Tool for Undertakings
DPM Database

Technical documentation

Table of Contents

1 Introduction

This document describes the database used by the Tool for Undertakings. The database is the core component of the solution. It contains metadata and data used or created by the T4U applications to fulfil the one of the main functional requirement for the tools which is data capture and presentation in form of tabular views (resembling layout and alignment of information requirements defined in the legal acts or regulations) as well as production of valid XBRL reports. In addition, the database is prepared to satisfy the secondary priority requirement, which is aiming to support translations and extension of metadata definitions and stored data.

2 Database Management System

The T4U database is available in two technologies:

- SQLite,
- Microsoft SQL Server.

SQLite is deployed within the T4U application on the side (machine) of the Undertaking. The reason for selection of this concrete database technology is mainly for the ease of deployment (no installation or port configuration is needed), the multiplatform support and the open source licence. The drawbacks are lack of certain functionalities typical for DBMSs like stored procedures, limited support for simultaneous multiuser work and potential performance issues for larger amount of data.

To overcome these shortcomings the T4U database is also implemented in Microsoft SQL Server technology which is a typical DBMS with all standard functionalities. This database may be used to perform the various maintenance tasks.

The structure of both SQLite and MS SQL Server in the core part is identical. The differences may occur for functionalities not supported by the SQLite but included in MS SQL Server.

3 Database components

The T4U database consist of four major components. Each of them fulfils different roles and satisfies various requirements. These components are:

1. **information requirements and validations metadata** – that resembles the DPM dictionary, annotated templates and validation rules metadata; it is the description of the model (similar to an XBRL taxonomy),
2. **placeholder for data storage** – where stored facts refer to DPM properties; it is the understood as the actual reported data (similar to an XBRL instance) and structured in a dimensional approach,
3. entities whose structure resembles information requirements and is based on tabular views – these are **placeholders for data storage in "classic" relational manner**; this component comes also with information on **mapping between DPM metadata/data description and classic relational structures**; this tabular oriented view of the information is composed by relational tables in a similar way as flattened information from the XBRL Table linkbase and therefore similar to the reference business templates view,
4. **validations definitions** – define validation rules to be performed on the data stored in "classic" relational manner,
5. **T4U applications information**, this is the specific set of information needed by the tool for undertaking applications (mainly user interfaces), for localisation purposes (translation of menu options, buttons, messages, etc. to national languages), etc.

6. **supporting views** used by various components of the application (XBRL parser, Windows tool, etc).

Components described above are schematically presented on Figure 1 below.

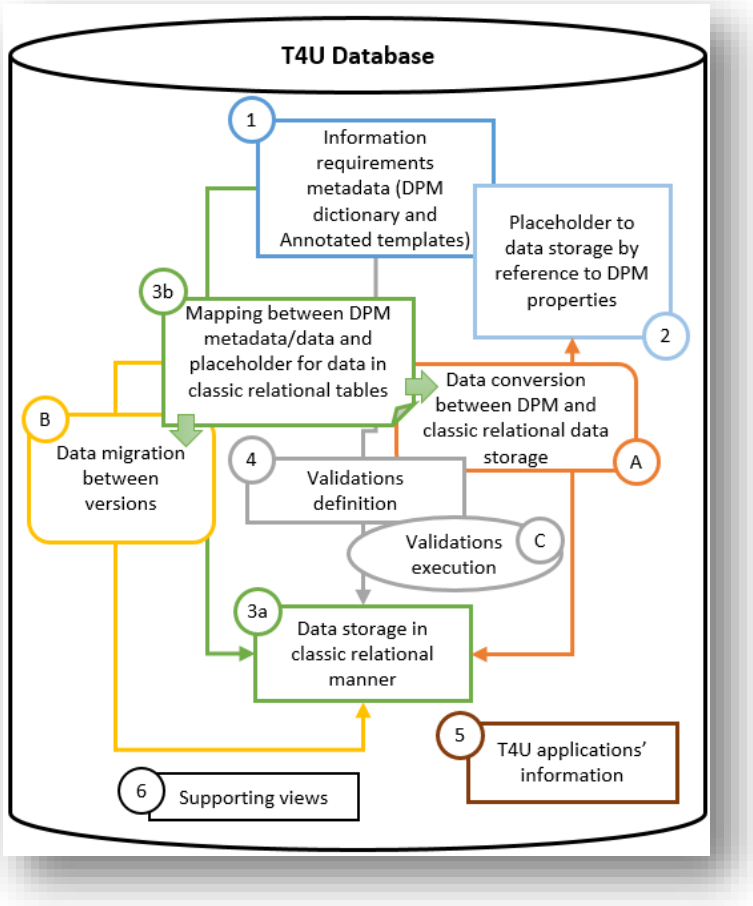


Figure 1. Components of the T4U database.

In addition to the components described above, the following processes (in form of internal ETL) occur in the T4U database:

A. **data conversion between DPM and classic relational data storage**,
B. **data migration between versions**,
C. **validations execution**.

Each component and process inside of the T4U database corresponds to particular functionality of the solution.

Information requirements metadata (component 1 on Figure 1) and validations definition (component 4 Figure 1) on is populated in the design/setup stage from the input materials (DPM dictionary and annotated templates of Solvency II or other scope in the same format and structure). Therefore it reflects all characteristics defined by these sources. Following the normalized DPM model, the structure (entities, their properties and relationships) of this component is relatively stable and able to accommodate any expected changes/modifications of information requirements in future versions. It is also flexible enough to enable storage of any other information requirements metadata. As described in more details later in this document, this component is used by the solution in various

stages and processes. One of the major tasks is to support navigation over the information requirements and present them in the tabular format as in the source materials.

Definition of information requirements in this component is both, data and from centric. On one hand it defines data cells by identifying their dimensional properties, on the other these cells are gathered in tables whose columns and rows represented in a very normalized manner (as ordinates on axis).

This data centric description enables reference to metadata from the placeholder for data storage according to the DPM properties (component 2 on Figure 1) which facilitates interaction with XBRL instance document files (where exchanged data is described in a similar style). Therefore this component interacts with XBRL parser in the process of loading of XBRL instance document data to the database as well as generation of XBRL instance documents from the data in the database in a fast and easy manner.

There are however two major problems with data centric approach. One is complexity of highly normalized model - the way in which tabular views are resembled is not very intuitive and easy to understand or query by users not familiar with the data point model and data point modelling methodology. The other issue is potential performance problems when accessing facts. All facts are stored in a single entity and are distinguished based on dimensional properties. This hinders prompt rendering of selected facts in tabular views and execution of validation rules (matching facts according to dimensional properties).

As a result, the T4U database contains also placeholders for data storage in “classic” relational manner (component 3a on Figure 1), i.e. the facts are stored according to their presentation in business templates by reference to row/column position. In consequence, “open” templates (i.e. these with unlimited/unknown number of rows) in the database look identical to their representation in the information requirements i.e. each column in a business template has its counterpart in the database entity for this template. Each “closed” templates (i.e. a template with known/defined columns and rows) results in a database entity whose columns represent cells from the template. Multiplications of a template (resulting for example from numerous z-axes properties or repeatable rows or columns) become separate columns in the database tables that are also keys for each row in these tables. Simplicity of design of this database component enables the T4U user interfaces (Windows Forms and Excel templates, iOS) or other connectors (like ODBC) to easily and quickly populate or access data (for their display or validation).

As explained in more details in later in this section, the placeholder for data storage in classic relational manner (component 3a) is generated automatically from DPM and annotated templates metadata (component 1). The link between the two is the mapping table (component 3b on Figure 1) which identifies the DPM properties hidden behind each row/column/page for every classic relational table. This mapping table and mapping data that it contains is used for multiple purposes.

One (marked as process A on Figure 1) is to convert the data stored in the classic relational manner to the DPM properties data storage placeholder (and further to XBRL) and vice versa (when XBRL instance document is loaded in the DPM properties data storage placeholder, its data is subsequently converted to the classic relational structures so that it can be accessed by T4U interfaces or validated).

Another process using the mapping table information relates to migration of data stored in classic relational manner between the versions of the database (marked as process B on Figure 1). One of the drawbacks of classic relational placeholders is their instability. Business templates tend to change in terms of their graphical tabular representation. For example rows and columns order may be rearranged, tables are split or merged without real change in their content, etc. Each such change

results in the new set of classic relational tables. As a consequence, data from previous periods need to be migrated to the new structures. Mapping information enables this process by providing a link to the stable component which is the DPM properties hidden behind every row, column and page of each template (and its counterpart database entity). Based on that information the data can be migrated to the new representation of classic relational structures maintaining the consistency of definitions in relation to previous versions (and thus allowing for example data comparison across time). Currently this process is run by means of generating XBRL instance documents and importing them back to a new container (with minor changes such as modules codes, etc.).

And last but not least, the mapping information may support the process of definition of executable data validations (marked as process C on Figure 1). Business rules defined in the source materials (provided by business users) are loaded to validations definition database metadata component (4 on Figure 1). At this stage they are stored in the normalized manner. Execution of the validations is performed on data stored in classic relational structures (component 3a). In this process the mapping table information may be harnessed to properly identify the involved facts based on the place of their occurrence in templates as well as representation in terms of the DPM properties.

Another separate component of the T4U database (marked with number 5 on Figure 1) is applications' information. In general it contains translation of the application's interfaces (menu, buttons, messages, etc.) and information about the version of the tool or supported containers.

A number of views is defined (6 on Figure 1) used by various components of the tool, e.g. XBRL parser, user interfaces, etc.

4 Model of DPM metadata storage

Entities and relationships of this component of the database resemble the artefacts of the Data Point Modelling methodology. Therefore it is recommended that readers of this section familiarize themselves with documents listed at http://www.eurofiling.info/dpm/index.shtml and in particular the following documentation: http://www.eba.europa.eu/documents/10180/632822/Description+of+DPM+formal+model.pdf explaining the DPM artefacts on UML diagrams.

Moreover, this part of the database closely follows the structure (entities and relationships) of the EBA DPM MS Access database. The latest version of this database at the moment of the T4U database designing and writing of this document is available under the following location: http://www.eba.europa.eu/documents/10180/1067579/DPM+Database+2.3.1.0.zip/4bbbb28d-3e3d-4262-b227-9daab3e008ec. It is recommended that the readers of this section read also *CRD4 DPM - Database description - v2.1.pdf* embedded in the compressed folder http://www.eba.europa.eu/documents/10180/781471/DPM+Database+2.2.zip/2dbcf8e5-5990-41e3-a068-5ddcb081ad08.

