Next**.
10. Add the required objects (tables or extractors) that you want to import.

**Note:**

If you want to add dependent tables of a selected table, select the required table in the Target pane and choose Add Dependent Tables from the context menu.

11. Select the schema where you want to import the metadata.
12. If the object type is extractor, select the package.
13. Choose **Next**.
14. Choose **Finish**.

**Result**

If source object is a table or non V-type extractor, system creates physical tables and stores them in the selected schema. If source object is a V-Type extractor, system creates content models and stores these models in the selected package, and the underlying physical tables in the schema.

## 4.4 Adding Connections

### Procedure

1. Choose **Manage Connections**.
2. Enter a name and description for the connection.
3. Enter the Application Server details.
4. Enter the username and password.
5. Enter the passphrase.
6. Enter the Client and System Number.
7. To test the connection, choose **Test**.
8. Save the connection details.
9. Choose **Close**.

## Loading Data into Table Definitions

You use this procedure to load data into your table definitions. Depending on your requirement, you can perform:

- **Initial Load** - to load all data from a source SAP ERP system into the SAP HANA database by using Load Controller or SAP Landscape Transformation (SLT). This is mostly applicable when you are loading data from the source for the first time.
- **Data Replication** - to keep the data of selected tables in the SAP HANA database up-to date with the source system tables by using SyBase Replication Server or SAP Landscape Transformation (SLT).

For more information on replication technologies, see *SAP HANA Replication Technologies* section in [SAP HANA 1.0 Master Guide](#).

### Prerequisites

- If you are using the Load Controller or Sybase Replication Server infrastructure, make sure that you have imported all table definitions into the SAP HANA database. For more information, see *Importing Metadata*.
- If you are using the SLT component, the source system(s), target system and the target schema, are configured by the administrator during the installation.

### Procedure

1. From the **Quick Launch** page, choose **Data Provisioning**.
2. If you are using SLT-based replication, choose **Source**.
3. Choose **Load** (for initial load) or **Replicate** (for data replication) as appropriate.
4. Select the required tables.
5. Choose **Add**.
6. If you are using the load controller infrastructure, choose **Next** and enter the operating system user name and password.
7. Choose **Finish**.

## 5.1 Suspending Data Load

You use this procedure to stop data replication for the selected tables temporarily if you are using SLT-based replication.

**Procedure**

1. On the **Quick Launch** tab page, choose **Data Provisioning**.
  2. Choose **Source**.
  3. Choose **Suspend**.
  4. Select the tables for which data replication is to be suspended in any of the following ways:
    - Search for the required tables.
      - a. Select the table from the object list.
      - b. Choose **Add**.
      - c. If you want to save the list of tables selected for suspension locally for future reference, perform the following steps:
        - a. Check **Save selected tables** checkbox.
        - b. Specify the target location.
    - Load the list of tables from a local file as follows:
      - a. Choose **Load from file**.
      - b. Select the required file.
- Note:**  
The file should be of type csv.
5. Choose **Finish**.

## 5.2 Resuming Data Load

You use this procedure to resume data replication for the suspended tables, if you are using SLT-based replication.

**Procedure**

1. On the **Quick Launch** tab page, choose **Data Provisioning**.
2. Choose **Source**.
3. Choose **Resume**.
4. Select the tables for which data replication is to be resumed in any of the following ways:
  - Search for the required tables.
    - a. Select the table from the object list.
    - b. Choose **Add**.
    - c. If you want to save the list of selected tables for future reference, perform the following steps:
      - a. Check **Save selected tables** checkbox.
      - b. Specify the target location.
  - Load the list of tables from a local file as follows:
    - a. Choose **Load from file**.
    - b. Select the required file.

**Note:**

The file should be of type csv.

5. Choose **Finish**.

## 5.3 Uploading Data from Flat Files

You use this procedure to upload data from flat files, available in a client file system to SAP HANA database. If the table schema corresponding to the file to be uploaded already exists in the SAP HANA database, the new data records are appended to the existing table. If the required table for loading the data does not exist in the SAP HANA database, create a table structure based on the flat file. The application suggests the column names and data types for the new tables and allows you to edit them. The new table always has a 1:1 mapping between the file and table columns. The application does not allow you to overwrite any columns or change the data type of existing data. The supported file types are .csv, .xls, and .xlsx.

### Procedure

1. In the **File** menu, choose **Import**.
2. In the **Select an import source** section, expand the **SAP HANA content** node.
3. Select **Import Flat File**, and choose **Next**.
4. In the **Target System** section, select the target system where you want to import the data using the flat file, and choose **Next**.
5. On the **Flat File Upload** screen, browse for the file containing the data you want to upload.
6. If you have selected a CSV file, select a delimiter.

**Note:**

A delimiter is used to determine columns and pick the correct data from them. In a csv file, the accepted delimiters are ',', ';' and '!':

7. If you have selected an .xls or .xlsx file, select a worksheet.
8. If you want to load the data into a new table, select the **New** option and perform the following substeps:
  - a. Choose **Next**.
  - b. On the **Manage Table Definition and Data Mapping** screen, map the source and target columns.

**Note:**

- Only 1:1 column mapping is supported. You can also edit the table definition by changing the data types, renaming columns, adding or deleting the columns, and so on.
  - You can choose to map the source and target columns using the Auto Map option. If you choose the one to one option then first column of the source is mapped with the first column at the target. If you choose the option Map by name, the source and target columns with same name are mapped.
9. If you want to append the data to an existing table, select the **Existing** option and perform the following substeps:

- a. Choose **Next**.
  - b. On the **Manage Table Definition and Data Mapping** screen, map the source and target columns.
10. Choose **Finish**.

# Copying Contents Delivered by SAP

You use this procedure to copy the objects available in a package to other packages in the same system based on your business use case.

You use this functionality in one of the most common scenarios, that is, to copy the standard content shipped by SAP or an SAP partner to your local package to meet your modeling and reporting use cases. For example, from <sap.ecc.fin> to <customer.ecc.fin>.

To copy the objects, you need to map the source root packages to the target root packages. You need to activate the copied objects in the target package to consume them for reporting purposes.

## Note:

- We recommended that you copy the content shipped by SAP or an SAP partner to your local package to avoid overwriting your changes during the next import.
- If you copy an object but not its dependent objects (if any), the copied object in the target package will have references to the dependent objects in the source package.
- For script-based calculation views and procedures, even if you copy the dependent objects, you need to change the script manually and adjust the dependent object references.

## Prerequisite

You have the following privileges:

- REPO.READ for the source package.
- REPO.MAINTAIN\_NATIVE\_PACKAGES and REPO.EDIT\_NATIVE\_OBJECTS for the root package.

## Procedure

1. From the **Quick Launch** tab page, choose **Mass Copy**.
2. To create a mapping between the source package and the target package, perform the following substeps:
  - a. Choose **Add**.
  - b. Select a source package and a target package.

### Note:

If you want to create more package mapping, select the source and target packages as required.

- c. Choose **Next**.
3. Select the required objects, and choose **Add**.
  4. Choose **Next** to view the summary.

### Note:

You can deselect an object to avoid copying it to the target package.

**5. Choose Finish.**

## Mapping the Authoring Schema to Physical Schema at the Target

You use this procedure to map the authoring schemas to the physical database schemas in the target system to access and deploy the transported objects.

A physical schema is the schema where the tables are available. It may differ in the source and target systems.

An authoring schema (logical schema) is the physical database schema in the source system with which the content objects are created.

Schema mapping is done when the physical schema in the target system is not the same as the physical schema in the source system.

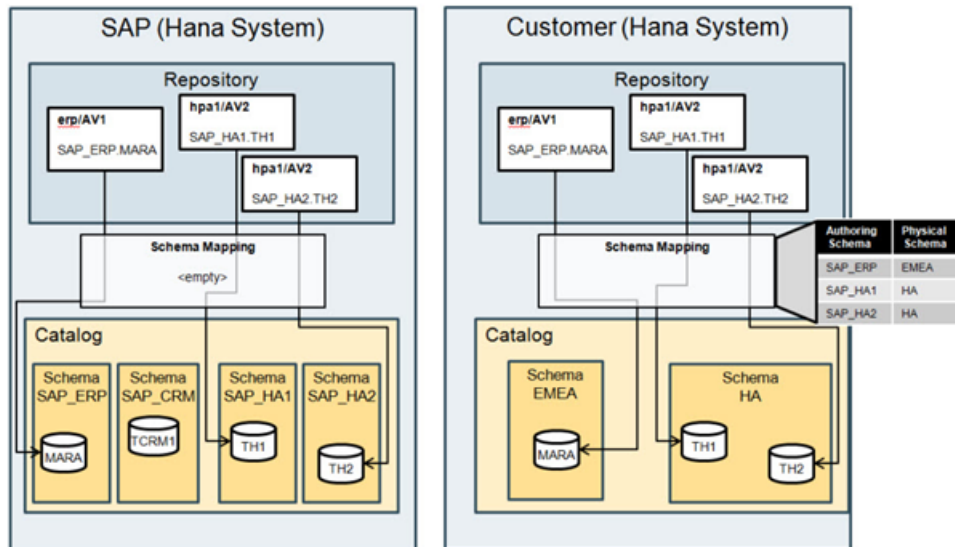
Content object definitions are stored in the repository, and contain references to the physical database schemas. When you copy the content objects to a different system, for example, from an SAP system to a customer system, or between customer systems, the object definition still refers to the physical database schemas at the source. To resolve this, you use schema mapping.

**Note:**

You need to map the references of script-based calculation views and procedures manually, that is, by changing the script.

You can map several authoring schemas to the same physical schema. For example, content objects delivered by SAP refer to different authoring schemas, whereas in the customer system, all these authoring schemas are mapped to a single physical schema where the tables are replicated.

**Example:**



Consider a source system, SAP, with an information object, AV1. This refers to the table MARA in the SAP\_ERP physical database schema. There is a target system, Customer, with the physical database schema EMEA.

After you import content, the object AV1 cannot be activated in the Customer system, since it still refers to SAP\_ERP schema.

