Working With Global Hierarchies

Overview

The SyWay project will make use of Global Hierarchies to organise dimension values into logical groupings for reporting. The candidates for this approach have been captured in this related document: Reporting Hierarchies - Google Sheets

The purpose of this document is to provide additional guidance as to how to set up and maintain hierarchies. The key steps are:

1. Update the central config workbook and create a request Jira for each hierarchy (Responsibility: POD)
2. Create a hierarchy type (Responsibility: POD)
3. Create the hierarchy itself (Responsibility: POD)
4. Create CDS views that represent the hierarchy. These will be used in S/4 reporting and for extraction to Datasphere. (Responsibility: ABAP team based on Jira above)
5. Create extraction to Datasphere and include in analytic models (Responsibility: Reporting team based on Jira above)
6. Transport hierarchy to subsequent systems
    a. Hierarchy type - configuration transport (Responsibility: POD)
    b. CDS Views - workbench transport (Responsibility: ABAP Team)
    c. Hierarchy itself - download / upload (Responsibility: POD)
    d. Datasphere developments - (Responsibility: Reporting Team)

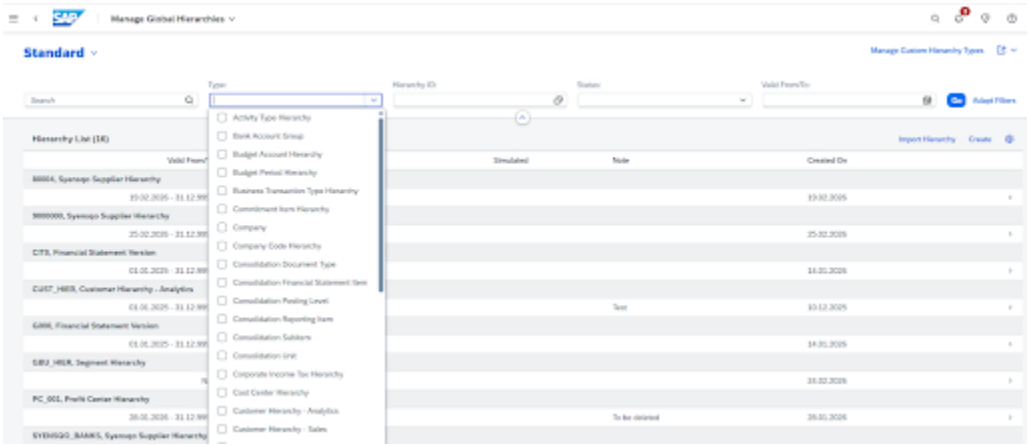
Please find more detailed information about these steps below.

The ‘Manage Global Hierarchies’ app

Hierarchies are created and maintained using the ‘Manage Global Hierarchies’ app, the standard SAP help for which can be found here.

Hierarchy Types

Before a hierarchy can be built a hierarchy type is needed. The hierarchy type defines the dimension (also known as characteristic and business entity) or combination of dimensions that can be included in the hierarchy. Multiple hierarchies can be built on each ‘hierarchy type’.



Hierarchies On Standard Hierarchy Types

There are a number of standard hierarchy types with standard hierarchies. These are automatically included in reports. This is achieved via a link between the hierarchy in the app and its inclusion in CDS dimension views e.g. I_PRODUCT

Custom hierarchies can be created on standard hierarchy types and they will also automatically be available for reporting.

Non-standard Hierarchy Types

Hierarchies may be required for dimensions, or combinations of dimensions, for which there is no existing hierarchy type. Custom hierarchy types must be defined to support these hierarchies. Eg vendor/supplier.

Non-standard hierarchy types will require custom development and therefore a request Jira will need to be created.

Simple (single dimension based) custom hierarchy types

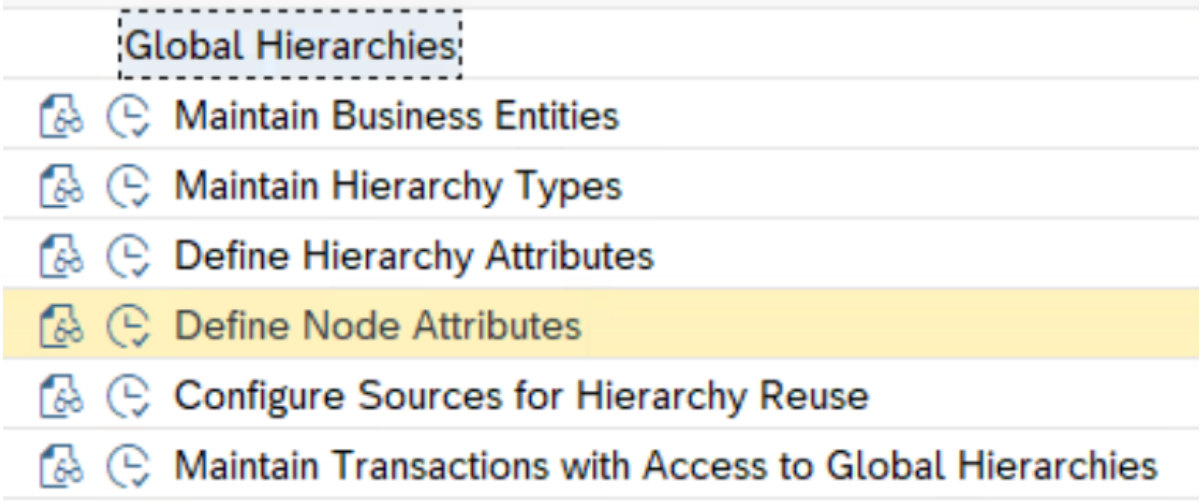
These can be created through the ‘Manage Global Hierarchies’ app. However, these non standard hierarchies will only be available in reports e.g. Custom Analytical Queries or Std CDS views after completing these steps:

