Parent-Child Hierarchy

Updates needed:

- Build automation in Tabular Editor for cross hierarchy calculation item which is needed for crossfiltering different unbalanced hierarchies. It should be sufficient to offer cross hierarchy calculation item for recursive hierarchies. The exisitng solution does only fully support one balanced hierarchy per dataset. More hierarchies are supported as well, except for crossfiltering between unbalanced hierarchies. Creating one Calculation Group per hierachy instead of one Calculation Item per hierarchy might be a suitable workaround.
- Add example for concatenating hierachies. The appended hierachy can be another unbalanced hierarchy or a balanced hierarchy. A table with appended hierarchies needs new surrogate keys representing all object types that are appendend, plus the change version if SCD history is applied.
    - Individual hierarchies are appended to each leaf (or any node), e.g. first hierachy is the work breakdown structure of a project, at each leaf it’s cost structure is appended.
    - Same hierarchy is appended to each leaf (or any node), like unfolding a matrix, e.g. first hierachy is the organization structure, then on each leaf (or any node) the cost element hierachy is appended, repeatedly.
- The Power BI DAX Measure part is not yet complete and automation thereof is also not yet complete. **Done:** Ready to use solutions for unbalanced standard hierarchies and unbalanced recursive hierachies are available. Balanced standard hierarchies do not need additional scriping, they are covered by the built-in behavior of Power BI.
- Writing an article about matrix-like hierachies as frequently used in financial reporting. Until then, refer to this very good description: Financial Modelling with Power BI - Bob Duffy - YouTube
- Automation for the periodic refresh to keep hierachies and calculation items in sync with new hierachy level columns, using XMLA endpoint.
- The provided stored procedures run on Azure SQL DB or self-hosted SQL Server. Synapse SQL distributed processing mode is not yet supported. Running the solution on **Synapse will require an adapted version** of the stored procedures that deal with the specifics of temporary tables in Synapse, as described here: 24. Temporary Tables in Synapse SQL in Azure Synapse Analytics - YouTube . When writing the Synapse version of the stored procedures, please also provide deployment scripts (“create” scripts).
- Building a ragged hierarchy in Power BI is not yet described.
- The fun part begins when the hierarchy shall have **SCD Type 2 history**. That means, e.g. a department can be somewhere in the structure in 2020 and can be somewhere else in the structure in 2021. This is solvable with valid from, valid to, is current and surrogate key columns. To be described and implemented when needed. Potentially, when any uptream node switches the parent then all its downstream nodes need a new version with recalculated generation columns (technically: these are all rows where the path starts with the path string of the changed node). Alternatively, at any date at which a restructuring in the hierarchy becomes effektive, the whole tree can get a new version and it’s recalculated from scratch. Needs analysis. Remark: The data model could contain an active relationship to the facts on the surrogate key to report in the original structure and an inactive (or in measure only) relationship on the business key (e.g. employee number) to report all (past) data in today's structure.

Motivation

Hierarchies are a common concept in data structures and BI. There are multiple recurring patterns of hierarchies and each needs to be treated accordingly.

In **Power BI**, any type of hierarchy needs to be transformed into a table structure with **one column per hierarchy level**. For each type of hierarchy you need specific transformations, so that each category item goes into its appropriate column. These transformations can be more or less elaborate to build and are error-prone if the actual type of hierachy is not properly understood.

First, we will have a look at hierachy data structures in general, then we will see what needs to be done in Power BI to implement each type of hierarchy, including existing Hubster.s templates and automation solutions.

The necessity to create one column per hierachy level is specific to the Power BI/Analysis Services tabular model. In the **SSAS multidimensional** cube it is possbile to build **parent-child hierachies natively**. Parent child hierarchy in SSAS Dimensions - YouTube

There is no Azure offering that supports multidimensional cubes. The only way to run them in Azure is a self-managed SQL Server SSAS Multidimensional installation on an Azure Virtual Machine.

Data Patterns

Hierarchy Types

Balanced Hierachy

In a balanced hierachy, each leaf node has the same number of levels to the root node or top level nodes. Be aware that the top level is not always a single root node, but there can be multiple nodes on top level. This is valid for all other types of hierarchy as well.

Balanced hierachies often come already in the required form of one column per level from the source system and can be loaded as is into Power BI. If they come as a parent-child hierachy they are harder to identify as a balanced hierachy - parent-child could also be unbalanced - and requires further data analysis or knowledge of the underlying business rules to identify the hierachy pattern.