The main modifications and dissimilarities of the T4U DPM metadata component comparing to the EBA model include:

- different patterns used for reflection of data point keys (in T4U database they are more XBRL oriented, with information on owners in form of canonical namespace prefixes),
- many-to-many relation between Table and Axis entities (allowing axes to be reused by tables which is a common case in some of the Solvency 2 templates),
- denormalization of the model by deletion of relationships and inclusion that information as enumerations in table columns (e.g. data type, period type, balance attribute),

- lack of listing and versioning of data points which representation is limited to correspondence with cells rather than enumerating all possible combinations (especially in case of semi open axes constrained by hierarchies where the number of data points in some temples may amount to several millions),
- EBA table groups, templates, tables and table versions are represented by T4U template groups, templates, template variants, business tables and annotated tables (versioned together with taxonomies) and tables (reused by taxonomies, linked with axes, cells, etc.),
- validation rules are limited to reference to row/column/sheet codes rather than DPM artefacts (axes, ordinates, cells, metrics, etc.).

All entities of the T4U database are described and explained in the next sections of this document.

4.1 Source (input) information and database DPM metadata population mechanisms

DPM metadata in the database is populated from DPM dictionary and annotated templates Excel files of EIOPA Solvency II or compatible (i.e. identical in structure).

Population of the database is performed using an automated process that is out of scope of the T4U. As this part of the process is currently used by EIOPA for creation of the Solvency II XBRL taxonomy, it is assumed that a database would be distributed by EIOPA.

4.2 Representation of DPM model artefacts and relationships

4.2.1 Entities

Names of entities (tables) of this component of the database start with letters “m” for information requirements metadata and “v” for validation rules metadata followed by the short description of the entities’ content.

The next sections introduces the entities defined for this component of the database by providing a general explanation of the entity’s content and a table identifying entity’s attributes (columns), their data type (in general, e.g. INTEGER, TEXT, DATE, …) and short description.

4.2.1.1 *mOwner*

This entity is used to identify the institution that "owns" (i.e. defines and manages) a concept (see mConcept table) representing DPM artefact such as domain, member, hierarchy, dimension, table, axis, ordinate, etc.

Attribute	Type	Description
OwnerID	INTEGER	Artificial ID.
OwnerName	TEXT	Institution (owner) name. E.g. European Insurance and Occupational Pensions Authority, …
OwnerNamespace	TEXT	Recommended namespace of an owner (the "core" part of the namespace, to be extended by suffixes representing different concepts).
OwnerLocation	TEXT	URI representing the root folder of the official location of taxonomy files.
OwnerPrefix	TEXT	Recommended prefix of an owner (used to construct XBRL codes of different concepts). E.g. "s2c", …
OwnerCopyright	TEXT	Copyright text. Used in comments in XBRL taxonomy files.
ParentOwnerID	INTEGER	Points to OwnerID of an institution in case of extensions of the model.
ConceptID	INTEGER	Owner is also a concept (its name can be translated, it can be versioned, etc.).

4.2.1.2 *mConcept*

This table is used to provide more information on various artefacts (identification of the owner, translations to national languages, references to legal acts and regulations, managing changes in the definitions by setting various currency dates, etc.).

Attribute	Type	Description
ConceptID	INTEGER	Artificial ID.
ConceptType	TEXT	Type of a concept: Axis, AxisOrdinate, Dimension, Domain, Hierarchy, Member, Module, ReportingFramework, Table, ConceptualTemplateOrTable, Taxonomy, Language, Owner…
OwnerID[1]	INTEGER	Points to mOwner.OwnerID.
CreationDate	DATE	Date when concept was first created.
ModificationDate	DATE	Date when concept was last modified.
FromDate	DATE	Date when concept starts to be used in expression of information requirements.
ToDate	DATE	Date when concept ends to be used in expression of information requirements.

4.2.1.3 *mDomain*

This table lists domains. Domains group values of a particular kind. Domain may have an explicit list of allowable values (members) or specify values of a particular type or pattern (a "typed" domain). Domains provides the allowable values for dimension the reference them.

Attribute	Type	Description
DomainID	INTEGER	Artificial ID.
DomainCode	TEXT	Short code (usually two capital letters).
DomainLabel	TEXT	Descriptive label (in English).
DomainDescription	TEXT	Longer description (in English).
DomainXBRLCode	TEXT	Code (QName) used in XBRL documents, consisting of canonical namespace prefix followed by a domain code.
DataType	INTEGER	Indicates the allowed type of values (for Typed domains). One of the following "boolean"/"date"/"integer"/"decimal"/"monetary"/ "percentage"/"code"/"string".
IsTypedDomain	BOOLEAN	"Typed" domains allow any value of a particular type (i.e. string, number, date etc.), "explicit" dimensions only allow a choice from a given list of members.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.4 *mDimension*

Category/aspect used to describe and differentiate data points, each relates to one specific feature. Allowed values are taken from a referenced domain.

Attribute	Type	Description
DimensionID	INTEGER	Artificial ID.
DimensionLabel	TEXT	Descriptive label (in English).
DimensionCode	TEXT	Short code (usually two or three capital letters).
DimensionDescription	TEXT	Longer description (in English).
DimensionXBRLCode	TEXT	Code (QName) used in XBRL documents consisting of canonical namespace prefix followed by a dimension code.

[1] In SQLite applies not existing OwnerID „0” for domain defining members representing metrics.

DomainID	INTEGER	Points to mDomain.DomainID. Domain from which the allowable values for this dimension are taken.
IsTypedDimension	BOOLEAN	"Typed" dimensions allow any value of a particular form (i.e. any string of certain length or pattern, any number, a date etc.), "explicit" dimensions only allow a choice from a given list of members.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.5 *mMember*

An explicit possible value within a domain.

Attribute	Type	Description
MemberID	INTEGER	Artificial ID.
DomainID	INTEGER	Points to mDomain.DomainID. Domain to which this member belongs.
MemberCode	TEXT	Short code (resembling XBRL local name).
MemberLabel	TEXT	Descriptive label (in English).
MemberXBRLCode	TEXT	Code (QName) used in XBRL documents consisting of canonical namespace prefix followed by a member code.
IsDefaultMember	BOOLEAN	Identifies if the member is a default value (1) for a domain it points to (and as a result for all dimensions that refer this domain).
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.6 *mHierarchy*

Hierarchies specify how members relate to each other, and can also define the aggregations from lower to upper levels in the hierarchy.

Attribute	Type	Description
HierarchyID	INTEGER	Artificial ID.
HierarchyCode	TEXT	Short code (often used also as @id on role definitions in XBRL). Usually contains referenced domain code followed by an underscore and sequential number.
HierarchyLabel	TEXT	Descriptive label (in English).
DomainID	INTEGER	Points to mDomain.DomainID. Domain this hierarchy relates to.
HierarchyDescription	TEXT	Description (in English) or application to dimensions or templates (for documentation purposes only).
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.7 mHierarchyNode

Represents a node in a hierarchy of members, specifying how members relate to each other, and can also define the aggregations from lower to upper levels in the hierarchy

Attribute	Type	Description
HierarchyID	INTEGER	Points to mHierarchy.HierarchyID. Hierarchy to which this node belongs.
MemberID	INTEGER	Points to mMember.MemberID. Member this node represents.
IsAbstract	BOOLEAN	Identifies if a member is in the hierarchy merely for the reason of grouping.
ComparisonOperator	TEXT	Indicates the comparison relationship between this node and the aggregation of its children.
UnaryOperator	TEXT	Indicates the contribution of this node to the aggregation of its siblings.

Order	INTEGER	Position of this node within its set of siblings.
Level	INTEGER	Level of this node, lower level numbered nodes contain higher numbered ones, i.e. lower levels are nearer the root (having level 1)
ParentMemberID	INTEGER	Indicates the parent of this node, if any - i.e. the level immediately above. Null for root nodes.
Path	TEXT	Path from the root node to this node, using MemberID split by comma.

4.2.1.8 mMetric

The fundamental conceptual meaning of a piece of information.

Attribute	Type	Description
MetricID	INTEGER	Artificial ID. Preferably it should match mMember.MemberID from which descriptive labels may be obtained (metric is a subtype of member).
CorrespondingMemberID	INTEGER	Points to mMember.MemberID (in case mMetric.MetricID does not match corresponding mMember.MemberID).
DataType	INTEGER	Type of data. One of the following (in brackets corresponding XBRL data types): “Date” (xbrli:dateItemType), “Percentage” (xbrli:pureItemType), “Integer” (xbrli:integerItemType), “Monetary” (xbrli:monetaryItemType), “Decimal” (xbrli:decimalItemType), “String” (xbrli:stringItemType), “Boolean” (xbrli:booleanItemType), “Enuneration/Code” (enum:enumerationItemType), “true” (xbrli:booleanItemType with restriction to true), “URI” (xbrli:anyURIItemType)
FlowType	TEXT	The time dynamics of the information, is it a value at a specific point in time (a "stock" or "level"), or measured over a time period (a "flow" or "change"). N.B. not necessarily the XBRL "period type" where all metrics are assumed to be mapped to instant periods (the reference date).
BalanceType	TEXT	"Credit"/"Debit"/Null.
ReferencedDomainID	INTEGER	Points to mDomain.DomainID. Domain of the allowed values for this Metric (for enumerated/code-typed Metrics).
ReferencedHierarchyID	INTEGER	Points to mHierarchy.HierarchyID/mHierarchyNode.HierarchyID. Indicates that the allowed values for this metric are restricted to those present in the referenced hierarchy.
HierarchyStartingMemberID	INTEGER	Points to mHierarchyNode.MemberID. Identifies starting member in the hierarchy (ReferenceHierarchyID) whose descendants (-or-self depending on IsStartingMemberIncluded) form valid values for a metric (taking into account mHierarchyNode.IsAbstract).
IsStartingMemberIncluded	BOOLEAN	Informs if the starting member identified by HierarchyStartingMemberID is also a valid value for a metric (or is it only its descendants).

4.2.1.9 mReportingFramework

Overall reporting framework. High level, stable concept. E.g. Solvency 2, …

Attribute	Type	Description
FrameworkID	INTEGER	Artificial ID.
FrameworkCode	TEXT	Short code of a framework (e.g. s2md, …)
FrameworkLabel	TEXT	Descriptive label (in English). E.g. solvency, …
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.10 mTaxonomy

A specific description of the classification of the tables and data points of a reporting framework, at a particular point/period in time.

Attribute	Type	Description
TaxonomyID	INTEGER	Artificial ID.
FrameworkID	INTEGER	Points to mReportingFramework.FrameworkID. Reporting framework this taxonomy describes.
TaxonomyCode	TEXT	Short code of a taxonomy, e.g. sol2, …
TaxonomyLabel	TEXT	Descriptive label (English), e.g. solvency2, …
Version	TEXT	E.g. 1.5.2.c, ….
PublicationDate	DATE	Taxonomy publication date (e.g. 2015-02-28). To be used in namespaces of taxonomy files.
TechnicalStandard	TEXT	Identifier of the prescriptive technical standard which this taxonomy describes/models.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.
FromDate	DATE	Date from which this taxonomy is/was valid.
ToDate	DATE	Date until which this taxonomy is/was valid.

