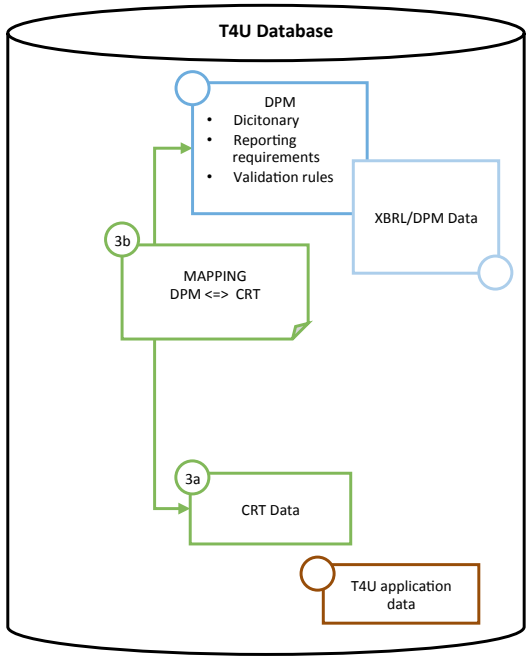
Tool for undertakings (T4U): Database and XBRL Integration

11:00 - 11:30

Agenda

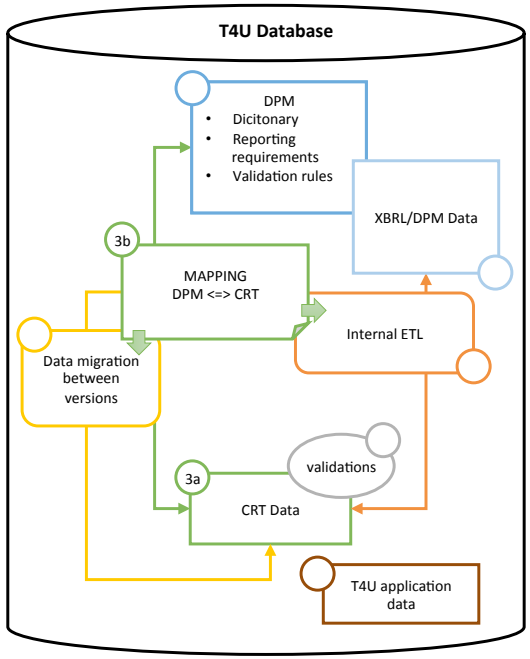
- Components and processes
- Technologies
- CRTs
- CRTs queries and T4U DB data validations

T4U Database Components



1. information requirements and validations rules metadata – resemble DPM dictionary, Annotated Templates and business checks information (description of the model, similar to an XBRL taxonomy)
   - populated from the DPM dictionary and Annotated Templates
   - similar to the EBA DPM MS Access database
   - stable structure, able to accommodate any changes (including NCA extension metadata)
   - used by the tool's application interfaces for navigation in information requirements and presentation of tabular forms
   - data (dimensional) and form (row-column) centric representation of metadata
   - problems: complexity of highly normalized model (not very intuitive and easy to understand or query by users not familiar with the data point model)
2. placeholder for data storage – stored facts referring to DPM properties (actual reported data, similar to an XBRL instance document and structured in a dimensional approach)
   - uses metadata definitions from component 1
   - facilitates interaction with XBRL instance document files (where exchanged data is described in a similar style)
   - potential performance problems when accessing facts (all facts stored in a single entity and are distinguished based on dimensional properties which hinders prompt rendering of selected facts in tabular views and execution of validation rules by matching facts according to dimensional properties)
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 (tabular oriented view of the information, similar to the reference business templates view, expressed as relational tables composed according to the normalized form defined by table linkbase)
   - generated from component 1
   - easier to work with (overcomes issues of component 1 and 2)
   - supports ETL (from external sources), data storage and validation
   - requires additional maintenance processes due to instability (data migration between versions), and redundancy (checks for duplicates, etc.)
4. T4U applications information, this is a specific set of information needed by the tool's applications (mainly user interfaces), for localisation purposes (translation of menu options, buttons, messages, etc. to national languages), validation results, etc.

T4U Database Processes



A. data conversion between DPM and classic relational data storage

- converts data stored in classic relational manner to the DPM properties data storage placeholder (and further to XBRL)
- 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 (and vice versa)
- 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)

B. migration of data stored in classic relational manner between the versions of the database

- help to overcome the problem of instability of classic relational placeholders due to changes to graphical views of business templates in time (even if they don't impact the definition of the exchanged data):
  - moving rows/columns between tables
  - table split/merge
- consequence: data from previous periods need to be migrated to the new structures whenever there is a change
- 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 and its attributes); 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)

C. validations (including queries supporting data validation and aggregations)

- business rules defined in the source materials (provided by business users) are loaded to the information requirements and validations metadata component where they are stored in a normalized manner
- execution of validations is performed on data stored in classic relational structures (using supportive database views if necessary)
- 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

