EIOPA DPM Database

Technical documentation

2026-06-30

Table of Contents

1 Introduction

This document describes the DPM database published by EIOPA (e.g. for Solvency II, Pension Funds, and other models).

2 Database Management System

The database is available in SQLite format.

3 Content

The database contains information requirements and validations metadata that resemble the DPM dictionary, annotated templates, and validation rules metadata; it is the description of the model (similar to an XBRL taxonomy).

4 Structure

4.1 Relation to DPM metamodel/methodology

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 familiarise themselves with documents listed at http://www.eurofiling.info/dpm/index.shtml and in particular the following EIOPA DPM documentation (explaining the DPM artefacts on diagrams). Relation to EBA DPM MS Access database.

EIOPA database closely follows the structure (entities and relationships) of the EBA DPM MS Access database, which can be found on EBA website under "Reporting frameworks" section. https://www.eba.europa.eu/risk-and-data-analysis/reporting/reporting-frameworks.

It is recommended that the readers of this section read also DPM Database documentation provided on EBA website. *CRD4 DPM - Database description - v2.1.pdf* embedded in the compressed folder https://www.eba.europa.eu/sites/default/files/2023-11/ff41ff26-f206-460a-8977-edf8a8883cba/DPM%20Database%20210.zip.

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

- different patterns used for reflection of data point keys (in EIOPA DPM 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),
- denormalisation of the model by deletion of relationships to dedicated entities and inclusion that information as attributes in entities' columns (e.g. data type, period type, balance attribute),
- lack of 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 EIOPA 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 syntax rather than referring to database object representing corresponding DPM artefacts (axes, ordinates, cells, metrics, etc.).

4.2 Relation to previous versions

EIOPA DPM Database structure and content has been changed comparing to releases prior 2.8.0 where they included artefacts needed for the decommissioned a few years ago Tool for Undertakings.

From the following T4U Database components:

- information requirements and validations metadata;
- placeholder for data storage;
- entities whose structure resembles information requirements and is based on tabular views;
- validations' representation in SQL;
- T4U applications information;
- supporting views used by various components of the application;

only the first one remains in 2.8.0 and its content is roughly identical apart from:

- missing links to Annotated templates Excel layout (e.g. TC/TT/TL… entries in mTemplateOrTable) resulting from 4.3,
- some fields required for the T4U may no longer be populated,
- Ordinate categorisations are only according to MD approach,
- tables grouping names do not distinguish between Template Variants/Groups/Template other than by level of nesting,
- entities containing definition of validations are now re-structured (simplified) and contain different syntactical representation of validation.

4.3 Source (input) information

DPM metadata in the database is generated from the metadata modelling platform used by EIOPA to define and maintain DPM models.

4.4 Representation of DPM model artefacts and relationships

4.4.1 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 brief description of the entities' content.

The next sections introduce 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.

4.4.1.1 *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).
OwnerCode	TEXT	Code assigned to the Institution (owner)
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.). Not populated at the moment.

4.4.1.2 *mConcept*

This table is used to provide more information on various artefacts (identification of the owner, translations to national languages, 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, HierarchyNode, Member, Module, ReportingFramework, Table, Taxonomy, TemplateOrTable, ValidationRule
OwnerID[1]	INTEGER	Points to mOwner.OwnerID.
ReleaseID	INTEGER	Identifier of the release
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.

4.4.1.3 *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.
IsNillable	BOOLEAN	Indicates if domain is nillable. A nillable element is valid with no content, even if its content type requires content.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner, and translation) information.

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

4.4.1.4 *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.
DomainID	INTEGER	Points to mDomain.DomainID. Domain from which the allowable values for this dimension are taken.
DefaultMemberID	INTEGER	Identifier of member that is set as default for this dimension.
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.

4.4.1.5 *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.

4.4.1.6 *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.

4.4.1.7 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.
HierarchyNodeID	INTEGER	Identifier of the hierarchy node.
ComparisonOperator	TEXT	Indicates the comparison relationship between this node and the aggregation of its children.
HierarchyNodeLabel	TEXT	Represents string value (label) assigned to the hierarchy node.
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.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner, and translation) information.
Path	TEXT	Path from the root node to this node, using MemberID split by comma.