AV1 in the Customer system currently refers to SAP\_ERP. To be able to activate the object, you need to modify the mapping of AV1 in the Customer system. To do this, a mapping is created in the target system Customer between the authoring and physical schema as follows:

Authoring Schema	Physical Schema
SAP_ERP	EMEA

**Procedure**

1. On the **Quick Launch** tab page, choose **Schema Mapping**.
2. Choose **Add**.
3. Enter the authoring schema name.
4. Select the physical schema.
5. Choose **OK**.

## Setting Preferences for Modeler

You use this procedure to set up the default settings that the system uses whenever you log on.

### **Procedure**

1. From the **Quick Launch** tab page, choose **Manage Preferences**.
2. Expand the **Modeler** node.
3. Identify the required preference and perform the corresponding substeps from the table below:

Requirement	Preference	Substeps
To specify the structure of content packages in Navigator pane	Content Presentation	<ol style="list-style-type: none"> <li>a. If you want to view the package structure in a hierarchical manner such that, the child folder inside the parent folder, select <b>Hierarchical</b>.</li> <li>b. If you want to view all the packages at the same level like, sap, sap.ecc, sap.ecc.ui, choose <b>Flat</b>.</li> <li>c. If you want to group together similar objects in a package like attribute views in Attribute View package, select <b>Show Object Type Folders</b>.</li> <li>d. Choose <b>Apply</b>.</li> <li>e. Choose <b>OK</b>.</li> </ol>
To set the preferences for loading data using flat file	Flat File	<ol style="list-style-type: none"> <li>a. Browse the location to save error log files for data load using flat files.</li> <li>b. Enter the batch size for loading data. For example, if you specify 2000 and a file has records of 10000 rows the data load will happen in 5 batches.</li> <li>c. Enter a decision maker count that will be used to propose data types based on the file. For example, enter 200 if you want the proposal to be made based on the 200 rows of file data.</li> </ol>
To set the default value for the client that will be used while previewing model data.	Default Model Parameters	Select the client from the <b>Default Client</b> drop-down list.
To enforce various rules on objects <b>Note:</b> Enforcing validation rules with severity as "Error" are mandatory.	Validation Rules	<ol style="list-style-type: none"> <li>a. Select the required rules to be applied while performing object validation.</li> <li>b. Choose <b>Apply</b>.</li> <li>c. Choose <b>OK</b>.</li> </ol>
To determine the numbers of rows to be displayed in a page	Data Preview	<ol style="list-style-type: none"> <li>a. Select the maximum rows for data preview as required.</li> <li>b. Choose <b>Apply</b>.</li> <li>c. Choose <b>OK</b>.</li> </ol>

Requirement	Preference	Substeps
To specify location for job log files	Logs	<ol style="list-style-type: none"> <li>a. Expand the <b>Logs</b> node.</li> <li>b. Select <b>Job Log</b>.</li> <li>c. Browse the location where you want to save the job log files.</li> <li>d. Choose <b>Apply</b>.</li> <li>e. Choose <b>OK</b>.</li> </ol>
To enable logging the repository calls and specify location for repository log files	Logs	<ol style="list-style-type: none"> <li>a. Expand the <b>Logs</b> node.</li> <li>b. Select <b>Job Log</b>.</li> <li>c. Select true from the drop-down list.</li> <li>d. Browse the location where you want to save the repository log files.</li> <li>e. Choose <b>Apply</b>.</li> <li>f. Choose <b>OK</b>.</li> </ol>
To enable search for the attributes used in the views	Search Options	<ol style="list-style-type: none"> <li>a. Select <b>Enable Search Attributes</b>.</li> <li>b. Choose <b>Apply</b>.</li> <li>c. Choose <b>OK</b>.</li> </ol>

4. To establish a connection with the server from where you need to import data, perform the following substeps:
  - a. In the **Quick Launch** tab page, choose **Configure Import Server**.
  - b. Enter the IP address of the server from which you want to import data.
  - c. Enter the repository name.
  - d. Enter the ODBC data source.
  - e. Choose **OK**.
5. If you want to generate time data, then perform the following substeps:
  - a. In the **Quick Launch** tab page, choose **Generate Time Data**.
  - b. In the **Calendar Type** field, select the required option.
  - c. Enter the period for which you want to generate data.
  - d. If the calendar type is **Fiscal**, select a variant table, and a fiscal variant.
  - e. If the calendar type is **Gregorian**, select the granularity for the data.
  - f. Choose **Generate**.

System populates the generated time data in M\_TIME\_DIMENSION\_MONTH, M\_TIME\_DIMENSION\_WEEK, M\_TIME\_DIMENSION, and M\_FISCAL\_CALENDAR tables present in the \_SYS\_BI schema.
6. To set vendor for the content you are developing, perform the following substeps:
  - a. Choose **Window > Open Perspective > Administration Console**.
  - b. In the **Navigator** toolbar, choose **Administration**.
  - c. Choose **Configuration** tab page.
  - d. Expand **indexserver.ini** node.

- e. Expand **repository** node.
- f. Select **content\_vendor**.
- g. From the context menu, choose **Change**.
- h. Enter a value.

**Note:**

It is recommended to use reserved DNS names for vendor, to avoid naming conflicts. For example, SAP content development systems should use the vendor sap.com.

## Creating Content Objects

You use this procedure to create the appropriate structural objects that are required to create and deploy the content. In addition to creating the information views, you create additional objects such as packages (for organizing content), procedures (for SQL routines you may want to call repeatedly), and analytic privileges (for applying restrictions on which data users can see), as appropriate.

**Note:**

You can find the different types of content objects in the table below:

<b>Content Object Type</b>	<b>Description</b>
Package	Groups together related content objects in SAP HANA studio.
Procedure	SQL procedures that can be reused by other content objects.
Analytic Privilege	Allows selective access control for the database views that are generated when modeled views (analytic views, calculation views, attribute views) are activated.
Attribute View	Defines joins between tables and selects a subset (or all) of their columns. The rows selected can also be restricted by filters. An additional application of attribute views is as dimensions in the star schema defined by an analytic view. In this use case, the attribute view adds more columns and also hierarchies as further analysis criteria to the analytic view. In the star schema of the analytic view the attribute view is shown as a single dimension table (although it might join multiple tables), that can be joined to a fact table. For example, attribute views can be used to join employees to organizational units and then the result can be joined to a sales fact table in the analytic view.
Analytic View	Retrieves data from single or joined database tables. Analytic views can contain two types of attributes (or columns): measures and normal attributes. All measures must reside in a single table (the fact table) and must be aggregated when queried. Normal attributes can be handled as regular columns and do not need to be aggregated. Attributes can either be picked from the underlying tables, or are added to the analytic view by joining attribute views (also known as dimensions).
Calculation View	Provides a composite of other views or tables. Inside the calculation view, these are referred to as data sources. The calculation view offers the nodes/operations: join, union, projection and aggregation. These operations work on Data Sources and/or on top of other operations. The calculation view can be defined graphically, by building a tree of operations, or by writing SQL script in a text-based editor.

## Procedure

### 1. Create a Package

More information: [Creating a Package](#)

### 2. Create an Attribute View

More information: [Creating an Attribute View](#)

### 3. Create an Analytic View

More information: [Creating an Analytic View](#)

### 4. Create a Calculation View

More information: [Creating a Calculation View](#)

### 5. Create a Procedure

More information: [Creating a Procedure](#)

## 6. Create an Analytic Privilege

More information: [Creating an Analytic Privilege](#)

### Note:

- To create, read, edit, delete, activate, or maintain an information object or package, you must have repository authorization at package level.
- To add a description for a model and its elements in the chosen language, specify the model language when logging on to the system.
- All imported objects open in read-only mode. You can choose to edit an imported object by selecting the **Edit Imported Object** option available in **Switch Version** on the editor toolbar. The changes made to the imported object are overridden if you import the same object again.
- If the logon locale is different from the original language of an object that is to be edited, the object opens with a warning, and you must maintain all the texts in the original language.

## 9.1 Creating a Package

You use this procedure to create packages to group related objects together in a structured way.

You can create either a structural package or a non-structural package. The two types of packages are described in the table below:

Type	Description
Structural	A structural package helps to organize the content in a logical package tree. It categorizes objects using the subpackages. This ensures that each object is associated with a particular package.
Non Structural	A non structural package contains subpackages and content objects.

### Procedure

1. On the **Quick Launch** page, choose **Package**.
2. Enter a name and description for the package.

#### Note:

Packages that use a dot in their name form a logical hierarchy. For example, "a.b" is considered to be a subpackage of "a" and "a.b.c" is considered to be a subpackage of "a.b".

3. If you want to specify the entity with which a package and its objects are exported to a server, select the delivery unit from the drop-down list.

#### Note:

The assignment of packages to delivery units is completely independent of the package hierarchy.

4. From the **Original Language** drop-down list, select the same language for the package as your logon language.
5. If you know the user responsible for maintaining the content in the package, select the user from the **Person Responsible** dropdown list.

6. If the object is relevant for translation, choose **Translation**.
7. Select the required terminology domain to determine the correct terminology of the text that needs translation.
8. Enter a **Text Collection** to associate a package with a collection in order to specify the language into which the package objects are to be translated.
9. To provide a suggestion regarding the translation of the package, enter text in **Hint**.
10. Enter a text status.
11. Choose **OK**.
12. To specify the package type, perform the following substeps:
  - a. In the **Navigator** pane, select the package.
  - b. In the **Properties** pane toolbar, choose **Edit Package Details**.
  - c. From the **Structural** drop-down list, choose the required option.
  - d. Choose **OK**.

**Note:**

- The package *system-local* and all of its subpackages are not transportable. All other packages are transportable. You can view the package property **Transportability** to check whether or not the package can be transported.
- All sub-packages of *system-local.generated* should be used for generated content (content that is not created by manual user interaction). You can view the package property **Content type** to check whether the package contains generated or edited content.

## 9.2 Creating an Attribute View

You use this procedure to create a view that is used to model descriptive attribute data (that does not contain measures) using attributes. For more information regarding attribute view, see section *Content*.

**Note:**

You need this view for creating a multidimensional view.

**Prerequisites**

You have imported T009 and T009B tables for creating a fiscal time attribute view.