Typical for a balanced hierachy is that the levels can be named. If this is the case, **apply names to the level columns, don't just count levels**.

Decide for yourself, which hierachies are better to understand:

- Level 1
- Level 2
- Level 3
- Level 4

Or

- Chief
- Director
- Head
- Employee

- Year
- Quarter
- Month
- Day

- Region
- Country
- State
- County
- Community

These are typical examples of balanced hierachies.

Unbalanced Hierachy

In an unbalanced hierachy, the number of levels between leaf and root node can be different for each leaf. That means, not all leafs are on the same - lowest - level. Often the levels cannot be named with functional names because it's just a recursive definition. A typical example is the work breakdown structure of a project.

This type of hierachy is usually stored as a prant-child data structure. In a parent-child hierarchy the levels are called "generation". Most implementation examples that you will find for parent-child hierachies in Power BI are dealing with unbalanced - ⚠️ not ragged - hierarchies (even though some called it "ragged").

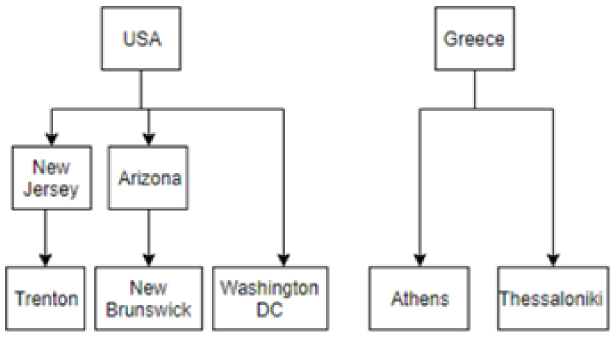
Ragged Hierachy

In BI literature, the term "ragged hierachy" is used **ambigiously**. Some sources call the unbalanced hierachy, as described above, a ragged hierachy, but these sources usually do then not describe at all what is called a "ragged hierachy" here - they do not even have a word for this. Some of the confusion comes from the different capabilities to support hierachies in different BI tools.

In a ragged hierachy, same as in an unbalanced hierachy, there can be a different number of nodes between a leaf and a root node.

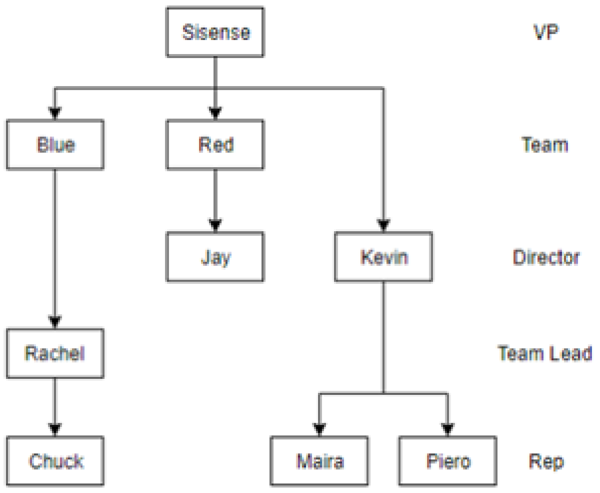
Other than in an unbelanced hierachy, each node and each leaf is assigned to a level. That means, while an unbalanced hierachy is "compact" in the way that each next node or leaf is also on the next level without gaps, in a ragged hierachy there can be gaps: empty or unfilled levels between two nodes or a leaf and a node.

Have a look at this example:



Ragged Hierarchy

A ragged hierachy can occur in a balanced or unbalanced flavor. It's called a balanced ragged hierachy if all leafs are on the same level. It's called an unbalanced ragged hierachy if leafs can occur on different levels.



Ragged and Unbalanced Hierarchy

In a ragged hierachy, levels are often named, so be careful when deciding which hierachy type you are actually dealing with: **balanced or ragged**.

- If the source system stores the data in a parant-child fashion then it’s more likely to be a ragged hierachy. (If the levels have names. Otherwise is more likely an unbalanced hierarchy.)
- If the data is stored in named columns per level and these columns have no blank values between root and leaf, then it’s usually a balanced hierachy.
- If the data is stored in named columns per level and there are blank values between root and leaf then it’s obvious that this is a ragged hierachy. You can turn this into a balanced hierachy by removing the levels that have blank values.

Ultimately, choosing the correct hierachy implementation requires an understanding of the business rules that are relevant for each hierachy. The data structure can give relevant hints but **be aware of false friends**.