Technologies

SQLite

- easy deployment on the side/machine of an undertaking
- multiplatform support (various environments and devices)
- open source license
- problems:
  - limited functionalities of tools for database managements (difficult migration from EBA DPM MS Access due to different structures)
  - lack of certain functionalities typical for DBMSs like stored procedures
  - partial support for simultaneous multiuser work
  - potential performance issues for larger amount of data and more complex computations

Microsoft SQL Server

- overcomes problems of the SQLite
- for internal (EIOPA and management/administrative tasks) usage
  - to conduct typical DBMS functions
  - to perform 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
- data is further transferred to SQLite in scope that is required by the standalone T4U applications

Relation to the Eurofiling Initiative deliverables and the EBA MS Access Database

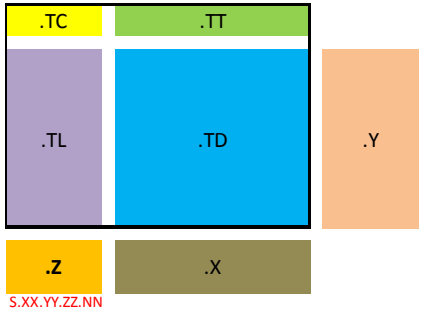
- T4U database DPM component follows the model expressed by:
  - http://www.eurofiling.info/dpm/index.shtml
  - http://www.eba.europa.eu/documents/10180/632822/Description+of+DPM+formal+model.pdf
  - http://www.eba.europa.eu/documents/10180/632822/DPM+Database.2.1.0.PC.7z
    (*CRD4 DPM - Database description - v2.1.pdf)*

- it is closely aligned with EBA DPM MS Access Database 2.1.0; differences include:
  - names of entities (tables) start with letters indicating their purpose, e.g. “m” for information requirements metadata and “v” for validation rules metadata
  - 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, etc),
  - many-to-many relation between Table and Axis entities (allowing axes to be reused by tables which is a common case in many of the Solvency II 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.),
  - validation rules are limited to reference to row/column/sheet codes rather than DPM artefacts (axes, ordinates, cells, metrics, etc.),

- T4U Database documentation lists all entities and their attributes together with relations to their EBA counterparts

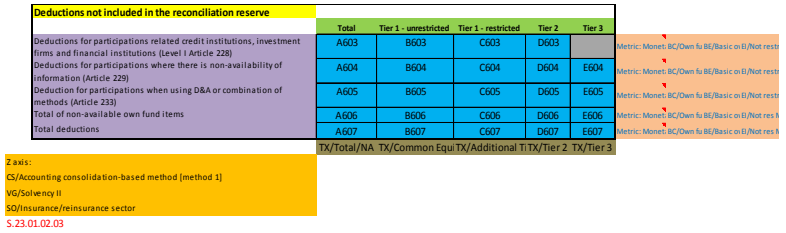
DPM Dictionary and Annotated templates conversion approach

Named ranges in annotated templates:

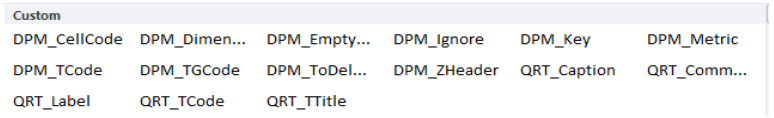


+ row/column codes where available

Example:



Cell styles:

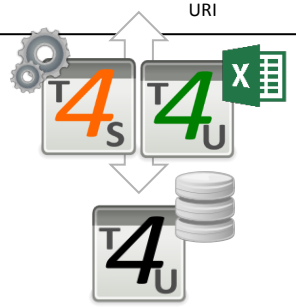


Change in annotation:

<Dimension>/<Domain member> → <Dimension>/{Domain[Hierarchy].[*,+]<MemberCode>}/<Domain member>