4.2.1.11 mTemplateOrTable

Identifies templates and tables (as defined in the Business and Annotated Templates).

Attribute	Type	Description
TemplateOrTableID	INTEGER	Artificial ID.
TaxonomyID	INTEGER	Points to mTaxonomy.TaxonomyID.
TemplateOrTableCode	TEXT	Short code, e.g. S.01.01.01.01
TemplateOrTableLabel	TEXT	Description (in English). Usually template/table title.
TemplateOrTableType	TEXT	One of the following: "TemplatesGroup", "Template", "TemplateVariant", "BusinessTable", "AnnotatedTable".
Order	INTEGER	Order (preferably global but not necessary) of a Template or Table for displaying templates and tables in tree structure.
Level	INTEGER	Level of a Template or Table for displaying templates and tables in tree structure, usually: 1 for "TemplatesGroup", 2 for "Template", 3 for "TemplateVariant", 4 for "BusinessTable", 5 for "AnnotatedTable".
ParentTemplateOrTableID	INTEGER	Parent template or table.
ConceptID	INTEGER	Points to mConcept.ConceptID.
TC	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: caption of the table.
TT	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: business labels on the top of the table (headers of columns)
TL	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: business labels on the left side of the table (headers of rows)
TD	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: rectangular area enclosing the data cells of the table.

YC	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: codes of rows.
XC	TEXT	Range of cells as in underlying Annotated Templates used to identify where are the components of each table and how tables relate to one another in graphical layout on one sheet: codes of columns.

4.2.1.12 mTable

The specific description of a particular table from a reporting framework, within a taxonomy, valid during a particular time period. Several "Tables" may represent the evolution of a particular "Business-"/"Annotated Table" over time.

Attribute	Type	Description
TableID	INTEGER	Artificial ID.
TableCode	TEXT	Short code of a table.
TableLabel	TEXT	Descriptive label (in English).
FromDate	DATE	Date from which this version of this table is/was valid.
ToDate	DATE	Date until which this version of this table is/was valid.
XbrlFilingIndicatorCode	TEXT	Code of the filing indicator used to indicate the reporting of this table (N.B. may be shared with other tables which form part of the same template, all of those tables will be considered filed or not filed as a single unit).
XbrlTableCode	TEXT	Table code used in XBRL documents.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.
YDimVal	TEXT	For open and semi-open tables – dimension codes (with wildcards * or hierarchy reference) used on open or semi open Y axes. In alphabetical order based on dimension code.
ZDimVal	TEXT	Metrics and dimension members (or wildcards * for open axis, hierarchy reference for open axis) used on Z axes. In alphabetical order based on dimension code.

4.2.1.13 mTaxonomyTable

Attribute	Type	Description
TaxonomyID	INTEGER	Points to mTaxonomy.TaxonomyID.
TableID	INTEGER	Points to mTable.TableID.
AnnotatedTableID	INTEGER	Points to mTemplateOrTable.TemplateOrTableID for TemplateOrTableType = "AnnotatedTable".
IsSimplyResuse	BOOLEAN	Indicates that a table from a previously released taxonomy is being directly reused without any modifications.

4.2.1.14 mAxis

Represents either a row, column or sheet of a particular table that it is linked to via mTableAxis.

Attribute	Type	Description
AxisID	INTEGER	Artificial ID.
AxisOrientation	TEXT	Either X, Y or Z for row, column or sheet respectively
AxisLabel	TEXT	Descriptive label (in English). Relevant for these axes with IsOpenAxis = 1, in particular Z axes (where it can be used e.g. to label a text or dropdown box for the user to enter/choose the Z axis value) and for open/semi-open Y axes (headers of columns in open tables).

IsOpenAxis	BOOLEAN	An "open" (1) ("closed" is 0) axis allows a variable number of entries, either chosen from a list of options or of a type of value. Used e.g. for vertical list tables, where a "line number" is used, and for "sheet per country/currency/sector" type tables.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.15 mTableAxis

Links axis and table to which it applies (enables reuse of axis for different tables).

Attribute	Type	Description
AxisID	INTEGER	Point to mAxis.AxisID.
TableID	INTEGER	Point to mTable.TableID.
Order	INTEGER	Required mainly for multiple Y or Z-axes. Indicates in what order the axes should be shown (i.e. in what order any text or dropdown boxes used to represent the axes should be displayed).

4.2.1.16 mAxisOrdinate

Represents a specific position on a closed axis (or the only ordinate on open axis referring to a typed dimension or semi-open axis pointing to a hierarchy). Tree structure of ordinates represents structure (indenting/nesting) of rows or columns.

Attribute	Type	Description
AxisID	INTEGER	Points to mAxis.AxisID.
OrdinateID	INTEGER	Artificial ID.
OrdinateLabel	TEXT	Descriptive label (in English). Text of a header.
OrdinateCode	TEXT	Row/column code (e.g. R0010, C0020, …)
IsDisplayBeforeChildren	BOOLEAN	Hint for display. If 1 then this ordinate is intended to be displayed above or to the left of any child ordinates, if 0/null it should be shown below or to the right of them.
IsAbstractHeader	BOOLEAN	If 1, then this ordinate does not represent any reportable data row or column or sheet, e.g. it may be displayed either as a completely grey row/column, or as just a heading with no row/column for values etc.
IsRowKey	BOOLEAN	Identifies (if 1) ordinate that is a key column in an open table. Otherwise it is not key (may be left empty/nilled).
Level	INTEGER	Level of this ordinate, lower level numbered ordinates "contain" higher numbered ones, i.e. lower levels are nearer the root (tree structure information).
Order	INTEGER	Position of this ordinate within its set of siblings.
ParentOrdinateID	INTEGER	Parent of this ordinate, if any - i.e. on the level immediately above.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

4.2.1.17 mOrdinateCategorisation

A pair of dimension and member describing one aspect of the categorisation of a particular position along an axis of a table.

Attribute	Type	Description
OrdinateID	INTEGER	Points to mAxisOrdinate.OrdinateID.

DimensionID	INTEGER	Points to mDimension.DimensionID. The dimension considered to describe (data in the cells that have) a specific position along an axis of a particular table.
MemberID	INTEGER	Points to mMember.MemberID. The relevant value of a dimension describing (data in the cells that have) a specific position along an axis of a particular table.
DimensionMemberSignature	TEXT	Signature for dimension and its member. Constructed as a component of mTableCell.DataPointSignature based on values of mDimension.DimensionXBRLCode, mMember.MemberXBRLCode and mOpenAxisValueRestriction (if applies).
DPS	TEXT	Same as DimensionMemberSignature but referring to XBRL codes rather than database IDs.
Source	TEXT	Identifies if categorisation is used for MD model or specific for HD model.

4.2.1.18 mOpenAxisValueRestricton

For table with semi open axes (i.e. those allowing a choice of a variable number of sheets/rows/columns each having one value from a particular domain), the values allowed to be reported may not be all the values from a domain, but only a subset. This table indicates the allowed subset by referencing a member in a hierarchy, all member below the referenced member are acceptable values, if IsStartingMemberIncluded is true, the referenced member is also a valid value, otherwise it is not.

Attribute	Type	Description
AxisID	INTEGER	Points to mAxis.AxisID (for semi-open axis). Axis to which this restriction applies.
HierarchyID	INTEGER	Points to mHierarchyNode.HierarchyID. Values for a semi open axis are restricted to those in the given hierarchy.
HierarchyStartingMemberID	INTEGER	Points to mHierarchyNode.MemberID. If provided values for a semi open axis are restricted to the descendants (or self) of this member in the given hierarchy.
IsStartingMemberIncluded	BOOLEAN	If 1, then the referred starting member is a valid value, if not, only it's descendants are.

4.2.1.19 mTableCell

Represents an individual intersection of row, column (and sheet) for a particular table.

Attribute	Type	Description
CellID	INTEGER	Artificial ID.
TableID	INTEGER	Points to mTable.TableID. A table this cell is part of.
IsRowKey	BOOLEAN	Same as mAxisOrdinate.IsRowKey but for cells.
IsShaded	BOOLEAN	Identifies if no data is expected to be entered into this cell, either because it is not required, or because this cell forms part of a heading, or the intersection of its row and column (and sheet) has no logical meaning.
BusinessCode	TEXT	Business code as assigned to a cell in the Business Templates (if applies).
DatapointSignature	TEXT	Signature of a data point represented by a cell. Identifies metric and dimension member pairs (sorted alphabetically base on dimension codes). In case of open values for metrics or dimensions uses wildcard (*), for semi-open axes refers information from mOpenAxisValueRestriction in square brackets [] and contains "?" character if default members is included in the referred hierarchy (and

		hence this dimension will be omitted in the XBRL instance document for this value). Created base on alphabetical concatenation of mOrdinateCategorisation.DimensionMemberSignarute (starting with the metric).
DPS	TEXT	Same as DatapointSignature but referring to XBRL codes rather than database IDs.

4.2.1.20 mCellPosition

Links a cell in a table to its position on the axes of that table by referring to ordinates on intersection of which the cell occurs.

Attribute	Type	Description
CellID	INTEGER	Points to mTableCell.CellID.
OrdinateID	INTEGER	Points to mAxisOrdinate.OrdinateID.

4.2.1.21 mConceptualModule

Represents modules in general, irrespective of taxonomy versions. Supportive table currenly unused.

Attribute	Type	Description
ConceptualModuleID	INTEGER	Artificial ID.
ConceptualModuleCode	TEXT	Short code for module. E.g. ARS, ARG, …
ConceptualModuleLabel	TEXT	English label of a module.

4.2.1.22 mModule

A module represents a reporting/filing unit, i.e. a set of tables that should be reported together in a single report (instance document).

Attribute	Type	Description
ModuleID	INTEGER	Artificial ID.
TaxonomyID	INTEGER	Points to mTaxonomy.TaxonomyID. Taxonomy to which this Module belongs.
ModuleCode	TEXT	Short code, e.g. ars, qrs, arg, …
ModuleLabel	TEXT	Descriptive label (in English).
ConceptualModuleID	TEXT	Points to mConceptualModule.ConceptualModuleID.
DefaultFrequency	TEXT	Frequency of reporting of a module (quarterly, annually, …).
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.
XBRLSchemaRef	TEXT	URI used for the schemaRef element in XBRL documents referring to this module. This is supposed to be the absolute URI to the official location of the taxonomy files in the domain of its owner.

4.2.1.23 mModuleBusinessTemplate

Indicates which Templates are included in each reporting module.

Attribute	Type	Description
ModuleID	INTEGER	Points to mModule.ModuleID. Module to which this entry relates.
Order	INTEGER	Sequence number to indicate (visual only) ordering of Templates within the Module. Templates and Tables within the module are presented as defined by tree structure information in TemplateOrTable table.

BusinessTemplateID	INTEGER	Template to be included in the Module. Points to mTemplateOrTable.TemplateOrTableID where TemplateOrTableType = "TemplateVariant".