**Procedure****1. Set Parameters**

1. In the **Modeler** perspective, expand the **Content** node of the required system.
2. Select the package where you want to save your information object.
3. From the context menu, choose **New > Attribute View**.
4. Enter a name and description for the view.
5. To create data foundation for the view, perform substeps of the required scenario given in the table below:

Scenario	Substeps
Create a view with table attributes.	<ol style="list-style-type: none"> <li>a. Choose <b>Standard</b>.</li> <li>b. Choose <b>Next</b>.</li> <li>c. Add the required tables to the <b>Selected</b> list.</li> </ol>
Create a view with time characteristics.	<ol style="list-style-type: none"> <li>a. Choose <b>Time</b>.</li> <li>b. In the <b>Calendar Type</b> field, select the required option.                             <ul style="list-style-type: none"> <li>• If the calendar type is <b>Fiscal</b>, select a variant table, and a fiscal variant.</li> <li>• If the calendar type is <b>Gregorian</b>, select the granularity for the data.</li> </ul> </li> <li>c. To select a table for defining the view's output structure, perform one of the following substeps:                             <ul style="list-style-type: none"> <li>• To use the default table, select <b>Auto Create</b>.</li> <li>• To specify tables, choose <b>Next</b> and select the required tables.</li> </ul> </li> </ol> <p><b>Note:</b> The tables used for attribute creation are M_TIME_DIMENSION, M_TIME_DIMENSION_MONTH, M_TIME_DIMENSION_WEEK and M_FISCAL_CALENDAR. If you want to do a data preview on the created attribute view, you need to generate time data into the mentioned tables from the Quick launch tab page.</p>
Derive a view from an existing view – in this case, you cannot modify the derived view.	<ol style="list-style-type: none"> <li>a. Choose <b>Derived</b>.</li> <li>b. Select the required attribute view.</li> </ol>
Copy a view from an existing view – in this case, you can modify the copied view.	<ol style="list-style-type: none"> <li>a. Choose <b>Copy From</b>.</li> <li>b. 2. Select the required attribute view.</li> </ol>

6. Choose **Finish**.

**2. Define Output Structure**

1. Select the required table field, and perform substeps of the required scenario given in the table below:

Scenario	Substeps
Include table field in the output structure.	From the context menu, choose <b>Add as Attribute</b> .
Include table field in the output structure as a key element.	From the context menu, choose <b>Add as Key Attribute</b> .
Include table field in the output structure and specify a filter condition based on which system must display data for this field in the output.	<ol style="list-style-type: none"> <li>a. From the context menu, choose <b>Apply Filter</b>.</li> <li>b. Select the required operator and enter the values.</li> </ol>

2. If you want to query data from more than one table, perform the following substeps:
  - a. In the editor pane, from the context menu, choose **Create Join**.
  - b. Select the required tables, columns, join type, and cardinality.
  - c. Choose **Create Join**.

**Note:**

- You can choose to add the same table again in **Data Foundation** pane using table aliases in the editor. For example, in cases where you want to create self join or to have different cardinalities from the same table.
- You can choose to create a hierarchy between attributes of the view. For more information, see [Creating a Hierarchy](#).

**3. (Optional) Create Calculated Attributes**

1. In the output pane, right-click **Calculated Attributes**.
2. From the context menu, choose **New**.
3. Enter a name and description for the calculated attribute.
4. Choose the **Key** tab page.
5. Select a data type for the calculated attribute.
6. Enter the length and scale.
7. Define the attribute using the required attributes, operator, and function.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

8. Choose **OK**.

**Note:**

You can specify a value for the **Default Client** property of the view to filter the table data that is relevant to a specific client as specified in the table fields, such as, MANDT or CLIENT, at runtime. The default value for the property is the one that is specified as preferences. If the property is set to **Dynamic** then at runtime, the value set for the **Session Client** property is considered to filter table data. The **Session Client** property is set while creating a user. For more information about how to set the default value for **Default Client**, see [Setting Preferences for Modeler](#).

**4. Activate the View**

1. Choose **File > Save**.
2. From the context menu of the view, select one of the following options as appropriate:

- Activate – to activate the current view only
- Cascade Activate - to activate the current view and its dependent objects

**Note:**

For more information about activation, see [Activating Objects](#).

**Result**

You can find the activated model in the related package. If you want to modify this model, from the context menu, choose **Open** and make the necessary changes.

If you want to view the creation job details, refer to the details on the **Job Log** tab page.

## 9.3 Creating an Analytic View

You use this procedure to define a multidimensional view (OLAP cube) based on database tables, attribute views and facts of a specific table.

**Procedure****1. Set Parameters**

1. In the **Modeler** perspective, expand the **Content** node of the required system.
2. Select the package where you want to save your information object.
3. From the context menu choose, **New > Analytic View**.
4. Enter a name and description for the view.
5. To create data foundation for the view, perform substeps of the required scenario given in the table below:

Scenario	Substeps
Create a new analytic view.	<ol style="list-style-type: none"> <li>a. Choose <b>Create New</b>.</li> <li>b. If you want to perform currency conversion for measures, select the required schema from the <b>Schema for Conversion</b>.</li> <li>c. Choose <b>Next</b>.</li> <li>d. Add the required tables and attribute views to the <b>Selected</b> list.</li> <li>e. Choose <b>Finish</b>.</li> </ol>
Create an analytic view from an existing analytic view.	<ol style="list-style-type: none"> <li>a. Choose Copy From.</li> <li>b. If you want to perform currency conversion for measures, select the required schema from the <b>Schema for Conversion</b>.</li> <li>c. Select the required analytic view.</li> <li>d. Choose <b>Finish</b>.</li> </ol>

## 2. Add Attributes and Measures

1. Select the required table field, and perform substeps of the required scenario given in the table below:

Scenario	Substeps
Include table field in the output structure.	From the context menu, choose <b>Add as Attribute</b> .
Add measures based on which you want to view data from the fact table for analysis purposes.	From the context menu, choose <b>Add as Measure</b> .
Include table field in the output structure and specify a filter condition based on which system must display data for this field in the output.	<ol style="list-style-type: none"> <li>a. From the context menu, choose <b>Apply Filter</b>.</li> <li>b. Select the required operator and enter the values.</li> </ol>
Hide attributes and measures that are not required for client consumption. <b>Note:</b> You can choose to hide only private attributes of an analytic view.	<ol style="list-style-type: none"> <li>a. Select the required attribute or measure.</li> <li>b. In the properties pane, assign value True to the Hidden property.</li> </ol>

## 3. Create Joins

1. If you want to define relationship between tables to query data from two or more tables, perform the following substeps:
  - a. In the editor pane, from the context menu, choose **Create Join**.
  - b. Select the required tables, columns, join type, and cardinality.

- c. Choose **Create Join**.

#### **4. (Optional) Create Calculated Attributes**

1. In the output pane, right-click **Calculated Attributes**.
2. From the context menu, choose **New**.
3. Enter a name and description for the calculated attribute.
4. Choose the **Key** tab page.
5. Select the data type for the calculated attribute.
6. Enter the length and scale.
7. Define the attribute using the required attributes, operator, and function.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

8. Choose **OK**.

#### **5. (Optional) Create Calculated Measures**

1. In the Output pane, right-click **Calculated Measures**.
2. From the context menu, choose **New**.
3. Enter a name and description for the calculated measure.
4. Select the required **Aggregation Type**.
5. If you want to hide the measure while previewing data, select **Hidden**.
6. Select the required data type for the calculated measure.
7. Enter the length and scale.
8. Define the measure by selecting the required measures, operator, and function.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

9. Choose **Validate**.

**Note:**

If you want to associate the calculated measure with a unit of currency or weight, choose the **Currency/Unit of Measure** tab page. For more information, see [Using Currency and Unit of Measure](#).

10. Choose **OK**.

#### **6. (Optional) Create Restricted Measures**

1. In the **Output** pane, select **Restricted Measures**.
2. From the context menu, choose **New**.
3. Enter a name and description for the restricted measure.
4. Select the required aggregation type.
5. Choose the measure from the drop-down list.
6. To add a restriction, choose **Add Restriction**.
7. Select the required parameter, operator, and value.
8. Choose **OK**.

**Note:**

- You can choose to create variables, and bind them to attributes for filtering data. The values you provide to the variables at the runtime determines which data records are selected for consumption. For more information, see [Assigning Variables](#).
- You can create input parameters for the analytic view that works as placeholder in the query. For more information, see [Creating an Input Parameter](#).
- You can specify a value for the **Default Client** property of the view to filter the table data that is relevant to a specific client as specified in the table fields, such as, MANDT or CLIENT, at runtime. The default value for the property is the one that is specified as preferences. If the property is set to **Dynamic** then at runtime, the value set for the **Session Client** property is considered to filter table data. The **Session Client** property is set while creating a user. For more information about how to set the default value for **Default Client**, see [Setting Preferences for Modeler](#).

**7. Activate the View**

1. Choose **File > Save** .
2. From the context menu of the view, select one of the following options as appropriate:
  - Activate – to activate the current view only.
  - Cascade Activate - to activate the current view and its dependent objects.

**Note:**

For more information regarding activation, see [Activating Objects](#).

**Result**

You can find the activated model in the related package. If you want to modify this model, from the context menu, choose **Open** and make the necessary changes.

If you want to view the creation job details, refer to the details on the **Job Log** tab page.

## 9.4 Creating a Calculation View

You create a calculation view when your business use case demands complex business logic. For queries that involve more than one analytic view or table, you use a calculation view. You can use calculation views to derive values and Key Performance Indicators(KPIs).

**Procedure**

1. In the **Modeler** perspective, expand the system node from the **Navigator** pane.
2. Expand the **Content** node.
3. Right-click the required package.
4. From the context menu, choose **New > Calculation View**.
  - a. Enter a name and description.
  - b. Select the required package.

**Create a Script-Based Calculation View**

1. Choose **SQL Script**.
2. Select the required schema from the **Default Schema** drop-down list, for unqualified access in SQL.

**Note:**

If you do not select a default schema, while scripting, you need to provide fully qualified names of the objects used.

3. Select the required option from the **Run With** drop-down list.

**Note:**

*Definer's right:* If you want to control the access to the model for other users.

*Invoker's right:* If you want system to grant access to the model for other users.

4. Choose **Finish**.
5. Define the Output Structure
  - a. Choose **Define Output Parameter** from the output pane.
  - b. To add the output parameters with the required data type and length, from the Output pane toolbar, choose **Define Output Parameter** button.