The **loading of a ragged hierachy** needs to take care of both, the assigning of each node to the correct parent plus the assigning of the node to the correct level. If there is a gap, then for each gap, you usually create an artificial placeholder to fill the gap that is used as the parent. A ragged hierarchy requires a decision for the implementation of the data loading how to fill the gaps: A copy of the parent value, a placeholder, Unfilled + level name, … Since in ragged hierarchies levels usually have names, use the level names as column headers, not level 1, level 2, etc.

Matrix

Sometimes, users want to visualize data that has a matrix structure as a hierachy. This structure is characterized by children that can appear below multiple different parents, depending on the use case with the same or a split value per each parent in the measures. The structure of a financial report or a cost center-cost element-reporting are frequent examples of this scenario. Template solutions include just using multiple separate dimension tables or a separate hierachy table with a one to many relationship to a dimension table (e.g. financial report line - account).

Before applying one of these structures, doublecheck whether your scenario really does not match one of the plain vanilla hierachy structures described above. A normal hierachy is never a separate table attached to a dimension, it’s included into a dimension table itself.

The matrix scenario is out fo scope of this article and needs to be addressed in a separate article.

Facts Assignment

Standard hierarchy and recursive hierachy describes to which nodes data can be assigned. Each hierachy structure can exist as a standard or recursive hierarchy.

Standard Hierachy

In a standard hierachy, **facts exist only for leafs**, not for nodes. For example, sales always happen on a specific date, as stated on the document. There are no sales assigned to a month without a day.

The dimension table of a standard hierachy does not need **separate rows for nodes**, and as long as they are not needed for other purposes should be **avoided**. Take a calendar table as an example: There is one row for each day, but there is no row for each month, without a day.

Recursive Hierarchy

In a recursive hierachy, **facts can exist for leafs and nodes.** As an example, take a look at the salary costs along the organization structure. The costs of a department are not just the salaries of the employees. The head of the department adds to these costs, although she is a node, not a leaf.

Further Reading

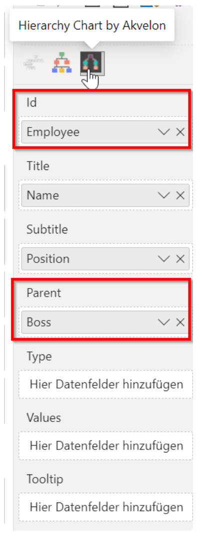
Handling Ragged and Unbalanced Data Hierarchies (sisense.com)

Implementation in Power BI

The implementation requires, depending on the hierachy type and visuals used, three parts:

1. The loading into a table with one column per hierachy level
2. Creating hierachy objects
3. DAX Measure code to hide unfilled levels in an unbalanced or recursive hierachy and to correctly treat standard or recursive facts assignment, preferably implemented as one calculation group.

The necessity to turn a parent-child-hierarchy into columns per level is the behavior of the buildt-in Power BI visuals. There are other visuals on the market that natively support parent-child-hierarchies. It's not about Power BI in general that parent child hierarchies are not supported, it's about the visuals.



Id: Employee
Title: Name
Subtitle: Position
Parent: Boss
Type: Hier Datenfelder hinzufügen
Values: Hier Datenfelder hinzufügen
Tooltip: Hier Datenfelder hinzufügen

Data Loading

Balanced Standard Hierachy

This is the most straight forward situation in Power BI. As long as the source system or data warehouse already provides columns per level, just load them. Make sure there are no rows with blank leafs. As long as it’s a balanced standard hierachy you don’t need them. You might need to provision an “unknown” category with the value “unknown” on all levels.

Balanced Recursive Hierachy

In contrast to the blanced standard hierachy, the balanced recursive hierachy needs seperate records for each node on any level. In these rows, the inferior levels are filled with blank.

In order to hide these blank rows on lower levels in visuals, some DAX code is required, as described below. Actually, a balanced recursive hierarchy needs all the same DAX code as an unbalanced recursive hierarchy.

Unbalanced Standard Hierachy

In an unbalanced standard hierachy you only need to load records for leafs and you can remove all rows that represent nodes, not leafs. Something like calculating a “Has Children” column and a filter on it could do the job.

A specific implementation might be added later.

 But since you need to write DAX code anyway - or better copy the **Hubster.s original template for unbalanced hierachies** (see below) - you can treat the hierachy records for nodes in the DAX code and just load them. Actually, the implementation of an unbalanced standard hierachy can be treated as an unbalanced recursive hierachy with just currently zero data on the nodes. So see below, and just implement an unbalanced recursive hierachy using the existing automation.

