X The T4U Database

Work in progress, Draft, 2014-05-14 19:35

This document describes the database used by the EIOPA 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 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) and production of valid XBRL reports. In addition, the database is prepared to satisfy the secondary priority requirement for the T4S, i.e. tool for supervisors, which is aiming to support national extension metadata and data.

Table of Contents

X.1 The 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 easy deployment (not installation of 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 is also used to perform the maintenance tasks like population of metadata from the DPM dictionary and annotated templates, generation of classic relational data structures from the DPM metadata, migration of data from the EBA DPM MS Access database, etc. This data is further transferred to SQLite in scope that is required by the standalone T4U applications.

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 as described in the paragraph above.

X.2 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. That could be understood as the description of the model (similar to a XBRL taxonomy)
2. **placeholder for data storage** – where stored facts refer to DPM properties. That could be understood as the actual reported data (similar to a 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 the table Linkbase and therefore similar to the reference business templates view.
4. **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.

Components described above are schematically presented on Figure 1 below.

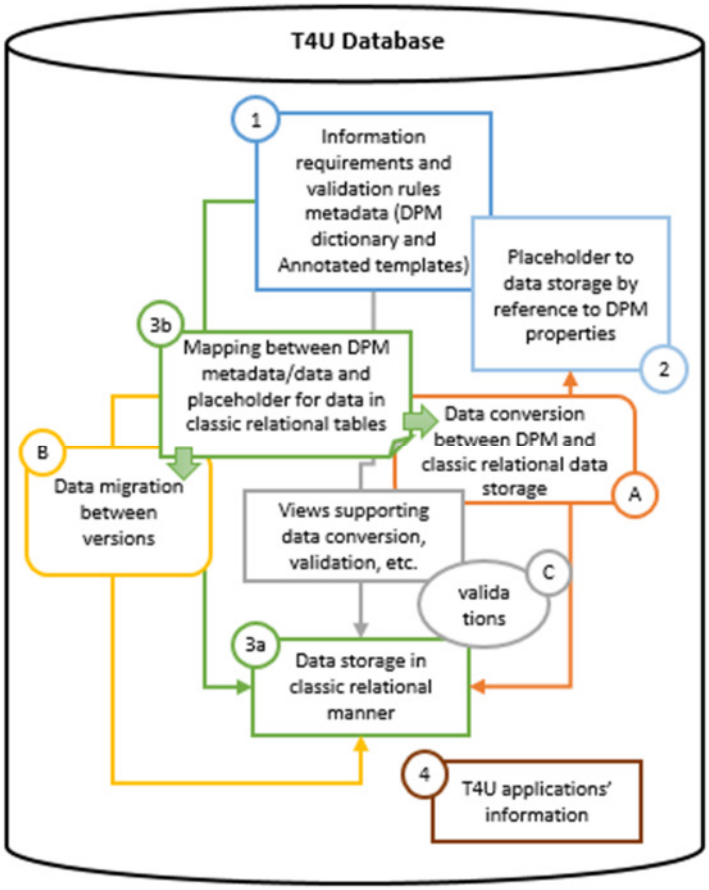


Figure 1. Components of the T4U database.

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 (including and views supporting data validation and aggregations)**.

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

Information requirements and validations metadata (component 1 on Figure 1) is populated in the design/setup stage from the input materials (DPM dictionary and annotated templates of Solvency II/national extensions or the EBA DPM MS Access database). 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 NCA extension 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 its dimensional properties, on the other the cells are gathered in tables, their columns and rows represented in a very normalized manner (as ordinates on axis).

This data centric description allowed for definition of placeholders for data storage according to the DPM properties (component 2 on Figure 1) which facilitate 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 load 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. For "closed" templates (with known/defined columns and rows), every cell becomes an attribute in the database entity (one entity for each template) and multiplication of a template (resulting for example from numerous z-axes properties) are row keys in these database entities. Simplicity of the design of this component of the database allows it to be used by the T4U graphical user interfaces (Windows Forms, Excel Add-in, iOS, …) or other connectors (like ODBC) to easily and quickly populate the T4U database with data as well as access data for their display or validation.

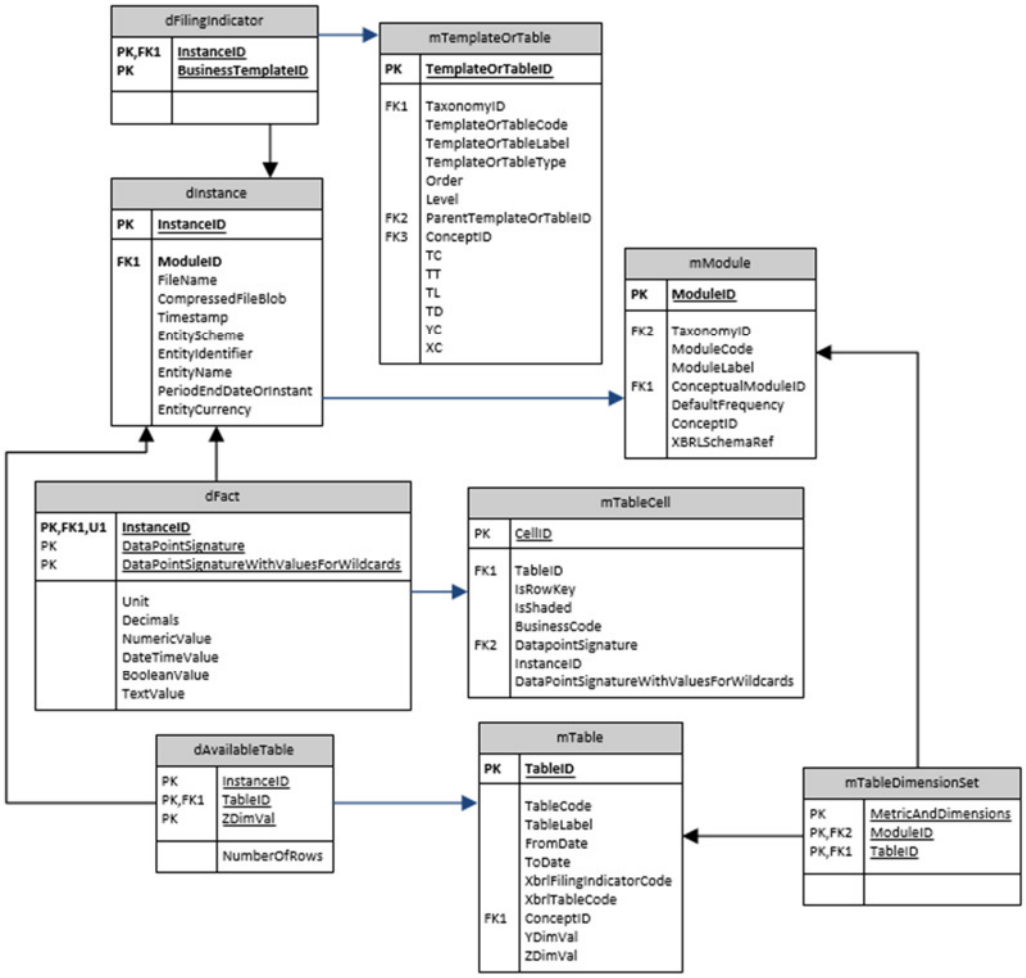
As explained in more details in section "

X.2.1 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 points to a signature of a data point (as represented by table cells where this fact belongs). In case when a table cell represents more than one data point, the concrete and full representation for a fact is provided in DataPointSignatureWithValuesForWildcards column.

Information on what tables were potentially reported is generated in dAvailableTable (based on metadata from mTableDimensionSet). This information is helpful in further processing and use of data (e.g. moving to classic relational structures as explained in the next chapter of this document).



Tables for classic relational data storage and mapping to DPM metadata properties" of this document, 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 the 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 GUI or validated). In this step (as well as the other described below) a set of database views is created to support for example identification of duplicates (i.e. duplicated facts) in the classic relational data storage (form centric data storage results in data duplication for facts shared between templates).

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 is rearranged, tables are split or merged, 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).

And last but not least, the mapping information supports the process of data validation (marked as process C on Figure 1). As explained above, business rules defined in the source materials (provided by business users) are loaded to the information requirements and validations metadata component (number 1 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) using supportive database views if necessary. In this process the mapping table information is 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 4 on Figure 1) is the applications' information. In general it contains translation of the application's interfaces (menu, buttons, messages, etc.) to national languages.

In the next sections of this document each component introduced above is explained in more details.

X.3 DPM, annotated templates and validations metadata

X.3.1 General model

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/10180/632822/DPM+Database.2.1.0.PC.7z. It is recommended that the readers of this document read first the *CRD4 DPM - Database description - v2.1.pdf* embedded in the compressed folder indicated in the link above. This is particularly important as some definitions and descriptions included in that document and applicable also for the T4U database are not repeated in this document. Moreover, the full comparison of entities of the EBA DPM MS Access and the T4U information requirements and validation rules metadata is provided in Annex 1 to this document.

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 recommended 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 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.),
- entities supporting relating facts to tables (by indicating dimensional properties used in tables),
- validation rules are limited to reference to row/column/sheet codes rather than DPM artefacts (axes, ordinates, cells, metrics, etc.)[1].

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

X.3.2 Source (input) information and database population mechanisms

This part of the database is populated from:

A. DPM dictionary and annotated templates Excel files of EIOPA Solvency II or compatible (i.e. identical in structure) defined by NCAs,
B. EBA DPM MS Access database.

In terms of A, the population of the database is performed using the mechanism similar to the one applied in the T4U Excel Add-In i.e. reverse engineered process of rendering of data input/entry forms. This process is automated and supports versioning and extensions (if defined in the expected manner). In addition to the artefacts declared in the DPM

[1] The ultimate format/manner of representation of validation rules metadata is subject to change depending on the input provided in the Solvency II templates and logs.

dictionary, annotated templates and validation rules, the supportive database content (e.g. data point signatures) is generated automatically.

For B, most of the migration is 1:1 with minor changes in entities' and their attributes' names (see Annex 1 for details). For T4U specific properties or more complex conversions a mechanism is developed to populate these information automatically based on the source material[2].

X.3.3 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.

X.3.3.1 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 Banking Authority, European Insurance and Occupational Pensions Authority, …
OwnerPrefix	TEXT	Recommended prefix of an owner (used to construct XBRL codes of different concepts). E.g. "eba", "s2c", …
OwnerNamespace	TEXT	Recommended namespace of an owner (the "core" part of the namespace, to be extended by suffixes representing different concepts).
OwnerCopyright	TEXT	Copyright text. Used in comments in XBRL taxonomy files.
OwnerLocation	TEXT	URI representing the root folder of the official location of taxonomy files.
ParentOwnerID	INTEGER	Points to OwnerID of an institution in case of extensions.
ConceptID	INTEGER	Owner is also a concept (its name can be translated, it can be versioned, etc.).