**Note:**

The order and data types of the output parameters should match the table columns order and data type which is used in the select query

- c. Choose **OK**.
6. Define the Function
  - a. Define the function using SQL Script commands.

**Note:**

- For more information about the syntax and examples of SQL functions, see [SQL Functions on Content Assist](#).
- You can create variables, and bind them to attributes for filtering data. The values you provide to the variables at the runtime determines which data records are selected for consumption . For more information, see [Assigning Variables](#).
- You can create input parameters for the calculation view that works as placeholder in the query. For more information, see [Creating an Input Parameter](#).
- You can create hierarchy between attributes of the view. For more information, see [Creating a Hierarchy](#).

7. Save and Activate
  - a. To save the view, choose **File > Save**.
  - b. From the context menu of the calculation view, choose **Activate**.

**Note:**

For more information about activation, see [Activating Objects](#).

### Create a Graphical Calculation View

1. Choose **Graphical**.
2. Select the required schema from the **Schema for Conversion** drop-down list.

**Note:**

The schema selected for conversion is used during the currency conversion. It list down all the schemas which has currency related tables and the same can be changed during design time from the properties.

3. Choose **Next**.
4. Follow the instructions on the wizard to select the required tables and content models.

### Adding Unions/Joins/ Projections/ Aggregation

- a. From the **Tools Palette**, choose the required option as follows:

View	Description
Union	Used to combine the result set of two or more data sources. For example, to know the names of all the employees of a store with different branches maintaining their own employee records table.
Join	Used to query data from two or more data sources, based on some condition. For example, to retrieve the sales of two stores maintaining individual table for sales based on the customer id.
Projection	Used to filter or create a subset of the required columns of a table or view for creating the model. For example, selecting the employee name and sales quantity from a table consisting of many more columns.
Aggregation	Used to summarize data of a group of rows by calculating values in a column. For example, to retrieve total sales of some product in a month. The supported aggregation types are sum, min, and max.

**Note:**

- The input for union, join, projection, and aggregation views can consist of data sources, union, join, projection or aggregation views.
  - You can have only one source of input for aggregation and projection view.
  - You can choose to create filters on projection and aggregation view attributes.
- b. Map the input to the selected option.

#### Mapping attributes

- a. To map attributes in a union view, drag and drop the required columns from **Source** to **Target**.  
 You can also modify the attribute mapping. More information: [Managing Attribute Mappings](#)  
 To create a system generated mapping, choose **Auto Map By Name**.
- b. In case of a join view, join the columns of the source data sources.

**Note:**

- The output of a union view is the attributes that you add to the target.
- The output of a join view is the joined columns. However, to add additional attributes to the view's output, from the context menu, choose **Add to Output**.

- To add attributes of projection or aggregation view to its output, choose **Add to Output** from the context menu.

### Creating Calculated Columns (optional)

The output of union, join or projection view is stored as a column in the output pane. To perform calculations on these columns, perform the following steps:

1. Right-click the **Calculated Columns** node.
2. From the context menu, choose **New**.
  - a. Enter the name.
  - b. Select the data type.
  - c. Enter the length and scale.
  - d. To perform calculations on the output columns, perform one of the following:
    - If you know how to write the formula, enter the expression to perform calculation.
    - From the list, select the required elements, operator and functions.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

- e. Choose **Add**.

### Applying Filter on Aggregation and Projection view Attributes (optional)

1. Right-click the required attribute.
2. From the context menu, choose **Apply Filter**.
3. Select the required operator.
4. Enter value.
5. Choose **OK**.

**Note:**

You can edit a filter using filter expressions from the output pane that offers more conditions to be used in the filter including AND, OR, and NOT. For example, to retrieve the sales of a product where (revenue  $\geq$  100 AND region = India) OR (revenue  $\geq$  50 AND region = Germany).

### Adding Attributes and Measures to Calculation View Output

- a. From the workflow pane, choose the **Output** node.
- b. To add an attribute, from the context menu, choose **Add as Attribute**.
- c. To add a measure, from the context menu, choose **Add as Measure**.
- d. If you want to create calculated attributes, perform the following substeps:
  - a. In the **Output** pane, right-click **Calculated Attributes**.
  - b. From the context menu, choose **New**.
  - c. Enter a name and description.
  - d. On the **Key** tab, define the formula for the calculated attribute as follows:
    - a. Select a data type for the calculated attribute from the dropdown list.
    - b. Enter the length and scale.
    - c. Select the required attributes, operator, and function.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

- d. To check the validity of the formula, choose **Validate**.
- e. Choose **Add**.
- e. To add a **Description** to the calculated attribute, write the formula as above.
- e. If you want to create calculated measures, perform the following substeps:
  - a. In the **Output** pane, right-click **Calculated Measures**.
  - b. From the context menu, choose **New**.
    - a. Enter a name and description.
    - b. Select the required **Aggregation Type**.
    - c. If you want to hide the measure while previewing data, choose **Hidden**.
    - d. Select the required data type.
    - e. Enter the length and scale.
    - f. Define the measure by selecting the required measures, operator, and function.

**Note:**

For more information about functions, see [Functions used in Expressions](#).

- g. Choose **Validate**.

**Note:**

If you want to associate the calculated measure with a unit of currency or weight, choose the **Currency/Unit of Measure** tab page. For more information, see [Using Currency and Unit of Measure](#).

- h. Choose **OK**.

**Note:**

If you set the calculation view property **Multidimensional Reporting** as disabled, you can create a calculation view without adding any measure to the calculation view output. A calculation view without any measure works like an attribute view and is not available for reporting purposes.

**Creating Counters (optional)**

To obtain the number of distinct values of an attribute, perform the following substeps:

1. In the Output pane, right-click **Counters**.
2. From the context menu, choose **New**.
3. Choose **Add Attribute**.
4. Choose **OK**.

**Note:**

1. You can choose to hide the attributes and measures that are not required for client consumption by assigning value true to the property *Hidden* in the properties pane.
2. You can choose to create variables, and bind them to attributes for filtering data. The values you provide to the variables at the runtime determines which data records are selected for consumption. For more information, see [Assigning Variables](#).
3. You can choose to create input parameters for the union, join, projection, aggregation and calculation view output. For more information, see [Creating an Input Parameter](#).

4. You can choose to create hierarchy between attributes of the view. For more information, see [Creating a Hierarchy](#).
5. You can specify a value for the **Default Client** property of the view to filter the table data that is relevant to a specific client as specified in the table fields, such as, MANDT or CLIENT, at runtime. The default value for the property is the one that is specified as preferences. If the property is set to **Dynamic** then at runtime, the value set for the **Session Client** property is considered to filter table data. The **Session Client** property is set while creating a user. For more information about how to set the default value for **Default Client**, see [Setting Preferences for Modeler](#).

#### **Save and Activate**

- a. To save the calculation view, choose **File > Save**.
- b. From the context menu of the Calculation View, choose **Activate**.

#### **Note:**

For more information about activation, see [Activating Objects](#).

#### **Note:**

You can choose to generate documentation for the calculation view. For more information, see [Generating Object Documentation](#).

## **9.4.1 Managing Attribute Mappings**

You use this procedure to map the source attribute to the target attribute if there is a big list of attributes, or to assign a constant value to the target attribute.

#### **Procedure**

1. Right-click the attribute in the target list.
2. From the context menu, choose **Manage Mappings**.
  - a. To map the source to the target column, select the required source from the dropdown list.
  - b. To assign a default value to the constant column, enter the value in the **Constant Value** field. For more information, see [Constant Column](#).
  - c. Select the required data type.
  - d. Enter the length and scale as required.
  - e. Choose **OK**.

## **9.4.2 Constant Column**

In a union view, a **Constant Column** is created if there are any target or output attributes for which there is no mapping to the source attributes. The default value for the constant column is NULL.

**Note:**

The target attribute is mapped to all the sources.

For example, you have two tables with similar structures, Actual Sales and Planned Sales, corresponding to the sales of products. You want to see the combined data in a single view, but differentiate between the data from the two tables for comparisons. To do so, you can create a union view between the two tables and have a constant column indicating constant values like A & P, as indicated below:

Actual Sales

Sales	Product
5000	A1
2000	B1

Planned Sales

Sales	Product
3000	A1
6000	B1

The result of this query can be as follows:

Actual Planned Indicator	Sales	Product
A	5000	A1
P	3000	A1
A	2000	B1
P	6000	B1

## 9.5 Creating an Analytic Privilege

You apply analytic privileges when business users access values with certain combinations of dimension attributes. You can use analytic privileges to partition data among various users sharing the same data foundation. You can assign analytic privileges in two ways:

- **Privileges for Selected Content Models** - to define restrictions for a selected group of models. For example, you have defined a restriction on an attribute CUST\_ID that belongs to the CUSTOMER attribute view. The restriction is applicable to all the selected content models that use this attribute view.

- **Privileges for All Content Models** - to define restrictions for all content models across multiple packages that are specific to a user. For example, you have defined a restriction on an attribute view CUSTOMER that belongs to the SALES package and the user EMP. The restriction is applicable to all content models across multiple packages that use CUSTOMER and that are specific to EMP.

**Note:**

The restrictions that you set for attributes belonging for attribute views also apply to analytic views, if the same attribute is reused. However, the restrictions set for attributes belonging to either analytic or calculation views apply only to the analytic or calculation views, if the same attribute is reused.

**Procedure****1. Set Parameters for the Analytic Privilege**

1. In the **Modeler** perspective, expand the system node from the **Navigator** pane.
2. Expand the **Content** node.
3. Select the required package.
4. From the context menu, choose **New > Analytic Privilege**.
  - a. Enter the name and description of the analytic privilege.
  - b. Choose **Next**.
  - c. Select the required information models.
  - d. Choose **Add**.
  - e. Choose **Finish**.

**Note:**

You can choose to add more content models in the editor pane.

**2. Add Attributes**

1. If you want to add restrictions for all content models, select the **Applicable to all Content Models** checkbox.

**Note:**

If you do not select this option, the restrictions you create apply only to the secured list of models.

2. In the **Associated Attributes Restrictions** pane, choose **Add**.
3. Select the attribute(s) for which you want to apply the restriction.
4. Choose **OK**.