This is different with the balanced hierachy, because building a blanced standard hierachy is significantly less effort than properly building a balanced recursive hierachy and results in an easier to use dataset. So for balanced hierachies it makes sense to choose wisely. More effort is described below in the DAX section.

Unbalanced Recursive Hierachy

Unbalanced hierachies are usually stored in a parant-child pattern. That means, the hierachy table has a “Parent” column that contains a reference to the parent row.

As long as this is the case, you can use the stored procedures provided in this chapter to automate transformation of the parent-child structure into a one column per generation structure.

What The Automation Does

1. Load the full dimension table, plus the flattened hierachy columns, plus a path column as used by Power BI PATH functions, plus the path length (hierachy depth) as a number as required for the DAX measure dealing with the unblanaced hierachy, plus count of direct children.
2. Automatically load as many hierachy columns as needed, auto-adjusted with each refresh. A minimum number of levels can be enforced to keep the table consistent with Power BI visuals and DAX code.
3. Any columns from the source table can be concatenated into a custom format to create the labels that shall be used in the visualization of the hierachy by providing a SQL expression that defines the string pattern. Alternatively, a view can be used as input table, allowing for even more customization of the resulting dimension with hierachy table or prefiltering of the hierachy, e.g. for certain object types only.

Setup

spCountHierarc…ons.sql
26 Okt. 2022, 08:12 PM

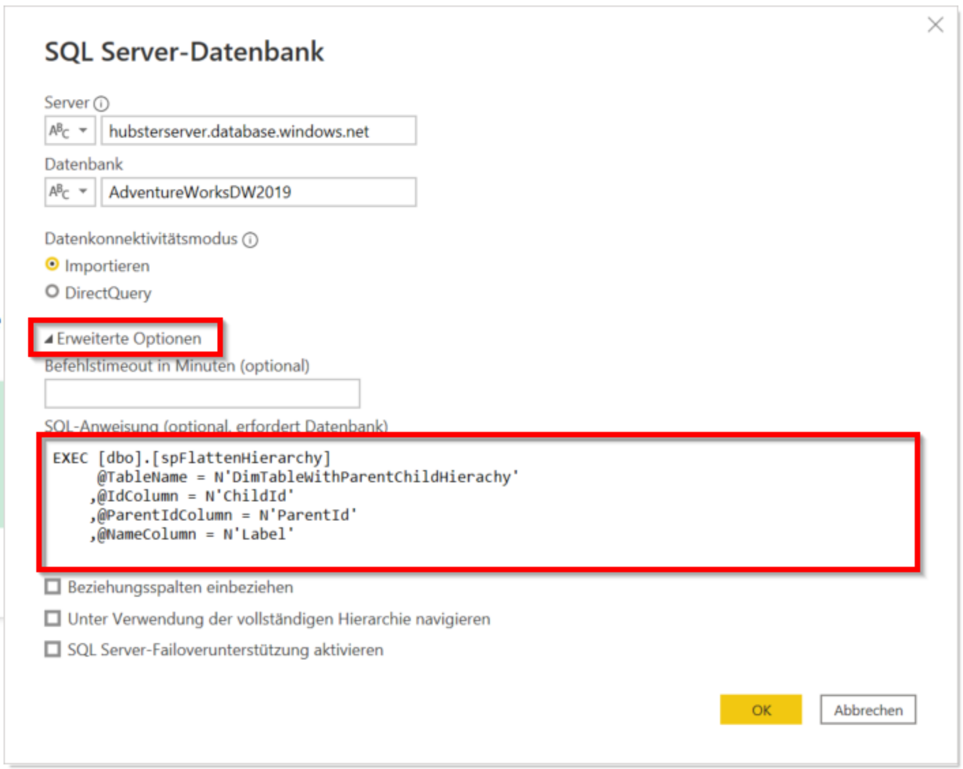
spFlattenHierarchy.sql
26 Okt. 2022, 08:12 PM

1. Download the files from above
2. Open the files in SQL Server Management Studio
3. Connect to the database containing the source table with the parent child hierachy
4. Per open file, select this database for the execution of the query.
5. Execute the two files in this order: first spCountHierachyGenerations, then spFlattenHierachy.

Now you can retrieve the tables that contain parent-child-hierachies including the columns per generation.