[2] Temporary/supportive entities may be included to facilitate this process, for example linking the IDs for keys of EBA DPM MS Access database with IDs of the T4U database.

X.3.3.2 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.
OwnerID	INTEGER	Points to mOwner.OwnerID.
ConceptType	TEXT	Axis, AxisOrdinate, Dimension, Domain, Hierarchy, Member, Module, ReportingFramework, Table, ConceptualTemplateOrTable, Taxonomy, Language, Owner…
CreationDate	DATE	Date when concept was first created.
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.
ModificationDate	DATE	Date when concept was last modified.

X.3.3.3 mDomain

This table lists domains. Domains group values of a particular kind/addressing a particular concept. May have an explicit list of allowable values (members), or else specify values of a particular type or pattern (a "typed" domain). Provides the allowable values for a dimension.

Attribute	Type	Description
DomainID	INTEGER	Artificial ID.
DomainLabel	TEXT	Descriptive label (in English).
DomainCode	TEXT	Short code (usually two capital letters).
DomainDescription	TEXT	Longer description (in English).
DomainXbrlCode	TEXT	Code (QName) used in XBRL documents.
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 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.

X.3.3.4 mDimension

Category/aspect used to describe and differentiate data points, each relates to one specific feature. Allowed values are taken from a domain. If these are explicitly listed they are called members.

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.
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.

X.3.3.5 mMember

An explicit possible value within a domain.

Attribute	Type	Description
MemberID	INTEGER	Artificial ID.
MemberLabel	TEXT	Descriptive label (in English).
MemberCode	TEXT	Short code (resembling XBRL local name).
MemberXbrlCode	TEXT	Code (QName) used in XBRL documents.
DomainID	INTEGER	Points to mDomain.DomainID. Domain to which this member belongs.
IsDefaultMember	BOOLEAN	Identifies if the member is a default value 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.

X.3.3.6 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).
HierarchyLabel	TEXT	Descriptive label (in English).
HierarchyDescription	TEXT	Longer description (in English).
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.
DomainID	INTEGER	Points to mDomain.DomainID. Domain this hierarchy relates to.

X.3.3.7 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.
Level	INTEGER	Level of this node, lower level numbered nodes contain higher numbered ones, i.e. lower levels are nearer the root (tree structure information).
Order	INTEGER	Position of this node within its set of siblings, if any (tree structure information).
ParentMemberID	INTEGER	Indicates the parent of this node, if any - i.e. the level immediately above (tree structure information).
UnaryOperator	TEXT	Indicates the contribution of this node to the aggregation of its siblings.
ComparisonOperator	TEXT	Indicates the relationship between this node and the aggregation of its children.

X.3.3.8 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 in the future).
DataType	INTEGER	Type of data. One of the following: "Monetary"/"Percent"/"Decimal"/"Integer"/"Date"/ "Boolean"/"Text"/"DomainBased".
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 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 (for code-typed/domain-based Metrics)
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).

X.3.3.9 mReportingFramework

Overall reporting framework. High level, stable concept. E.g. COREP, FINREP, Asset Encumbrance, Solvency 2, …

Attribute	Type	Description
FrameworkID	INTEGER	Artificial ID.
FrameworkCode	TEXT	Short code of a framework (e.g. COREP, FINREP, AE, Sol2 …).
FrameworkLabel	TEXT	Descriptive label (in English). E.g. Common Reporting, Financial Reporting, Asset Encumbrance, Solvency 2 …
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

X.3.3.10 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.
TaxonomyCode	TEXT	Short code of a taxonomy, e.g. FINREP 2.1.
TaxonomyLabel	TEXT	Descriptive label (English), e.g. FINREP 2014 (2.1) PC DRAFT
Version	TEXT	E.g. 2.1.0
PublicationDate	DATE	Taxonomy publication date (e.g. 2014-03-31). To be used in namespaces of taxonomy files.
FromDate	DATE	Date from which this taxonomy is/was valid.
ToDate	DATE	Date until which this taxonomy is/was valid.
FrameworkID	INTEGER	Points to mReportingFramework.FrameworkID. Reporting framework this taxonomy describes.
TechnicalStandard	TEXT	Identifier of the prescriptive technical standard which this taxonomy describes/models. E.g. ITS 2013 02.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

X.3.3.11 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.
TemplateOrTableLabel	TEXT	Description (in English). Usually template/table title.
TemplateOrTableType	TEXT	This is one of the following: "TemplatesGroup", "Template", "TemplateVariant", "BusinessTable", "AnnotatedTable".
Level	INTEGER	Level of a Template or Table for displaying templates and tables in tree structure.
Order	INTEGER	Order (preferably global) of a Template or Table for displaying templates and tables in tree structure.

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.
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.
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.

X.3.3.12 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 tables - dimensions (with wildcards *) used on open Y axes.

ZDimVal	TEXT	Metrics and dimension members (or wildcards * for open axis) used on Z axes.

X.3.3.13 mTaxonomyTable

Links taxonomies with tables (allowing for reuse of the latter by different taxonomies).

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.

X.3.3.14 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.
AxisLabel	TEXT	Descriptive label (in English). Relevant for 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 Y axes (that don't have ordinates and link directly to OpenAxisValueRestriction). This column is not null mainly for IsOpenAxis = 1.
AxisOrder	INTEGER	Used 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).
AxisOrientation	TEXT	Either X, Y or Z for row, column or sheet respectively
IsOpenAxis	BOOLEAN	An "open" (vs. "closed") 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.

X.3.3.15 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.

X.3.3.16 mAxisOrdinate

Represents a specific position on a "closed" axis (or only ordinate on "open" axis referring to typed dimension). Tree structure of ordinates represent indenting / nesting of rows or columns (used for e.g. "of which" type breakdowns).

Attribute	Type	Description
AxisID	INTEGER	Points to mAxis.AxisID.
OrdinateID	INTEGER	Artificial ID.
OrdinateLabel	TEXT	Descriptive label (in English).
OrdinateCode	TEXT	Row/column code (e.g. 010, 020, …)
IsDisplayBeforeChildren	BOOLEAN	Hint for display. If yes/true then this ordinate is intended to be displayed above or to the left of any child ordinates, if false it should be shown below or to the right of them.
IsAbstractHeader	BOOLEAN	If true, this ordinate does not represent any "reportable data", e.g. it may either be displayed as a completely "grey row/column", or as just a heading with no row/column for values etc.
IsRowKey	BOOLEAN	Identifies ordinate that is also repeated as the key axis in open tables.
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, if any (tree structure information).
ParentOrdinateID	INTEGER	Parent of this ordinate, if any - i.e. the level immediately above (tree structure information).
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

X.3.3.17 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 it was a component of mTableCell.DataPointSignature base on values of mDimension.DimensionXBRLCode, mMember.MemberXBRLCode, etc.

X.3.3.18 mOpenAxisValueRestriction

For table with "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 open axis). Axis to which this restriction applies.
HierarchyID	INTEGER	Points to mHierarchyNode.HierarchyID. Values for the open axis are restricted to those in the given hierarchy.
HierarchyStartingMemberID	INTEGER	Points to mHierarchyNode.MemberID. Values for the open axis are restricted to the descendants of this member in the given hierarchy.
IsStartingMemberIncluded	BOOLEAN	If yes, the linked member is a valid value, if not, only it's descendants are.

X.3.3.19 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	Identifies if this cell represent an artificial code/ID used to identify a row of data (in an open table/list). Applies to cells under mAxisOrdinate.IsRowKey column.
IsShaded	BOOLEAN	Identifies if no data 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.
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 (*). Created base on alphabetical concatenation of mOrdinateCategorisation.DimensionMemberSignarute (starting with the metric).

X.3.3.20 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.

X.3.3.21 mConceptualModule

Represents modules in general, irrespective of taxonomy versions.

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

X.3.3.22 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.
ModuleCode	TEXT	Short code, e.g. COREP_LCR_Con, FINREP_Con_IFRS, ARS, ARG, QRS, …
ModuleLabel	TEXT	Descriptive label (in English).
ConceptualModuleID	TEXT	Points to mConceptualModule.ConceptualModuleID.
DefaultFrequency	TEXT	Frequency of reporting of a module (quarterly, annually, …).
TaxonomyID	INTEGER	Points to mTaxonomy.TaxonomyID. Taxonomy to which this Module belongs.
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.
ConceptID	INTEGER	Points to mConcept.ConceptID. Reference to concept (change, owner and translation) information.

X.3.3.23 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.
BusinessTemplateID	INTEGER	Template to be included in the Module. Points to mTemplateOrTable.TemplateOrTableID where TemplateOrTableType = "TemplateVariant".
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.

X.3.3.24 mTableDimensionSet

Identifies which metrics and dimensions are used in which tables and modules.

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).

X.3.3.25 mReference

Identifies regulation describing concepts or other artefacts.

Attribute	Type	Description
ReferenceID	INTEGER	Artificial ID.
SourceCode	TEXT	Short code of a regulation (e.g. CRR, IFRS, …).
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.

X.3.3.26 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.

X.3.3.27 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.

X.3.3.28 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.

X.3.3.29 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.

X.3.3.30 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.

X.3.3.31 vExpression

Test expressions used by the validation rules.

Attribute	Type	Description
ExpressionID	INTEGER	Artificial ID.
ErrorMessage	TEXT	Message displayed when expression is evaluated to an error.
ExpressionType	TEXT	One of the following: empty, "Intrinsic in XBRL", "Not implemented in XBRL"
LogicalExpression	TEXT	Expression referring to variable names (vVariableOfExpression.VariableCode), e.g. $a = +$b + $c.
TableBasedFormula	TEXT	Expression referring to table row/columns, e.g. {r100} = +{r110} + {r120} + {r130} + {r140}. In combination with vValidationRule.Scope identifies all potential cells involved in the rule.

X.3.3.32 vPrecondition

Precondition for a rule.

Attribute	Type	Description
PreconditionExpressionID	INTEGER	Points to vExpression.ExpressionID representing the test of a precondition.

ValidationRuleID	INTEGER	Links precondition to a validation rule (vValidationRule.ValidationRuleID).

X.3.3.33 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).
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.
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)
Severity	TEXT	Either "ERROR" or "WARNING".
ConceptID	INTEGER	Points to mConcept.ConceptID.

X.3.3.34 vValidationRuleSet

Identifies modules that include a validation rule.

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

X.3.3.35 vValidationRuleScope

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
TableID	INTEGER	Points to mTable.TableID.
ValidationRuleID	INTEGER	Points to vValidationRule.ValidationRuleID.