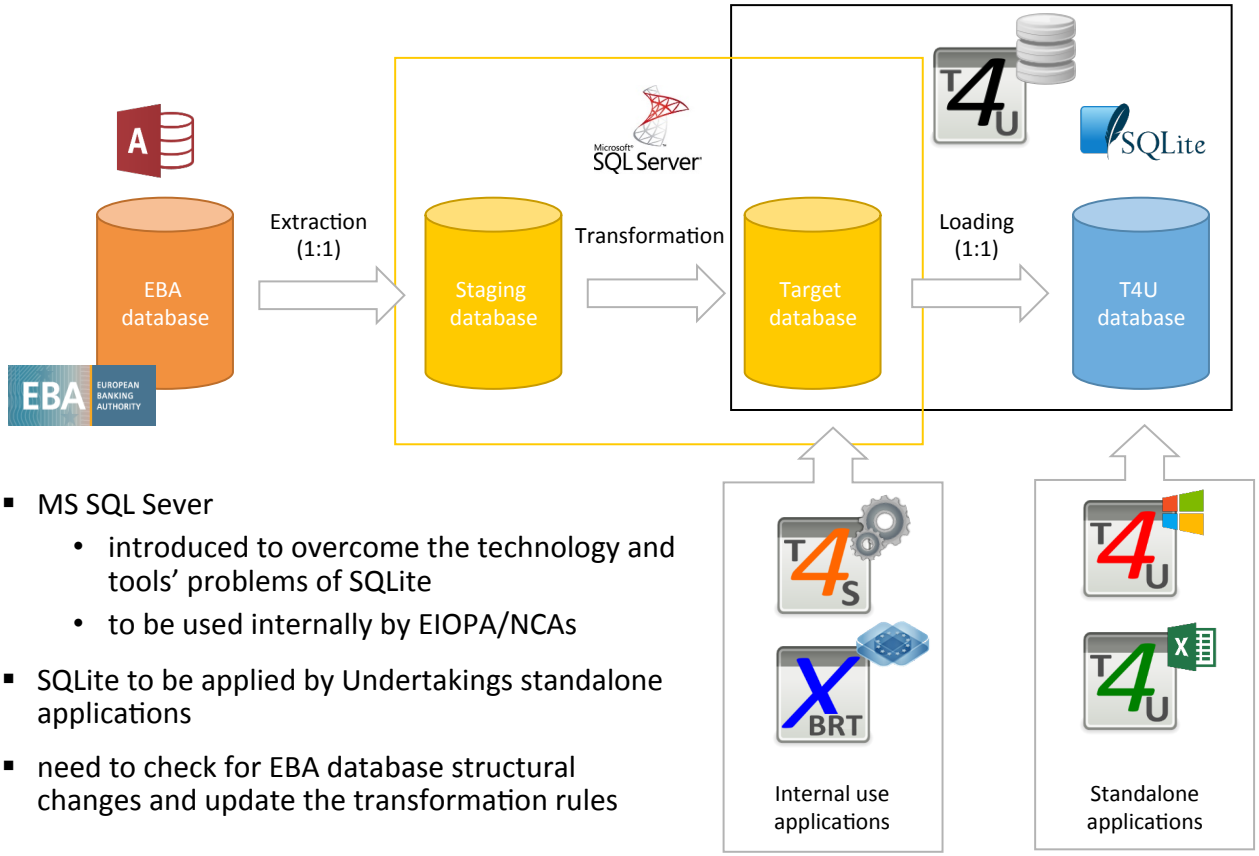
e.g.:

TP/Basic information	→ TP/{TP:x1}/Basic information
BA/All members (Total/NA)	→ BA/{LB[4.+]:x0}/All members (Total/NA)
LW/All member	→ LW/{CG[0.*]:x0}/All members
URI	→ UI/{ID:?}/URI



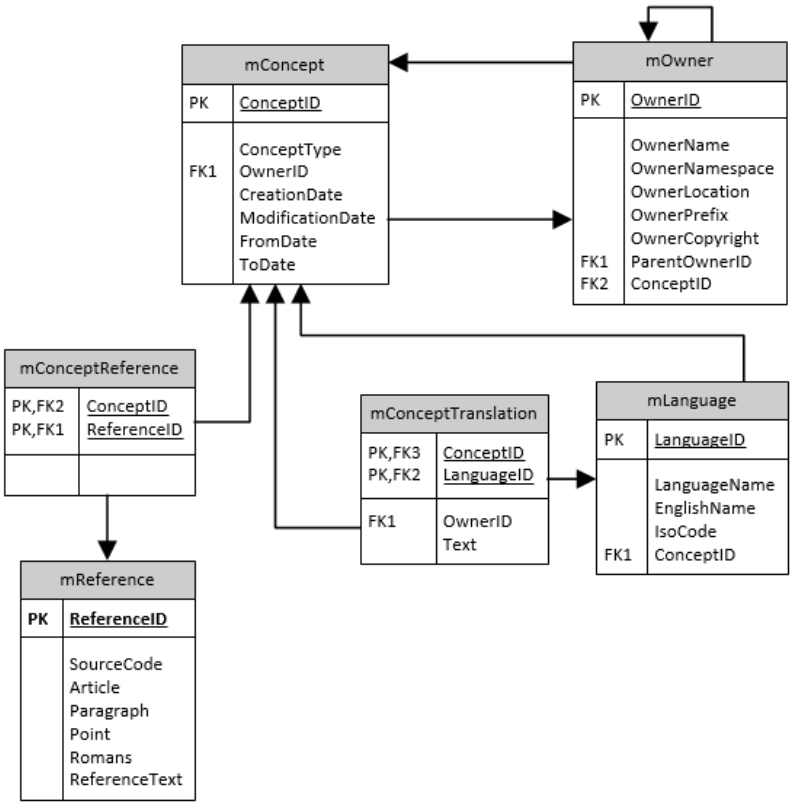
- migration of DPM dictionary data rather straightforward
- annotated templates extended by named ranges, styles and proposal of change in annotation
- T4S/reverse process of Excel Add-In to be used in repopulating the DPM dictionary and annotated templates with required structures and formats as well as for maintenance (in the future)