- Configuration of Business Entities: Global Hierarchies
  - Configure Hierarchy Types
- Creation of the necessary cds views
  - Hierarchy specific views for the Hierarchy, Nodes and texts.
  - CDS view for a custom dimension e.g. zsupplier
  - Extending reporting cds views with the custom dimension.
  - These custom field extensions must be performed at the correct base level so that they are included in the datasphere extraction
- Creating the Global Hierarchy using the Maintain Global Hierarchies app.
  - Activated hierarchies will appear in tables hrrp_node and hrrp_directory

More complex needs

If a hierarchy needs to have levels that are other dimensions, this can be accommodated via configuration (transaction UHIMG).

E.g. a product hierarchy that is based on both material type and material group, or customer hierarchy based on sales org, distr channel, division



Additional reading

SAP Manage Global Hierarchies Help

2922493 - Extensibility with Hierarchies

ExtendGlobalHierarchiesWithCustomHierarchy.pdf

Note that there is a where clause not explained in the document to avoid reading the not assigned node.

Sample Code

Example of code to be used in CDS views for custom hierarchy types. N.B. the standard documentation does NOT include the last where clause which is essential for the code to function.

```
define view Z_SUPPLIER_HIERARCHY_NODE
  as select from hrrp_node
    inner join   Z_SUPPLIER_HIERARCHY as Header on  Header.SupplierHierarchy = hrrp_node.hryid
                                                and Header.ValidityEndDate   = hrrp_node.hryvalto
  association [0..*] to Z_SUPPLIER_HIERARCHY_NODE_T as _Text      on  $projection.SupplierHierarchy = _Text.SupplierHierarchy
                                                                  and $projection.HierarchyNode     = _Text.HierarchyNode
                                                                  and $projection.Supplier          = ''
  association [0..1] to Z_SUPPLIER                  as _Supplier  on  $projection.Supplier = _Supplier.Supplier
  association [1..1] to Z_SUPPLIER_HIERARCHY        as _Hierarchy on  $projection.SupplierHierarchy = _Hierarchy.SupplierHierarchy
                                                                  and $projection.ValidityEndDate   = _Hierarchy.ValidityEndDate
{
      @Consumption.filter: { mandatory: true, selectionType: #SINGLE, multipleSelections: false }
      @ObjectModel.foreignKey.association: '_Hierarchy'
  key hrrp_node.hryid                                as SupplierHierarchy,
      @ObjectModel.text.association: '_Text'
  key hrrp_node.hrynode                              as HierarchyNode,
      @Consumption.filter: { mandatory: true, selectionType: #SINGLE, multipleSelections: false }
      @Semantics.businessDate.to: true
  key cast( hrrp_node.hryvalto as fis_datbi )        as ValidityEndDate,
      @Semantics.businessDate.from: true
      cast( hrrp_node.hryvalfrom as fis_datab )      as ValidityStartDate,
      hrrp_node.parnode                              as ParentNode,
      hrrp_node.hryver                               as HierarchyVersion,
      concat( hrrp_node.hryseqnbr, hrrp_node.hrynode ) as SequenceNumber,
      hrrp_node.hrylevel                             as HierarchyNodeLevel,
      hrrp_node.nodetype                             as NodeType,
      @ObjectModel.foreignKey.association: '_Supplier'
      cast(
        case hrrp_node.nodetype
          when 'L' then hrrp_node.nodevalue
          else ''
        end as lifnr )                               as Supplier,
      _Text,
      _Supplier,
      _Hierarchy
}

where
  nodetype <> 'D';
```

```
define view Z_SUPPLIER
  as select from I_Supplier as isupplier


  association [0..*] to Z_SUPPLIER_HIERARCHY_NODE as _SupplierHierarchyNode
    on $projection.Supplier = _SupplierHierarchyNode.Supplier

{
  @ObjectModel.text.element: 'SupplierFullName'
  @ObjectModel.hierarchy.association: '_SupplierHierarchyNode'
  key isupplier.Supplier as Supplier,

  isupplier.SupplierAccountGroup,
  isupplier.SupplierFullName,


  _SupplierHierarchyNode
}
```