DPM 2.0 – Refit project

Outline

1. DPM background (EBA-EIOPA collaboration)
2. EBA experience with DPM and DPM Refit role towards integration
3. EIOPA experience with DPM and DPM Refit uses
4. EC supervisory data strategy and links with DPM
6. DPM 2.0 timelines

DPM background

- The EBA Data Point Model development **started in 2012**, to support the EBA reporting framework 2.0
- The DPM database was first **published by end 2013**, as part of the ITS 2.0 technical package
- Over the last **8 years**, the DPM database has been accumulating all the successive releases of the EBA data dictionary, from **version 2.0 to 3.3**
- The data dictionary **tracks changes** and maintains the full historization of all templates structure, data points categorisation, validation rules, and taxonomies, **across all releases**
- The core structure of the DPM database has not changed significantly since its first publication
- The DPM database is being used as a **main component of EUCLID** (as it had been already with the previous ESP reporting system)
- The DPM database is also at the core of the EBA solutions for regulatory data analysis

- The EIOPA on 2011 decided to implement the xBRL format for data exchange of regulatory reporting data
- The Data Point Model development **started in 2012**, to support the business & technical development of the Solvency 2 reporting framework
- The first DPM model was **published on 2013**, both on Excel and xBRL formats
- **In 2014** EIOPA developed the **Tool for Undertakings** to support the insurance companies on the creation of Solvency 2 on xBRL, **adopting the EBA's DPM database** as core central piece of the software solution
- **In 2015** EIOPA published the **Solvency 2, DPM and xBRL taxonomies**
- The EIOPA DPM has being evolving since then, covering in the **single glossary all the EIOPA's reporting frameworks**, including the **ECB add-ons** (insurance and pension funds) with the define-once approach

DPM background

EBA-EIOPA collaboration

Use of DPM methodology

The EBA and EIOPA collaboration on data standardisation since both authorities started to use DPM methodology

Integration through a single data dictionary

Both ESAs have being very successful on integrating their regulatory frameworks using a single data dictionary

Building a metadata driven strategy

Both, adopted the metadata-driven strategy as the way to respond to changes and reduce costs

Facing challenges together

Similar future challenges, made evident the advantages of more strong collaboration and harmonisation

Working together in the development

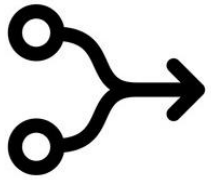
Experts from both ESAs have been working on the new common data model, envisaging the common development of Data Definition standards and tools

Building a common long-term goal

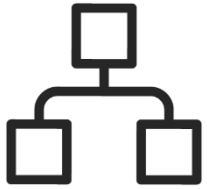
As a long-term goal, this should play a key role to enable a semantic integration of a single financial dictionary for the whole financial sector

Total convergence of EBA and EIOPA *methods, models, processes, and tools used for the development of data dictionaries and related regulatory products.*

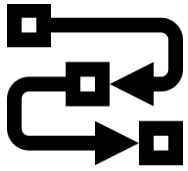
DPM Refit project goals



Total convergence of EBA and EIOPA methods, models, processes, and tools used for the development of data dictionaries and related regulatory products



Unified and versatile metamodel applicable to all regulatory data exchanges, from highly aggregated data points to very granular data sets of prudential, statistical or transactional information



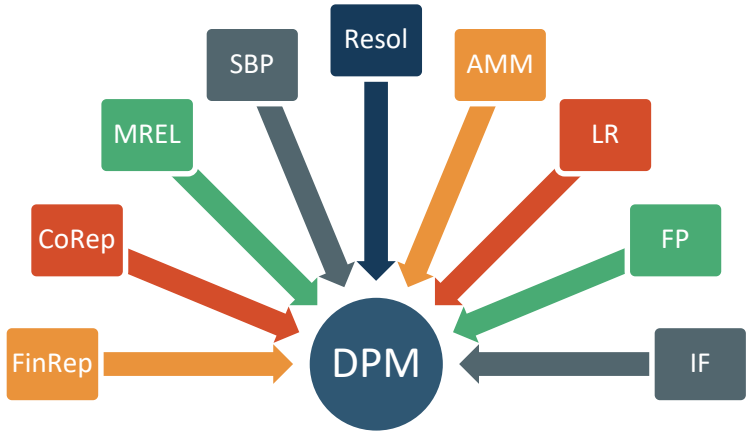
Content extensible and interoperable for defining, reusing and exchanging metadata for regulatory data requirements

Enabling the possibility of subsequent **semantic integration** of data dictionaries across different regulatory domains

EBA experience with DPM and DPM Refit role towards

The DPM experience

The goal of integrating the EBA reporting frameworks was always a priority in the definition of supervisory and resolution regulation



The EBA approach was guided by:

- Use the DPM, as a unique common data dictionary including all elements needed to define the regulatory data
- Publish always the regulatory text accompanied with formal and standard data definitions
- Produce a standardised technical package fully aligned with regulation and envisaging its direct use by CAs and institutions in their digital processing
- Enable systems communication by sharing DPM as a common platform of understanding across different areas and users

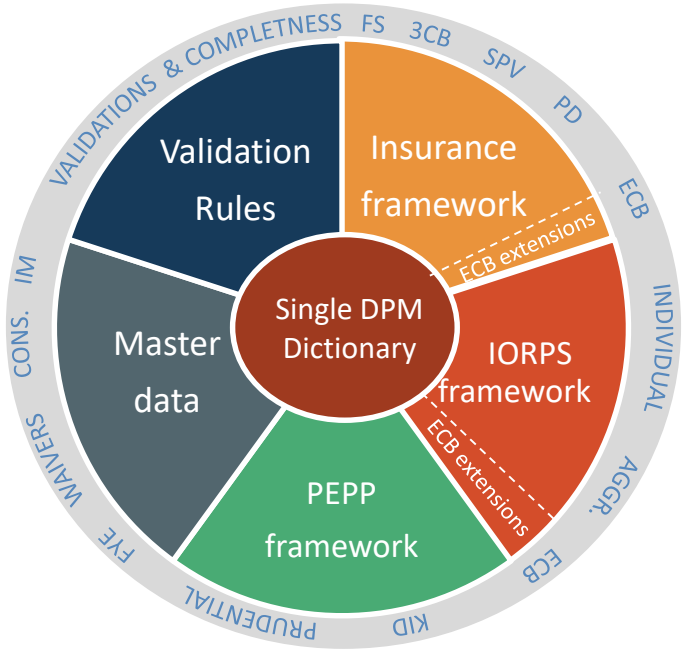
DPM Refit role towards further integration

- ⓾Standard and comparable definitions
- ⓾Ready to support digital processing
- ⓾Ready to support systems connectivity

- *Review of the data model and glossary*
- *Integrated reporting and works towards semantic integration – work with the ESAs, ESCB, SRB and EC*
- *Pilar III data hub and pillar 3 integration with reporting*