4.2.1.24 mModuleLargeDimension

Identifies large dimensions for a modules for later use by the XBRL Parser (to populate dInstanceLargeDimensionMember) and subsequently by the internal ETL (migrating data from dFact to classic relational tables).

Attribute	Type	Description
ModuleID	INTEGER	Points to mModule.ModuleID.
DimensionID	INTEGER	Points to mDimension.DimensionID.

4.2.1.25 mTableDimensionSet

Identifies which metrics and dimensions are used in which tables and modules. Supportive table currently unused.

Attribute	Type	Description
ModuleID	INTEGER	Points to mModule.ModuleID.
TableID	INTEGER	Points to mTable.TableID.
MetricAndDimensions	TEXT	Identification of a metric and dimensions (sorted alphabetically) that appear in a table for a given module. There could be many entries for a table if different sets of dimensions are used (excluding default values).

4.2.1.26 mReference

Identifies regulation describing concepts or other artefacts.

Attribute	Type	Description
ReferenceID	INTEGER	Artificial ID.
SourceCode	TEXT	Short code of a regulation.
Article	TEXT	Article in the regulation structure.
Paragraph	TEXT	Paragraph in the regulation structure.
Point	TEXT	Point in the regulation structure.
Romans	TEXT	Roman number in the regulation structure.
ReferenceText	INTEGER	Text of a regulation.

4.2.1.27 mReferenceCategorisation

A pair of dimension and member describing one aspect of the categorisation of a particular position along an axis of a table.

Attribute	Type	Description
ReferenceID	INTEGER	Points to mReference.ReferenceID. The reference for which a dimensional categorisation is being provided.
DimensionID	INTEGER	Points to mDimension.DimensionID. The dimension considered to describe the applicability of the reference.
MemberID	INTEGER	Points to mMember.MemberID. The relevant value of a dimension describing the applicability of the reference.

4.2.1.28 mCellReference

Links a cell to a reference (e.g. regulations).

Attribute	Type	Description
CellID	INTEGER	Points to mTableCell.CellID.
ReferenceID	INTEGER	Points to mReference.ReferenceID.

4.2.1.29 mConceptReference

Links concept (and whatever it represents) to a reference (e.g. legal regulation).

Attribute	Type	Description
ConceptID	INTEGER	Points to mConcept.ConceptID.
ReferenceID	INTEGER	Points to mReference.ReferenceID.

4.2.1.30 mLanguage

Stores information on languages that can be used for translation of concepts.

Attribute	Type	Description
LanguageID	INTEGER	Artificial ID.
LanguageName	TEXT	Name of a language in that language.
EnglishName	TEXT	Name of a language in English.
IsoCode	TEXT	Language ISO (639-1) code.
ConceptID	INTEGER	Points to mConcept.ConceptID. Enables translation of language names to different languages.

4.2.1.31 mConceptTranslation

Links concept (and whatever it represents) to its translation in different languages.

Attribute	Type	Description
ConceptID	INTEGER	Points to mConcept.ConceptID.
LanguageID	INTEGER	Point to mLanguage.LanguageID.
OwnerID	INTEGER	Enables translations of a concept to the same language by different owners.
Text	TEXT	Text of the translation.

4.2.1.32 vExpression

Test expressions used by the validation rules.

Attribute	Type	Description
ExpressionID	INTEGER	Artificial ID.
ExpressionType	TEXT	One of the following: null, "Intrinsic in XBRL" (e.g. for identical data points), "Not implemented in XBRL".
TableBasedFormula	TEXT	Expression referring to table row/columns, e.g. {S.27.01.01.02, R0760,C0090} = {S.27.01.01.02, R0600,C0090} + {S.27.01.01.02, R0750,C0090}. In combination with vValidationRule.Scope identifies all potential cells involved in the rule.
LogicalExpression	TEXT	Expression referring to variable names (vVariableOfExpression.VariableCode), e.g. $a = +$b + $c.
ErrorMessage	TEXT	Message displayed when expression is evaluated to an error.

4.2.1.33 vPrecondition

Precondition for a rule.

Attribute	Type	Description

ValidationRuleID	INTEGER	Links precondition to a validation rule (vValidationRule.ValidationRuleID).
PreconditionExpressionID	INTEGER	Points to vExpression.ExpressionID representing the test of a precondition.

4.2.1.34 vValidationRule

Defines validation rules.

Attribute	Type	Description
ValidationRuleID	INTEGER	Artificial ID.
ValidationCode	TEXT	Code of validation (as defined by business users in the input materials or in the taxonomy files). E.g. S.02.01.03_A19B_x84
Severity	TEXT	Either "Error" or "Warting". May include “(deactivated)” to identify deactivated rules.
Scope	TEXT	Identification of where the rule applies in terms of tables and their row/columns, e.g. S.07.00.01.a (r010;020;030;040;050;060;070;080;090;110;130, All sheets)
ValidationType	TEXT	One of the following: "Allowed values for metric" (enumerations), "Coherence check" (related to introductory table), "Hierarchy", "Identity" (not implemented in XBRL as it is XBRL intrinsic), "Manual", "Sign", "Unique identifier" (probably not in XBRL, used for example to check uniqueness of row key in open tables).
ExpressionID	INTEGER	Points to vExpression.ExpressionID defining test expression for the validation rule.
ConceptID	INTEGER	Points to mConcept.ConceptID.

4.2.1.35 vValidationRuleSet

Identifies modules that include a validation rule.

Attribute	Type	Description
ModuleID	INTEGER	Points to mModule.ModuleID.
ValidationRuleID	INTEGER	Points to vValidationRule.ValidationRuleID.

4.2.1.36 vValidationScope

Identifies tables to which a validation rule applies (i.e. refers content of this table - metrics, data points, ordinates, cells, …). Also used to identify tables where it should not apply.

Attribute	Type	Description
ValidationRuleID	INTEGER	Points to vValidationRule.ValidationRuleID.
TableID	INTEGER	Points to mTable.TableID.

4.2.1.37 vVariableOfExpression

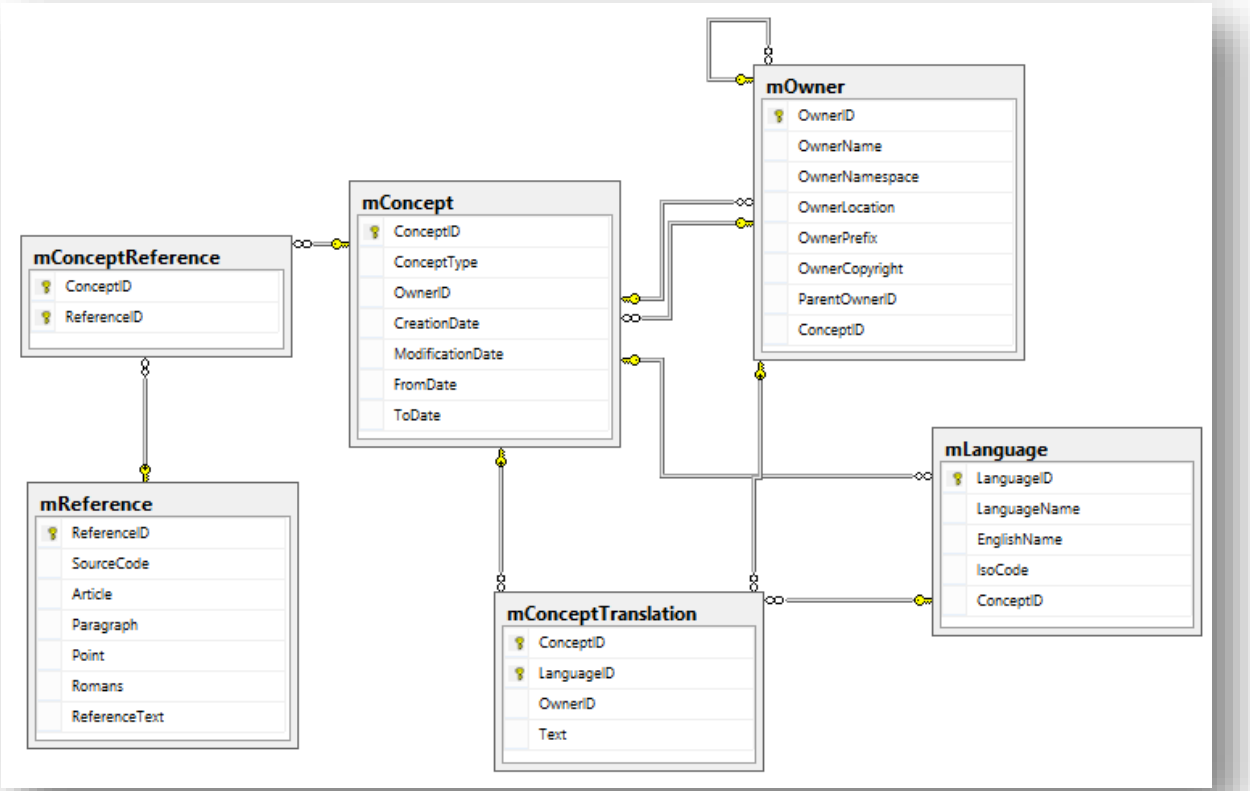
Names of variables and identification of their fall-back values in an expression.

Attribute	Type	Description
ExpressionID	INTEGER	Points to vExpressionID.ExpressionID.
VariableCode	TEXT	Code of a variable, e.g. a, b, c, …
IfMissing	TEXT	Either "treat as zero", or "do not run rule". Specifies the action to be taken if a value for a variable is not found (but the templates involved in the rule are reported, i.e. a blank cell rather than a missing table).