Retrieve The Table in Power BI

In Power BI, you create a new SQL Server query using import mode and advanced options. You need to provide the parameters to configure the hierachy in the SQL code box in the connector window:


SQL Server-Datenbank

Server: hubsterserver.database.windows.net

Datenbank: AdventureWorksDW2019

Datenkonnektivitätsmodus
[✓ Importieren] DirectQuery

Erweiterte Optionen

Befehlstimeout in Minuten (optional)

SQL-Anweisung (optional, erfordert Datenbank)

```
EXEC [dbo].[spFlattenHierarchy]
     @TableName = N'DimTableWithParentChildHierachy'
    ,@IdColumn = N'ChildId'
    ,@ParentIdColumn = N'ParentId'
    ,@NameColumn = N'Label'
```

☐ Beziehungsspalten einbeziehen
☐ Unter Verwendung der vollständigen Hierarchie navigieren
☐ SQL Server-Failoverunterstützung aktivieren

From there, press OK, then Load, and you are good.

Parameters

Parameter	Required	Datatype	Default	Description
@TableName	mandatory	nvarchar(MAX)		Can be the name of a table or view, with or without schema, with or without square brackets. . Input table may not have a column named [Path]. Use a view to rename a [Path] column in the input table.
@IdColumn	mandatory	nvarchar(MAX)		Name of the column containing the (child) ID.
@ParentIdColumn	mandatory	nvarchar(MAX)		Name of the column containing the reference to the parent ID.

@NameColumn	mandatory	nvarchar(MAX)		Column that contains the texts that shall be used in the per-generation column. Can be a column name, a SQL string expression or even the same as @IdColumn.
@MinGenerations	optional	int	1	Minimum number of generation columns that shall be created.
@PathDelimiter	optional	nchar	|	Separator for the concatenated path. Choose a character that does not exist in the ID and Name columns. DAX PATH functions require this to be |.
@GenerationLabel	optional	nvarchar(MAX)	Generation	Text used to name the generation columns.
@StartGenerationNumber	optional	int	0	The number used for the topmost generation.
@Assignment	optional	nvarchar(10)	standard	Valid values: 'standard', 'recursive' Defines where data from the fact tables is assigned. standard: leafs only recursive: leafs and nodes

Examples

Call with mandatory parameters only:

```sql
EXEC [dbo].[spFlattenHierarchy]
     @TableName = N'DimTableWithParentChildHierachy'
    ,@IdColumn = N'ChildId'
    ,@ParentIdColumn = N'ParentId'
    ,@NameColumn = N'Label'
```

Call with all optional parameters:

```sql
EXEC [dbo].[spFlattenHierarchy]
     @TableName = N'[gold].[vCustomViewWithParentChildHierachy]'
    ,@IdColumn = N'ChildId'
    ,@ParentIdColumn = N'ParentId'
    ,@NameColumn = N'[No.] + '' '' + Label'
    ,@MinGenerations = 10
    ,@PathDelimiter = '^'
    ,@GenerationLabel = N'Department Level'
    ,@StartGenerationNumber = 1
    ,@Assignment = N'recursive'
```

Ideas for debugging

- Special characters in the ID or name columns might confuse the SQL code. If you observe problems, try these things first to identify the root cause:
  - Provide the ID column name also to the @NameColumn parameter.
  - Choose a different path delimiter character.

⚠️ This solution does not support ragged hierachies. Don't use it as is for ragged hierachies, just because your source table has this nice, tempting "parent" column.

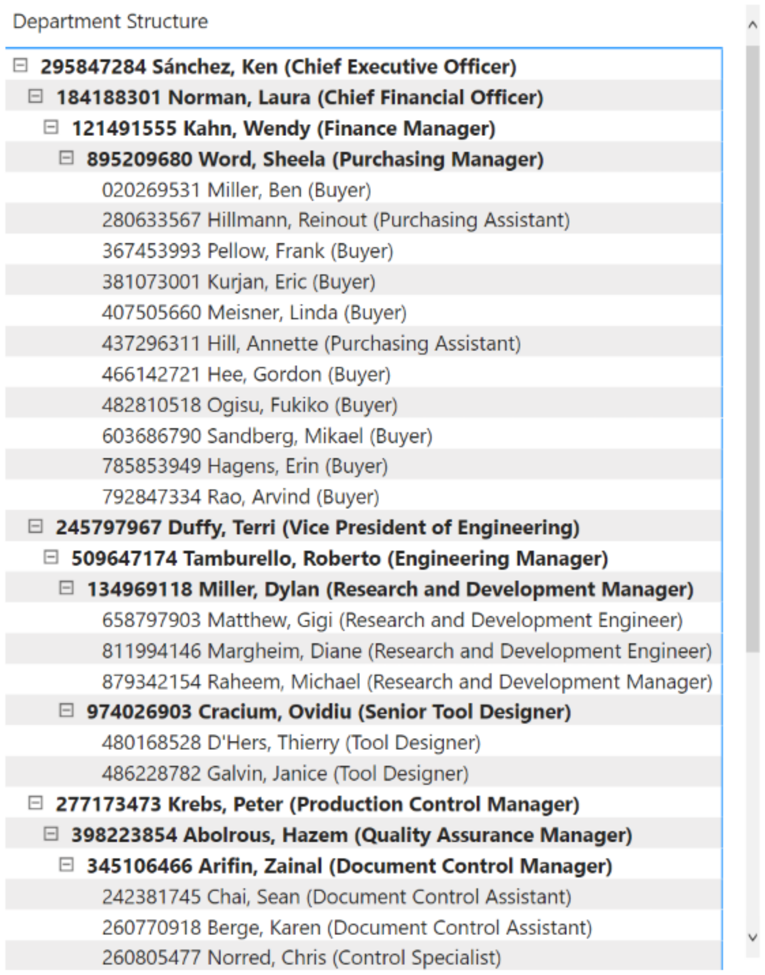
⚠️ This solution does not support Synapse SQL Serverless Pool. Code needs adaption to CTAS for handling of temporary tables.