4.4.1.8 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), “Enumeration/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 ("Stock"), or measured over a time period ("Flow"). N.B. not necessarily resembles the XBRL "period type" where all metrics are assumed to be mapped to instant period (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).
CustomDataTypeID	INTEGER	Identifier of the corresponding custom data type
IsStartingMemberIncluded	BOOLEAN	Informs if the starting member identified by HierarchyStartingMemberID is also a valid value for a metric (or is it only its descendants).

4.4.1.9 mReportingFramework

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

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

4.4.1.10 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.
ExcelTemplate	BLOB	Not used.

4.4.1.11 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: "TableGroup", “Business Table”.
Order	INTEGER	Order (preferably global but not necessary) of a Template or Table for displaying templates and tables in tree structure.
Level	INTEGER	Level for displaying templates and tables in tree structure, usually: 1 for Templates Group, 2 for Template, 3 for Template Variant, 4 for Annotated Template or if content is limited only to the last two:

		1 for Template Variant, 2 for Annotated Template.
ParentTemplateOrTableID	INTEGER	Parent template or table ID.
ConceptID	INTEGER	Points to mConcept.ConceptID.
IsTableGroupSource	BOOLEAN	Reflects object relation with respective table group
TC	TEXT	Not used.
TT	TEXT	Not used.
TL	TEXT	Not used.
TD	TEXT	Not used.
YC	TEXT	Not used.
XC	TEXT	Not used.

4.4.1.12 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	Not used.
JsonBlob	BLOB	Technical JSON attribute
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.

4.4.1.13 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.

4.4.1.14 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.

4.4.1.15 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).

4.4.1.16 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.
RelatedDimensionTableID	INTEGER	Points to mTable.TableID where related dimension is applied.
TypeOfKey	TEXT	Not used. Information about relations between keys in open tables along with their types and indication if they are mandatory or optional can be found in mConceptTranslation as *description* Role for these headers.

4.4.1.17 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	Not used.

4.4.1.18 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 its descendants are.

4.4.1.19 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	Not used (this information is in mAxisOrdinate.IsRowKey).
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) otherwise calculated as template and r/c codes.
DatapointSignature	TEXT	Signature of a data point represented by a cell. Identifies metric and dimension member pairs (sorted alphabetically based 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 are included in the referred hierarchy (and hence this dimension will be omitted in the XBRL instance document for this value). Created based on alphabetical concatenation of mOrdinateCategorisation.DimensionMemberSignarute (starting with the metric).
DPS	TEXT	Same as DatapointSignature but referring to XBRL codes rather than database IDs.

4.4.1.20 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.

4.4.1.21 mConceptualModule

Represents modules in general, irrespective of taxonomy versions.

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

4.4.1.22 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, …). Currently unused.
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.
AutogenerateRefs	BOOLEAN	Technical parameter.
JSONBlob	BLOB	Technical JSON attribute
JSONSchemaRef	TEXT	schemaREF represented in JSON format

4.4.1.23 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 TemplateOrTable represents TemplateVariant i.e. its Level = 3 (if all levels provided) or 2 (if content starts from TemplateVariant).

4.4.1.24 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.

4.4.1.25 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.
Text	TEXT	Text of the translation.
Role	TEXT	Type of documentation, e.g. *label*, *description*

4.4.1.26 vValidationRuleExpressions

Details of each validation rules.

Attribute	Type	Description
ValidationID	INTEGER	Artificial ID.
ValidationCode	TEXT	Code of validation rule, e.g. “BV1325”
ErrorMessage	TEXT	Error message of validation rule.
Rule	TEXT	Test expression, e.g. *{ m: [s2md_met:ei1024], seq: False, id: v0} != [s2c_CU:x5]* or *{t: S.02.01.01.01, r: R0100, dv: 0, seq: False, id: v1, f: solvency, fv: solvency2} i= {t: S.02.01.01.01, r: R0110, dv: 0, seq: False, id: v2, f: solvency, fv: solvency2} i+ {t: S.02.01.01.01, r: R0120, dv: 0, seq: False, id: v3, f: solvency, fv: solvency2}*
Filter	TEXT	Filter (if applies), e.g. *matches({t: S.06.02.04.02, c: C0290, z: Z0001, dv: emptySequence(), seq: True, id: v3, f: solvency, fv: solvency2}, "^..4.$") and {t: S.06.02.04.02, c: C0310, z: Z0001, dv: emptySequence(), seq: True, id: v4, f: solvency, fv: solvency2} = [s2c_PU:x16] and {t: S.06.02.04.01, c: C0090, z: Z0001, dv: emptySequence(), seq: True, id: v5, f: solvency, fv: solvency2} = [s2c_LB:x91]*