EBA DPM MS Access Database conversion approach



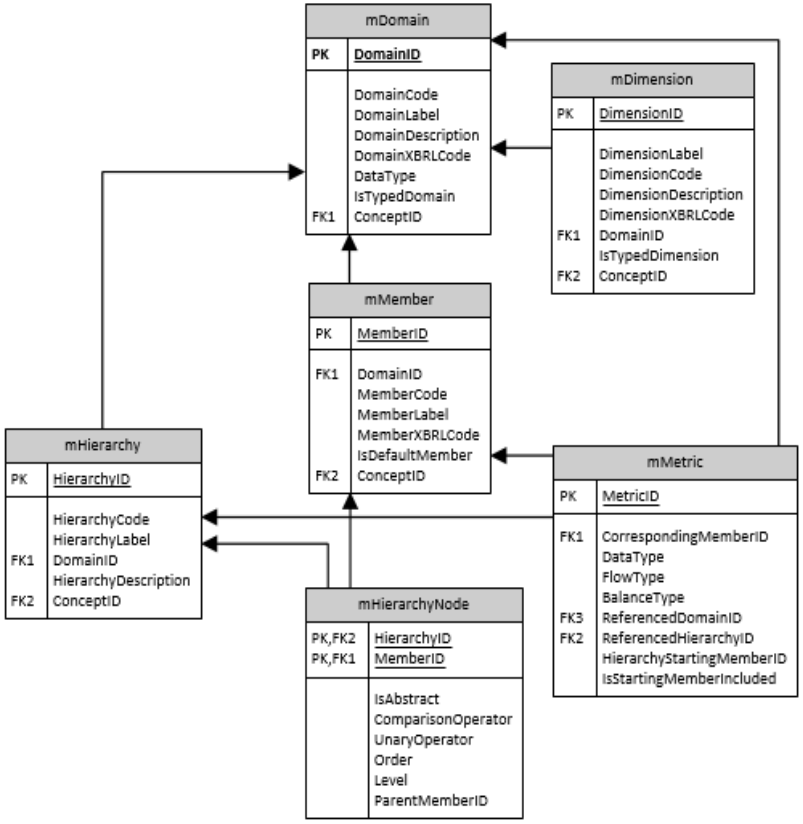
- MS SQL Sever
  - introduced to overcome the technology and tools' problems of SQLite
  - to be used internally by EIOPA/NCAs
- SQLite to be applied by Undertakings standalone applications
- need to check for EBA database structural changes and update the transformation rules

Concepts, owners, translations and references



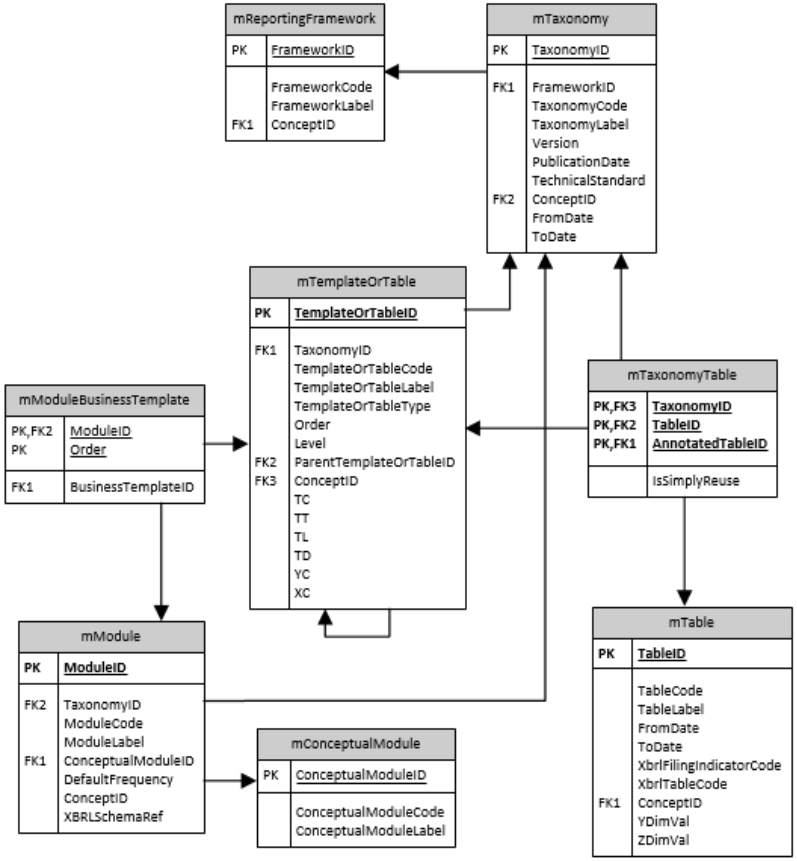
- concepts are DPM artefacts from both
  - dictionary:
    - domains,
    - members,
    - dimensions,
    - hierarchies,
    - metrics) and,
  - information requirements structures:
    - frameworks,
    - taxonomies,
    - templates and tables,
    - axes,
    - ordinates,
    - etc.
- concepts can:
  - be defined by various institutions (owners)
  - have multilingual labels (translations)
  - be described in multiple legal regulations (references)

Dictionary (domains, members, hierarchies and dimensions)