X.3.3.36 vVariableOfExpression

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

Attribute	Type	Description
ExpressionID	INTEGER	Points to vExpressionID.ExpressionID.
IfMissing	TEXT	Either "treat as zero", or "do not run rule". Specifies the

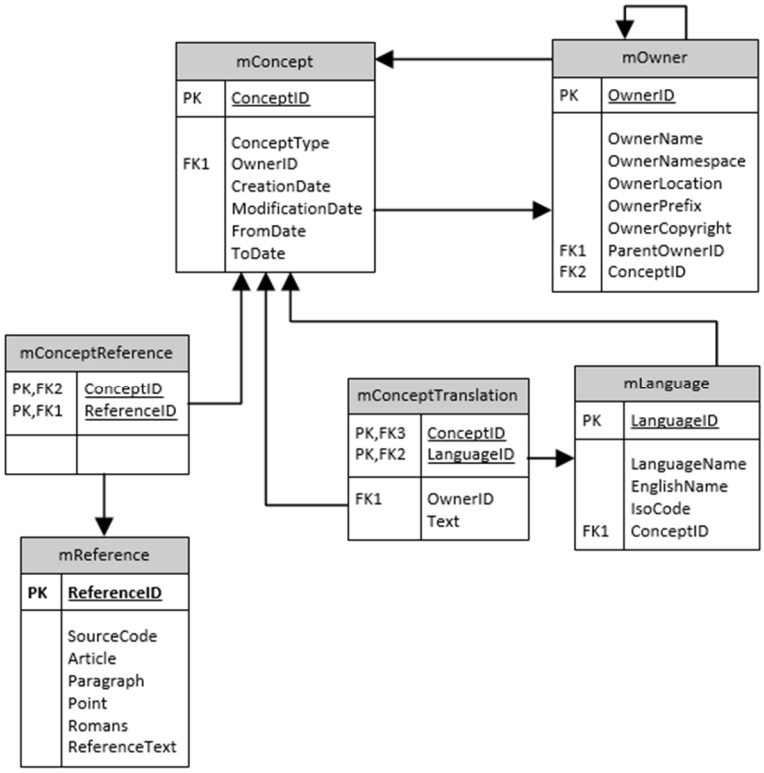
		action to be taken if a value for a variable is not found (but the templates involved in the rule are reported, i.e. a blank cell rather than a missing table).
VariableCode	TEXT	Code of a variable, e.g. a, b, c, …

X.4 Relationships

Diagrams bellow present and describe entity-relationship model of this component of the database.

X.4.1 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).

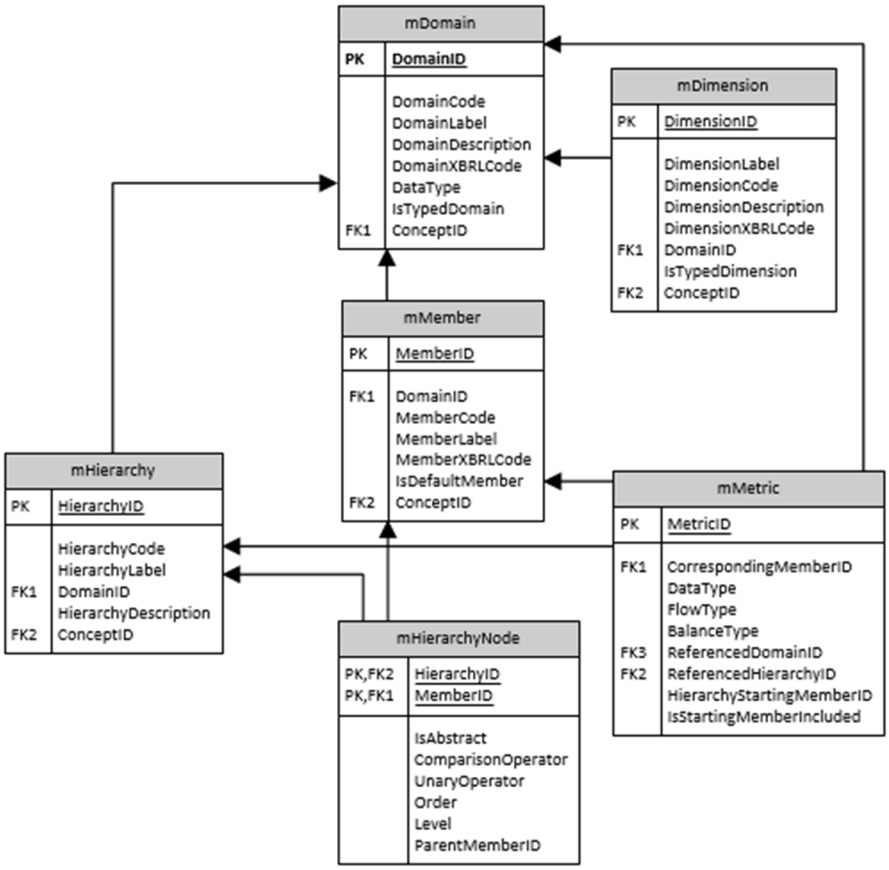


Moreover, owners and languages are also concepts (for example language names can be translated in various languages, properties of owners can modified, etc.).

X.4.2 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).



X.4.3 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.

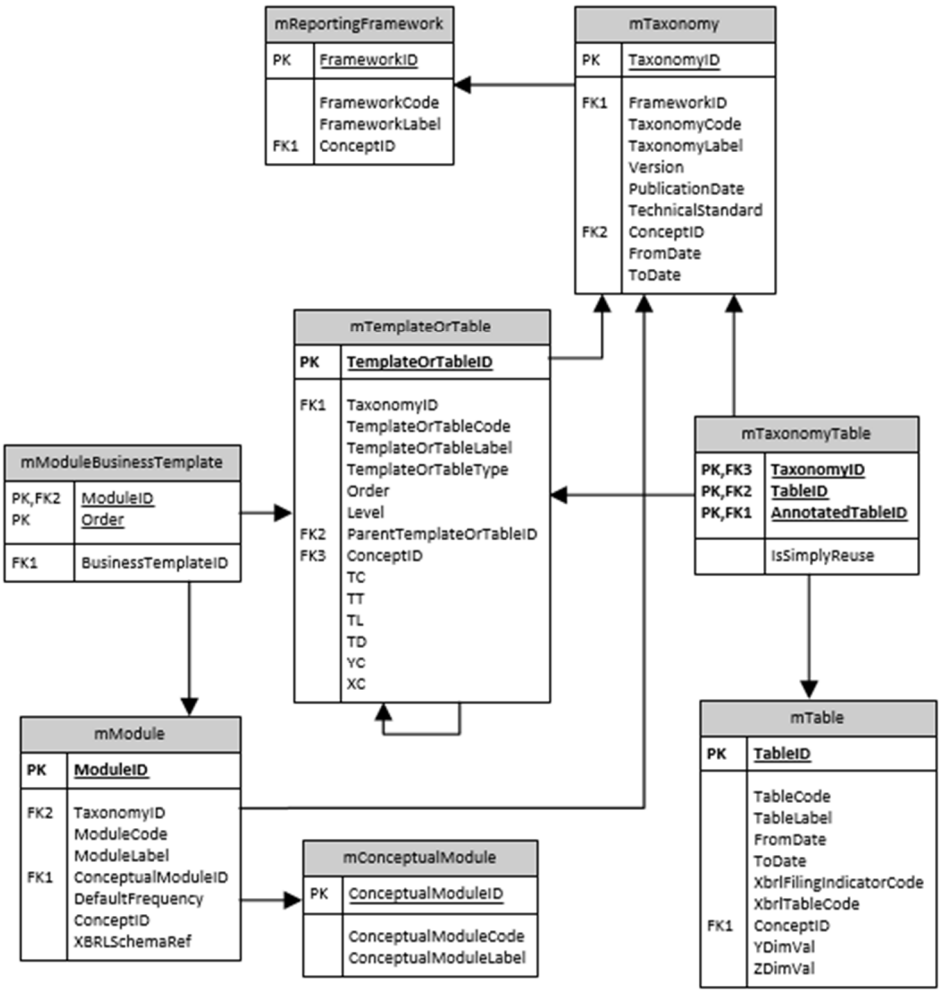
Frameworks are versioned as taxonomies (mTaxonomy) that identify data sets required at the particular moment of time (previous, current and potentially also future versions).

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")[3].

[3] For details see section "Representation of templates and tables".

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).

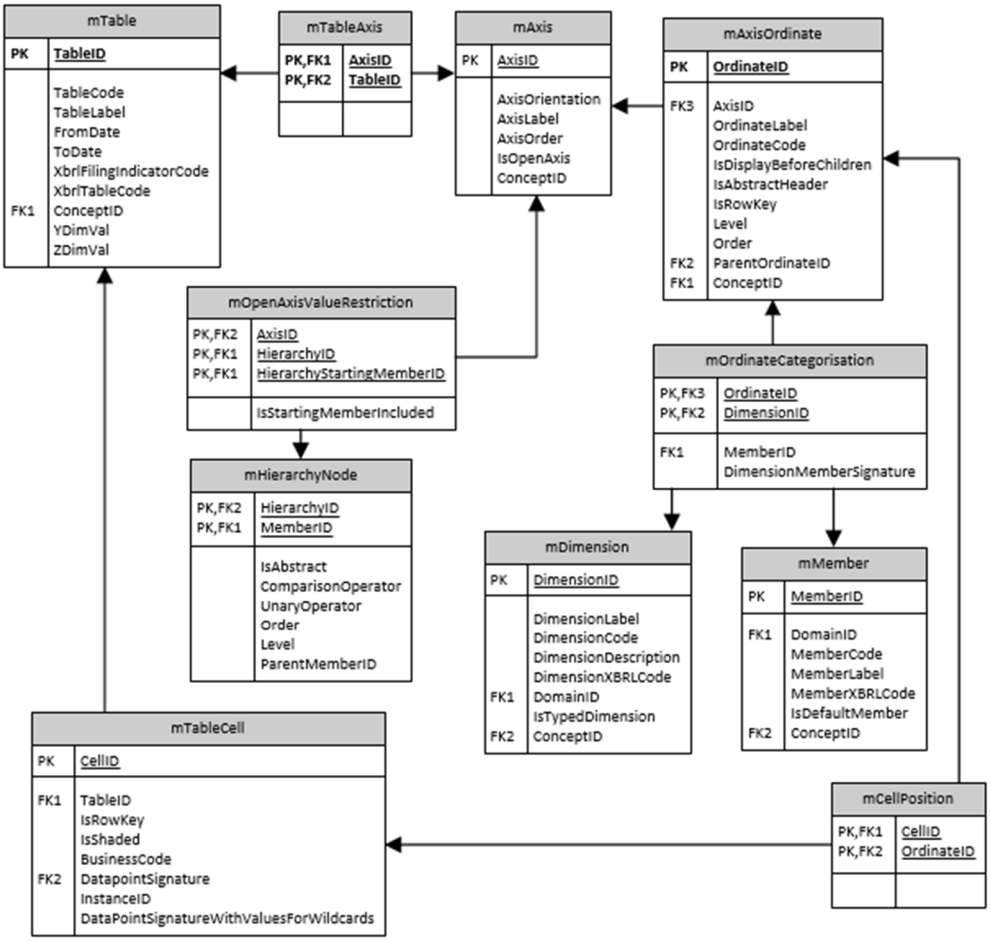
Taxonomies may define numerous templates. Depending on reporting period or type of a report not all templates must be filed under certain reporting scenario. Therefore modules (mModule) gather templates that are shall be submitted in one set (mModuleBusinessTemplate).