Prerequisites	TEXT	Not used.
Join	TEXT	Join (if applies), e.g. *join(dim({t: SR.02.01.07.01, r: R0780, c: C0010, z: Z0001, dv: 0, filter: dim(this(), [s2c_dim:PO]) = [s2c_PU:x40], seq: False, id: v5, f: solvency, fv: solvency2},[s2c_dim:NF]) = dim({t: SR.01.01.07.01, r: R0790, c: C0010, z: Z0001, filter: dim(this(), [s2c_dim:PO]) = [s2c_PU:x60], seq: False, id: v1, f: solvency, fv: solvency2},[s2c_dim:FN]))*
Scope	TEXT	Scope (if applies), e.g. *scope({t: SR.26.01.04.02, c:C0060;C0080, f: solvency, fv: solvency2})*
SQL	TEXT	Not used.
IsEnabled	BOOLEAN	Indication is the rule is active.
IsPoint	BOOLEAN	Not used.
AlwaysOn	BOOLEAN	Not used.
IncludeInXBRL	BOOLEAN	Indication if rule is part of XBRL taxonomy.
ToleranceMargin	TEXT	Not used.
VariableNames	TEXT	Not used.
ConceptID	INTEGER	Points to mConcept.ConceptID.
ExpertMode	BOOLEAN	Technical parameter.
SourceTaxonomyID	INTEGER	Not used.
PublishToParticles	BOOLEAN	Technical parameter.
Precondition	TEXT	Not used.

4.4.1.27 vValidationRuleTables

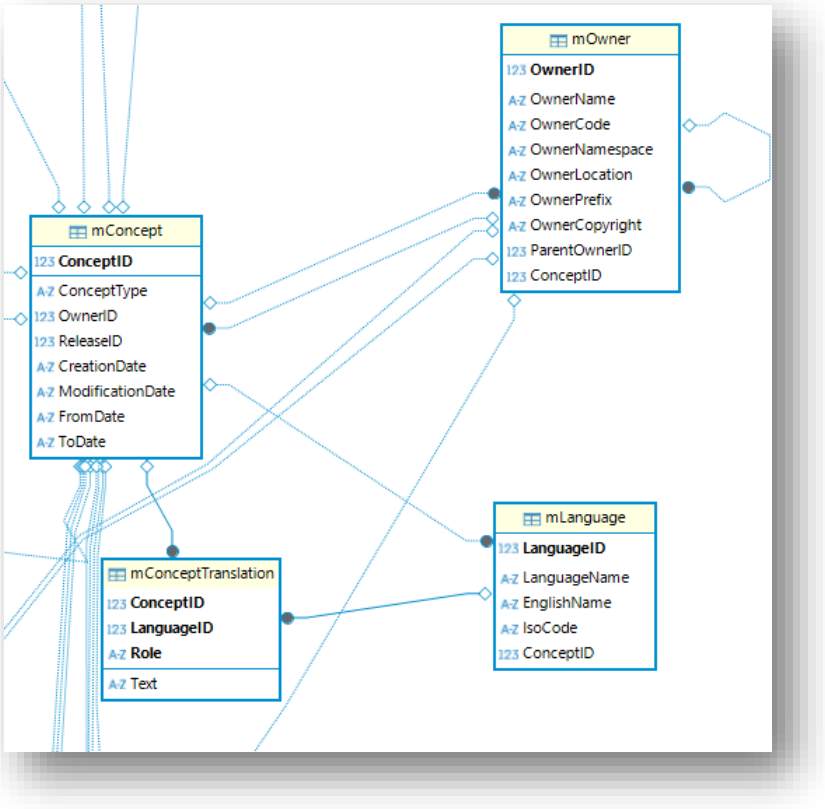
Application of rules to tables and modules.

Attribute	Type	Description
ValidationRuleID	INTEGER	Artificial ID.
ModuleID	INTEGER	mModule.ModuleID of module to which rule applies.
TableID	INTEGER	mTable.TableID of table to which rule applies (if applicable)
Severity	TEXT	Either "Error" or "Warning".
IsValidationSource	BOOLEAN	Not used.

4.4.2 Relationships

4.4.2.1 *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) and have multilingual labels (translations). 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).