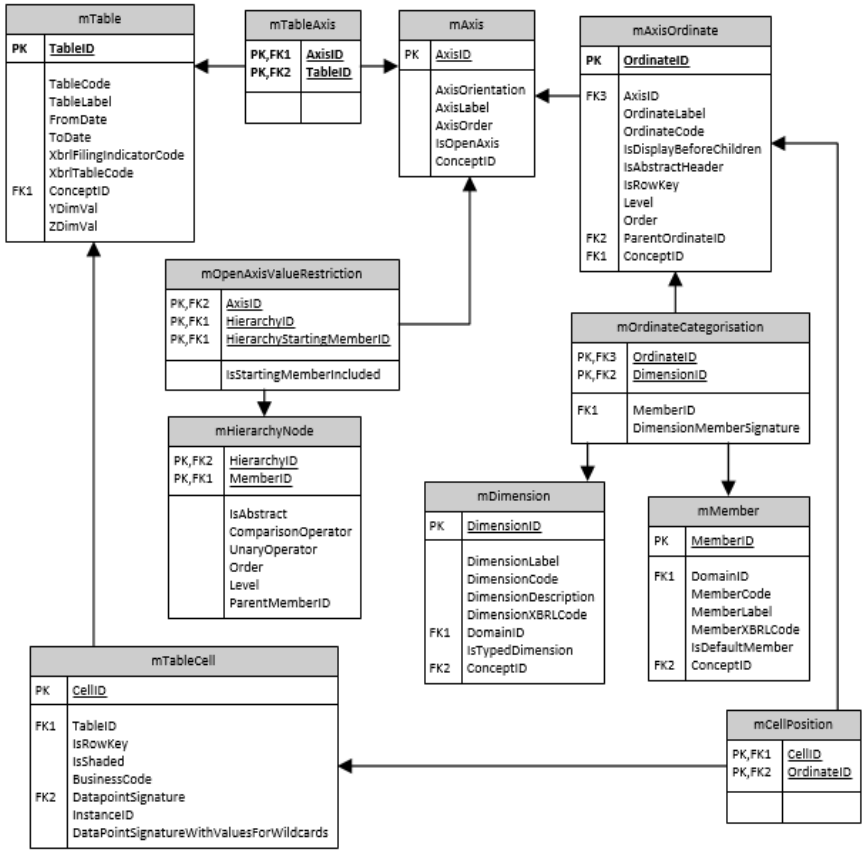
- dictionary contains definitions of domains (mDomain)
- each domain:
  - consists of members (mMembers), and
  - is associated with dimensions (mDimensions) that further contextualize members in the information requirements section of the model (database)
- members are gathered in hierarchies (mHierarchy and mHierarchyNode)
  - for documentation purposes
  - in order to support management of the dictionary
  - to 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 attributes:
  - period type
  - data type (which could take form of a list of members of another domain by referencing a hierarchy of its members)

Structure of information requirements (frameworks, taxonomies, modules, templates, and tables)



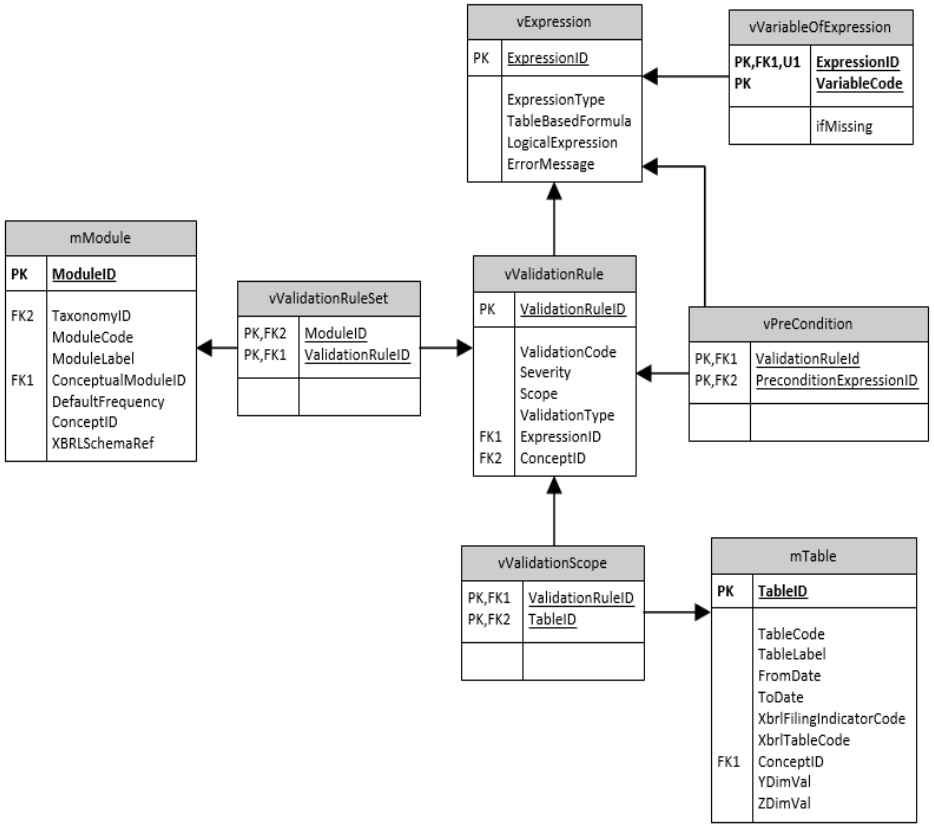
- information requirements are split in frameworks (mReportingFramework) that represent separate areas of interests/subject topics of collected data
- frameworks are versioned as taxonomies (mTaxonomy) that identify data sets required at the particular moment of time (previous, current and potentially also future versions)
- taxonomies consists of templates (mTemplateOrTable with TemplateOrTableType = “TemplatesGroup”, “Template” or “TemplateVariant”) which are graphical tabular representations of information requirements as defined in the legal regulations
- templates may consist of multiple individual tables, being defined as such originally or split as a result of normalization of the original tables (mTemplateOrTable with TemplateOrTableType = “BusinessTable” or “AnnotatedTable”)
- description of actual tables starts with mTable entity (as in the EBA DPM MS Access database, tables can be reused by taxonomies if they remain unchanged between different versions of frameworks - mTaxonomyTable)
- depending on reporting period or type of a report, not all templates defined by a taxonomy must be filed under certain reporting scenario; therefore modules (mModule) gather templates that are shall be submitted in one set (mModuleBusinessTemplate)