**Note:**

If you do not add any attributes for restrictions then it means unrestricted access to the selected/all content models, depending on the status of the check-box.

**3. Assign Restrictions**

1. In the **Assign Restrictions** pane, choose **Add**.
2. You can choose to change the value defined for **Operator**.
3. You can choose to enter a value or use the Value Help Dialog to search for a value.

**Note:**

To activate the analytic privilege, you must assign a minimum of one restriction to each attribute.

**4. Activate the Privilege**

1. Choose **File > Save**.
2. From the context menu of the privilege, select one of the following options as appropriate:
  - Activate – to activate the current view only.
  - Cascade Activate - to activate the current view and impacted objects.

**Result**

The administrator associates user with an analytic privilege. You can view the users and roles assigned to the analytic privilege on the **Users/Roles** tab page.

## 9.6 Creating a Procedure

You use this procedure to create SQL procedures that can be reused by other content objects.

**Note:**

The catalog object that is generated on procedure activation is stored in the `_SYS_BIC` schema.

**Prerequisites**

You have repository authorization at the package level.

**Procedure**

1. On the **Quick Launch** tab page, choose **Procedure**.
2. Enter a name and description for the procedure.
3. For unqualified access in SQL, select the required schema from the **Default Schema** dropdown list.

**Note:**

If you do not select a default schema, while scripting you need to provide fully qualified names of the catalog objects like `<SchemaName>.<TableName>`. If you do specify a default schema, and write SQL script such as "select \* from `<TableName>`", the specified default schema is used at runtime to refer to the table.

4. Select the package in which you want to save the procedure.
5. Select the required option from the **Run With** drop-down list.

**Note:**

*Definer's right:* If you want to control access to the procedure for other users.

*Invoker's right:* If you want the system to grant access to the procedure for other users.

6. Select the required access mode as follows:

Access Mode	Purpose
Read Only	Use this mode to create procedures for fetching table data.
Read Write	Use this mode to create procedures for fetching and updating table data.

7. Select the language in which you are writing the procedure.
8. Choose **Finish**.
9. In the function editor pane, write a script for the function using the following data types:
  - Table or scalar data types for input parameters.
  - Table data type for output parameters.

**Note:**

- You can only write one function in the function body. However, you can refer to other functions. For more information about the syntax and examples of SQL functions, see [SQL Functions on Content Assist](#).

10. Choose **File > Save**.
11. From the context menu of the view, choose the appropriate option:
  - To activate only the current view, choose **Activate**.
  - To activate the current view and impacted objects, choose **Cascade Activate**.

**Note:**

For more information about activation, see [Activating Objects](#).

## 9.7 Using Currency and Unit of Measure

You use this procedure to define a measure as an amount or weight in the analytical space and to perform currency conversion.

To simplify the process of currency conversion, the system provides a list of currencies, and exchange rates based on the tables imported for currency. Currency conversion is performed based on source currency, target currency, exchange rate, and date of conversion. You can also select currency from the attribute data used in the view.

**Note:**

Currency conversion is only enabled for analytic views.

For example, you need to generate a sales report for a region in a particular currency, and you have sales data in database tables in a different currency. You can create an analytic view by selecting the table column containing the sales data in this other currency as a measure, and perform currency conversion. Once you activate the view, you can use it to generate reports.

**Prerequisites**

- You have imported tables T006 and T006A for Unit of Measure.

- You have imported TCURC, TCURF, TCURN, TCURR, TCURT, TCURV, TCURW, and TCURX for currency.

**Procedure**

- In the **Output** pane, select a measure.
- In the **Properties** pane, select **Measure Type**.
- If you want to associate the measure with a currency, perform the following substeps:
  - In the **Measure Type** dropdown list, select the value **Amount with Currency**.
  - In the **Currency Dialog**, select the required **Type** as follows:

Type	Purpose
Fixed	To select currency from the currency table TCURC.
Attribute	To select currency from one of the attributes used in the view.

- Select the required value, and choose **OK**.
- If you want to convert the value to another currency, choose **Enable for Conversion**.
  - To select the target currency, choose **Currency**.

**Note:**

In addition to the types **Fixed** and **Attribute**, you can select **Input Parameter** to provide the currency at runtime.

- Select the source currency.
  - Select the exchange type for exchange rates.
  - Select the date for conversion.
- From the dropdown list, select the required value that is used populate data if the conversion fails:

Option	Result
Fail	In data preview, the system displays an error for conversion failure.
Set to NULL	In data preview, the value for the corresponding records is set to NULL.
Ignore	In data preview, you view the unconverted value for the corresponding records.

- If you want to associate a measure with a unit of measure other than currency, perform the following substeps:
  - Select the value **Quantity with Unit of Measure** in the **Measure Type** drop-down list.
  - In the **Unit Dialog**, select the required **Type** as follows:

Type	Purpose
Fixed	To select a unit of measure from the unit tables T006 and T006A.
Attribute	To select a unit of measure from one of the attributes used in the view.

c. Select the required value, and choose **OK**.

5. Choose **OK**.

**Note:**

You can associate Currency or Unit of Measure with a calculated measure, and perform currency conversion for a calculated measure by editing it.

## 9.8 Assigning Variables

You use this procedure to assign variables to a filter at design time for obtaining data based on the values you provide for the variable. At runtime, you can provide different values to the variable to view the corresponding set of attribute data. You provide values to the variables either by entering the values manually, or by selecting them from the Value Help dialog.

**Note:**

You can apply variables to attributes of analytic and calculation views.

The following types of variables are supported:

Type	Description
Single Value	Use this to filter and view data based on a single attribute value. For example, to view the sales of a product where the month is equal to January.
Interval	Use this to filter and view a specific set of data. For example, to view the expenditure of a company from March to April.
Range	Use this to filter and view data based on the conditions that involve operators such as "=" (equal to), ">" (greater than), "<" (less than), ">=" (greater than or equal to), and "<=" (less than or equal to). For example, to view the sales of all products in a month where the quantity sold is $\geq 100$ .

Each type of variable can be either mandatory or non-mandatory. For a mandatory variable, you need to provide a value at runtime. However, for a non-mandatory variable, if you have not specified a value at runtime, you view unfiltered data.

**Note:**

You can check whether a variable is mandatory or not from the properties of the variable in the properties pane.

**Procedure****1. Create a Variable**

1. In the **Output** pane, right-click the **Variables** node.
2. From the context menu, choose **New** and perform the following substeps:
  - a. Enter a name and description.
  - b. Select the required attribute from the drop-down list.

**Note:**

At runtime, the value for the variable is fetched from the selected attribute's data.

- c. Choose the required **Selection Type** from the drop-down list.
- d. Choose **OK**.

**Note:**

You can also choose to create a variable using the **Create Variable** option from the context menu of an attribute. In this case, the details of variable is pre-filled.

**2. Assign a Variable to a Filter**

1. On the **Logical View** tab page, right-click the attribute.
2. From the context menu, choose **Apply Filter**.
3. From the **Operator** drop-down list, choose **Variable**.
4. From the **Variable** drop-down list, choose the required variable.
5. Choose **OK**.

**Note:**

You can also choose to create a variable and apply a filter using the **Create Variable - Apply Filter** option from the context menu of an attribute.

## 9.9 Searching Related Tables

While building models, you need to select tables for creating the master data, where, one table can have multiple related tables. You use this procedure to search and add the associated tables for a table in the data foundation.

**Note:**

System will suggest related tables from your schema only.

**Prerequisites**

You have imported DD08L table in any one of the schemas.

**Procedure**

1. In the **Data Foundation** tab page, select the required table.

2. From the context menu, choose **Propose Tables**.
3. From the drop-down list, select the table for which you want to find the related tables.
4. From the drop-down list, select the schema in which DD08L metadata table is present.
5. To add related table(s) in Data Foundation tab page, perform the following steps:
  - a. Select the required table(s) from the list.
  - b. Choose **OK**.

**Note:**

If you want to build joins suggested by system between related tables, check the **Automatically Propose Joins** checkbox.

6. Choose **OK**.

## 9.10 Using System Generated Joins

While building models, you use number of tables and define relationship between them using joins. You use this procedure to build joins suggested by the system for tables based on the metadata information. In a business scenario where, tables have many columns and many associated tables, system generated joins helps to define the relationship between the tables used.

**Note:**

- The system will suggest join only if there is any relationship between the tables.
- You can edit the joins suggested by the system.

**Prerequisites**

You have imported DD08L table in any one of the schemas.

**Procedure**

1. In the **Data Foundation** tab page, select the required tables.
2. From the context menu, choose **Suggested Joins** to create joins among tables.

**Note:**

You can overwrite the existing joins with the system generated joins.

## 9.11 Activating Objects

You activate objects available in your workspace to expose the objects for reporting and analysis. Based on your requirements, you can:

- **Activate** - To deploy the inactive objects.
- **Redeploy** - To deploy the active objects in one of the following scenarios:

- If your runtime object gets corrupted or deleted, and you want to create it again.
- If an object goes through client-level activation and server-level activation but fails at MDX, and the object status is still active.
- **Cascade Activate** - To activate the inactive object along with the impacted objects. For example, consider an analytic view AN1 being used in a calculation view C1. If you make changes to AN1, and choose **Cascade Activate**, it activates C1 as well.

Depending on where you invoke the activation, redeployment or cascade activation, the behavior is as follows:

Context	Activate	Redeploy	Cascade Activate
<b>Quick Launch</b> tab page	A dialog box appears with a preselected list of all inactive objects in your workspace.	A dialog box appears with a list of active objects in your workspace.	Not Applicable
<b>Pack-age</b> context menu	A dialog box appears with a preselected list of all inactive objects in your workspace.	A dialog box appears with a list of active objects in your workspace.	Not Applicable
<b>Con-tent</b> context menu	A dialog box appears with a preselected list of all inactive objects in your workspace.	Not Applicable	Not Applicable
<b>Editor</b>	A dialog box appears with a preselected list of the selected object along with all the dependent objects that are inactive in your workspace.	A redeployment job is submitted for the selected object.	Not Applicable
Object context menu	A dialog box appears with a preselected list of the selected object along with all the dependent objects that are inactive in your workspace.	A redeployment job is submitted for the selected object.	A dialog box appears with a preselected list of the inactive object and impacted objects in your workspace..