X.4.4 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

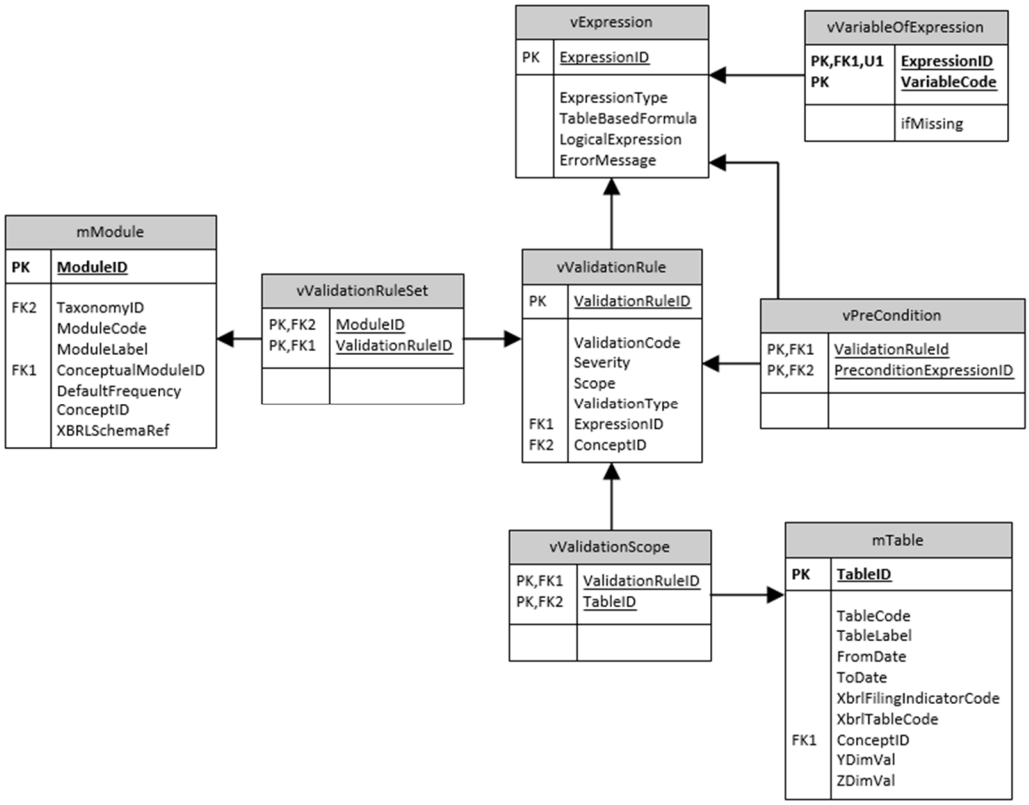
occur on the intersection of axis ordinates (mCellPosition). Each cell represents none (grey shaded/criss-crossed), one or many data points (mTableCell).



X.4.5 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.



X.5 Derivation of supportive entities or their attributes

As explained in the section "Source (input) information and database population mechanisms", part of the database content that is used by the T4U applications is not defined explicitly in the input materials and needs to be derived. This process has been automated and the algorithms are described below.

X.5.1 Representation of templates and tables

Information requirements of the Solvency II are divided in multiple thematic/subject subsets.

The first level of division is groups of templates represented in the database as TemplateOrTable.TemplateOrTableType = "TemplatesGroup". These are for example collections of templates related to balance sheet, technical provisions, details on assets, etc. The second level are individual templates (TemplateOrTable.TemplateOrTableType = "Template"), e.g. "Balance sheet", "Assets D1, Investement Data, Portfolio list". The next level is template variants (TemplateOrTable.TemplateOrTableType = "TemplateVariant") where templates are differentiated based on their application for certain reporting scenario (e.g. solo/group, frequency of reporting, etc.). Template variants may consist of one or more business tables (TemplateOrTable.TemplateOrTableType = "BusinessTable"), i.e. graphical tabular views as defined in the legal regulations (i.e. Business Logs and Business Templates in case of Solvency II or ITS for COREP and FINREP). These in turn in some cases must be normalized in the process of annotation with the DPM properties (as a consequence of identification of functional relationships within the table or the same property represented in headers of rows and columns). This last level of annotated templates has TemplateOrTable.TemplateOrTableType = "AnnotatedTemplate". These are the normalized table views that correspond to tables in a given version of taxonomy (mTaxonomyTable entity).

The hierarchy of template and table levels described above and resembled in mTemplateOrTable entity of the database is also reflected in the codes used in the annotated templates of Solvency II which are constructed according to the following pattern: S.NN.MM.XX.YY, where

- NN corresponds to a group of templates,
- MM identifies particular template,
- XX is a template variant,
- YY represents annotated table.

Business tables are not included in this codification as they are not reflected in the annotate templates. Information on this level shall be added in either the annotated templates or the process of population of the database from the annotated templates.

Due to different business requirements (i.e. dissimilar arrangement of information requirements in the EBA ITS) the EBA DPM MS Access database structure for these artefacts was not follow by the T4U database model. As a result, in order to support the EBA ITS (which is one of the aims of the T4U) a migration mechanism for these entities is defined.

T4U database mTemplateOrTable entity includes information on the EBA Template, Table, TableGroup and TableVersion entities which, in contrast to the T4U database, apart from the last two are not all versioned together with the taxonomy (i.e. there is no reference to TaxonomyID from these entities). As a result, entries from these entities need to be duplicated according to the EBA many-to-many relationships entities (TableGroupTemplates and TaxononomyTableVersion). This requires including new rows with new IDs and impacts other tables such as mModuleBusinessTemplate (which is not 1:1 representation of the EBA mModuleTableOrGroup as it links to mTemplateOrTable.TemplateOrTableID with TemplateOrTableType = “TemplateVersion”).

In the first step the EBA TableGroup is migrated. Entries from this entity have TemplateOrTable.TemplateOrTableType set to “TemplateGroup” and mTemplateOrTable.Level equal 1.

The next step is to migrate EBA Template entity. In this case mTemplateOrTable.TemplateOrTableType is set to “Template” and Level equal 2. The relation between table groups and templates is expressed by mTemplateOrTable.ParentTemplateOrTable as in the EBA TemplateGroupTemplates (hence the need to duplicate these EBA templates which are reused bye EBA table groups).

The following step is repeat migration of EBA Templates once again, this time with mTemplateOrTable.TemplateOrTableType set to “TemplateVariant” and Level equal 3 (the order is always 1). These entries should be subsequently linked in mModuleBusinessTemplate.

In the last step the EBA Table and Table Version are migrated. This time mTemplateOrTable.TemplateOrTableType is set to “BusinessTable”/”AnnotatedTable” and Level is 4 or 5 respectively. As in the previous cases, many-to-many EBA TaxonomyTableVersion needs to be used for this purpose.

mTable (corresponding to the EBA TableVersion) links to mTemplateOrTable entries with TemplateOrTableType set to ”AnnotatedTable” via mTaxonomyTable.

X.5.2 Cell/Data point signatures

Each single piece of information requirements corresponds to a data point (or a property of a data point in case for example of row keys in open tables). Data points are described in the data centric manner by identification of a metric and dimension member pairs. From the form-centric perspective, data points are identified by reportable table cells. To combine the two, each reportable table cell is given the signature of a data point or a set of data points in case of tables with z-axis referring to hierarchies of members. This signature is generated based on the properties already included in the database, in particular position of a cell in the table (mCellPosition) by reference to table axis ordinates and dimensional properties hidden behind these ordinates (mOrdinateCategarisation). Data point signature is constructed according to the following pattern:

MET({XBRL code of a metric}|{XBRL code of dimension|({XBRL code of a member}).

XBRL code consist of a recommended prefix followed by a colon and local name/code of an element (member corresponding to a metric, dimension or member). All these components can be accessed by queries as they are defined in the source (input) materials used to populate its content.

When more than one dimension described a data point, dimension member pairs are sorted alphabetically in ascending order based on dimension name.

In case one of the ordinates determining position of a table cell belongs to an open axis or axis reusing the hierarchy of domain members, the {XBRL code of a member} component is replaced by a wildcard: * (asterisk).

This first step of generating DataPointSignatures is to define signatures of ordinate categorisations (mOrdinateCategorisation.DimensionMemberSignature) according to the following query:

```
SELECT oc.OrdinateID,
      CASE
            WHEN dom.DomainLabel = 'Metric' or dom.DomainLabel = 'Metrics (HD)' or
dom.DomainLabel = 'Metrics (MD)'
          THEN 'MET(' || own.OwnerPrefix || '_met:' || mem.MemberCode || ')'
          ELSE CASE
          WHEN oc.MemberID = '999' or oc.MemberID = '9999'
          THEN dim.DimensionXBRLCode || '(*)'
          ELSE dim.DimensionXBRLCode || '(' || mem.MemberXBRLCode || ')'
      END
      END AS DimensionMemberSignature
  FROM mOrdinateCategorisation oc
      INNER JOIN mDimension dim
            ON dim.DimensionID = oc.DimensionID
      INNER JOIN mMember mem
            ON mem.MemberID = oc.MemberID
      LEFT OUTER JOIN mConcept con
                ON con.ConceptID = mem.ConceptID
      LEFT OUTER JOIN mOwner own
                ON own.OwnerID = con.OwnerID
      INNER JOIN mDomain dom
            ON dom.DomainID = dim.DomainID;
```

Using this information, population of mTableCell.DatapointSignature should be conducted basing on the following query:

```
SELECT tc.CellID,
     oc.DimensionMemberSignature
 FROM mTableCell tc
     INNER JOIN mCellPosition cp
          ON cp.CellID = tc.CellID
     INNER JOIN mOrdinateCategorisation oc
          ON oc.OrdinateID = cp.OrdinateID
ORDER BY cp.CellID,
        oc.DimensionMemberSignature;
```