Sample File

This file contains a ready-to-use live example on the hubsterserver for you to try (if it doesn't work, add your current IP number to the firewall whitelist of the SQL Server hubsterserver and ask @Martin Bubenheimer for database privileges.)

Parent-Child Hi... le.pbix
22 Okt. 2022, 07:34 PM

```
EXEC [dbo].[spFlattenHierarchy]
    @TableName = N'DimEmployee',
    @IdColumn = N'EmployeeKey',
    @ParentIdColumn = N'ParentEmployeeKey',
    @NameColumn = N'
        CONCAT(
            RIGHT(''000000000'' + EmployeeNationalIDAlternateKey, 9), '' '',
            LastName, '', '',
            FirstName, '' ('' + Title + '')''
        )',
    @StartGenerationNumber = 1,
    @GenerationLabel = 'Department Level'
```



Department Structure

⊟ **295847284 Sánchez, Ken (Chief Executive Officer)**
- ⊟ **184188301 Norman, Laura (Chief Financial Officer)**
  - ⊟ **121491555 Kahn, Wendy (Finance Manager)**
    - ⊟ **895209680 Word, Sheela (Purchasing Manager)**
      - 020269531 Miller, Ben (Buyer)
      - 280633567 Hillmann, Reinout (Purchasing Assistant)
      - 367453993 Pellow, Frank (Buyer)
      - 381073001 Kurjan, Eric (Buyer)
      - 407505660 Meisner, Linda (Buyer)
      - 437296311 Hill, Annette (Purchasing Assistant)
      - 466142721 Hee, Gordon (Buyer)
      - 482810518 Ogisu, Fukiko (Buyer)
      - 603686790 Sandberg, Mikael (Buyer)
      - 785853949 Hagens, Erin (Buyer)
      - 792847334 Rao, Arvind (Buyer)
- ⊟ **245797967 Duffy, Terri (Vice President of Engineering)**
  - ⊟ **509647174 Tamburello, Roberto (Engineering Manager)**
    - ⊟ **134969118 Miller, Dylan (Research and Development Manager)**
      - 658797903 Matthew, Gigi (Research and Development Engineer)
      - 811994146 Margheim, Diane (Research and Development Engineer)
      - 879342154 Raheem, Michael (Research and Development Manager)
    - ⊟ **974026903 Cracium, Ovidiu (Senior Tool Designer)**
      - 480168528 D'Hers, Thierry (Tool Designer)
      - 486228782 Galvin, Janice (Tool Designer)
- ⊟ **277173473 Krebs, Peter (Production Control Manager)**
  - ⊟ **398223854 Abolrous, Hazem (Quality Assurance Manager)**
    - ⊟ **345106466 Arifin, Zainal (Document Control Manager)**
      - 242381745 Chai, Sean (Document Control Assistant)
      - 260770918 Berge, Karen (Document Control Assistant)
      - 260805477 Norred, Chris (Control Specialist)

Next step: Tabular Editor scripts to create the necessary DAX code can be found in the Tabular Editor tool page. Tabular Editor | Calculation Item for Unbalanced Standard Hierachies

More hierarchy examples from Financial Reporting:

Hierarchies.pbix
28 Okt. 2022, 07:15 AM

To be continued.