DPM Refit in EIOPA

EIOPA Evolution of regulatory reporting with the current data point model



The EIOPA architecture outline on DPM

- ✓Solvency II group and solo reporting
- ✓Financial stability
- ✓Pilar III public disclosure templates
- ✓IORP reporting
- ✓PEPP reporting
- ✓ECB insurance and pension funds add-ons

- The DPM implements the **uniform and consistent definitions** included in the implementing technical standards (ITS), guidelines and Board of Supervisors decisions on **reporting and disclosure**
- Provides a **structured representation of the information**, identifying all the **business concepts** and **their relations**, as well as **validation rules**
- **One model for all data** reporting requirements under EIOPA remit (insurance, pensions, PEPP providers, public disclosure) and for the ECB reporting extensions
- The model facilitates the appliance of waivers and completeness information reporting via the basic information – Master data
- The common syntactic format will further support data harmonisation for both EBA and EIOPA, providing synergies for a **common development of tools for the digital regulatory reporting.**

What's next for EIOPA on DPM Refit

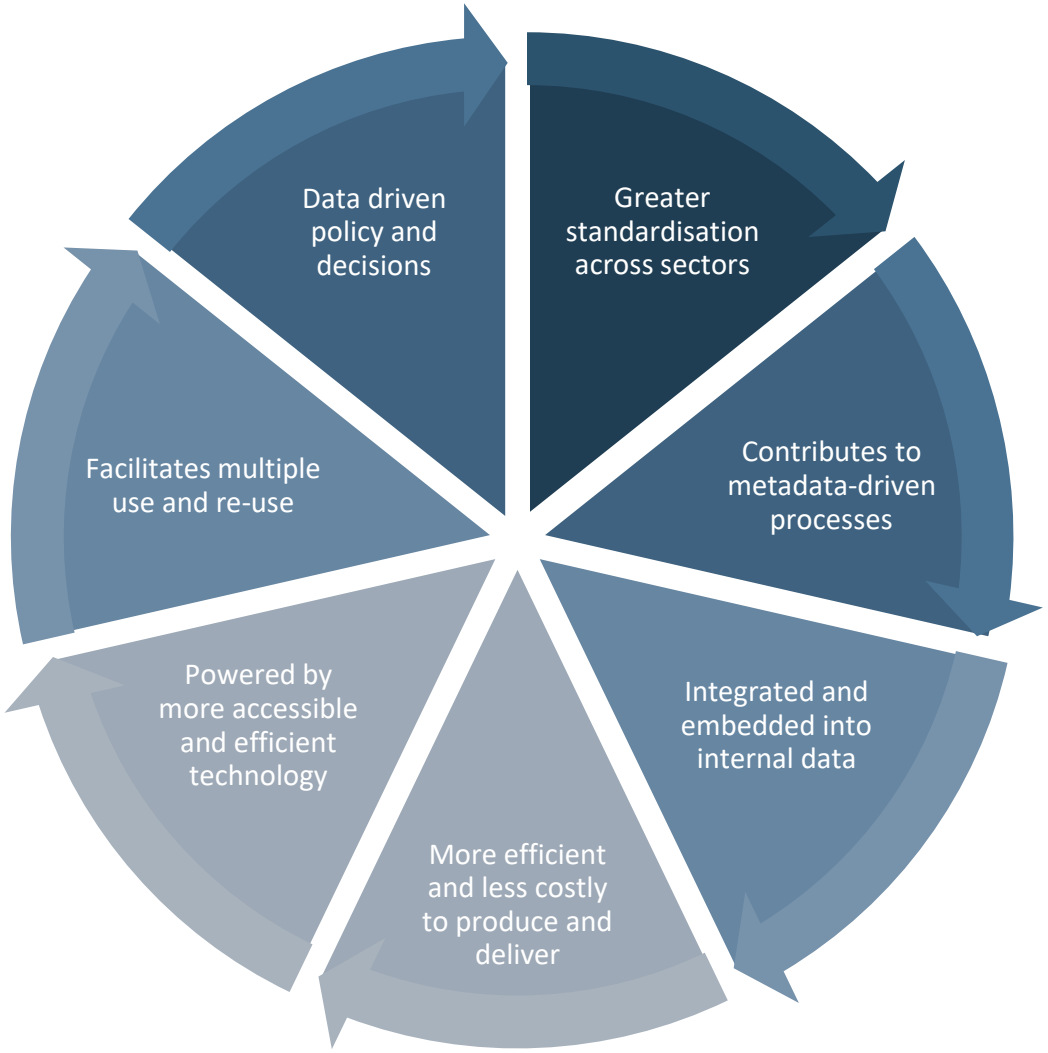
- *Impact on taxonomies and data exchange*
- *Impact on insurance ECB add-ons*
- *EC data strategy*
- *Business changes (Financial Conglomerates, Decision on IORPs, 2020 Review, EU-US agreement*
- *Monitoring development of new formats (CSV)*
- *Use of platform Atome:Matter*
- *Digital Reporting Tool under development*

Technical and business changes affect the outputs expected hence require Communication and gradual implementation to allow preparation

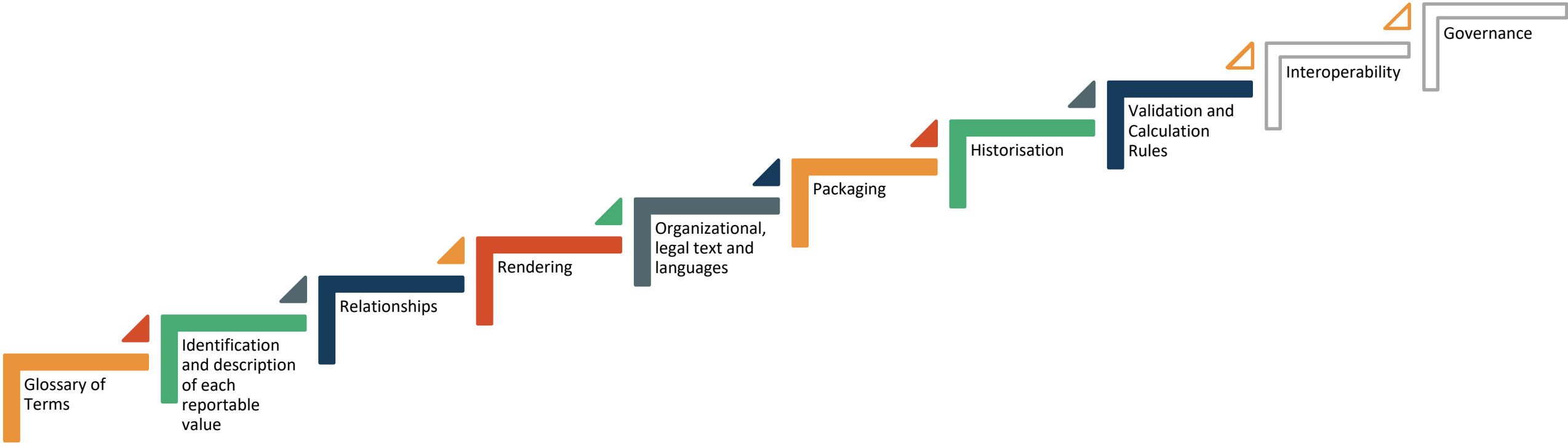
EC supervisory data strategy and links with DPM