4.2.2 Relationships

4.2.2.1 *Concepts, owners, translations and references*

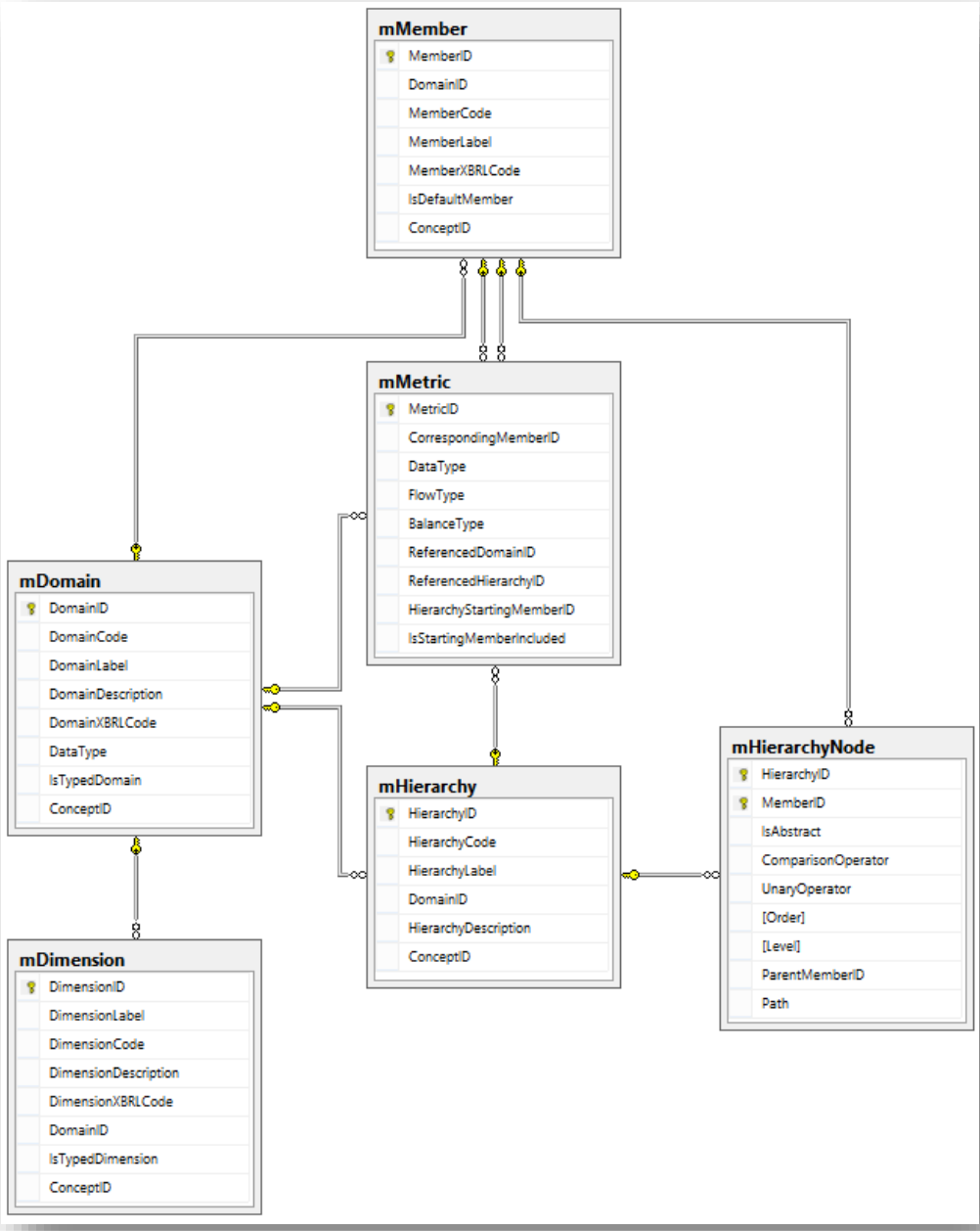
DPM artefacts from both, the dictionary (i.e. domains, members, dimensions, hierarchies, metrics) and the current information requirements (i.e. frameworks, taxonomies, templates and tables, axes, ordinates, etc.) can be defined by various institutions (owners), have multilingual labels (translations) and be described in multiple legal regulations (references). Therefore, these artefacts defined in various entities of the database refer to mConcept entity which supports metadata management, links to mOwner entity (in order to identify the institution that defined each artefact) and provide translations (mConceptTranslation) or references (mConceptReference and mReference).



Moreover, owners and languages are also concepts (for example language names can be translated in various languages, properties of owners can modified, etc.).

4.2.2.2 *Dictionary (domains, members, hierarchies and dimensions)*

Dictionary contains definitions of domains (mDomain). Each domain consists of members (mMembers) and is associated with dimensions (mDimensions) that further contextualize members in the information requirements section of the database. For documentation purposes and in order to support management of the dictionary, members are gathered in hierarchies (mHierarchy and mHierarchyNode). These tree-like structures may also describe basic arithmetical relationships between members (following the nesting and values of mHierarchyNode.ComparisonOperator and mHierarchyNode.UnaryOperator). Metrics (mMetric) are members of a selected domain that are further associated with period type, balance and data type attributes. In some cases, the latter could take form of a list of members of another domain (by reference to this domain and hierarchy of its members).



4.2.2.3 *Information requirements (frameworks, taxonomies, modules, templates, and tables)*

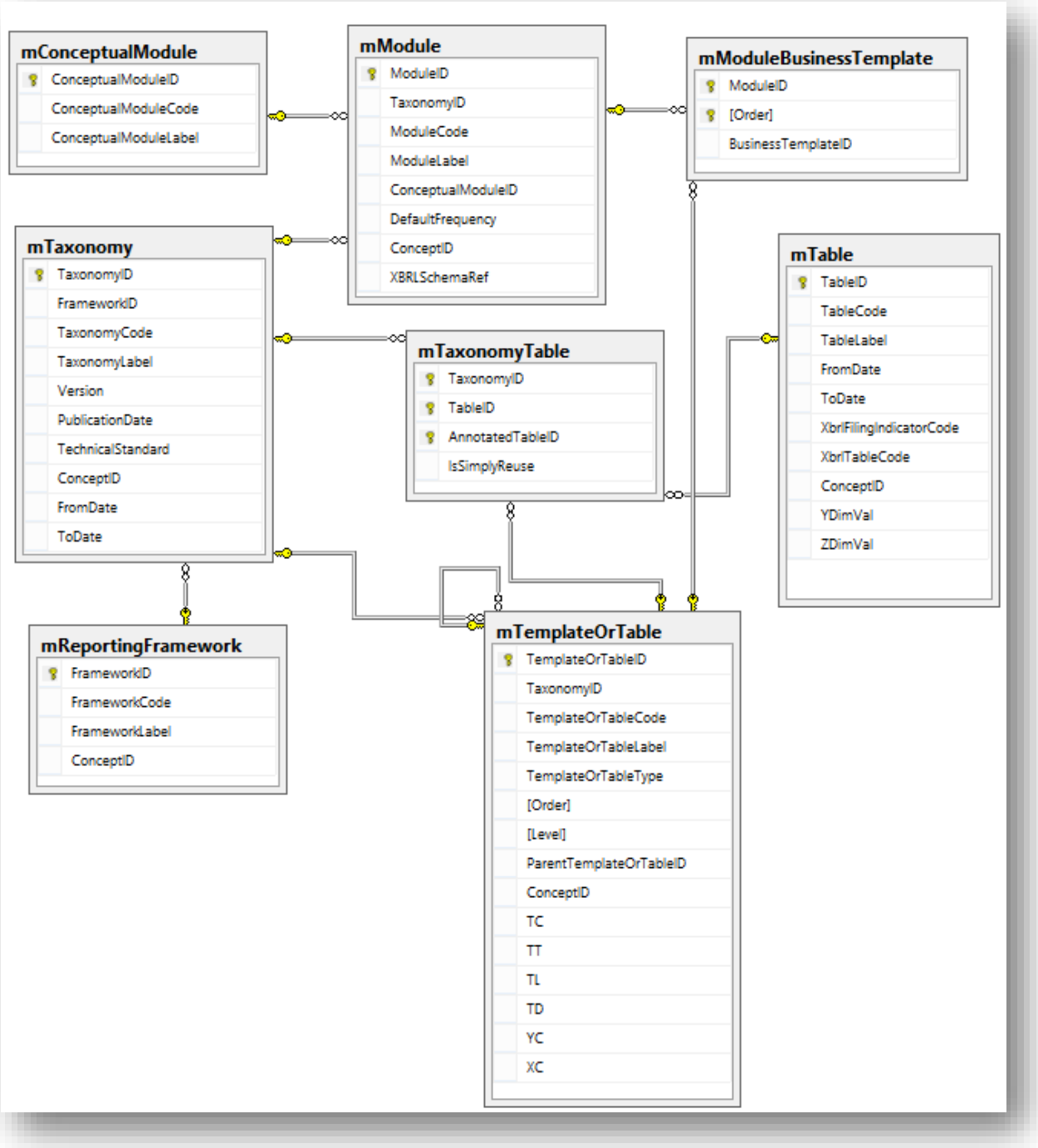
Information requirements are split in frameworks (mReportingFramework) that represent separate areas of interests/subject topics of collected data.

Frameworks are versioned as taxonomies (mTaxonomy) that identify data sets required at the particular moment of time (previous, current and potentially also future versions).

Taxonomies consists of templates (mTemplateOrTable with TemplateOrTableType = “TemplatesGroup”, “Template” or “TemplateVariant”) which are graphical tabular representations of information requirements as defined in the legal regulations. Templates may consist of multiple individual tables, being defined as such originally or split as a result of normalization of the original tables (mTemplateOrTable with TemplateOrTableType = “BusinessTable” or “AnnotatedTable”).

Description of actual tables starts with mTable entity. As in the EBA DPM MS Access database, tables can be reused by taxonomies if they remain unchanged between different versions of frameworks (mTaxonomyTable).

Taxonomies may define numerous templates. Depending on reporting period or type of a report not all templates must be filed under certain reporting scenario. Therefore modules (mModule) gather templates that are shall be submitted in one set (mModuleBusinessTemplate).



4.2.2.4 *Tables and their components: axes, ordinates, cells*

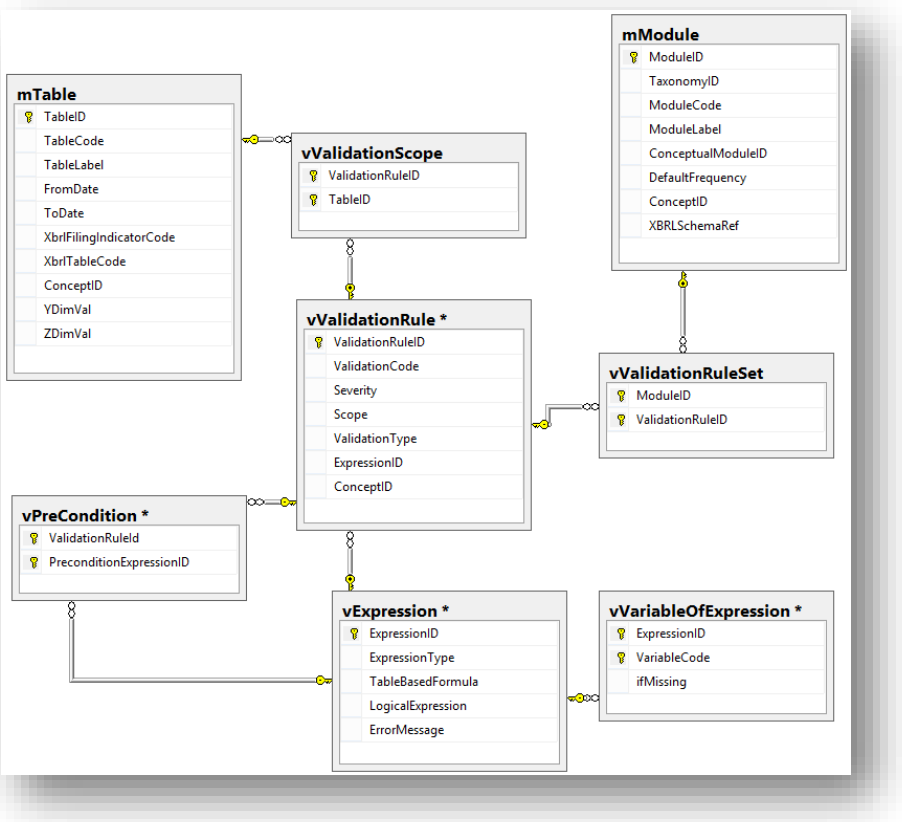
As described in the previous sections, actual tables are defined in mTable entity. There are linked with axes (mAxis) using mTableAxis entity. Each axis defines a section of the table represented as headers

of columns (AxisOrientation = “X”), headers of rows (AxisOrientation = “Y”) or pages/sheets (AxisOrientation = “Z”) multiplying the table, typically represented as a drop-down combo box above the table or reproductions of table views in separate window tabs. Axis consist of ordinates representing each individual header of a row, column or page/sheet (depending on disposition of the axis their belong to). Similarly to headers, ordinates can be nested and result in graphs/tree-structures. Ordinates may be associated with the dictionary concepts (mOrdinateCategorisation) identifying dimensions and members hidden behind the row/column/page header they represent (usually corresponding to, but not always identical as the text of a header). Axes whose ordinates are identical to the member structures defined in domains can link to hierarchies and reuse them as a whole or in parts (mOpenAxisValueRestriction). Table cells occur on the intersection of axis ordinates (mCellPosition). Each cell represents none (grey shaded/criss-crossed), one or many data points (mTableCell).