Tables and their components: axes, ordinates, cells



- actual tables are defined in mTable entity
- tables 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
- 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)

Validation rules



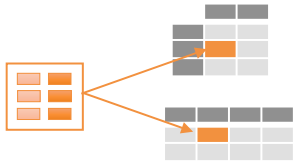
- reflection of validation rules metadata is much less elaborate in this component of the T4U database comparing to the EBA DPM MS Access database; the reason for is still unknown format of business rules definition in the input/source materials for the Solvency II and to which extend their execution 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
- 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)
- validations are gathered in sets (vValidationRuleSet) based on their application to specific modules

Representation of templates and tables

- structures and codes of templates (S.NN.MM.XX.YY/mTemplateOrTable.TemplateOrTableType):
  - NN - TemplatesGroup - collections of templates related to balance sheet, technical provisions, details on assets, etc.
  - MM - Template - individual templates, e.g. “Balance sheet”, “Assets D1, Investment Data, Portfolio list”
  - XX - TemplateVariant - templates differentiated based on their application for certain reporting scenario (e.g. solo/group, frequency of reporting, etc.)
  - BusinessTable - graphical tabular views under a template variant as defined in the legal regulations (i.e. Business Logs and Business Templates in case of Solvency II or ITS for COREP and FINREP)
    - not included in table codification (not reflected by any artefact 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
  - AnnotatedTable - business table 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), correspond to tables in a given version of taxonomy (mTaxonomyTable entity)
- EBA DPM MS Access database structure for these artefacts was not follow by the T4U database model due to different business requirements (i.e. dissimilar arrangement of information requirements in the EBA ITS)
  - 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 - 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”)

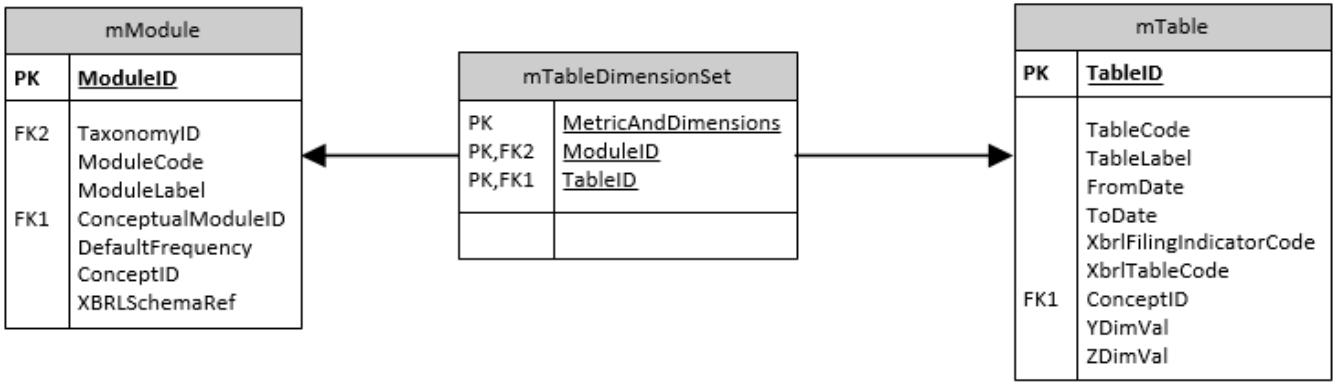
Data Point Signatures

- 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)