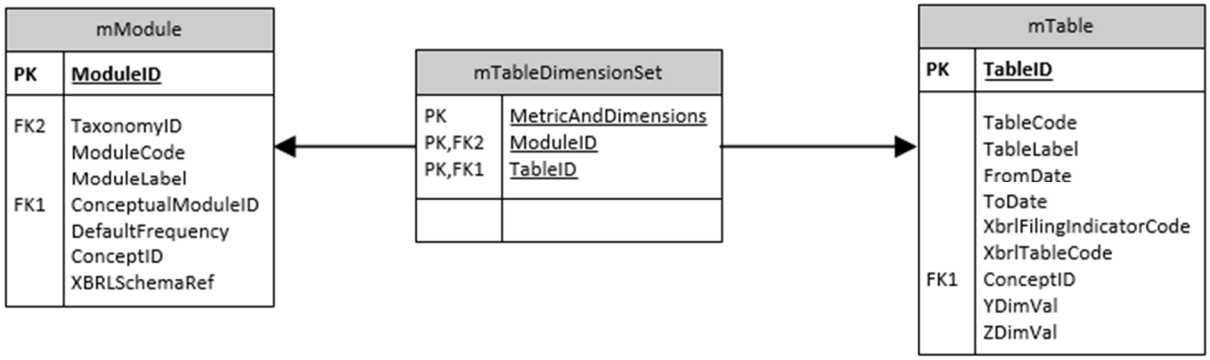
The results of this query for a given cell need to be concatenated in alphabetical order with pipe ('|') as the separator. For example, the following exemplary results for two cells:

tc.CellID	oc.DimensionMemberSignature
1	MET(eba_met:mi76)
1	eba_dim:BAS(eba_BA:x11)
1	eba_dim:MCU(eba_MC:x274)
1	eba_dim:MCY(eba_MC:x374)
1	eba_dim:OFS(eba_OF:x9)
2	MET(eba_met:mi76)
2	eba_dim:BAS(eba_BA:x11)
2	eba_dim:INV(eba_PL:x50)
2	eba_dim:MCU(eba_MC:x131)
2	eba_dim:MCY(eba_MC:x367)
2	eba_dim:OFS(eba_OF:x9)
2	eba_dim:RPR(eba_RP:x2)

would results in DataPointSignature code *MET(eba_met:mi76)|eba_dim:BAS(eba_BA:x11)|eba_dim:MCU(eba_MC:x274)|eba_dim:MCY(eba_MC:x374)|eba_dim:OFS(eba_OF:x9)* for tc.CellID = *1* and *MET(eba_met:mi76)|eba_dim:BAS(eba_BA:x11)|eba_dim:INV(eba_PL:x50)|eba_dim:MCU(eba_MC:x131)|eba_dim:MCY(eba_MC:x367)|eba_dim:OFS(eba_OF:x9)|eba_dim:RPR(eba_RP:x2)* for tc.CellID = *2*.

X.5.3 Entities supporting association of facts with tables

Some entities or attributes of entities in the model of the T4U database are created to be used when associating facts with tables to which they potentially belong. This is to accelerate performance of queries or applications working with the database that defines and contains thousands of cells and millions of potential data points and facts. One of such entities is mTableDimensionSet identifying metrics and dimensions (without specifying explicitly the members) that describe data points belonging to each table. Using this information, it should be relatively easy and fast to assign a fact (reported in an instance document referring to a known module) to a table where is would potentially appear. For similar purpose the mTable entity contain YDimVal and ZDimVal attributes.



mTableDimensionSet entity lists distinct/unique metric and dimension sets used by any cell of a given table. The following query provides a list of tables together with cells and their signatures:

```
SELECT t.TableID,
      cp.CellID,
      CASE
            WHEN( d.DimensionLabel = 'Metric' OR d.DimensionLabel = 'Metric (HD)' OR
d.DimensionLabel = 'Metric (MD)' ) THEN 'MET(' || o.OwnerPrefix || '_met:' || m.MemberCode || ')'
          ELSE d.DimensionXBRLCode
      END AS MetricAndDimensions
 FROM mTable t
      INNER JOIN mTableCell tc
            ON tc.TableID = t.TableID
      INNER JOIN mCellPosition cp
            ON cp.CellID = tc.CellID
      INNER JOIN mOrdinateCategorisation oc
            ON oc.OrdinateID = cp.OrdinateID
      INNER JOIN mDimension d
            ON d.DimensionID = oc.DimensionID
      INNER JOIN mMember m
            ON m.MemberID = oc.MemberID
      LEFT OUTER JOIN mConcept c
                ON c.ConceptID = m.ConceptID
      LEFT OUTER JOIN mOwner o
                ON o.OwnerID = c.OwnerID
ORDER BY t.TableID,
             cp.CellID,
        CASE WHEN( d.DimensionLabel = 'Metric' OR d.DimensionLabel = 'Metric (HD)' OR
d.DimensionLabel = 'Metric (MD)' )
       THEN 'MET(' || o.OwnerPrefix || '_met:' || m.MemberCode || ')'
       ELSE d.DimensionXBRLCode END;
```

Table below presents the exemplary results of the query above:

t.TableID	cp.CellID	MetricAndDimensions
486	62094	MET(eba_met:mi53)
486	62094	eba_dim:BAS
486	62094	eba_dim:MCY
486	62095	MET(eba_met:mi53)
486	62095	eba_dim:BAS
486	62095	eba_dim:CPS
486	62095	eba_dim:MCY
486	62096	MET(eba_met:mi53)
486	62096	eba_dim:APL
486	62096	eba_dim:BAS
486	62096	eba_dim:MCY
486	62097	MET(eba_met:mi53)
486	62097	eba_dim:APL
486	62097	eba_dim:BAS
486	62097	eba_dim:MCY
486	62098	MET(eba_met:mi53)
486	62098	eba_dim:APL
486	62098	eba_dim:BAS
486	62098	eba_dim:MCY
486	62099	MET(eba_met:mi53)
486	62099	eba_dim:APL
486	62099	eba_dim:BAS
486	62099	eba_dim:MCY
486	62100	MET(eba_met:mi53)
486	62100	eba_dim:APL
486	62100	eba_dim:BAS
486	62100	eba_dim:MCY

The first cell in the results (cp.CellID = 62094) effects in the following entry to mTableDimensionSet.MetricAndDimensions: *MET(eba_met:mi53)|eba_dim:BAS|eba_dim:MCY* (alphabetical concatenation based on dimension code with pipe as a separator). The next cell (cp.CellID = 62095) results in another row in mTableDimensionSet for a given table (t.TableID = 486) with MetricAndDimensions = *MET(eba_met:mi53)|eba_dim:BAS| eba_dim:CPS|eba_dim:MCY*. The following one (cp.CellID = 62096) is another row, this time mTableDimensionSet.MetricAndDimensions = *MET(eba_met:mi53)|eba_dim:APL|eba_dim:BAS|eba_dim:MCY*.

The next results i.e. cp.CellID = 62097, 62098, 62099, 62100 **DO NOT** result in new entries as they define the same MetricAndDimensions as already added by cp.CellID = 62096. And so on.

The above needs to be combined with information on Module (mTableDimensionSet.ModuleID) that the table can be reported for (as the starting point for the use of this data is an instance document that links to a module). A table can be associated to more than one module. This following query should provide a full list of modules for tables:

```
SELECT t.TableID AS Tables,
    --tot.TemplateOrTableID AS AnnotatedTables,
    --tot2.TemplateOrTableID AS BusinessTables,
    --tot3.TemplateOrTableID AS TemplateVariants,
    mbt.ModuleID AS Modules
 FROM mTable t
    INNER JOIN mTaxonomyTable tt
        ON tt.TableID = t.TableID
    INNER JOIN mTemplateOrTable tot
        ON tot.TemplateOrTableID = tt.AnnotatedTableID
    INNER JOIN mTemplateOrTable tot2
        ON tot2.TemplateOrTableID = tot.ParentTemplateOrTableID
    INNER JOIN mTemplateOrTable tot3
        ON tot3.TemplateOrTableID = tot2.ParentTemplateOrTableID
    INNER JOIN mModuleBusinessTemplate mbt
        ON mbt.BusinessTemplateID = tot3.TemplateOrTableID
  Order by t.TableID, mbt.ModuleID
```

As one can see from the results:

Tables	Modules
486	9
486	21

t.TableID 486 is reported for ModuleID 9 and 21. Therefore the exemplary entries in the mTableDimensionSet would be as follows:

MetricAndDimensions	TableID	ModuleID
MET(eba_met:mi53)|eba_dim:BAS|eba_dim:MCY	486	9
MET(eba_met:mi53)|eba_dim:BAS| eba_dim:CPS|eba_dim:MCY	486	9
MET(eba_met:mi53)|eba_dim:APL|eba_dim:BAS|eba_dim:MCY	486	9
MET(eba_met:mi53)|eba_dim:BAS|eba_dim:MCY	486	21
MET(eba_met:mi53)|eba_dim:BAS| eba_dim:CPS|eba_dim:MCY	486	21
MET(eba_met:mi53)|eba_dim:APL|eba_dim:BAS|eba_dim:MCY	486	21

mTable.YDimVal stores information about key dimensions in open tables. Its values can be populated using the following query:

```
SELECT t.TableID,
    oc.DimensionMemberSignature AS YDimVal
 FROM mTable t
    INNER JOIN mTableAxis ta
        ON ta.TableID = t.TableID
```

```
      INNER JOIN mAxis a
            ON a.AxisID = ta.AxisID
      INNER JOIN mAxisOrdinate ao
            ON ao.AxisID = ta.AxisID
      INNER JOIN mOrdinateCategorisation oc
            ON oc.OrdinateID = ao.OrdinateID
WHERE a.AxisOrientation = 'Y'
      AND
      a.IsOpenAxis = 1
ORDER BY t.TableID,
         oc.DimensionMemberSignature;
```

If more than one result is returned for a table then the values need to be sorted alphabetically and concatenated. E.g. given these are the result:

t.TableID	YDimVal
553	eba_dim:LEC(*)
560	eba_dim:OGR(*)
566	eba_dim:OGR(*)
570	eba_dim:SRN(*)
585	eba_dim:INC(*)
586	eba_dim:INC(*)
593	eba_dim:LEC(*)
594	eba_dim:LEC(*)
594	eba_dim:STC(*)
602	eba_dim:GCC(*)
602	eba_dim:INC(*)
607	eba_dim:INC(*)
608	eba_dim:GCC(*)
608	eba_dim:INC(*)
658	eba_dim:LEC(*)

for table 608, the value to be stored in YDimVal is *eba_dim:GCC(*)|eba_dim:INC(*)*.