4.2.2.5 *Validation rules*

Reflection of the validation rules metadata is much less elaborate in this component of the T4U database comparing to the EBA DPM MS Access database. The reason for that is still unknown (at the moment of writing of this document) what would be the format of business rules definition in the input/source materials for the Solvency II and to which extend their execution would be conducted in the database (rather than outputting and evaluating them as XBRL taxonomy linkbase according to the Formula specification).

Currently in the database the validation rules are identified in vValidationRule entity. Their test expression and variables used are defined in vExpression and vVariableOfExpression entities. Validations can be linked to preconditions (vPreCondition) that can also have test expression and refer to variables. Scope of validations is defined in respect to tables where it applies (vValidationScope). Moreover validations are gathered in sets (vValidationRuleSet) based on their application to specific modules.



5 Database validations execution definition

There are two tables defining execution of validations (based on other table starting with “v”). Both hold definition of the SQL scripts generated for quick, run-time execution of the validation rules. Generated scripts, as a source of data for validation, use dynamic structures for data storage. Both tables that store validation rules scripts, contain two common columns:

- SQL – script of validation rules, that returns:
    - INSTANCE – identification of instance in which validation rule failed

- PK_ID – identification of the row of dynamic structures table, in which evaluation of validation rules failed
- E[i]_FORMULA – formula of validation rule evaluation (where [i] is the initial number of the validation rule that is evaluated in this SQL script), return null if rule is not broken, and returns string with formula containing fact values, when rule is broken
- CONTEXT – column returning additional information about

- CELLS – representation of cells that are included in evaluations of particular SQL script, pointing to the validation rule, initial number of the formula and set of cells that were part of evaluation {[ValidationRuleId],[i],[TableCode.rcCode]|[TableCode.rcCode]}

5.1 Entities

5.1.1 vValidationRuleSQL

Stores SQL scripts that execute rules working on multiple dynamic tables each.

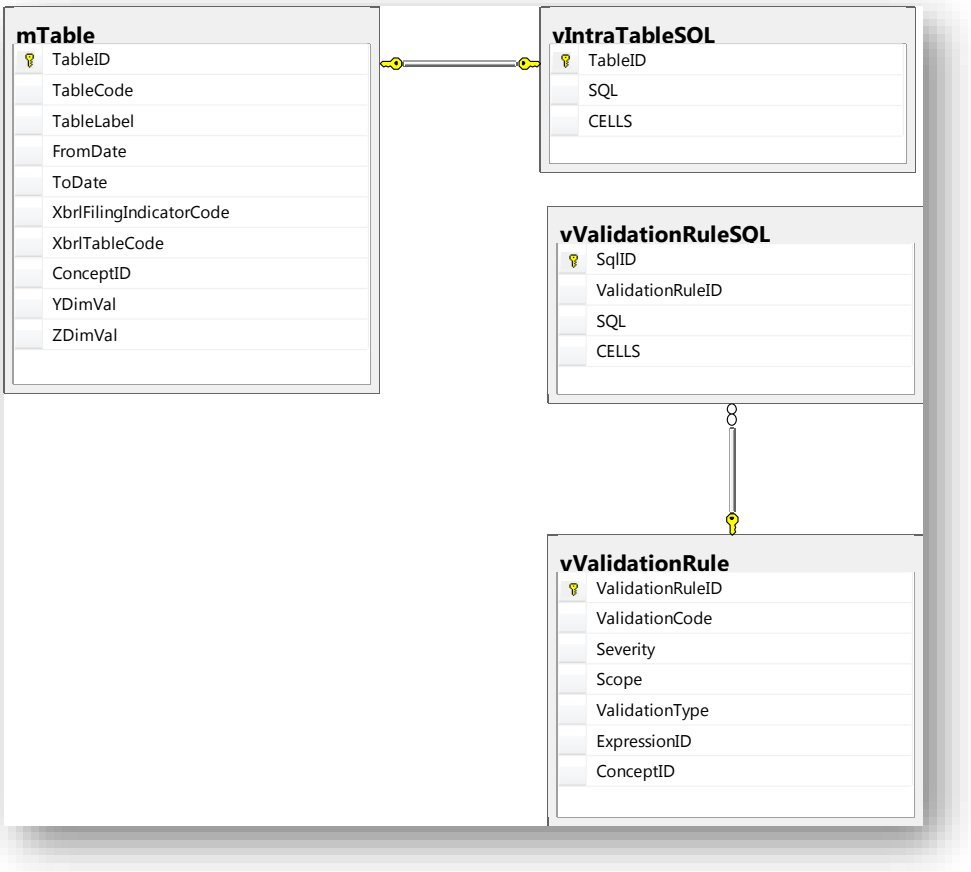
Attribute	Type	Description
SqlID	INTEGER	Artificial Primary Key of Validation rules SQL
ValidationRuleID	INTEGER	Foreign key, referencing ValidationRuleID in vValidationRule table
SQL	TEXT	SCL script of validation rule
CELLS	TEXT	Representation of cells that are included in evaluations of particular SQL script

5.1.2 vIntraTableSQL

Stores SQL scripts that execute rules working on single dynamic table each.

Attribute	Type	Description
TableID	INTEGER	Primary key referencing table id
SQL	TEXT	SCL script of validation rules
CELLS	TEXT	Representation of cells that are included in evaluations of particular SQL script

5.2 Relationships



6 Model of data storage

T4U database structure has two placeholders for storage of data:

- according to the DPM metadata for XBRL read/write,
- dynamic structures to be used in data validation or access (for rendering purposes).

6.1 Data storage according to DPM metadata for XBRL read/write

The DPM methodology is mainly focused on description of metadata for clear communication of information requirements by defining each data point fully, explicitly and consistently across the model. From this standpoint, reported facts refer to DPM properties (metrics and dimension-member pairs) describing the exchanged piece of information.

The simplest and the most natural manner of modelling this in the database is associating fact values with the data point signatures (as described in the previous chapter of this document).

6.1.1 Entities

There are only few entities used in this component of the database. Their purpose and description of their attributes is provided below in the next sections of this chapter.

6.1.1.1 *dInstance*

Stores information on instance documents.

Attribute	Type	Description

InstanceID	INTEGER	Artificial ID.
ModuleID	INTEGER	Points to mModule.ModuleID based on schema reference in the instance document (mModule.XbrlSchemaRef).
FileName	TEXT	Instance document file name.
CompressedFileBlob	BLOB	BLOB of compressed instance document file.
Timestamp	DATETIME	Date and time of instance creation (load or last modification in data entry interfaces).
EntityScheme	TEXT	Entity identifier scheme.
EntityIdentifier	TEXT	Entity identifier.
EntityName	TEXT	Entity name.
PeriodEndDateOrInstant	DATE	Date of facts in instance document.
EntityCurrency	TEXT	Default ISO 4217 currency code for reported monetary facts.

6.1.1.2 *dFact*

Reported facts.

Attribute	Type	Description
FactID	INTEGER	Artificial ID.
InstanceID	INTEGER	Instance document in which the fact was reported. Points to dInstance.InstanceID.
DataPointSignature	TEXT	Same as mTableCell.DataPointSignature but with wildcards replaced with actually reported values (for typed dimensions and dimensions referring to hierarchies of members).
Unit	TEXT	Content of unit measures. Usually: iso4217:EUR or other currency code for monetary and xbrli:pure for other numeric facts.
Decimals	TEXT	Precision of reported numeric fact as defined in XBRL specification by @decimals attribute.
NumericValue	REAL	Value of a fact if numeric.
DateTimeValue	DATE	Value of a fact if date. ISO Format i.e. YYYY-MM-DD.
BooleanValue	BOOLEAN	Value of a fact if boolean.
TextValue	TEXT	Value of a fact if not numeric, date or boolean.

6.1.1.3 *dFilingIndicator*

Identifies filing indicators sent in a report.

Attribute	Type	Description
InstanceID	INTEGER	Points to dInstance.InstanceID.
BusinessTemplateID	INTEGER	Points to mTemplateOrTable.TemplateOrTableID where TemplateOrTableType = "TemplateVariant".
Filed	BOOLEAN	0 for find:filed=”false”, 1 for find:filed=”true”, null for missing (“true” by default)

6.1.1.4 *dInstanceLargeDimensionMember*

Contains members for large dimensions.

Attribute	Type	Description
InstanceID	INTEGER	Points to dInstance.InstanceID.
DimensionID	INTEGER	Points to mDimension.DimensionID.
MemberID	INTEGER	Points to mMember.MemberID

6.1.1.5 *dAvailableTable*

Identifies tables where facts from an instance document could belong to. Unsused.

Attribute	Type	Description
InstanceID	INTEGER	Instance document in which the facts were reported (dInstance.InstanceID).
TableID	INTEGER	Table where facts could belongs to (mTable.TableID).
ZDimVal	TEXT	Values of Z axes properties (metrics, dimension member pairs, typed dimension values) of a table for facts in the instance (helps in selection from multiple values on Z axes).
NumberOfRows	INTEGER	Total number of rows for an open table and given Z axes values.

6.1.1.6 *dProcessingContext*

Temporary/staging phase table for storing information on contexts from a loaded instance document. Unused.

Attribute	Type	Description
InstanceID	INTEGER	Points to dInstance.InstanceID.
ContextID	TEXT	@id of a context from instance document.
SortedDimensions	TEXT	Dimensions and their values. Sorted.
IsNotValid	INTEGER	Identifies a context that for some reason is not valid.