EU Supervisory Data Strategy

- Facilitate digital transformation
- Consistent and standardised data
- Improved design of reporting requirements
- Data sharing and reuse

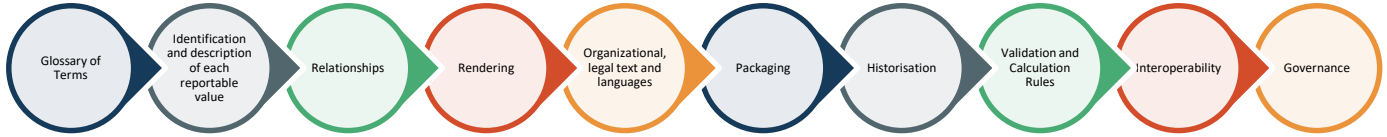


DPM Refit status of building blocks



- Most of the building blocks are done/decided. From now only minor changes can be expected mainly due to experience gained when doing the migration of metadata and software
- Two building blocks still ahead to be developed: interoperability and governance
- EBA, EIOPA and other EU component authorities are analyzing the following DPM Refit implementation phases

DPM Refit pending building blocks



Interoperability API

- Goal: to enable meta-exchange among DPM Refit repositories
- Decentralized approach
- Based on open standards (WS/REST)
- First version read only
- Based on DPM Refit ID and GUIDs to enable cross-domain interoperability

Governance

- Clear and transparent decision making process for the maintenance and evolution of the DPM Refit
- Open standard to European and National competent authorities and other standard user stakeholders
- Focused on the scope of DPM methodology (syntactical integration)
- Focused on operational aspects and complementary to the promotion and maintenance of the Refit as ISO standard

Process has not started yet, so all still to be discussed

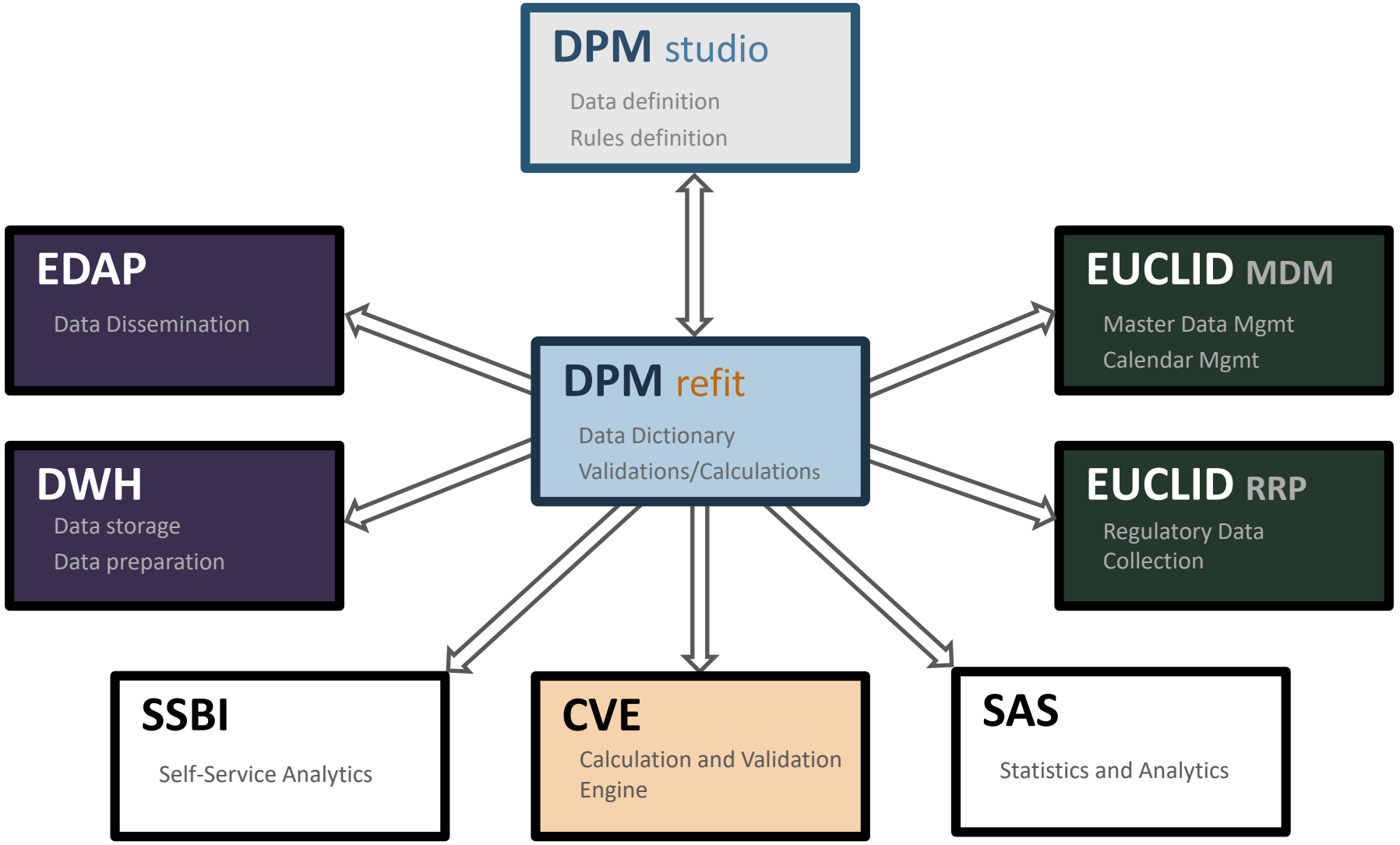
What is next on EIOPA's DPM implementation

Taxonomy - application date	Publication	Business changes	Current approach	Atome	DPM Refit	CSV	DPM St
2.8.0 – 12.2023	Current approach: July 2022 Validations added in Jan 2023 only via ATOME: Matter.	SII (new ITS on reporting and disclosure)	Yes – for the dictionary and annotated templates for comparison No – for validations would be only provided in ATOME: Matter outputs	Yes, for information and adaptation; Validations only from ATOME: Matter	No	No	No
2.8.1 - 12.2023?	PWD in July 2023 (tentative)	Financial Conglomerates (FICOD)	No	Yes	No	No	No
2.9.0 – 12.2024 (or Q12025?)	PWD1 in Q1 2023 (tentative)	IORPs Decision	No	Yes	Yes, for comparison purposes only	To be confirmed (only if adequate engines are available to execute rules/validations in csv))	No
2.10.0 – 12.2025	1 June 2024 without validations; 15 July 2024 with validations	SII Review	No	Yes	Yes	Yes	No - Potentially for info
2.11.0 - 12.2026	1 June 2025 without validations; 15 July 2025 with validations	If needed	No	Yes, for comparison	Yes	Yes	Yes
Following years		If needed	No	No (to be confirmed)	Yes	Yes	Yes