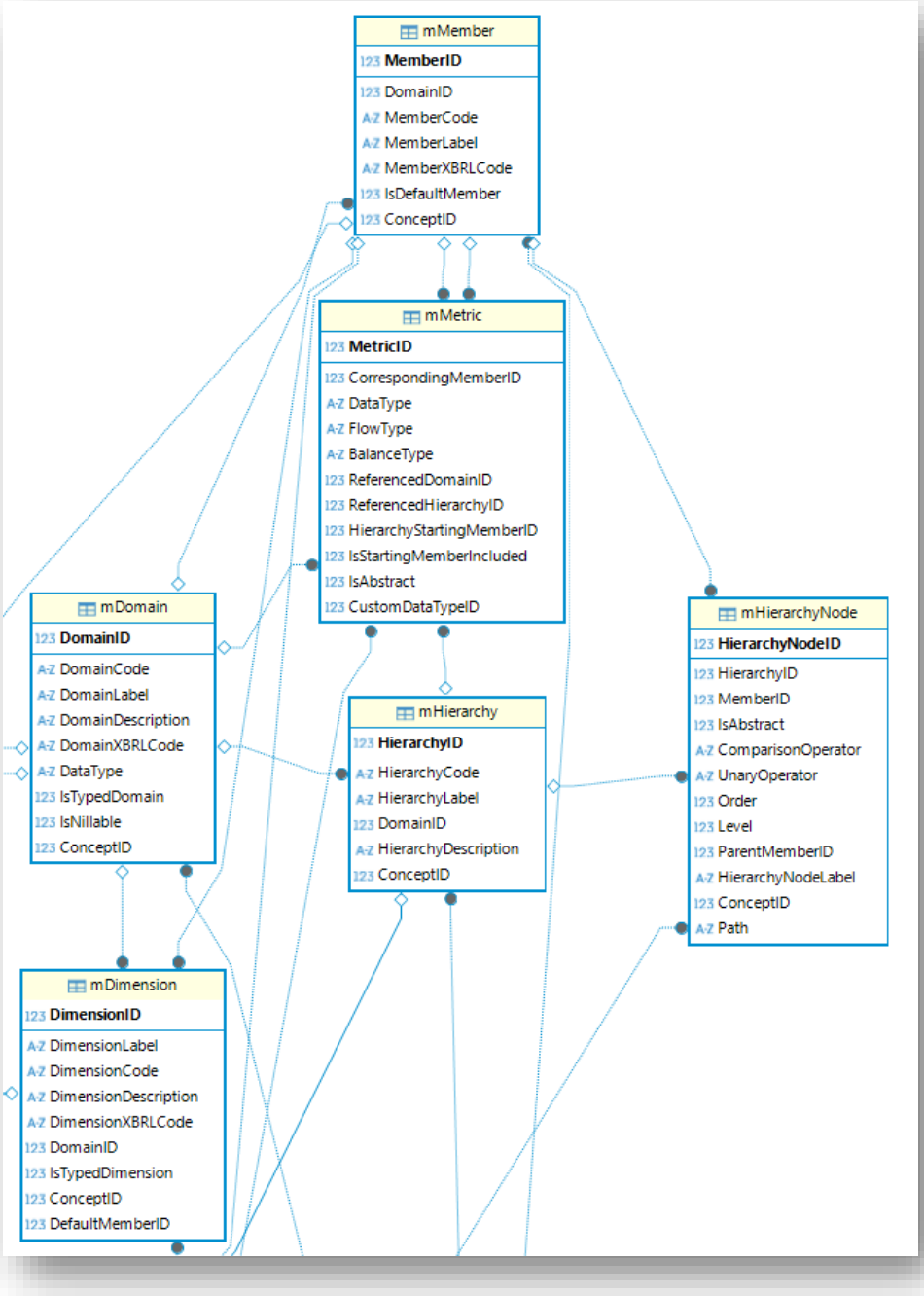


Moreover, owners and languages are also concepts (for example language names can be translated in various languages, properties of owners can modify, etc.).

4.4.2.2 *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 contextualise 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).

4.4.2.3 *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.