6.1.1.7 *dProcessingFact*

Temporary/staging phase table for storing information on facts from a loaded instance document. Unused.

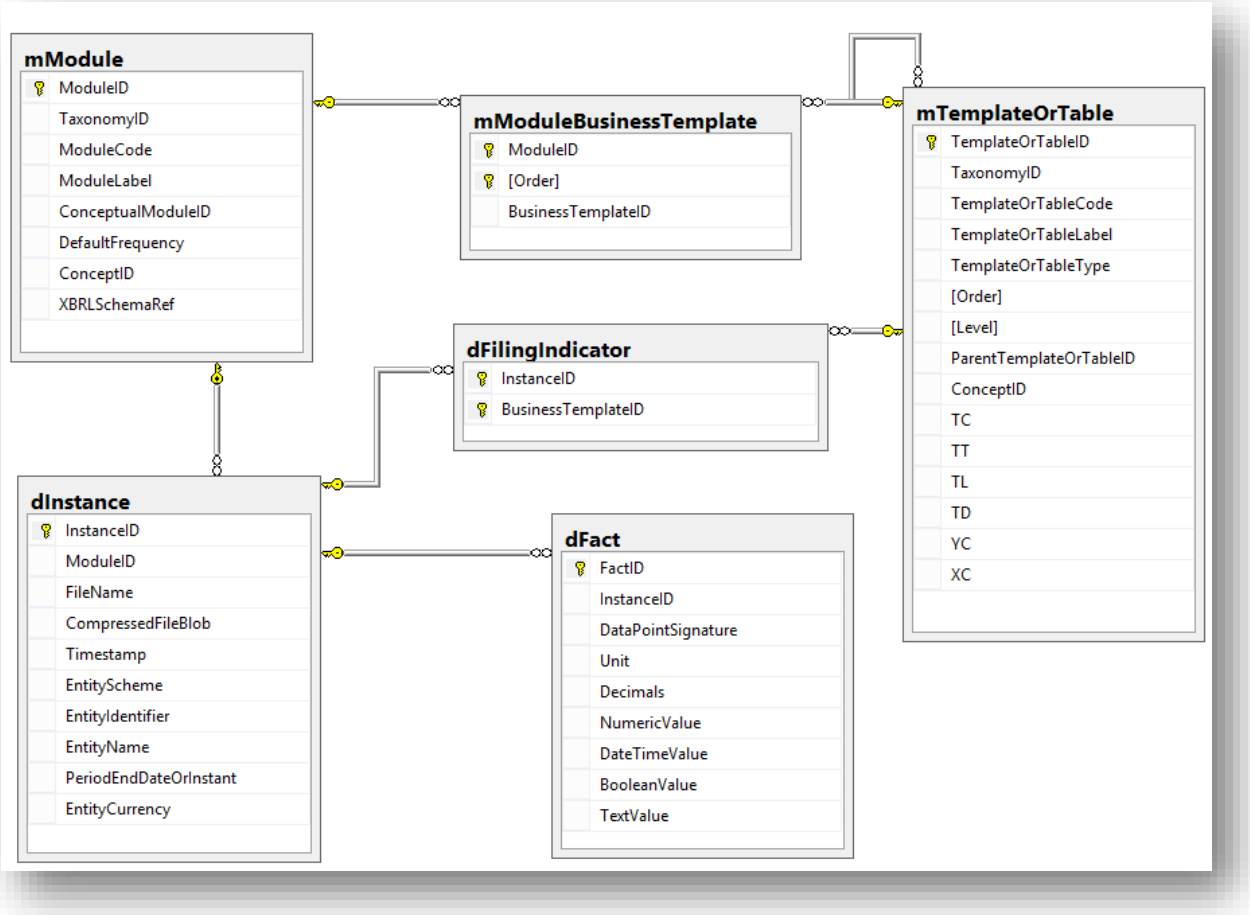
Attribute	Type	Description
InstanceID	INTEGER	Points to dInstance.InstanceID.
Metric	TEXT	Metric QName.
ContextID	TEXT	Points to dProcessingContext.ContextID.
Text	TEXT	Value of a fact if not numeric or date.
Number	REAL	Value of a fact if numeric.
Date	REAL	Value of a fact if date.
Error	TEXT	Identifies an error in fact declaration.

6.1.2 Relationships

Information about stored instance documents is represented in dInstance entity. It identifies a module (mModule) that was the basis for creation of a report (selecting from various available reporting scenarios). Indication of templates that were submitted for a given module is stored in dFilingIndicator entity.

Values for the reported facts are warehoused in dFact entity. Each fact is identified by a data point signature in DataPointSignature column.

Information on what tables were potentially reported may be generated in dAvailableTable (based on metadata from mTableDimensionSet). This information may be helpful in further processing and use of data.



6.2 Dynamic structures for data storage

This sections describes in more detail the principle of operation of the classic relational data storage placeholders and the related processes of data migration and validation.

6.2.1 General idea, goals and alternatives considered

As explained in the introduction to this document storage of facts according to the DPM properties is not flawless. Therefore and attempt was made to assess alternative approaches. The main goals for consideration during this assessment were related to:

- data validation:
  - automatic generation of checks based in business formulation (modelled in annotated templates, most likely in a table centric manner i.e. referring to business codes or rows/columns/sheet notation) or using the formulation of the EBA DPM MS Access database assertions (mix of table centric and data centric),
  - performance, especially in case of rules for open table,
  - duplicates (i.e. same fact appearing in different tables which is inevitable in table centric manner for data storage),
- rendering and data access (CRUD) for data entry tools:
  - handling of duplicates,
  - populating of open tables and hints for reported z-axes values,

- facilitate the understanding of the database for non DPM/XBRL experts and providing alternative structure for ETLs.

The following alternative solutions were considered in the process of assessment:

1. Dynamically created standard (classic) relational structures - for each taxonomy table a relational table is created; this table resembles row column structure of a taxonomy table it corresponds to.
2. Schema-star like structure - relational table which each row stores information about one fact from taxonomy table, additionally identifying from which table/row/column/page it comes.
3. Stable flat structure - one structure of relational table for representation of all taxonomy tables (with predefined attributes for various purposes).

The analysis of the three alternatives above led to the conclusion that:

- alternative number two is similar to the storage of facts in DPM properties placeholders, but instead of dimension-member pairs it uses row/column codes to identify the facts,
- solution three may encounter unexpected cases which would be hard to deal with (unknown and potentially different to the assumed number of columns for various purposes),
- option one is least stable but probably the closest to the desired result.

Selection of alternative one does not mean that it is flawless. The main drawbacks are:

- maintenance process includes regeneration of tables in case of changes in information requirements and migration of data,
- database structure is more complex and less stable,
- another steps in processing and validation of data (move data to another tables to interact with XBRL parser, detect duplicates, etc.),
- need to think of how to accommodate precision if can be different for each fact.

On the positive side, selected alternative one it is easy to understand by DPM unaware users, the queries on the data are relatively simple and it is expected that performance of validations and rendering should increase.

6.2.2 Example explaining principle of operation

Example explaining the principle of operation of classic relational structures placeholder is based on two “dummy” sample tables – closed table S.99.12.31.01 with z-axis and open table S.44.01.02.01:

S.99.12.31.01

Page	…

	C10	C20	C30	C40	C50
R10					
R20					
R30					
R40					
R50					

S.44.01.02.01

C10	C20	C30	C40
…	…	…	…

6.2.2.1 *Generating of classic relational tables*

DPM metadata of these tables is populated from the annotated templates and stored in DPM metadata components of the database. Please mind that for the sake of simplicity only these attributes necessary to explain the use case are presented and populated in the entities below.

In the first stage the information about the tables together with their codes is populated in mTable entity.

mTable

TableID	TableCode
1365	S.99.12.31.01
1699	S.44.01.02.01

Following this, information about the axes and their dispositions is stored in mAxis:

mAxis

AxisID	Orientation
122	X
123	Y
124	Z
131	Y
132	Y
133	X

and the axes are linked to tables via many-to-many mTableAxis:

mTableAxis

TableID	AxisID
1365	122
1365	123
1365	124
1699	131
1699	132
1699	133

Every row and column header is given representation in mAxisOrdinate table:

mAxisOrdinate

AxisID	OrdinateID	OrdinateCode	IsRowKey
122	201	10	
122	202	20	
122	203	30	
122	204	40	
122	205	50	
123	210	10	
123	211	20	
123	212	30	
123	213	40	
123	214	50	
124	215		
131	428	10	true
132	429	20	true
133	439	30	
133	440	40	

Axes referring to members lists from the dictionary (z-axis in closed table and one of y axes in open table) link to hierarchies in mOpenAxisValueRestriction.

mOpenAxisValueRestriction

AxisID	HierarchyID
124	12
132	12

All ordinates are assigned with dimension member codes hidden behind their description in annotated templates. This is done in mOrdinateCategorisation:

mOrdinateCategorisation

OrdinateID	DimensionCode	MemberCode
201	MET	mi2
201	BAS	x26
202	MET	mi5
203	MET	mi10
204	MET	mi12
205	MET	mi1
210	PFL	x12
211	PFL	x24
212	PFL	x32
213	PFL	x43
214	PFL	x23
215	CTP	open
428	IDC	open
429	CTP	open
439	MET	mi67
439	BAS	x12
440	MET	pi68

This information is enough to generate the classic relational structure data placeholders.

For every table in mTable and a given taxonomy (based on mTaxonomyTable entity) a separate relational table is created. The first column in this table is reference to dInstance entity, followed by a column for each z-axis (page) and a column for every cell in table (based on mTableCell, excluding criss-crossed/gray shaded cells):

1365_S.99.12.31.01

InstanceID	Page	R10C10	R10C20	R10C30	R10C40	R10C50	R20C10	…

For open tables, entities' columns resemble columns of the underlying table:

1699_S.44.01.02.01

InstanceID	C10	C20	C30	C40

6.2.2.2 *Mapping table*

In the process of generating classic relational tables from the DPM metadata container a mapping table is defined linking form centric representation with DPM properties hidden behind table row/column/page. Extract from the mapping table for the analysed example is presented below:

mMapping

TableID	RSTableName	RowColumnCode	Signature
1365	S.99.12.31.01	PAGE1	s2c_CTP(*)
1365	S.99.12.31.01	R10C10	MET(s2md_mi2)|s2c_BAS(s2c_BL:x26)|s2c_PFL(s2c_PL:x12)
1365	S.99.12.31.01	R10C20	MET(s2md_mi2)|s2c_BAS(s2c_BL:x26)s2c_PFL(s2c_PL:x12)
…			

1399	S.44.01.02.01	C10	s2c_IDC(*)
1399	S.44.01.02.02	C20	s2c_CTP(*)
1399	S.44.01.02.03	C30	MET(s2md_mi67)|s2c_BAS(s2c_BA:x12)
1399	S.44.01.02.04	C40	MET(s2md_pi68)

6.2.2.3 *Data migration*

Using information from the mapping table it is possible to move data between classic relational structures and the DPM properties fact storage (and vice versa).