What is next on EBA's DPM implementation

Main Milestones	Target date
DPM Refit finalised	End 2022
DPM Studio delivered	Q2 2023
Calculation/Validation engine delivered	Q2 2023
DPM semantic review	Q3 2023
DPM content migration from old to new model	Q3 2023
Start using DPM Studio/DPM Refit for the development of new Framework releases	Q3 2023
Internal dependent EBA systems adapted to DPM Refit	Q4 2023
Start DPM transition period, where new DPM releases are published both in the old and new formats	Start 2024
End DPM transition period, aligned with XBRL transition period. Only the new DPM refit will be published, and only XBRL-CSV files will be accepted by the EBA	End 2025

The DPM in the EBA reporting ecosystem



EBA EUROPEAN BANKING AUTHORITY

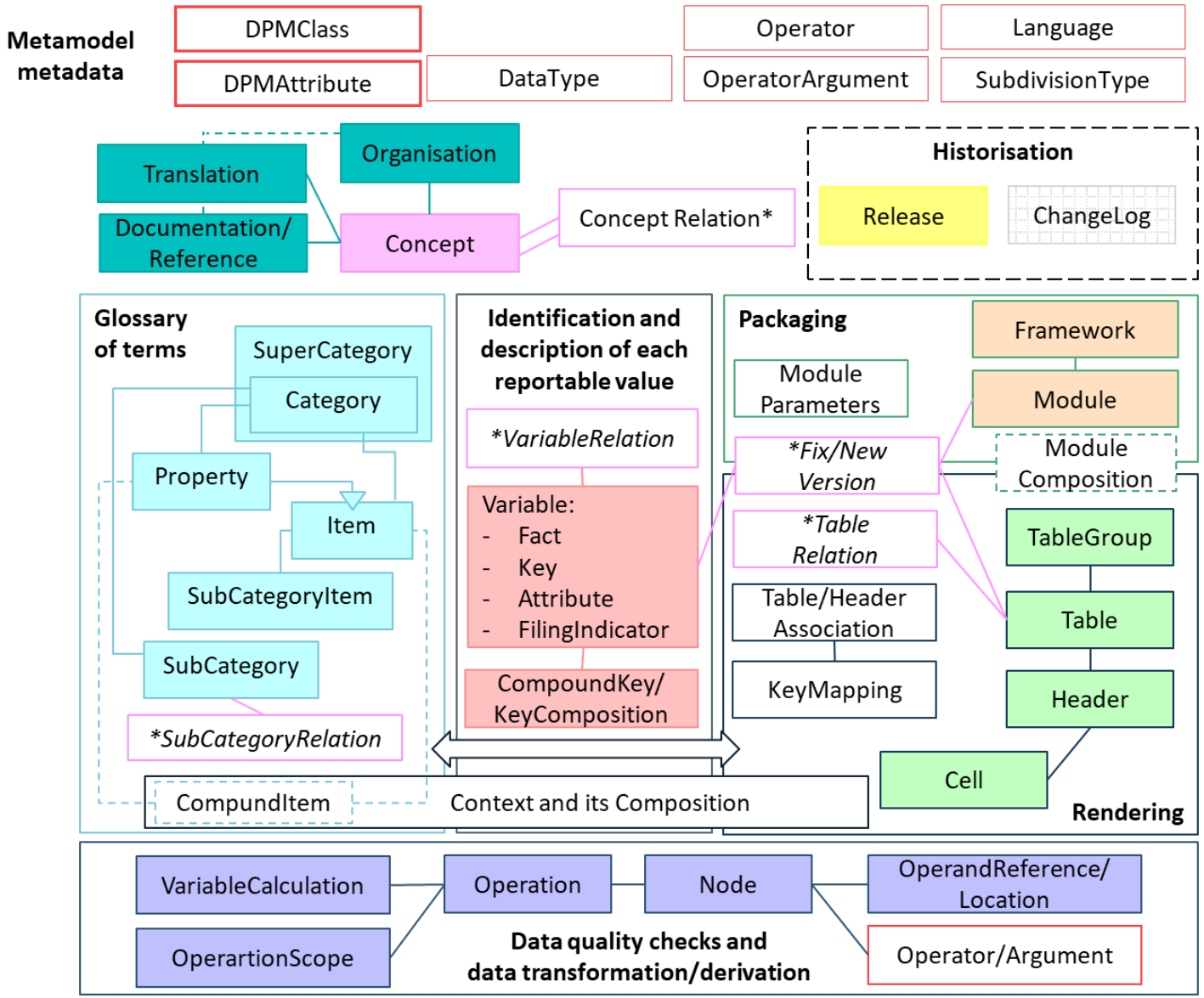
eiopa European Insurance and Occupational Pensions Authority

DPM 2.0 Refit project –Metamodel Intro

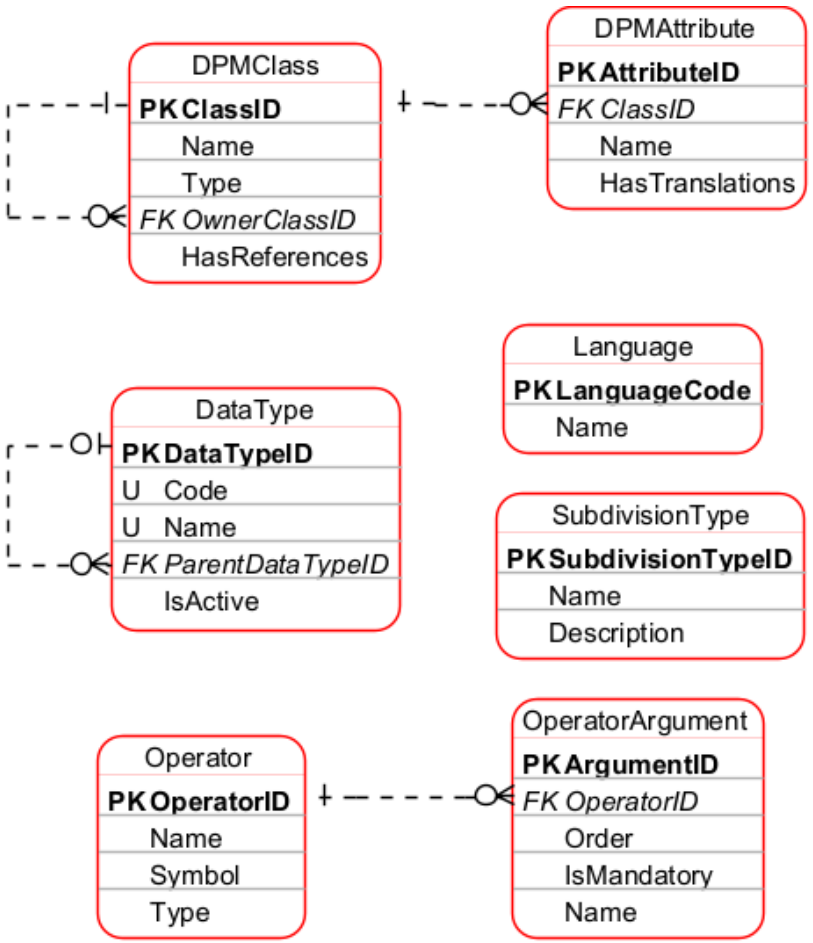
DPM standard

Outline:

1. DPM meta-model overview
2. Meta-model metadata
3. Administration and documentation
4. Historization: releases, deactivations, application dates
5. Glossary: basic and advanced modelling concepts
6. Representation of information requirements
    - Application of glossary terms
    - Rendering: tables, headers, cells
    - Role and definition of variables
    - Packaging: frameworks, modules, table groups
7. Operations: data quality checks and data derivation rules


Metamodel metadata
DPMClass
DPMAttribute
DataType
Operator
OperatorArgument
Language
SubdivisionType
Translation
Organisation
Documentation/ Reference
Concept
Concept Relation*
Historisation
Release
ChangeLog
Glossary of terms
SuperCategory
Category
Property
Item
SubCategoryItem
SubCategory
*SubCategoryRelation
Identification and description of each reportable value
*VariableRelation
Variable:
- Fact
- Key
- Attribute
- FilingIndicator
CompoundKey/ KeyComposition
Packaging
Framework
Module Parameters
Module
*Fix/New Version
Module Composition
*Table Relation
TableGroup
Table/Header Association
Table
KeyMapping
Header
Cell
Rendering
CompundItem
Context and its Composition
VariableCalculation
Operation
Node
OperandReference/ Location
OperartionScope
Data quality checks and data transformation/derivation
Operator/Argument

Metamodel: Metadata



Class
Organisation
Category
Subcategory
Property
Item
Framework
Module
ModuleVers
TableGroup
Table
TableVersion
TableAssocia
Header
HeaderVersi
Cell
Variable
VariableVers
CompoundKey
Context
Operation
OperationVersion
OperationScope
Document
DocumentVersion
Subdivision
Release
SubCategoryItem

Name	Description
Chapter	For a publication that uses chapters, this part should be used to capture this information. Because chapters are not necessarily numbers, this is a string.
Article	Article refers to a statutory article in legal material.
Section	Section is used to capture information typically captured i docum
Subsection	Subsec
Paragraph	Paragr
Subparagraph	Subpar
Clause	Subcor
Subclause	Subcor
Appendix	Refers
Example	Examp Exhibit
Page	Page n
Exhibit	Exhibit
Footnote	Footnc
Sentence	In som referer
URI	Full UR
Requirement	Somet

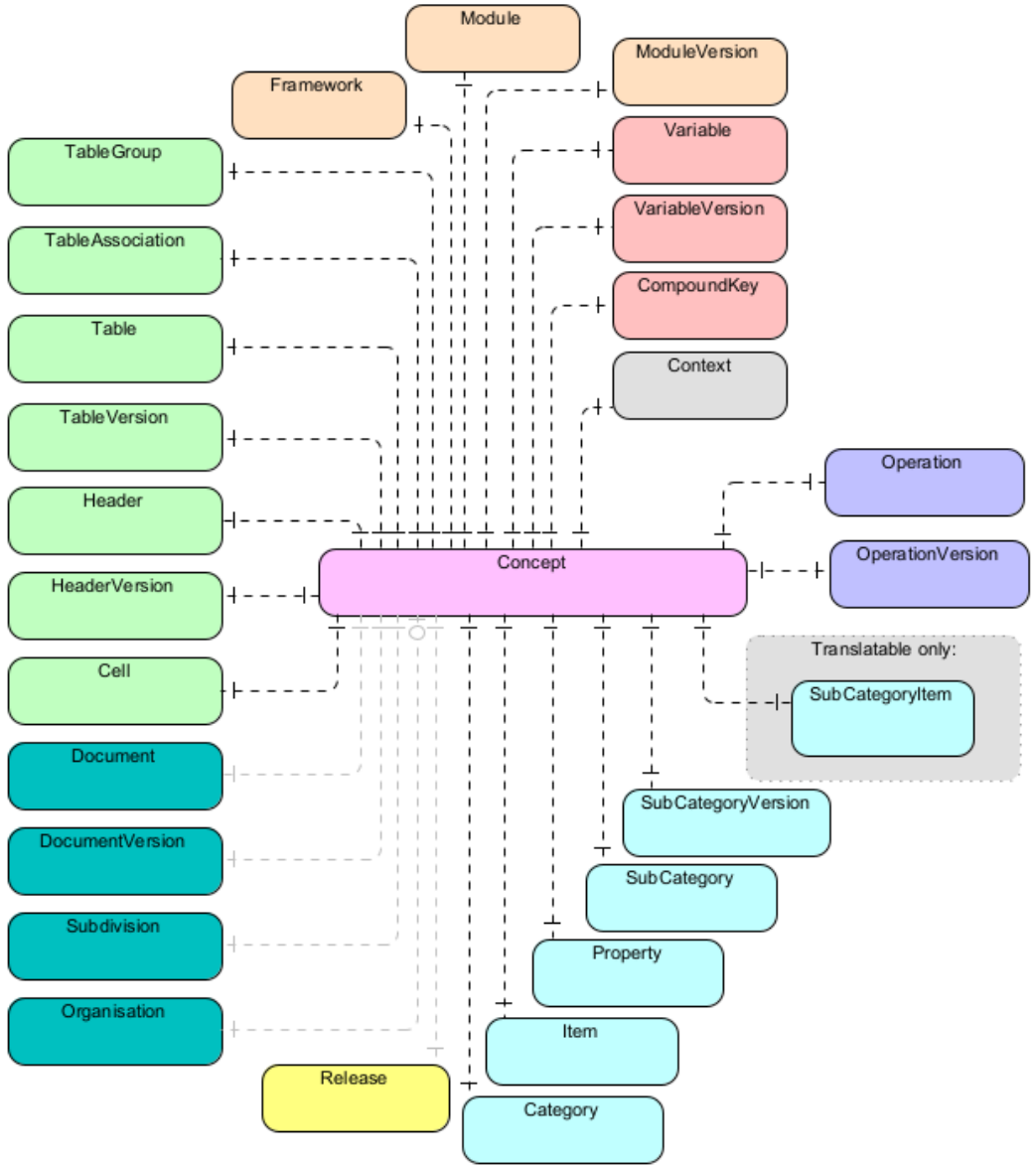
Attribute	DPMClass
Name	Category
Description	Category
Name	SubCategory
Description	SubCategory
Name	Item
Description	Item
Label	SubCategoryItem
Name	Framework
Description	Framework
Name	ModuleVersion
Description	ModuleVersion
Name	TableGroup
Description	TableGroup
Name	TableVersion
Description	TableVersion
Name	HeaderVersion
Name	TableAssociation
Description	TableAssociation