**Note:**

- If an object is the only inactive object in the workspace, the activation dialog box is skipped and the activation job is submitted.
- If an object is inactive and you want to revert back to the active version, from the editor or object context menu, choose **Revert To Active**.

## 9.12 Creating an Input Parameter

You use this procedure to allow you to provide input for the parameters within stored procedures, to obtain a desired functionality when the procedure is executed. You use input parameters as placeholders during currency conversion, input parameters of the script node, filter expressions, formulas like calculated measures, and calculated attributes. The calculation of the formula is based on the input you provide at runtime during data preview.

You can apply input parameters in analytic and calculation views. If a calculation view is created using an analytic view with input parameters, those input parameters are also available in the calculation view but you cannot edit them.

The following types of input parameters are supported:

Type	Description
Attribute Value	Use this when the value of a parameter comes from an attribute.
Currency	Use this when the value of a parameter is in a currency format, for example, to specify the target currency during currency conversion.
Date	Use this when the value of a parameter is in a date format, for example, to specify the date during currency conversion.
Static List	Use this when the value of a parameter comes from a user-defined list of values.
Empty	Use this when the value of a parameter could be anything of the selected data type.

Each type of input parameter can be either mandatory or non-mandatory. For a mandatory input parameter, it is necessary to provide a value at runtime. However, for a non-mandatory input parameter, if you have not specified a value at runtime, the data for the column where the input parameter is used is blank.

**Note:**

You can check whether an input parameter is mandatory or not from the properties of the input parameter in the properties pane.

**Example:**

- If you want to create a formula to analyze the annual sales of a product in various regions, you can use *Year* and *Region* as input parameters.
- If you want to preview a sales report with data for various countries in their respective currency for a particular date for correct currency conversion, you can use *Currency* and *Date* as input parameters.

**Procedure**

1. In the **Output** pane, right-click the **Input Parameters** node.
2. From the context menu, choose **New**.
  - a. Enter a name and description.
  - b. Select the type of input parameter from the drop-down list.

**Note:**

For the **Attribute Value** type of input parameter, you need to select the attribute from the drop-down list. At runtime the value for the input parameter is fetched from the selected attribute's data.

- c. Select a data type.
- d. Enter the length and scale for the input parameter.
- e. Choose **OK**.

## 9.13 Creating a Hierarchy

You use this procedure to create hierarchies between attributes to enhance analysis by displaying attributes according to their defined hierarchical relationships. Hierarchies can exist cross attributes (that is, Country - State - City) or within the values of a single attribute (that is, Employee manager - employee direct report).

You structure and define relationships between attributes in the attribute view and calculation view using the following hierarchy types:

- **Level Hierarchy**

A level hierarchy is rigid in nature, and the root and the child nodes can be accessed only in the defined order.

Level hierarchies consist of one or more levels of aggregation. Members roll up into the next higher level in a many-to-one relationship, and members at this higher level roll up into the next higher level, and so on to the top level.

For example: an address hierarchy comprised of region, country, state, and so on.

- **Parent/Child Hierarchy**

A parent/child hierarchy is a hierarchy in a standard view that contains a parent attribute. A parent attribute describes a self-referencing relationship, or self-join, within the main table. Parent-child hierarchies are constructed from a single parent attribute.

For example: a bill of materials hierarchy (parent and child) or an employee master (employee and manager) hierarchy.

### **Procedure**

#### **Creating a Level Hierarchy**

1. In the **Output** pane, right-click **Hierarchy**.
2. Choose **New Level Hierarchy**.
3. Enter a name and description for the hierarchy.
4. Add the required attributes from the drop-down list.

**Note:**

You can select attributes from the required table columns in the drop-down list to add to the view.

5. Select the required **Level Type**.

**Note:**

The level type is used to specify formatting instructions for the level attributes.

For example, a level of the type LEVEL\_TYPE\_TIME\_MONTHS can indicate that the attributes of the level should have a text format such as "January", and LEVEL\_TYPE\_REGULAR indicates that a level does not require any special formatting.

6. Choose **OK**.

**Creating a Parent/Child Hierarchy**

1. In the **Output** pane, right-click **Hierarchy**.
2. Choose **New Parent Child Hierarchy**.
3. Enter a name and description for the hierarchy.
4. If you have not specified the principal key attribute, select the **Child Attribute** from the drop-down list.

**Note:**

Since you have not selected the principal key attribute while adding attributes, the selected child attribute is added as the principal key attribute of the view.

5. Select the **Parent Attribute** from the dropdown list.

**Note:**

The hierarchies belonging to an attribute view are available in an analytic view that reuses the attribute view, in read-only mode. However, the hierarchies belonging to an attribute view are not available in a calculation view that reuses the attribute view.

## 9.14 Maintaining Search Attributes

You use this procedure to enable attribute search for an attribute used in a view. Various properties related to attribute search are as follows:

- **Freestyle Search:** Set to True if you want to enable the freestyle search for an attribute. You can exclude attributes from freestyle search by setting the property to False.
- **Weights for Ranking:** To influence the relevancy of items in the search results list, you can vary the weighting of the attribute. You can assign a higher or lower weighting (range 0.0 to 1.0). The higher the weighting of the attribute, the more influence it has in the calculation of the relevance of an item. Items with a higher relevance are located higher up the search results list. Default value: 0.0.

**Note:**

To use this setting the property Freestyle Search must be set to True.

- **Fuzziness Threshold:** This parameter is reserved for future usage of Fault-tolerant search.



**Note:**

It is recommended to use the default values for Ranking and Fuzziness at the beginning. You can tune the search settings based on the experiences that you gained after some searches. Also collect the feedback of your users and use it for fine-tuning.

## 9.15 Managing Data Foundation Layout

You use this procedure to adjust the data foundation and logical view layout comprising user interface controls like, tables and attribute views in a more readable manner. This functionality is supported for attribute views and analytic views.

The options available are as follows:

Option	Purpose	Substeps
Auto Arrange	Use this option to arrange the user interface elements automatically.	In the editor tool bar, choose  .
Show outline	Use this option to view an outline of the elements arranged so that , you do not have to navigate in the editor using horizontal and vertical scrollbars.	In the editor tool bar, choose  .
Highlight related tables	Use this option if you want to view only those tables that are related to a table selected in the editor.	<ol style="list-style-type: none"> <li>1. In the editor, right-click the selected table.</li> <li>2. From the context menu, choose <b>Highlight related tables</b>.</li> </ol>
Display	Use this option if you have a table with a large number of columns in the editor, and you want to view them in a way that meets your needs: for example, only the table name, or only joined columns, or the expanded form with all the columns.	<ol style="list-style-type: none"> <li>1. In the editor, right-click the relevant table.</li> <li>2. From the context menu, choose <b>Display</b>.</li> <li>3. If you want to view only the table name, choose <b>Collapsed</b>.</li> <li>4. If you want to view all the columns of the table, choose <b>Expanded</b>.</li> <li>5. If you want to view only the joined columns of the table, choose <b>Joins only</b>.</li> </ol>

## 9.16 SQL Functions on Content Assist

This topic covers the functions that are supported using the content assist (ctrl + space) in the SQL editor. You use this section to view the syntax and examples for the supported functions.

Function	Syntax	Example
call	<ul style="list-style-type: none"> <li>CALL - Internal Procedure Call</li> <li>CALL Procedure Called From Client</li> <li>CALL...WITH OVERVIEW From Client</li> </ul>	Not Applicable
case	CASE	Not Applicable
ce_aggregation	CE_AGGREGATION	CE_AGGREGATION
ce_calc	<ul style="list-style-type: none"> <li>CE_CALC</li> <li>CE_CALC_VIEW</li> </ul>	<ul style="list-style-type: none"> <li>CE_CALC usage in CE_PROJECTION</li> <li>CE_CALC_VIEW</li> </ul>
ce_column_table	CE_COLUMN_TABLE	CE_COLUMN_TABLE
ce_conversion	CE_CONVERSION	CE_CONVERSION
ce_join	<ul style="list-style-type: none"> <li>CE_JOIN</li> <li>CE_JOIN_VIEW</li> </ul>	<ul style="list-style-type: none"> <li>CE_JOIN</li> <li>CE_JOIN_VIEW</li> </ul>
ce_left_outer_join	CE_LEFT_OUTER_JOIN	CE_LEFT_OUTER_JOIN
ce_olap_view	CE_OLAP_VIEW	CE_OLAP_VIEW
ce_projection	CE_PROJECTION	CE_PROJECTION
ce_right_outer_join	CE_RIGHT_OUTER_JOIN	CE_RIGHT_OUTER_JOIN
ce_union_all	CE_UNION_ALL	CE_UNION_ALL
ce_vertical_union	CE_VERTICAL_UNION	CE_VERTICAL_UNION
close	CLOSE CURSOR	Not Applicable
drop procedure	DROP PROCEDURE	Not Applicable
drop type	DROP TYPE	Not Applicable
exec	EXEC Dynamic Sql	EXEC usage
explain plan	EXPLAIN PLAN	EXPLAIN PLAN usage
export	EXPORT as Binary	Not Applicable
fetch	FETCH CURSOR	Not Applicable

Function	Syntax	Example
for	<ul style="list-style-type: none"> <li>FOR LOOP</li> <li>FOR Looping over Resultsets</li> </ul>	<ul style="list-style-type: none"> <li>FOR LOOP Nested</li> <li>FOR Looping over resultsets using Procedure</li> </ul>
if	IF	<ul style="list-style-type: none"> <li>IF for NULL</li> <li>IF for NULL check</li> </ul>
import	<ul style="list-style-type: none"> <li>IMPORT FROM</li> <li>IMPORT as Binary</li> </ul>	Not Applicable
open	OPEN CURSOR	Not Applicable
while	WHILE LOOP	WHILE LOOP nested loop
create	<ul style="list-style-type: none"> <li>CREATE TYPE as TABLE</li> <li>CREATE PROCEDURE Read Only Procedure</li> </ul>	<ul style="list-style-type: none"> <li>CREATE PROCEDURE - Implementing Functional Logic</li> <li>CREATE PROCEDURE - Read only Procedure</li> <li>CREATE PROCEDURE with Cursor</li> <li>CREATE PROCEDURE with IF as Control Statement</li> <li>CREATE PROCEDURE with Scalar Variable</li> </ul>
cursor	CURSOR	Not Applicable

**Note:**

For more information about the functions, see [SAP HANA Database – SQLScript Guide](#).