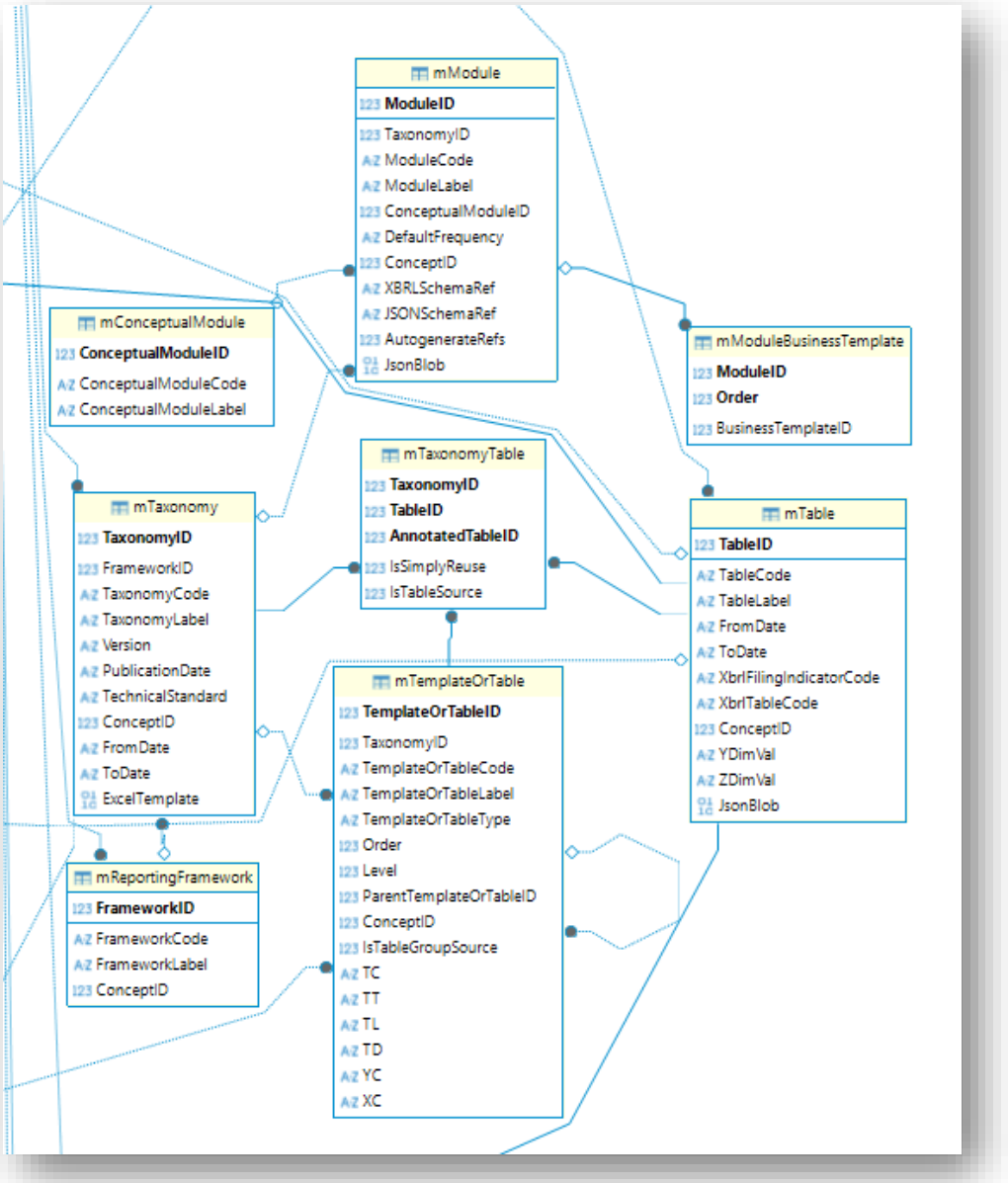
Frameworks are versioned as taxonomies (mTaxonomy) that identify data sets required at the particular moment of time (previous, current and potentially also future versions).

Taxonomies consist 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 because of normalisation 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).