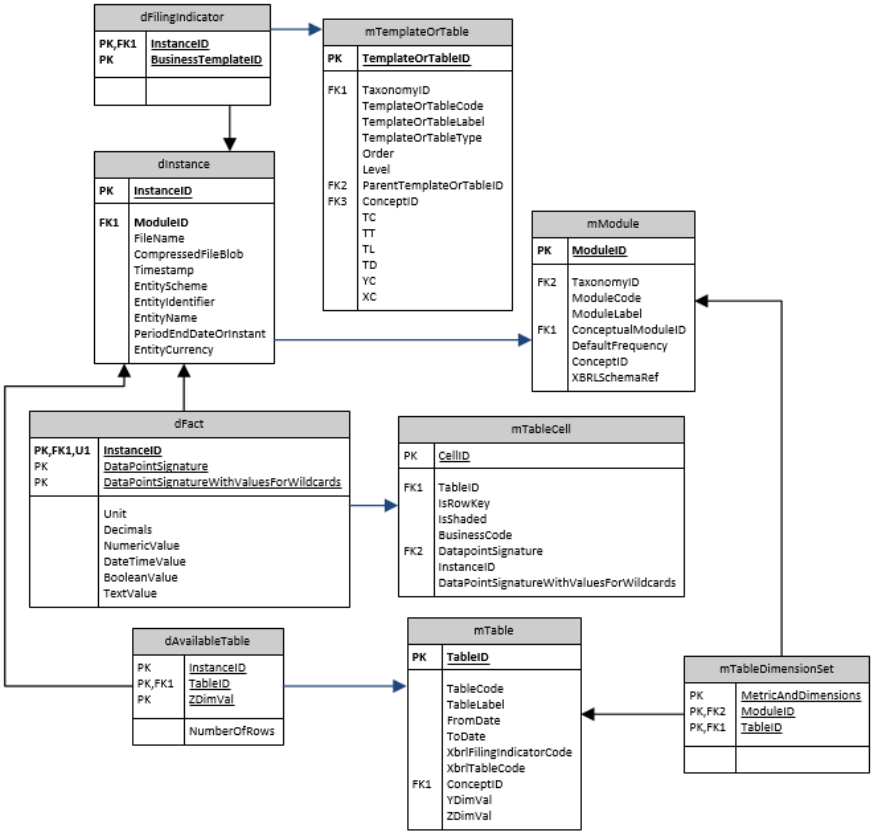
- signature is generated based on the properties already included in the database, in particular
  - position of a cell in the table (mCellPosition) by reference to table axis ordinates and
  - dimensional properties hidden behind these ordinates (mOrdinateCategarisation)
- data point signature is constructed 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)
  - when more than one dimension describes 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)
- Example of data point signatures:
  - *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)*
  - *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:ISA(*)*

Entities supporting association of facts with tables



- some entities or attributes of entities are created to be used when associating facts with tables to which they potentially belong in order 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
  - mTableDimensionSet identifies metrics and dimensions (without specifying explicitly the members) that describe data points belonging to each table
  - mTable entity contain YDimVal and ZDimVal attributes
- using this information, it is 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

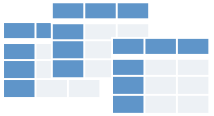
DPM component facts (and their relations to metadata)

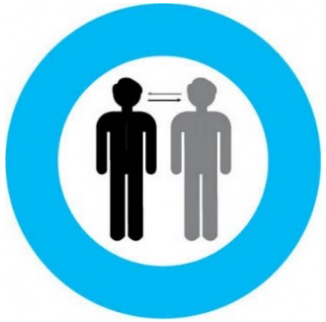


- information about stored instance documents is represented in dInstance entity which identifies a module (mModule) that was the basis for creation of a report (selecting from various available reporting scenarios)
- indication of templates that were submitted for a given module is stored in dFilingIndicator 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
- 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 but as this is also easily available in Classic Relational Structures placeholder it may be removed in the next versions
- additional tables will be included to store information on failed assertions in a parsed (imported or exported) XBRL instance document, in particular:
  - dErrors table would hold information about the rule and facts involved in erroneous evaluation

Classic relational data storage
General idea, goals and alternatives considered (1)

- alternative approaches assessed to overcome the problems of DPM metadata component
- goals for consideration/judgement criteria:
  - 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





Classic relational data storage
General idea, goals and alternatives considered (2)

- 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)
- 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
- selected alternative one:
  - pros:
    - easy to understand by DPM unaware users
    - relatively simple queries on data
    - expected increase in performance of validations and rendering should increase
  - cons:
    - 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

DPM and Classic Relational Structures
Principle of operation

Templates **S.99.12.31.01**

Page	PL

	C10	C20	C30	C40	C50
R10	2345		345	436	
R20					
R30	345				
R40		4567			
R50			234		

S.44.01.02.01

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

- for every table (mTable) a separate relational table is created containing:
  - a column referencing dInstance
  - a column for each z-axis (page) and
  - a column for every cell in table (based on mTableCell, excluding criss-crossed/grey shaded cells)
- 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)

DPM Annotated Templates Metadata

mTable

TableID	TableCode
1365	S.99.12.31.01
1699	S.44.01.02.01

mTableAxis

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

mAxis

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

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	

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

mOpenAxisValueRestriction

AxisID	HierarchyID
124	12
132	12

DPM and Relational Structures Mapping:

mMapping

TableID	RSTableName	RowColumnCode	Signature
1365	S.99.12.31.01	PAGE124	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)