## 9.17 Functions used in Expressions

This topic covers the functions that you can use while creating expressions like, calculated attributes and calculated measures.

### Conversion Functions

Function	Syntax	Purpose	Example
int	int int(arg)	convert arg to int type	int(2)
float	float float(arg)	convert arg to float type	float(3.0)
double	double double (arg)	convert arg to double type	double(3)
sdfloat	sdfloat sdfloat (arg)	convert arg to sdfloat type	
decfloat	decfloat decfloat (arg)	convert arg to decfloat type	
fixed	fixed fixed (arg, int, int)	arg2 and arg3 are the intDigits and fractdigits parameters, respectively. Convert arg to a fixed type of either 8, 12, or 16 byte length, depending on intDigits and fractDigits	fixed(3.2, 8, 2) + fixed(2.3, 8, 3)
string	string string (arg)	convert arg to string type	
raw	raw raw (arg)	convert arg to raw type	

Function	Syntax	Purpose	Example
date	date date(stringarg) date date(fixedarg) date date(int, int) date date(int, int, int) date date(int, int, int, int) date date(int, int, int, int, int) date date(int, int, int, int, int, int)	convert arg to date type. The first version parses a string in the format "yyyy-mm-dd hh:mi:ss" where trailing components except for the year may be omitted. The version with one fixed number arg strips digits behind the comma and tries to make a date from the rest. The other versions accept the individual components to be set.	date(2009) -> date('2009') date(2009, 1, 2) -> date('2009-01-02') date(fixed(20000203135026.1234567, 10, 4)) -> date('2000-02-03 13:50:26')
longdate	longdate longdate(stringarg) longdate longdate(fixedarg) longdate longdate(int, int, int) longdate longdate(int, int, int, int, int) longdate longdate(int, int, int, int, int, int) longdate longdate(int, int, int, int, int, int)	convert arg to longdate type, similar to date function above.	longdate(fixed(20000203135026.1234567, 10, 5)) -> longdate('2000-02-03 13:50:26.1234500') longdate(2011, 3, 16, 9, 48, 12, 1234567) -> longdate('2011-03-16 09:48:12.1234567')
time	time time(stringarg) time time(fixedarg) time time(int, int) time time(int, int, int)	convert arg to time type, similar to date function above	

### String Functions

Function	Syntax	Purpose
strlen	int strlen(string)	returns the length of a string in bytes, as an integer number.
midstr	string midstr(string, int, int)	returns a part of the string starting at arg2, arg3 bytes long. arg2 is counted from 1 (not 0)
leftstr	string leftstr(string, int)	returns arg2 bytes from the left of the arg1. If arg1 is shorter than the value of arg2, the complete string will be returned.
rightstr	string rightstr(string, int)	returns arg2 bytes from the right of the arg1. If arg1 is shorter than the value of arg2, the complete string will be returned.
instr	int instr(string, string)	returns the position of the first occurrence of the second string within the first string ( $\geq 1$ ) or 0, if the second string is not contained in the first.
hextoraw	string hextoraw(string)	convert a hexadecimal representation of bytes to a string of bytes. The hexadecimal string may contain 0-9, upper or lowercase a-f and no spaces between the two digits of a byte; spaces between bytes are allowed.
rawtohex	string rawtohex(string)	convert a string of bytes to its hexadecimal representation. The output will contain only 0-9 and (upper case) A-F, no spaces and is twice as many bytes as the original string.

Function	Syntax	Purpose
ltrim	string ltrim(string) string ltrim(string, string)	removes a whitespace prefix from a string. The Whitespace characters may be specified in an optional argument. This functions operates on raw bytes of the UTF8-string and has no knowledge of multi byte codes (you may not specify multi byte whitespace characters).
rtrim	string rtrim(string) string rtrim(string, string)	removes trailing whitespace from a string. The Whitespace characters may be specified in an optional argument. This functions operates on raw bytes of the UTF8-string and has no knowledge of multi byte codes (you may not specify multi byte whitespace characters).
trim	string trim(string) string trim(string, string)	removes whitespace from the beginning and end of a string.
lpad	string lpad(string, int) string lpad(string, int, string)	add whitespace to the left of a string. A second string argument specifies the whitespace which will be added repeatedly until the string has reached the intended length. If no second string argument is specified, chr(32) (' ') will be added. This function operated on UTF-8 bytes and has no knowledge of unicode characters (neither for the whitespace string nor for length computation).
rpad	string rpad(string, int) string rpad(string, int, string)	add whitespace to the end of a string. A second string argument specifies the whitespace which will be added repeatedly until the string has reached the intended length. If no second string argument is specified, chr(32) (' ') will be added. This function operated on UTF-8 bytes and has no knowledge of unicode characters (neither for the whitespace string nor for length computation).
replace	string replace(string, string, string)	replace every occurrence of arg2 in arg1 with arg3 and return the resulting string

### Mathematical Functions

Function	Syntax	Purpose	Example
sign	int sign(double) int sign(time) int sign(date)	Sign returns -1, 0 or 1 depending on the sign of its argument. Sign is implemented for all numeric types, date and time.	
abs	double abs(double) decfloat abs(decfloat) decfloat abs(decfloat) time abs(time)	Abs returns arg, if arg is positive or zero, -arg else. Abs is implemented for all numeric types and time.	
round	double round(double, int)	round does rounding of absolute values toward zero while the sign is retained	round(123.456, 0) = 123 round(123.456, 1) = 123.5 round(-123.456, 1) = -123.5 round(123.456, -1) = 120
round-down	double rounddown(double, int)	rounddown rounds toward negative infinity making rounddown(-1.1, 0) = -2	rounddown(123.456, -1) = 120 rounddown(-123.456, -1) = -130

### Date Functions

Function	Syntax	Purpose
utctolocal	utctolocal(datearg, timezonearg)	interprets datearg (a date, without timezone) as utc and convert it to the timezone named by timezonearg (a string)
localtoutc	localtoutc(datearg, timezonearg)	converts the local datetime datearg to the timezone specified by the string timezonearg, return as a date
weekday	weekday(date)	returns the weekday as an integer in the range 0..6, 0 is monday.
now	now()	returns the current date and time (localtime of the server timezone) as date

Function	Syntax	Purpose
daysbetween	<p>daysbetween(date1, date2)</p> <p>daysbetween(daydate1, daydate2)</p> <p>daysbetween(seconddate1, seconddate2)</p> <p>daysbetween(longdate1, longdate2)</p>	<p>returns the number of days (integer) between date1 and date2. The first version is an alternative to date2 - date1.</p> <p>Instead of rounding or checking for exactly 24 hours distance, this will truncate both date values today precision and subtract the resulting day numbers, meaning that if arg2 is not the calendar day following arg1, daysbetween will return 1 regardless of the time components of arg1 and arg2.</p>
secondsbetween	<p>secondsbetween(seconddate1, seconddate2)</p> <p>secondsbetween(longdate1, longdate2)</p>	<p>returns the number of seconds the first to the second arg, as a fixed point number. The returned value is positive if the first argument is less than the second. The return values are fixed18.0 in both cases (note that it may prove more useful to use fixed11.7 in case of longdate arguments).</p>
component	<p>component(date, int)</p>	<p>the int argument may be int the range 1..6, the values mean year, day, month, hour, minute, second, respectively. If a component is not set in the date, the component function will return a default value, 1 for the month or the day, 0 for other components. The component function may also be applied to longdate and time types.</p>
addseconds	<p>addseconds(date, int)</p> <p>addseconds(seconddate, decfloat)</p> <p>addseconds(longdate, decfloat)</p>	<p>Return a date plus a number of seconds. Fractional seconds will also be used in case of longdate. Null handling is (in opposition to the default done with adds) to return null if any argument is null.</p>
adddays	<p>adddays(date, int)</p> <p>adddays(daydate, int)</p> <p>adddays(seconddate, int)</p> <p>adddays(longdate, int)</p>	<p>Return a date plus a number of days. Null handling is (in opposition to the default done with adds) to return null if any argument is null.</p>

**Misc Functions**

Func-tion	Syntax	Purpose	Example
if	if(intarg, arg2, arg3)	return arg2 if intarg is considered true (not equal to zero), else return arg3. Currently, no shortcut evaluation is implemented, meaning that both arg2 and arg3 are evaluated in any case. This means you cannot use if to avoid a divide by zero error which has the side effect of terminating expression evaluation when it occurs.	if("NETWR"<=500000,'A', if("NETWR"<=1000000,'B','C') )
in	in(arg1, ...)	return 1 (= true) if arg1 is equal to any of the remaining args, return 0 else	
case	case(arg1, default) case(arg1, cmp1, value1, cmp2, value2, ..., default)	return value1 if arg1 == cmp1, value2 if arg1 == cmp2 etc, default if there no match	case("CATEGORY", 'A', 'LV', 'B', 'MV', 'HV')
isnull	isnull(arg1)	return 1 (= true), if arg1 is set to null and null checking is on during Evaluator run (EVALUATOR_MAY_RETURN_NULL)	



# Processing Content Objects

## 10.1 Previewing Data of Content Objects

You use this procedure to preview the content of content models for analysis purposes.

### Procedure

1. In the **Modeler** perspective, expand the **Content** node of the required system.
2. Select the object from a package for which you want to view the content.
3. From the context menu, choose **Data Preview**.

System displays the content in different formats as mentioned in the table below.

Tab Page	Displays...
Raw Data	All attributes along with data in a table format.
Distinct values	All attributes along with data in a graphical format.
Analysis	All attributes and measures in a graphical format.

4. Navigate to the required tab page and view the content.

## 10.2 Validating Models

You use this procedure to check if there are any errors in an information object and if the object is based on the rules that you specified as part of preferences. For example, the "Check join: SQL" rule checks that the join is correctly formed.

For more information about setting preferences, see [Setting Preferences for Modeler](#) .

### Procedure

1. On the **Quick Launch** page, choose **Validate**.
2. From the **Available** list, select the required models that system must validate.
3. Choose **Add**.
4. Choose **Validate**.