mTable.ZDimVal stores information about dimensions and members appearing on the z-axes of a table. It can be populated using the following query:

```
SELECT distinct t.TableID,
      oc.DimensionMemberSignature AS ZDimVal
  FROM mTable t
      INNER JOIN mTableAxis ta
            ON ta.TableID = t.TableID
      INNER JOIN mAxis a
            ON a.AxisID = ta.AxisID
      INNER JOIN mAxisOrdinate ao
            ON ao.AxisID = ta.AxisID
      INNER JOIN mOrdinateCategorisation oc
            ON oc.OrdinateID = ao.OrdinateID
WHERE a.AxisOrientation = 'Z'
 ORDER BY t.TableID,
         oc.DimensionMemberSignature;
```

Apart from the alphabetical order concatenation according to dimension codes (as above for YDimVal) there is one more step in the process for population of ZDimVal. An example is table 554 for which the results are as follows:

t.TableID	ZDimVal
554	eba_dim:APR(eba_AP:x42)
554	eba_dim:CPS(eba_CT:x13)
554	eba_dim:CPS(eba_CT:x15)
554	eba_dim:CPS(eba_CT:x16)

554	eba_dim:CPS(eba_CT:x19)
554	eba_dim:CPS(eba_CT:x2)
554	eba_dim:CPS(eba_CT:x22)
554	eba_dim:CPS(eba_CT:x51)
554	eba_dim:CPS(eba_CT:x6)
554	eba_dim:EXC(eba_EC:x1)
554	eba_dim:EXC(eba_EC:x12)
554	eba_dim:EXC(eba_EC:x13)
554	eba_dim:EXC(eba_EC:x14)
554	eba_dim:EXC(eba_EC:x15)
554	eba_dim:EXC(eba_EC:x16)
554	eba_dim:EXC(eba_EC:x17)
554	eba_dim:EXC(eba_EC:x18)
554	eba_dim:EXC(eba_EC:x19)
554	eba_dim:EXC(eba_EC:x20)
554	eba_dim:EXC(eba_EC:x21)
554	eba_dim:EXC(eba_EC:x22)
554	eba_dim:EXC(eba_EC:x23)
554	eba_dim:EXC(eba_EC:x24)
554	eba_dim:EXC(eba_EC:x25)
554	eba_dim:EXC(eba_EC:x26)

In this case, i.e. when the same dimension is returned for a given table more than once (here CPS and EXC) with different domain members, in the concatenated string members code is replaced by asterisk. The resulting entry in the mTable.ZDimVal for TableID 554 should be then: *eba_dim:APR(eba_AP:x42)|eba_dim:CPS(*)|eba_dim:EXC(*)*.

X.6 Structures for storage of data according to the DPM properties

X.6.1 General model

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).

X.6.2 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.

X.6.2.1 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	DATETIM	Date and time of instance creation (load or last

	E	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.

X.6.2.2 dProcessingContext

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

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.

X.6.2.3 dProcessingFact

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

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.

X.6.2.4 dFact

Reported facts.

Attribute	Type	Description
InstanceID	INTEGER	Instance document in which the fact was reported. Points to dInstance.InstanceID.
DataPointSignature	TEXT	Same as mTableCell.DataPointSignature.
DataPointSignatureWithValuesForWildcards	TEXT	Same as mTableCell.DataPointSignature but with wildcards replaced with actually reported values (for typed dimensions and other "as open"

		axis).
Unit	TEXT	Content of unit measures taking into account possible use of 'xbrli:divide' element. Usually: iso4217:EUR or other currency code for monetary and xbrli:pure for other numeric facts.
Decimals	TEXT	Accuracy 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. Time is currently not supported.
BooleanValue	BOOLEAN	Value of a fact if boolean.
TextValue	TEXt	Value of a fact if not numeric, date or boolean.

X.6.2.5 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".

X.6.2.6 dAvailableTable

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

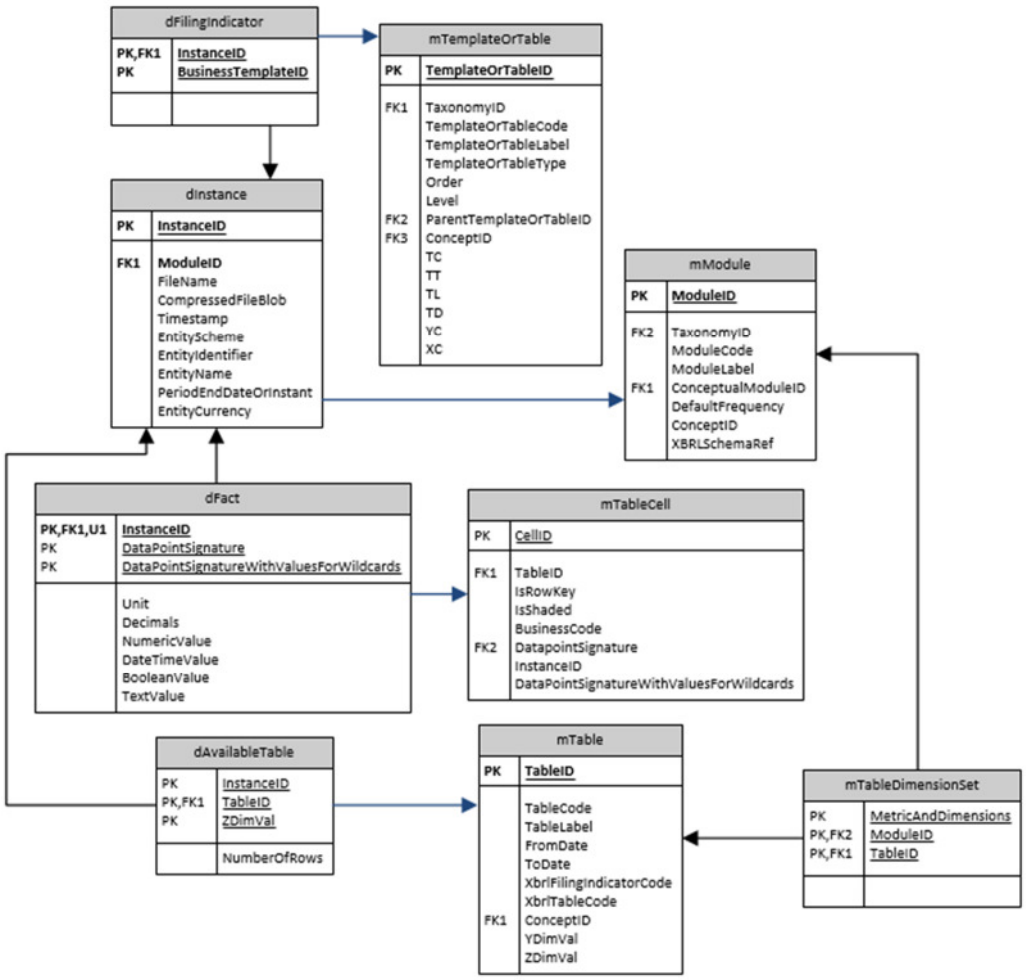
Attribute	Type	Description
InstanceID	INTEGER	Instance document in which the facts were reported (dInstance.InstanceID).
TableID	INTEGER	Table where facts could belongs to (mTable.TableID).
ZDimVal	TEXT	Values of Z axes properties (metrics, dimension member pairs, typed dimension values) of a table for 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.

X.6.3 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 points to a signature of a data point (as represented by table cells where this fact belongs). In case when a table cell represents more than one data point, the concrete and full representation for a fact is provided in DataPointSignatureWithValuesForWildcards column.

Information on what tables were potentially reported is generated in dAvailableTable (based on metadata from mTableDimensionSet). This information is helpful in further processing and use of data (e.g. moving to classic relational structures as explained in the next chapter of this document).



X.7 Tables for classic relational data storage and mapping to DPM metadata properties

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.

X.7.1 General idea, goals and alternatives considered

As explained in the introduction to this document (see chapter on "

Database components"), 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.

X.7.2 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
...	...	...	...

X.7.3 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 as explained in chapter X.3 of this document. 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

X.7.4 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)

X.7.5 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.

X.8 Application interface information

X.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.

X.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.

X.8.2.1 aApplication

Tool for Undertaking applications (e.g. Windows Forms, Excel Add-In).

Attribute	Type	Description
ApplicationID	INTEGER	Artificial ID.
ApplicationName	TEXT	Tool name, e.g. Excel Add-In, Windows Forms, …)

X.8.2.2 aInterfaceComponent

Translation of interface components (e.g. buttons, messages, etc.) into EU languages. An minimum English will be provided. Tool for undertaking applications (Windows Forms, Excel Add-In, etc) 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.

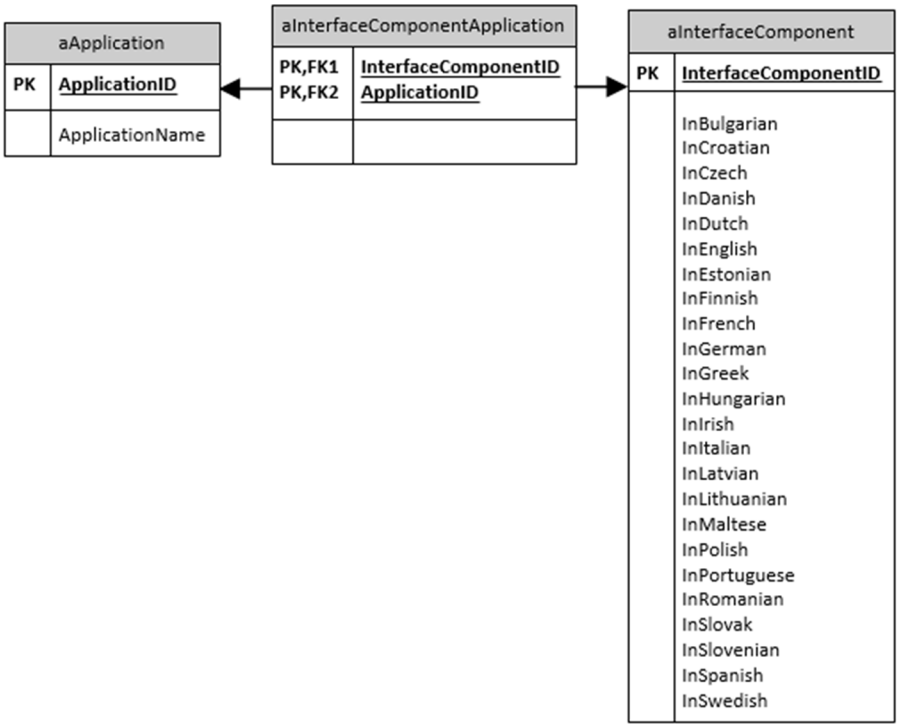
X.8.2.3 aInterfaceComponentApplication

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

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

X.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.


aApplication
PK
ApplicationID
ApplicationName
aInterfaceComponentApplication
PK,FK1
PK,FK2
InterfaceComponentID
ApplicationID
aInterfaceComponent
PK
InterfaceComponentID
InBulgarian
InCroatian
InCzech
InDanish
InDutch
InEnglish
InEstonian
InFinnish
InFrench
InGerman
InGreek
InHungarian
InIrish
InItalian
InLatvian
InLithuanian
InMaltese
InPolish
InPortuguese
InRomanian
InSlovak
InSlovenian
InSpanish
InSwedish

Annex 1: Relation between EBA DPM MS Access database and T4U entities for information requirements and validation rules metadata description

The comparison table presented below identifies the links and discrepancies between the EBA DPM MS Access database (version 2.1.0 PC) and T4U SQLite DPMdb (version 7).