DPM Data:

dFact

InstanceID	Signature	Value	Unit	Decimals
1	MET(s2md_mi2)|s2c_BAS(s2c_BA: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

Data in Relational Structures:

Table: 1365_S.99.12.31.01

InstanceID	Page124	R10C10	R10C20	R10C30	R10C40	R10C50	R20C10	...
1	eu_GA:PL	2345		345	436			

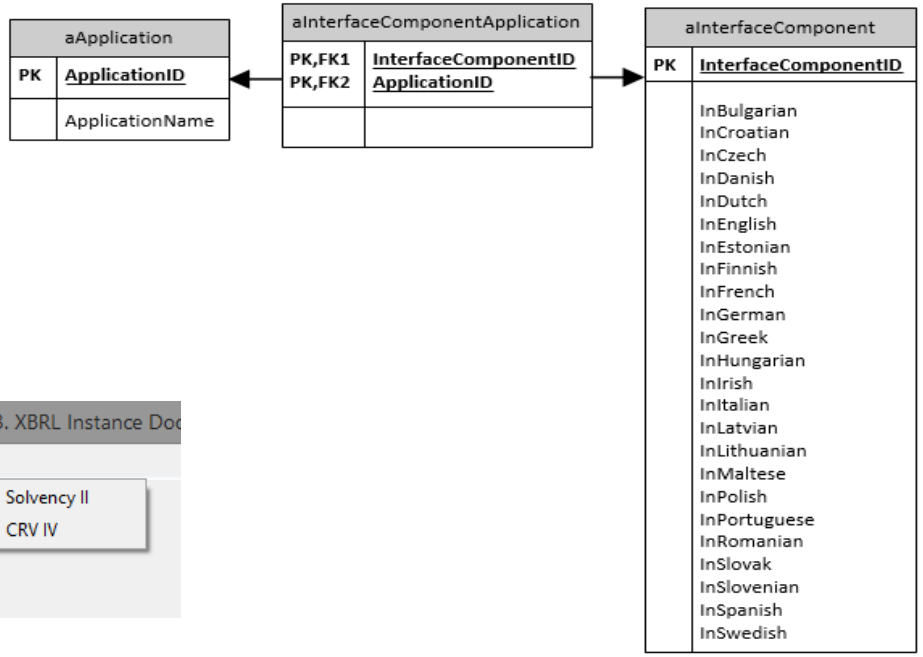
Table: 1699_S.44.01.02.01

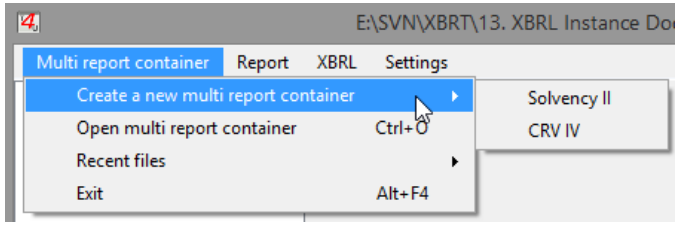
InstanceID	C10	C20	C30	C40
1	12	PL	1001	0.15
1	322	ES	2034	0.34

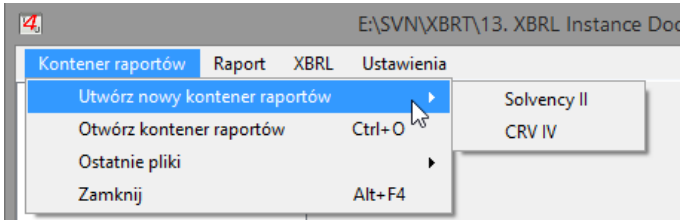
```
select R0020C0020, R0070C0020
from T__S_02_01_03_01__s2md__1_5_2
where INSTANCE = 1;

select
   T1.R0440C0010 LEFT_SIDE,
   T2.R0070C0030 + T2.R0070C0040 RGHT_SIDE
from T__S_02_01_03_01__s2md__1_5_2 t1
   inner join T__S_12_01_01_01__s2md__1_5_2 t2 on t1.INSTANCE = t2.INSTANCE
where t1.INSTANCE = 1;

select
   t1.r0530c0010 LEFT_SIDE,
   sum( t2.r0930c0040 ) + t3.r0930c0030 RGHT_SIDE
from T__S_02_01_03_01__s2md__1_5_2 t1
   inner join T__S_02_02_01_04__s2md__1_5_2 t2 on t1.INSTANCE = t2.INSTANCE
   inner join T__S_02_02_01_03__s2md__1_5_2 t3 on t1.INSTANCE = t3.INSTANCE
where t1.INSTANCE = 1;
```

Application's interface information

- names of the entities used by applications start with letter "a"
- it is expected that the application based on different solution/supporting different technologies (e.g. Windows Forms, Excel Add-in, iOS, …) 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
- aApplication now includes also application version and database type







Thank you!

Karol Minczyński