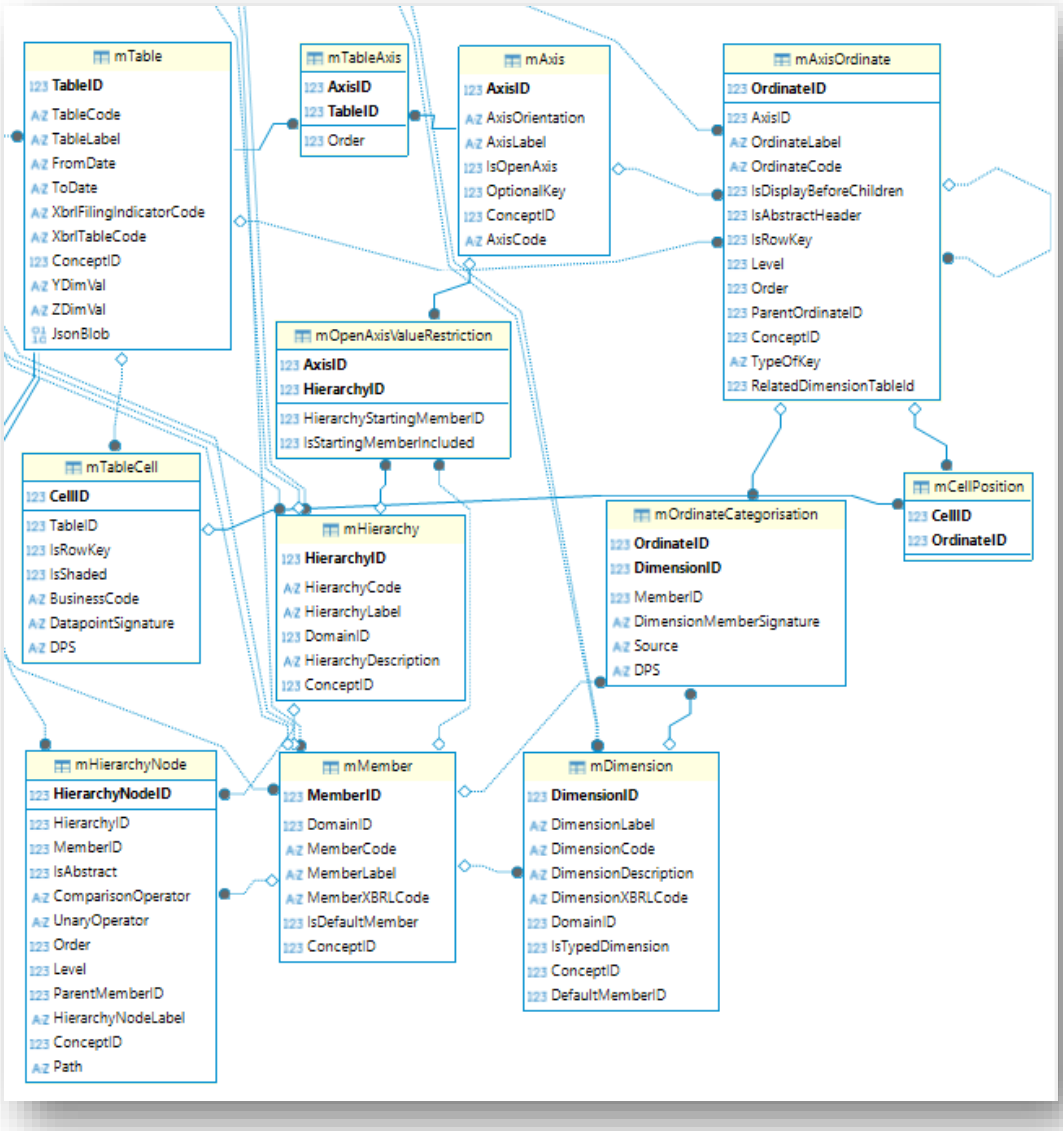
Taxonomies may define numerous templates. Depending on reporting period or type of a report not all templates must be filed under certain reporting scenario. Therefore modules (mModule) gather templates that are shall be submitted in one set (mModuleBusinessTemplate).



4.4.2.4 *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 consists 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).



4.4.2.5 *Validation rules*

Details on each validation rule are contained in vValidationRuleExpression entity. Application of each rule to modules and tables is indicated in mValidationRuleTables entity.

5 Model of data storage

Database structure, based on T4U implementation, 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).

5.1 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).

5.1.1 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.

5.1.1.1 *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.

5.1.1.2 *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)

5.1.2 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.

5.2 Dynamic structures for data storage

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

5.2.1 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, an 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 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 other 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.

5.2.2 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
…	…	…	…

5.2.2.1 *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 are 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 crisscrossed/grey 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

5.2.2.2 *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)

5.2.2.3 *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
R10	2345		345
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.

5.2.3 Entities

5.2.3.1 *MAPPING*

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

5.2.3.2 *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