## 10.3 Comparing Versions of Content Objects

You use this procedure to compare the active version of an information object with its inactive version.

### Procedure

1. In the **Modeler** perspective, expand the **Content** node of the required system.
2. Select the required object from a package.
3. From the context menu, choose **Open**.
4. In the editor pane, choose **Show Active Version**.
5. Compare the inactive and active versions of the object.
6. Choose **OK**.

## 10.4 Viewing Version History of Content Objects

You use this procedure to view the version details of an information model for tracking purposes.

### Procedure

1. In the **Modeler** perspective, expand the **Content** node of the required system.
2. Select the required object from a package.
3. From the context menu, choose **History**.

## 10.5 Switching Ownership of Inactive Objects

You use this procedure to take over the ownership of the inactive version of an object from other users' workspace.

Objects in edit mode in other workspaces are not available for modification. In order to modify such objects you need to own the inactive object.

The options available for changing the inactive object ownership are as follows:

Option	Purpose
Switch Ownership	To take over multiple inactive objects from other users. Inactive objects that do not have an active version are also available for take over using this option
Take Over	To take a single inactive object from other workspace that you wish to edit using editor.

**Note:**

Using this functionality you can only own the inactive version of the object. The active version is owned by the user who created and activated the object.

**Prerequisite**

You have obtained the **Work in Foreign Workspace** authorization.

**Procedure**

1. If you want to own multiple inactive objects from other workspaces, perform the following substeps:
  - a. In the **Quick Launch** tab page, choose **Switch Ownership**.
  - b. In the **Source User** field, select the user who owns the inactive objects.
  - c. Add the required inactive objects to the **Selected Models** section.
  - d. Choose **OK**.
2. If an object opens in read-only mode and want to edit it, perform the following substeps:
  - a. In the editor toolbar, select **Switch Version**.
  - b. Choose **Take Over**.

**Note:**

You can choose to save the changes made by the other user (owning the inactive version earlier) to the inactive version of the object.

## 10.6 Generating Object Documentation

You use this procedure to capture the details of an information model or a package into a single document. This helps you in viewing the necessary details from the document, instead of referring to multiple tables. The following table specifies the details that you can view from the document.

Type	Description
Attribute View	General object properties, attributes, calculated attributes, data foundation joins, cross references, and where-used
Analytic View	General object properties, private attributes, calculated attributes, attribute views, measures, calculated measures, restricted measures, variables, input parameters, data foundation joins, logical view joins, cross references, and where-used
Calculation View	General object properties, attributes, calculated attributes, measures, calculated measures, counters, variables, input parameters, calculation view SQL script, cross references, and where-used
Package	Sub-packages, general package properties, and list of content objects

### Procedure

1. From the **Quick Launch** page, choose **Auto Documentation**.
2. In the **Select Content Type** field, select one of the following options as required:

Option	Description
Model Details	To generate documentation for models such as attribute, analytic, and calculation views.
Model List	To generate documentation for packages.

3. Add the required objects to the **Target** list.
4. Browse the location where you want to save the file.
5. Choose **Finish**.

## 10.7 Enabling Multilanguage Support for Objects

You use this procedure to enable translation of text pertaining to objects and their elements that did not have Multilanguage support.

For example, you can enable Multilanguage support for models along with their elements like attributes and measures in different languages.

### Procedure

1. From the **Quick Launch** tab page, choose **Migrate**.
2. Select the required objects.
3. Choose **Add**.
4. Choose **OK**.

**Result**

Object texts along with the corresponding elements are flagged for translation. These objects can be viewed in multiple languages provided that the object texts are translated.

## 10.8 Checking Model References

You use this procedure to identify whether an information model is referenced by any other information model(s).

**Procedure**

1. In the **Modeler** perspective, expand the system node in the **Navigator** pane.
2. Expand the **Content** node.
3. Expand the required package node.
4. Select the required object.
5. From the context menu, choose **Where Used**.

## 10.9 Viewing the Job Log

The job log displays information related to requests entered for a job. Job log comprises of two tab pages as follows:

- Current: Lists all waiting, running, and last five jobs.
- History: Lists all the jobs.

**Note:**

You can terminate the job only if it is in the waiting state.

You can perform the following operations using the job log:

- Open Job Details: You use this to view the job summary in the current tab page.
- Open Job Log File: You use this to view the information pertaining to a job in detail using the internal browser.
- Clear Log Viewer: You use this to delete all the job from the current tab page.
- Export Log File: You use this to export the log file to a target location other than the default location for further reference.
- Delete Job: You use this to delete single job from the current tab page.



# Exporting Objects

## 11.1 Creating a Delivery Unit

You use this procedure to create a delivery unit, which is a group of transportable objects for content delivery. For example, to transport the design time objects that are stored in the SAP HANA database repository to other SAP HANA database installations.

**Note:**

A delivery unit is identified by a vendor and a name.

**Prerequisite**

You have ensured that the vendor property has been set. For more information about how to set the vendor property, see [Setting Preferences for Modeler](#).

**Procedure**

1. On the **Quick Launch** page, choose **Delivery Units**.
2. Choose **Create**.
3. Enter the details as follows:

Field Name	Value
Name	Name of the delivery unit.
Responsible	Name of the responsible user.
Version (optional)	Version of the delivery unit as defined by the responsible user.
Support Package Version (optional)	Support Package level of the delivery unit.
Patch Version (optional)	Patch level of the delivery unit.

**Note:**

The version information of a delivery unit is helpful for shipment use cases like, from SAP to customer.

4. Choose **OK**.
5. In the **Assigned Packages** section, add the required packages that you want to export.
6. Choose **Close**.

## 11.2 Exporting Objects using Delivery Unit

You use this procedure to export all packages that comprises a delivery unit and the relevant objects contained in the packages, to the client or to the SAP HANA server file system. The exported objects are available for import in the form of .tgz file at the specified location. Users who want to reuse these objects can import the related file.

### Prerequisite

You have created a delivery unit. More information: [Creating a Delivery Unit](#).

### Procedure

1. In the **File** menu, choose **Export**.
2. Expand the **SAP HANA content** node.
3. Select **Delivery Unit**.
4. Choose **Next**.
5. Select the system from where you want to export the models, and choose **Next**.

#### Note:

If you choose to export objects using **Export** option on the **Quick Launch** tab page, you do not have to select the source system as your current system is considered as the source.

6. Select the delivery unit.
7. If you want to export the packages that are created or modified during a time range, select the **Filter by time** checkbox.
8. Select the export location as one of the following:
  - To export objects on server to a specific file directory specified by an administrator, select **Export To Server**.
  - To export objects to a client file system, select **Export to Client**.
9. Choose **Next**.
10. Choose **Finish**.

## 11.3 Exporting Objects using Developer Mode

You use this procedure to export individual objects to a directory on your client computer. For example, developers working on content creation can use this mode to share objects in a collaborative development environment.

### Note:

This mode of export should only be used in exceptional cases since this does not cover all aspects of an object like, the translatable texts are not copied.

**Procedure**

1. In the **File** menu, choose **Export**.
2. Expand the **SAP HANA content** node.
3. Select **Developer Mode**, and choose **Next**.
4. Select the system from where you want to export the objects, and choose **Next**.

**Note:**

If you choose to export objects using **Export** option on the **Quick Launch** tab page, your current system is considered as the source.

5. Add the required objects that you want to export.
6. Browse the target folder where you want to export the objects.
7. Choose **Finish**.

Selected objects along with the properties file are stored in a package at the specified location under a <System ID> package. Properties file gets created with the same name as the package in the format <PackageName.properties>.

## 11.4 Exporting Objects using SAP Support Mode

You use this procedure to export an object, along with other associated objects and data, and provide them to the SAP support. You should use this mode only when you report an issue to the SAP Support and you receive a request to export the specific objects along with the associated objects and data to reproduce the issue.

**Procedure**

1. In the **File** menu, choose **Export**.
2. Expand the **SAP HANA content** node.
3. Select **SAP Support Mode**.
4. Choose **Next**.
5. Select the system from where you want to export the models, and choose **Next**.

**Note:**

If you choose to export objects using **Export** option on the **Quick Launch** tab page, your current system is considered as the source.

6. Select an object that you want to export along with all the relevant details for SAP support purpose.
7. Choose **Finish**.



# Importing Objects

## 12.1 Importing Objects using Delivery Unit

You use this procedure to import objects, which are grouped into a delivery unit, in the form of .tgz file from the server or client location to a target system.

For example, customers can use this option to bring repository content from one system to another

**Procedure**

1. In the **File** menu, choose **Import**.
2. Expand the **SAP HANA content** node.
3. Choose **Delivery Unit**, and choose **Next**.
4. Select the target system where you want to import the models, and choose **Next**.

**Note:**

If you choose to import objects using **Import** option on the **Quick Launch** tab page, your current system is considered as the target.

5. Select the required file from the server or client.
6. Select the following options as required:

Option	Requirement
Overwrite inactive versions	To replace inactive objects in the target system with the inactive objects of the selected file.
Activate objects	To activate objects of the selected file in the target system.

7. Choose **Finish**.

## 12.2 Importing Objects using Developer Mode

You use this procedure to import objects from a client location to your SAP HANA modeling environment.

**Procedure**

1. In the **File** menu, choose **Import**.

2. Expand the **SAP HANA content** node.
3. Choose **Developer Mode**, and choose **Next**.
4. Select the target system where you want to import the objects, and choose **Next**.

**Note:**

If you choose to import objects using **Import** option on the **Quick Launch** tab page, your current system is considered as the target.

5. Browse the required source folder from where you want to import the objects.

**Note:**

Select the source <System ID> folder that contains the folder with the required objects. For example, to import objects of package P1 exported from system A1, select the folder A1 and not P1.

6. Select the required objects.
7. If you want to replace the existing objects in the target system with the selected objects, select **Overwrite Existing Objects** checkbox.
8. Choose **Finish**.

The selected objects get imported as following:

- If at the target, a package with the same name as the one to be imported exists, however, the package languages differ, objects get imported with a warning message.
- If at the target, a package with the same name as the one to be imported does not exist, then the respective package with its original language (fetched from the properties file) is created.
- In case of old objects for which properties file does not exist, a package with its language set to the logon language gets created.