The following sample numbers were entered in the exemplary tables in order to describe this process:

S.99.12.31.01

Page	PL

	C10	C20	C30	C40	C50
R10	2345		345	436	
R20					
R30					
R40					
R50					

S.44.01.02.01

C10	C20	C30	C40
12	PL	1001	0.15
322	ES	2034	0.34

These would be stored in the classic relational structures as follows:

1365_S.99.12.31.01

InstanceID	Page	R10C10	R10C20	R10C30	R10C40	R10C50	R20C10	…
1	eu_GA:PL	2345		345	436			

1699_S.44.01.02.01

InstanceID	C10	C20	C30	C40
1	12	PL	1001	0.15
1	322	ES	2034	0.34

Using the information from the mapping table it is relatively easy to migrate this data to the storage placeholder according to the DPM properties:

dFact

InstanceID	Signature	Value	Unit	Decimals
1	MET(s2md_mi2)|s2c_BAS(s2c_BL:x26)|s2c_CTP(eu_GA:PL)|s2c_PFL(s2c_PL:x12)	2345	EUR	0
1	MET(s2md_mi10)|s2c_CTP(eu_GA:PL)|s2c_PFL(s2c_PL:x12)	345	EUR	0
1	MET(s2md_mi12)|s2c_CTP(eu_GA:PL)|s2c_PFL(s2c_PL:x12)	436	EUR	0
…				
1	MET(s2md_mi67)|s2c_BAS(s2c_BA:x12)|s2c_CTP(eu_GA:PL)|s2c_IDC("12")	1001	EUR	0
1	MET(s2md_pi68)|s2c_CTP(eu_GA:PL)|s2c_IDC("12")	0.15	pure	2
1	MET(s2md_mi67)|s2c_BAS(s2c_BA:x12)|s2c_CTP(eu_GA:Es)|s2c_IDC("322")	2034	EUR	0
1	MET(s2md_pi68)|s2c_CTP(eu_GA:ES)|s2c_IDC("322")	0.34	pure	2

It should be equally easy to migrate the data in the other direction.

6.2.3 Entities

6.2.3.1 *dMapping*

Stores mapping information allowing for matching dynamic tables columns and dimensional characteristics of facts.

Attribute	Type	Description
TABLE_VERSION_ID	NUMERIC	Artificial ID.

DYN_TABLE_NAME	VARCHAR	Dynamic table name (code + framework + taxonomy version), e.g. "S_01_01_01_01__sol2__1_5_2_c"
DYN_TAB_COLUMN_NAME	VARCHAR	Row column code (R0010C0010) or Page column name (PAGEs2c_CS)
DIM_CODE	VARCHAR	Signature of a metric or dimension
DOM_CODE	VARCHAR	Typed domain signature (if applicable)
MEM_CODE	VARCHAR	Domain member signature (if applicable)
ORIGIN	VARCHAR	F, C, …
REQUIRED_MAPPINGS	INTEGER	Number of mappings of DYN_TAB_COLUMN_NAME that is required to be satisfied in order for fact to fit into the DYN_TAB_COLUMN_NAME
PAGE_COLUMNS_NUMBER	INTEGER	Number of page column for particular DYN_TABLE_NAME
DATA_TYPE	VARCHAR	Data type name abbreviation
IS_PAGE_COLUMN_KEY	NUMERIC	If row is a PAGE column key has value of 1
IS_DEFAULT	NUMERIC	If member from DIM_CODE a default member
IS_IN_TABLE	NUMERIC	For dimensional characteristics dummy columns

6.2.3.2 *Example of dynamic table: T__S_12_01_01_02__sol2__1_5_2_c*

Table T__S_12_01_01_02__sol2__1_5_2_c is based on S.12.01.01.02 table from taxonomy version 1.5.2.c of solvency2 framework.

Attribute	Type	Description
PK_ID	INTEGER	Artificial primary key
INSTANCE	INTEGER	Foreign key referencing id in distance table
R0010C0020	NUMERIC	Value column storing fact for cell R0010C0020
R0010C0030	VARCHAR	Value column storing fact for cell R0010C0030
R0010C0040	DATE	Value column storing fact for cell R0010C0040
PAGES2C_LX	VARCHAR	Column storing context information for dimension S2C_LX

7 Storage of validation results

Description of existing validation rules (based on the taxonomy) and storing of the validation results for each validated report. When run-time DB validation is triggered, results of validation are managed in memory, on level of source code. But when validations are performed by Arelle parsing component, during XBRL import or export, results are stored in dMessage and dMessageReference tables. These tables are also used for storage of duplicated data point exceptions, occurring during internal ETL from dynamic tables to dFact.

7.1 Entities

7.1.1 dMessage

Stores content of single results message.

Attribute	Type	Description
MessageID	INTEGER	Artificial primary key of message
InstanceID	INTEGER	Foreign key referencing InstanceID in dInstance table
SequenceInReport	INTEGER	Sequential number of message in processed report

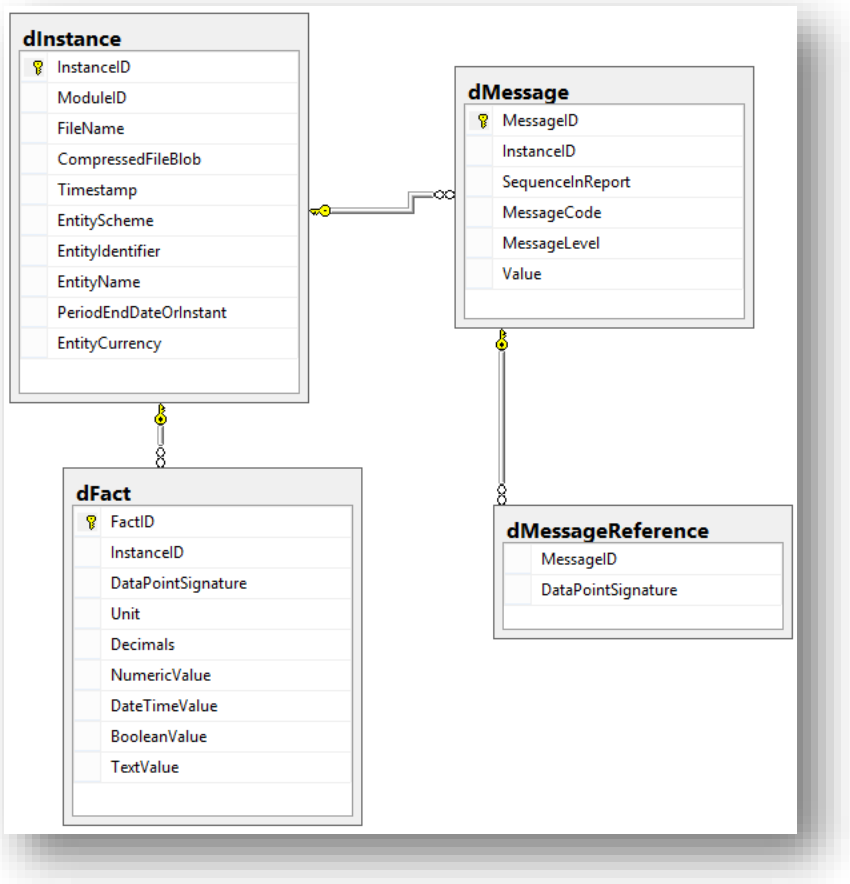
MessageCode	TEXT	Code identifying type of the message
MessageLevel	TEXT	Identification of message origin
Value	TEXT	String representing text of message

7.1.2 dMessageReferrence

Stores reference of message and fact for messages that are related with facts stored in dFact table.

Attribute	Type	Description
MessageID	INTEGER	Foreign key referencing MessageID in dMessage table
DataPointSignature	TEXT	String representing data point signature of fact related with message

7.2 Relationships



8 Application interface information

8.1 General model

T4U is expected to be used in multiple European countries with users speaking various languages. Therefore it is necessary to enable translation of the user interfaces menus, buttons and messages to these languages.

To accommodate this requirement the database contains three entities.

8.2 Entities

Names of the entities used by applications start with letter “a”. Their content is described in the following sections of the document.

8.2.1 aApplication

Tool for Undertaking applications (e.g. Windows Forms) and version.

Attribute	Type	Description
ApplicationID	INTEGER	Artificial ID.
ApplicationName	TEXT	Tool name, e.g. Windows Forms, …
ApplicationVersion	TEXT	
DatabaseType	TEXT	

8.2.2 aInterfaceComponent

Translation of interface components (e.g. buttons, messages, etc.) into national languages. At minimum English is provided. Tool for undertaking applications (Windows Forms, …) will use appropriate entries in this table in their interfaces.

Attribute	Type	Description
InterfaceComponentID	INTEGER	Artificial ID.
InBulgarian	TEXT	Text associated to the component in Bulgarian.
InCroatian	TEXT	Text associated to the component in Croatian.
InCzech	TEXT	Text associated to the component in Czech.
InDanish	TEXT	Text associated to the component in Danish.
InDutch	TEXT	Text associated to the component in Dutch.
InEnglish	TEXT	Text associated to the component in English.
InEstonian	TEXT	Text associated to the component in Estonian.
InFinnish	TEXT	Text associated to the component in Finnish.
InFrench	TEXT	Text associated to the component in French.
InGerman	TEXT	Text associated to the component in German.
InGreek	TEXT	Text associated to the component in Greek.
InHungarian	TEXT	Text associated to the component in Hungarian.
InIrish	TEXT	Text associated to the component in Irish.
InItalian	TEXT	Text associated to the component in Italian.
InLatvian	TEXT	Text associated to the component in Latvian.
InLithuanian	TEXT	Text associated to the component in Lithuanian.
InMaltese	TEXT	Text associated to the component in Maltese.
InPolish	TEXT	Text associated to the component in Polish.
InPortuguese	TEXT	Text associated to the component in Portuguese.
InRomanian	TEXT	Text associated to the component in Romanian.
InSlovak	TEXT	Text associated to the component in Slovak.
InSlovenian	TEXT	Text associated to the component in Slovenian.
InSpanish	TEXT	Text associated to the component in Spanish.
InSwedish	TEXT	Text associated to the component in Swedish.

8.2.3 aInterfaceComponentApplication

Links applications with interface components (buttons, messages, etc.).

Attribute	Type	Description
InterfaceComponentID	INTEGER	Points to aInterfaceComponent.InterfaceComponentID.
ApplicationID	INTEGER	Points to aApplication.ApplicationID.

8.3 Relationships

It is expected that the application based on different solution/supporting different technologies will share at least some components of the interface. Therefore the relation between the translation and each interface component was defined to be many-to-many as presented on the diagram below.