Code	Name	Parent Data Type
i	integer	
r	decimal	
s	string (non empty)	
b	boolean	
t	true	boolean
dt	date time	
d	date	date time
e	enumeration	string
m	monetary	
p	percentage	
u	URI	
o	ordinals	
es	string (including empty string)	

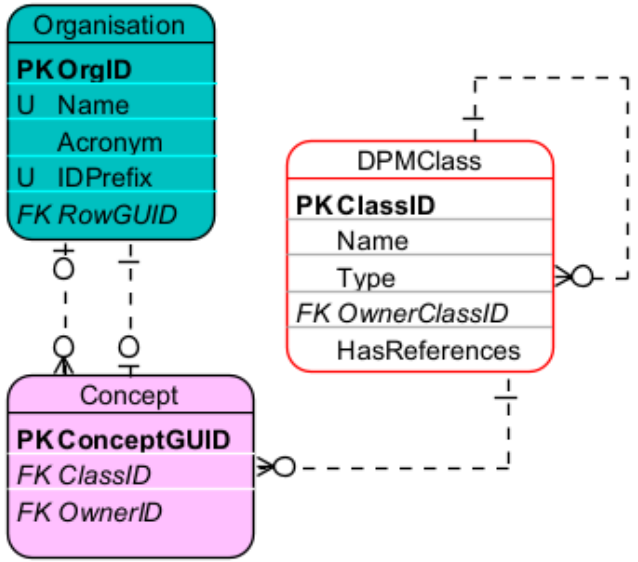
Name	Symbol	Type
Unary plus	+	Numeric
Addition	+	Numeric
Division	/	Numeric
Unary minus	-	Numeric
Subtraction	-	Numeric
Absolute value	abs	Numeric
Numeric minimum	min	Numeric
Multiplication	*	Numeric
Numeric maximum	max	Numeric
Square root	sqrt	Numeric
Aggregate maximum	max_aggr	Aggregate
Aggregate minimum	min_aggr	Aggregate
Equal to	=	Comparison
Less than equal to	<=	Comparison
Greater than equal to	>=	Comparison
Element of	in	Comparison
Is null	isnull	Comparison
Greater than	>	Comparison
Less than	<	Comparison
Not equal to	!=	Comparison
haracters	match	Comparison
	and	Logical
	or	Logical
	not	Logical
e or	xor	Logical
	sum	Aggregate
	count	Aggregate
	where	Clause
	get	Clause
lse	if-then-else	Conditional
	filter	Conditional
ft	time_shift	Time
	rename	Clause
Node	node	Clause

Concepts

- Identification
- Ownership
- Translations
- References
- Relations



Concepts: Owners



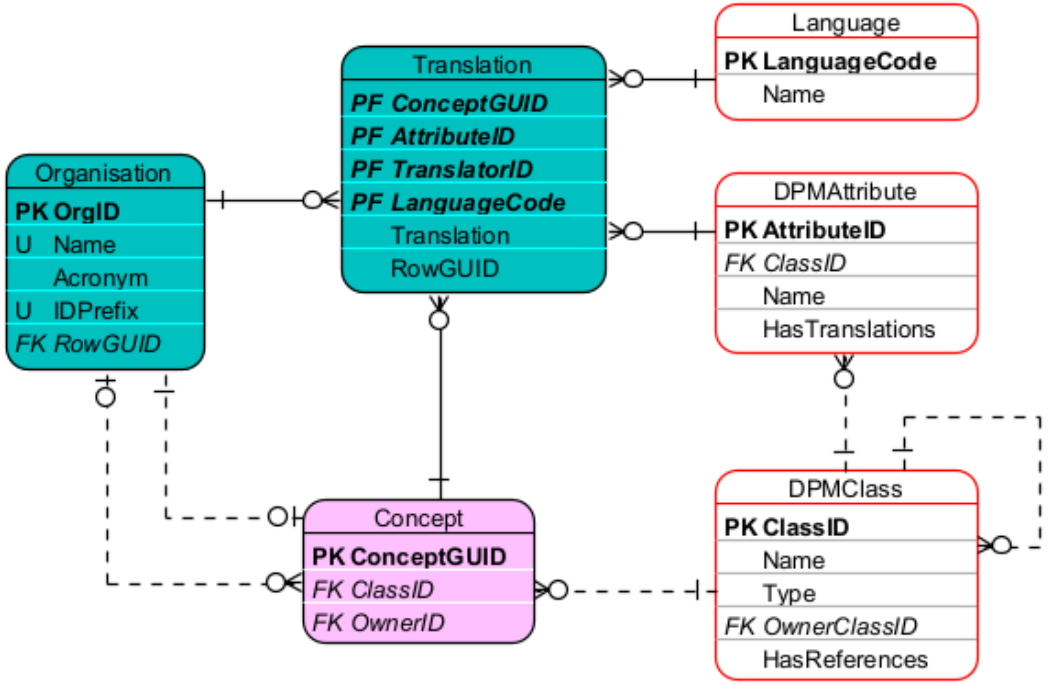
Predefined organisations:

Name	Acronym	IDPrefix
DPM Metamodel	DPMM	100
European Banking Authority	EBA	101
European Insurance and Occupational Pensions Authority	EIOPA	102

Implementation aspect: *IDPrefix* to ensure global uniqueness of IDs (to facilitate physical merging to models in one database)

Class	OwnerClass
Organisation	
Category	
Subcategory	
Property	
Item	
Framework	
Module	Framework
ModuleVersion	Module
TableGroup	
Table	
TableVersion	Table
TableAssociation	
Header	Table
HeaderVersion	Header
Cell	Table
Variable	
VariableVersion	Variable
CompoundKey	
Context	
Operation	
OperationVersion	Operation
OperationScope	Operation
Document	
DocumentVersion	Document
Subdivision	DocumentVersion
Release	
SubCategoryItem	SubCategory