Where rows are aligned, the content of the databases is identical. For missing rows the content does not exist in the other database or is defined in a different place (see description of entities and relationships in DPM, annotated template and validation rules metadata "Entities" and "Relationships" sections of this document.

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
Axis	-	mAxis	-
Axis	AxisID	mAxis	AxisID
Axis	AxisLabel	mAxis	AxisLabel
Axis	AxisOrder	mAxis	AxisOrder
Axis	AxisOrientation	mAxis	AxisOrientation
Axis	IsOpenAxis	mAxis	IsOpenAxis
Axis	ConceptID	mAxis	ConceptID
Axis	TableVID		
AxisOrdinate	-	mAxisOrdinate	-
AxisOrdinate	AxisID	mAxisOrdinate	AxisID
AxisOrdinate	OrdinateID	mAxisOrdinate	OrdinateID
AxisOrdinate	OrdinateLabel	mAxisOrdinate	OrdinateLabel
AxisOrdinate	OrdinateCode	mAxisOrdinate	OrdinateCode
AxisOrdinate	DisplayBeforeChildren	mAxisOrdinate	IsDisplayBeforeChildren
AxisOrdinate	IsAbstractHeader	mAxisOrdinate	IsAbstractHeader
AxisOrdinate	IsRowKey	mAxisOrdinate	IsRowKey
AxisOrdinate	Level	mAxisOrdinate	Level
AxisOrdinate	Order	mAxisOrdinate	Order
AxisOrdinate	ParentOrdinateID	mAxisOrdinate	ParentOrdinateID
AxisOrdinate	ConceptID	mAxisOrdinate	ConceptID
AxisOrdinate	CategorisationKey		
AxisOrdinate	DataSign		
AxisOrdinate	Path		
		mTableAxis	-
		mTableAxis	AxisID
		mTableAxis	TableID
BalanceType	-		
BalanceType	BalanceTypeCode		
BalanceType	BalanceTypeID		
BalanceType	BalanceTypeLabel		
BaseMember	-		
BaseMember	BalanceTypeID		
BaseMember	MemberID		
CellPosition	-	mCellPosition	-
CellPosition	CellID	mCellPosition	CellID
CellPosition	OrdinateID	mCellPosition	OrdinateID
CellReference	-	mCellReference	-
CellReference	CellID	mCellReference	CellID
CellReference	ReferenceID	mCellReference	ReferenceID
Concept	-	mConcept	
Concept	ConceptID	mConcept	ConceptID
Concept	OwnerID	mConcept	OwnerID
Concept	ConceptType	mConcept	ConceptType
Concept	CreationDate	mConcept	CreationDate
Concept	FromDate	mConcept	FromDate
Concept	ToDate	mConcept	ToDate
Concept	ModificationDate	mConcept	ModificationDate
ConceptReference	-	mConceptReference	-
ConceptReference	ConceptID	mConceptReference	ConceptID
ConceptReference	ReferenceID	mConceptReference	ReferenceID

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
ConceptTranslation	-	mConceptTranslation	-
ConceptTranslation	ConceptID	mConceptTranslation	ConceptID
ConceptTranslation	LanguageID	mConceptTranslation	LanguageID
ConceptTranslation	OwnerID	mConceptTranslation	OwnerID
ConceptTranslation	Text	mConceptTranslation	Text
ConceptTranslation	FieldName		
ConceptualModule	-	mConceptualModule	-
ConceptualModule	ConceptualModuleID	mConceptualModule	ConceptualModuleID
ConceptualModule	ConceptualModuleCode	mConceptualModule	ConceptualModuleCode
ConceptualModule	ConceptualModuleLabel	mConceptualModule	ConceptualModuleLabel
ContextDefinition	-		
ContextDefinition	ContextID		
ContextDefinition	DimensionID		
ContextDefinition	MemberID		
ContextOfDataPoints	-		
ContextOfDataPoints	ContextID		
ContextOfDataPoints	ContextKey		
ContextOfDataPoints	XbrlContextKey		
DataPoint	-		
DataPoint	DataPointID		
DataPoint	PeriodOffsetID		
DataPoint	RelatedDataPointID		
DataPointVersion	-		
DataPointVersion	CategorisationKey		
DataPointVersion	CategorisationKeyWithoutDefaults		
DataPointVersion	ContextID		
DataPointVersion	DataPointID		
DataPointVersion	DataPointVID		
DataPointVersion	FromDate		
DataPointVersion	MetricID		
DataPointVersion	ToDate		
DataPointVersionVariable	-		
DataPointVersionVariable	ExpressionID		
DataPointVersionVariable	VariableCode		
DataPointVersionVariable	DataPointVID		
DataType	-		
DataType	DataTypeCode		
DataType	DataTypeID		
DataType	DataTypeLabel		
Dimension	-	mDimension	-
Dimension	DimensionID	mDimension	DimensionID
Dimension	DimensionLabel	mDimension	DimensionLabel
Dimension	DimensionCode	mDimension	DimensionCode
Dimension	DimensionDescription	mDimension	DimensionDescription
Dimension	DimensionXbrlCode	mDimension	DimensionXbrlCode
Dimension	DomainID	mDimension	DomainID
Dimension	IsTyped	mDimension	IsTypedDimension
Dimension	ConceptID	mDimension	ConceptID
Dimension	IsImpliedIfNotExplicitlyModelled		
DimensionalCoordinate	-		
DimensionalCoordinate	DimensionID		
DimensionalCoordinate	MemberID		
Domain	-	mDomain	-
Domain	DomainID	mDomain	DomainID
Domain	DomainLabel	mDomain	DomainLabel
Domain	DomainCode	mDomain	DomainCode
Domain	DomainDescription	mDomain	DomainDescription
Domain	DomainXbrlCode	mDomain	DomainXbrlCode
Domain	DataTypeID	mDomain	DataType
Domain	IsTypedDomain	mDomain	IsTypedDomain
Domain	ConceptID	mDomain	ConceptID
Domain	IsExternalRefData		
Domain	ReferenceDataSource		
Expression	-	vExpression	-
Expression	ExpressionID	vExpression	ExpressionID

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
Expression	ErrorMessage	vExpression	ErrorMessage
Expression	ExpressionType	vExpression	ExpressionType
Expression	LogicalExpression	vExpression	LogicalExpression
Expression	TableBasedFormula	vExpression	TableBasedFormula
ExpressionScope	-		
ExpressionScope	ExpressionID		
ExpressionScope	OrdinateID		
FlowType	-		
FlowType	FlowTypeID		
FlowType	FlowTypeCode		
FlowType	FlowTypeLabel		
Hierarchy	-	mHierarchy	-
Hierarchy	HierarchyID	mHierarchy	HierarchyID
Hierarchy	HierarchyCode	mHierarchy	HierarchyCode
Hierarchy	HierarchyLabel	mHierarchy	HierarchyLabel
Hierarchy	HierarchyDescription	mHierarchy	HierarchyDescription
Hierarchy	ConceptID	mHierarchy	ConceptID
Hierarchy	DomainID	mHierarchy	DomainID
HierarchyNode	-	mHierarchyNode	-
HierarchyNode	HierarchyID	mHierarchyNode	HierarchyID
HierarchyNode	MemberID	mHierarchyNode	MemberID
HierarchyNode	IsAbstract	mHierarchyNode	IsAbstract
HierarchyNode	Level	mHierarchyNode	Level
HierarchyNode	Order	mHierarchyNode	Order
HierarchyNode	ParentMemberID	mHierarchyNode	ParentMemberID
HierarchyNode	UnaryOperator	mHierarchyNode	UnaryOperator
HierarchyNode	ComparisonOperator	mHierarchyNode	ComparisonOperator
HierarchyNode	ParentHierachyID		
HierarchyNode	Path		
HierarchyScope	-		
HierarchyScope	HierarchyID		
HierarchyScope	TableVID		
HierarchyValidationRule	-		
HierarchyValidationRule	HierarchyID		
HierarchyValidationRule	MemberID		
HierarchyValidationRule	ValidationID		
InstanceLevelConcept	-		
InstanceLevelConcept	ConceptLabel		
InstanceLevelConcept	InstanceLevelConceptID		
InstanceLevelDataPoint	-		
InstanceLevelDataPoint	DataPointID		
InstanceLevelDataPoint	InstanceLevelConceptID		
InstanceLevelDimension	-		
InstanceLevelDimension	DimensionID		
InstanceLevelDimension			
InstanceLevelDimension	InstanceLevelConceptID		
Language	-	mLanguage	-
Language	LanguageID	mLanguage	LanguageID
Language	LanguageName	mLanguage	LanguageName
Language	EnglishName	mLanguage	EnglishName
Language	IsoCode	mLanguage	IsoCode
		mLanguage	ConceptID
Member	-	mMember	-
Member	MemberID	mMember	MemberID
Member	MemberLabel	mMember	MemberLabel
Member	MemberCode	mMember	MemberCode
Member	MemberXbrlCode	mMember	MemberXbrlCode
Member	DomainID	mMember	DomainID
Member	IsDefaultMember	mMember	IsDefaultMember
Member	ConceptID	mMember	ConceptID
Metric	-	mMetric	-
Metric	MetricID	mMetric	MetricID
		mMetric	CorrespondingMemberID
Metric	DataTypeID	mMetric	DataType
Metric	FlowTypeID	mMetric	FlowType
		mMetric	BalanceType
Metric	CodeDomainID	mMetric	ReferencedDomainID

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
Metric	CodeSubdomainID	mMetric	ReferencedHierarchyID
		mMetric	HierarchyStartingMemberID
		mMetric	IsStartingMemberIncluded
Metric	StringListID		
MetricVariable	-		
MetricVariable	ExpressionID		
MetricVariable	MetricID		
MetricVariable	VariableCode		
Module	-	mModule	-
Module	ModuleID	mModule	ModuleID
Module	ModuleCode	mModule	ModuleCode
Module	ModuleLabel	mModule	ModuleLabel
Module	ConceptualModuleID	mModule	ConceptualModuleID
		mModule	DefaultFrequency
Module	TaxonomyID	mModule	TaxonomyID
Module	XbrlSchemaRef	mModule	XbrlSchemaRef
Module	ConceptID	mModule	ConceptID
ModuleDataPointRestriction	-		
ModuleDataPointRestriction	ModuleID		
ModuleDataPointRestriction	InstanceLevelConceptID		
ModuleDataPointRestriction	DataPointID		
ModuleDataPointRestriction	RequiredMemberID		
ModuleDataPointRestriction	RequiredValue		
ModuleDimensionImpliedValue	-		
ModuleDimensionImpliedValue	ModuleID		
ModuleDimensionImpliedValue	InstanceLevelConceptID		
ModuleDimensionImpliedValue	ImpliedMemberID		
ModuleDimensionImpliedValue	ImpliedValue		
ModuleTableOrGroup	-	mModuleBusinessTemplate	-
ModuleTableOrGroup	ModuleID	mModuleBusinessTemplate	ModuleID
ModuleTableOrGroup	TableGroupID	mModuleBusinessTemplate	BusinessTemplateID
ModuleTableOrGroup	Order	mModuleBusinessTemplate	Order
ModuleTableOrGroup	TableVID		
ModuleTableVersion	-		
ModuleTableVersion	ModuleID		
ModuleTableVersion	TableVID		
ModuleTableVersion	IsSimpleReuse		
		mTableDimensionSet	-
		mTableDimensionSet	ModuleID
		mTableDimensionSet	TableID
		mTableDimensionSet	MetricAndDimensions
OpenAxisValueRestriction	-	mOpenAxisValueRestriction	-
OpenAxisValueRestriction	AxisID	mOpenAxisValueRestriction	AxisID
OpenAxisValueRestriction	HierarchyID	mOpenAxisValueRestriction	HierarchyID
OpenAxisValueRestriction	MemberID	mOpenAxisValueRestriction	HierarchyStartingMemberID
OpenAxisValueRestriction	MemberIncluded	mOpenAxisValueRestriction	IsStartingMemberIncluded
OrdinateCategorisation	-	mOrdinateCategorisation	-
OrdinateCategorisation	OrdinateID	mOrdinateCategorisation	OrdinateID
OrdinateCategorisation	DimensionID	mOrdinateCategorisation	DimensionID

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
OrdinateCategorisation	MemberID	mOrdinateCategorisation	MemberID
		mOrdinateCategorisation	DimensionMemberSignature
OrdinateVariable	-		
OrdinateVariable	ExpressionID		
OrdinateVariable	OrdinateID		
OrdinateVariable	IsScopeFilter		
OrdinateVariable	VariableCode		
Owner	-	mOwner	-
Owner	OwnerID	mOwner	OwnerID
Owner	OwnerName	mOwner	OwnerName
Owner	OwnerPrefix	mOwner	OwnerPrefix
Owner	OwnerNamespace	mOwner	OwnerNamespace
Owner	OwnerCopyright	mOwner	OwnerCopyright
Owner	OwnerLocation	mOwner	OwnerLocation
Owner	ParentOwnerID	mOwner	ParentOwnerID
Owner	ConceptID	mOwner	ConceptID
PreCondition	-	vPrecondition	-
PreCondition	PrecondExprID	vPrecondition	PreconditionExpressionID
PreCondition	ValidationId	vPrecondition	ValidationRuleID
ReferenceCategorisation	-	mReferenceCategorisation	-
ReferenceCategorisation	ReferenceID	mReferenceCategorisation	ReferenceID
ReferenceCategorisation	DimensionID	mReferenceCategorisation	DimensionID
ReferenceCategorisation	MemberID	mReferenceCategorisation	MemberID
ReferencePeriodOffset	-		
ReferencePeriodOffset	OffsetValue		
ReferencePeriodOffset	PeriodOffsetID		
ReferencePeriodOffset	PeriodOffsetLabel		
ReferencePeriodOffset	PeriodType		
References	-	mReference	-
References	ReferenceID	mReference	ReferenceID
References	SourceCode	mReference	SourceCode
References	Article	mReference	Article
References	Paragraph	mReference	Paragraph
References	Point	mReference	Point
References	Romans	mReference	Romans
References	ReferenceText	mReference	ReferenceText
References	CategorisationKey		
References	WildcardCategorisationKey		
ReferenceSources	-		
ReferenceSources	SourceCode		
ReferenceSources	SourceDescription		
ReferenceSources	SourceLabel		
ReportingFramework	-	mReportingFramework	-
ReportingFramework	FrameworkID	mReportingFramework	FrameworkID
ReportingFramework	FrameworkCode	mReportingFramework	FrameworkCode
ReportingFramework	FrameworkLabel	mReportingFramework	FrameworkLabel
ReportingFramework	ConceptID	mReportingFramework	ConceptID
Source	-		
Source	SouceID		
Source	SourceCode		
Source	SourceLabel		
SpecificCellVariable	-		
SpecificCellVariable	CellID		
SpecificCellVariable	ExpressionID		
SpecificCellVariable	VariableCode		
StringList	-		
StringList	StringListCode		
StringList	StringListID		
StringListValues	-		
StringListValues	Description		

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
StringListValues	Note		
StringListValues	StringListID		
StringListValues	ValidValue		
SumOfManyOrdinates	-		
SumOfManyOrdinates	ExpressionID		
SumOfManyOrdinates	OrdinateID		
SumOfManyOrdinates	VariableCode		
SumOverOpenAxis	-		
SumOverOpenAxis	ExpressionID		
SumOverOpenAxis	OpenAxis		
SumOverOpenAxis	OpenDimensionID		
SumOverOpenAxis	VariableCode		
Table	-		
Table	ConceptID		
Table	FrameworkID		
Table	OriginalTableCode		
Table	OriginalTableLabel		
Table	TableID		
Table	Template		
TableCell	-	mTableCell	
TableCell	CellID	mTableCell	CellID
TableCell	TableVID	mTableCell	TableID
TableCell	DataPointVID		
TableCell	IsRowKey	mTableCell	IsRowKey
TableCell	IsShaded	mTableCell	IsShaded
		mTableCell	BusinessCode
		mTableCell	DataPointSignature
TableGroup	-	mTemplateOrTable	-
TableGroup	TableGroupID	mTemplateOrTable	TemplateOrTableID
TableGroup	TaxonomyID	mTemplateOrTable	TaxonomyID
TableGroup	TableGroupCode	mTemplateOrTable	TemplateOrTableCode
TableGroup	TableGroupLabel	mTemplateOrTable	TemplateOrTableLabel
		mTemplateOrTable	TemplateOrTableType
		mTemplateOrTable	Level
TableGroup	Order	mTemplateOrTable	Order
		mTemplateOrTable	ParentTemplateOrTableID
TableGroup	ConceptID	mTemplateOrTable	ConceptID
		mTemplateOrTable	TC
		mTemplateOrTable	TT
		mTemplateOrTable	TL
		mTemplateOrTable	TD
		mTemplateOrTable	YC
		mTemplateOrTable	XC
TableGroupTemplate	-		
TableGroupTemplate	TableGroupId		
TableGroupTemplate	TemplateID		
TableGroupTemplate	Order		
TableVersion	-	mTable	-
TableVersion	TableVID	mTable	TableID
TableVersion	TableVersionCode	mTable	TableCode
TableVersion	TableVersionLabel	mTable	TableLabel
TableVersion	FromDate	mTable	FromDate
TableVersion	ToDate	mTable	ToDate
TableVersion	XbrlFilingIndicatorCode	mTable	XbrlFilingIndicatorCode
TableVersion	XbrlTableCode	mTable	XbrlTableCode
TableVersion	ConceptID	mTable	ConceptID
		mTable	YDimVal
		mTable	ZDimVal
TableVersion	TableID		
Taxonomy	-	mTaxonomy	-
Taxonomy	TaxonomyID	mTaxonomy	TaxonomyID
Taxonomy	TaxonomyCode	mTaxonomy	TaxonomyCode
Taxonomy	TaxonomyLabel	mTaxonomy	TaxonomyLabel
Taxonomy	Version	mTaxonomy	Version
Taxonomy	PublicationDate	mTaxonomy	PublicationDate
Taxonomy	FromDate	mTaxonomy	FromDate

EBA DPM MS Access database (2.1.0 PC)		T4U SQLite DPMdb (ver7)	
Entity	Attribute	Entity	Attribute
Taxonomy	ToDate	mTaxonomy	ToDate
Taxonomy	FrameworkID	mTaxonomy	FrameworkID
Taxonomy	TechnicalStandard	mTaxonomy	TechnicalStandard
Taxonomy	ConceptID	mTaxonomy	ConceptID
TaxonomyTableVersion	-	mTaxonomyTable	-
TaxonomyTableVersion	TaxonomyID	mTaxonomyTable	TaxonomyID
TaxonomyTableVersion	TableVID	mTaxonomyTable	TableID
TaxonomyTableVersion	TableGroupID	mTaxonomyTable	AnnotatedTableID
TaxonomyTableVersion	IsSimplyResuse	mTaxonomyTable	IsSimplyResuse
TaxonomyTableVersion	TemplateID		
Template	-		
Template	TemplateID		
Template	FrameworkID		
Template	TemplateCode		
Template	TemplateLabel		
Template	ConceptID		
ValidationRule	-	vValidationRule	-
ValidationRule	ValidationId	vValidationRule	ValidationRuleID
ValidationRule	ValidationCode	vValidationRule	ValidationCode
ValidationRule	ValidationType	vValidationRule	ValidationType
ValidationRule	ExpressionID	vValidationRule	ExpressionID
ValidationRule	Scope	vValidationRule	Scope
ValidationRule	Severity	vValidationRule	Severity
ValidationRule	ConceptID	vValidationRule	ConceptID
ValidationRuleSet	-	vValidationRuleSet	-
ValidationRuleSet	ModuleID	vValidationRuleSet	ModuleID
ValidationRuleSet	ValidationRuleId	vValidationRuleSet	ValidationRuleID
ValidationScope	-	vValidationScope	-
ValidationScope	TableVID	vValidationScope	TableID
ValidationScope	ValidationID	vValidationScope	ValidationRuleID
VariableOfExpression	-	vVariableOfExpression	-
VariableOfExpression	ExpressionID	vVariableOfExpression	ExpressionID
VariableOfExpression	IfMissing	vVariableOfExpression	IfMissing
VariableOfExpression	VariableCode	vVariableOfExpression	VariableCode
zzTableChanges	-		
zzTableChanges	ID		
zzTableChanges	TableCode		
zzTableChanges	ComponentTypeName		
zzTableChanges	ComponentCode		
zzTableChanges	ComponentLabel		
zzTableChanges	HeaderFlag		
zzTableChanges	Level		
zzTableChanges	Change		
zzTableChanges	LastChangedIn		