Concepts: Translations

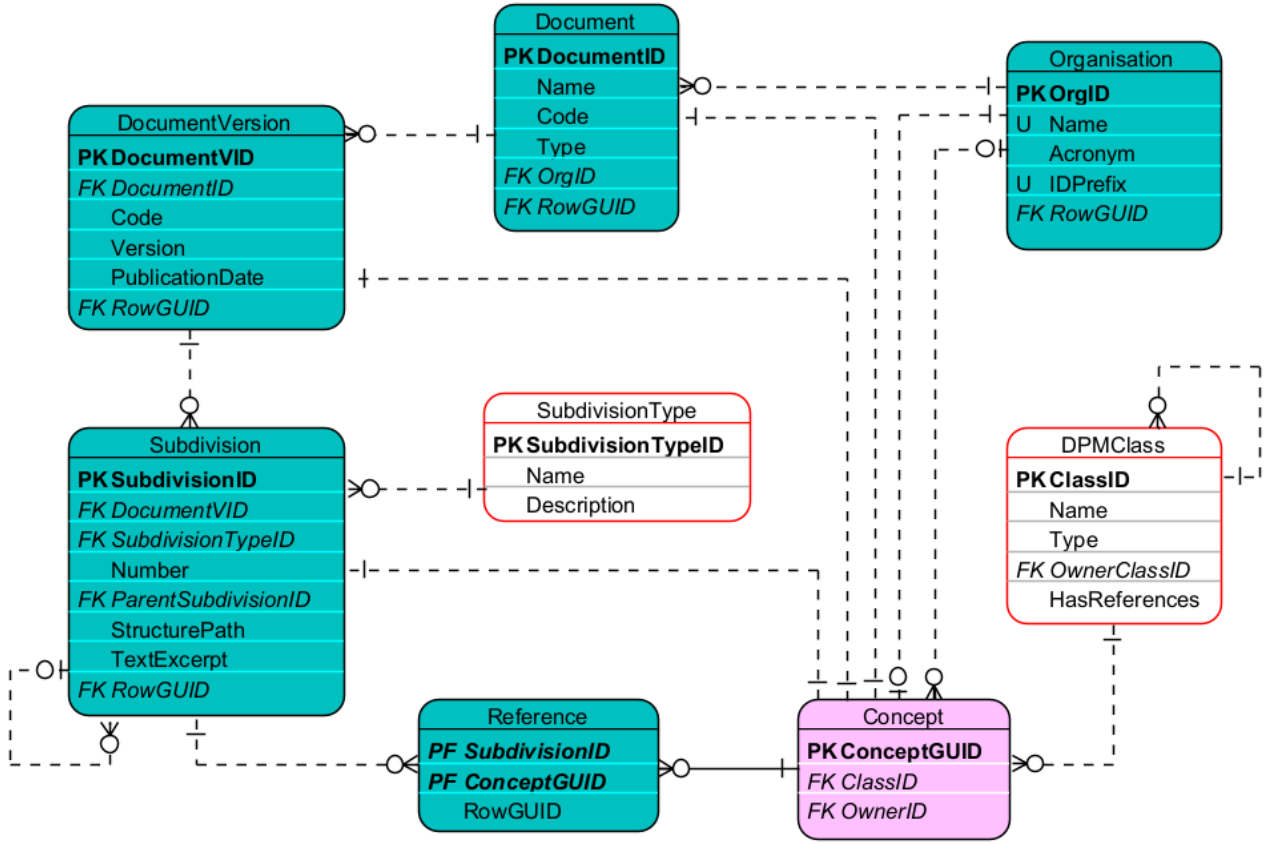


Enables translating various attributes (*Names*, *Descriptions*, *Labels*, *Values*, *Texts*, …) to various languages by various organisations (there can be multiple translations in one language by different organisations)

Enables representation of *Expressions* of *Operations* in various syntaxes/formats

Attribute	DPMClass
Name	Category
Description	Category
Name	SubCategory
Description	SubCategory
Name	Item
Description	Item
Label	SubCategoryItem
Name	Framework
Description	Framework
Name	ModuleVersion
Description	ModuleVersion
Name	TableGroup
Description	TableGroup
Name	TableVersion
Description	TableVersion
Name	HeaderVersion
Name	TableAssociation
Description	TableAssociation
Name	VariableVersion
Description	OperationVersion
Expression	OperationVersion
Name	Document
TextExcerpt	Subdivision
Name	Organisation
Acronym	Organisation
Description	Release

Concepts: Documentation/References



Ability to identify references of created concepts within any document structure

Handling legal references (e.g. to regulations, standards) but also change requests

Name	Description
Chapter	For a publication that uses chapters, this part should be used to capture this information. Because chapters are not necessarily numbers, this is a string.
Article	Article refers to a statutory article in legal material.
Section	Section is used to capture information typically captured in sections of legislation or reference documents.
Subsection	Subsection is a subsection of the section part.
Paragraph	Paragraph is used to refer to specific paragraphs in a document.
Subparagraph	Subparagraph of a paragraph.
Clause	Subcomponent of a sub paragraph.
Subclause	Subcomponent of a clause in a paragraph.
Appendix	Refers to the name of an Appendix, which could be a number or text.
Example	Example captures examples used in reference documentation; there is a separate element for Exhibits.
Page	Page number of the reference material.
Exhibit	Exhibit refers to exhibits in reference documentation; examples have a separate element.
Footnote	Footnote is used to reference footnotes that appear in reference information.
Sentence	In some reference material individual sentences can be referred to, and this allows them to be referenced.
URI	Full URI of the reference such as "http://www.fasb.org/fas133".
Requirement	A suggestion of a new model entry for consideration / to be addressed in the next releases.

Concepts: Relationships

Relating concepts: two or many

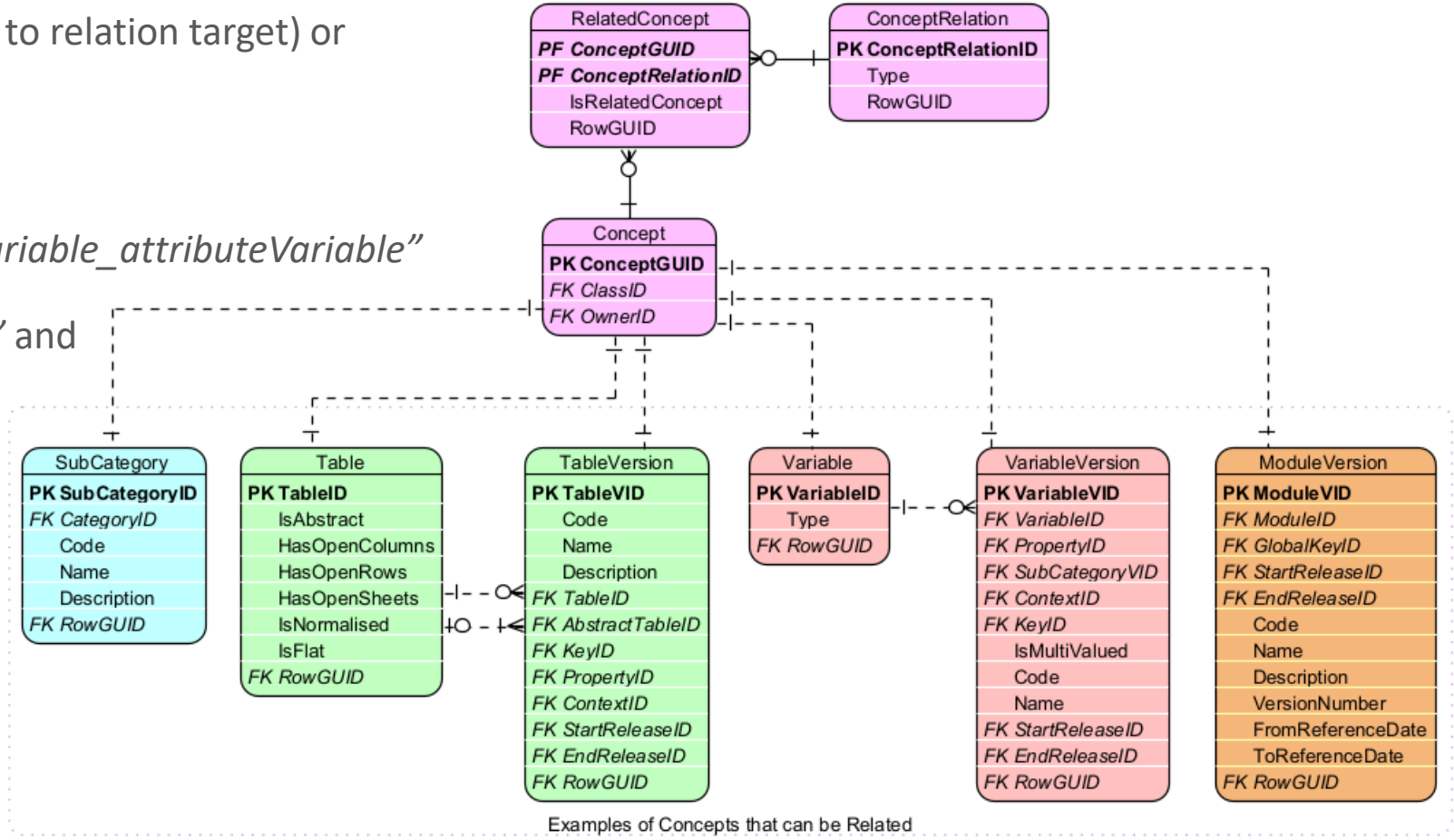
Relation types can be directed (*IsRelated* pointing to relation target) or undirected (working both ways)

Predefined relation types for:

- **Variables** - *"factVariable_keyVariable"* and *"variable_attributeVariable"*
- **SubCategories** - *"subCategoryMaster_version"* and *"subCategoryRendering_version"*
- **Tables** - *"table_variant"*

Generic relation types:

- *"equivalent_concept"*
- *"version_fix"/"version_new"*



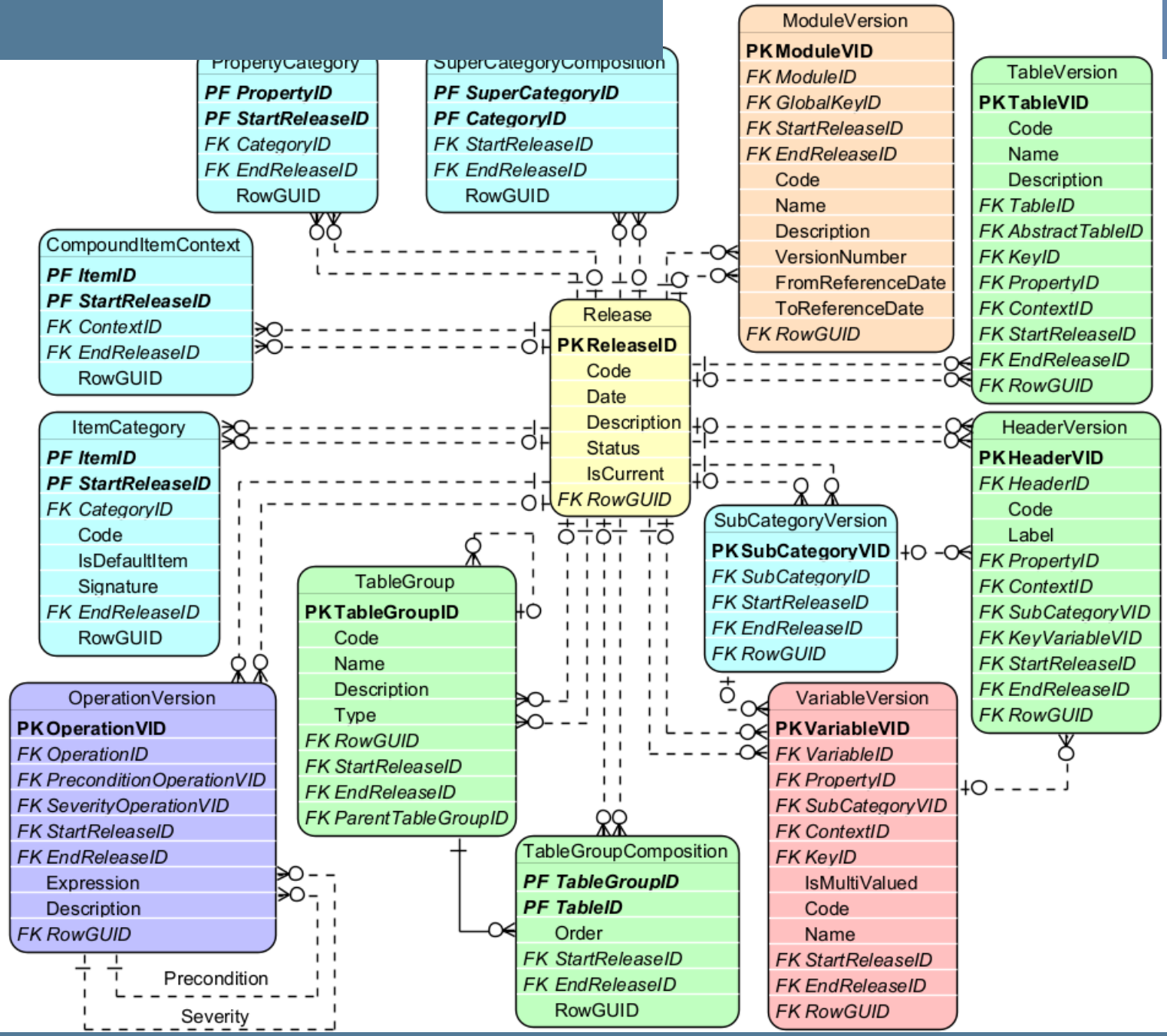
Historization: Releases

Release – publication of the model

Indicated on objects or connections between objects (to enable changing composition)

Start Release is mandatory, *End Release* is optional

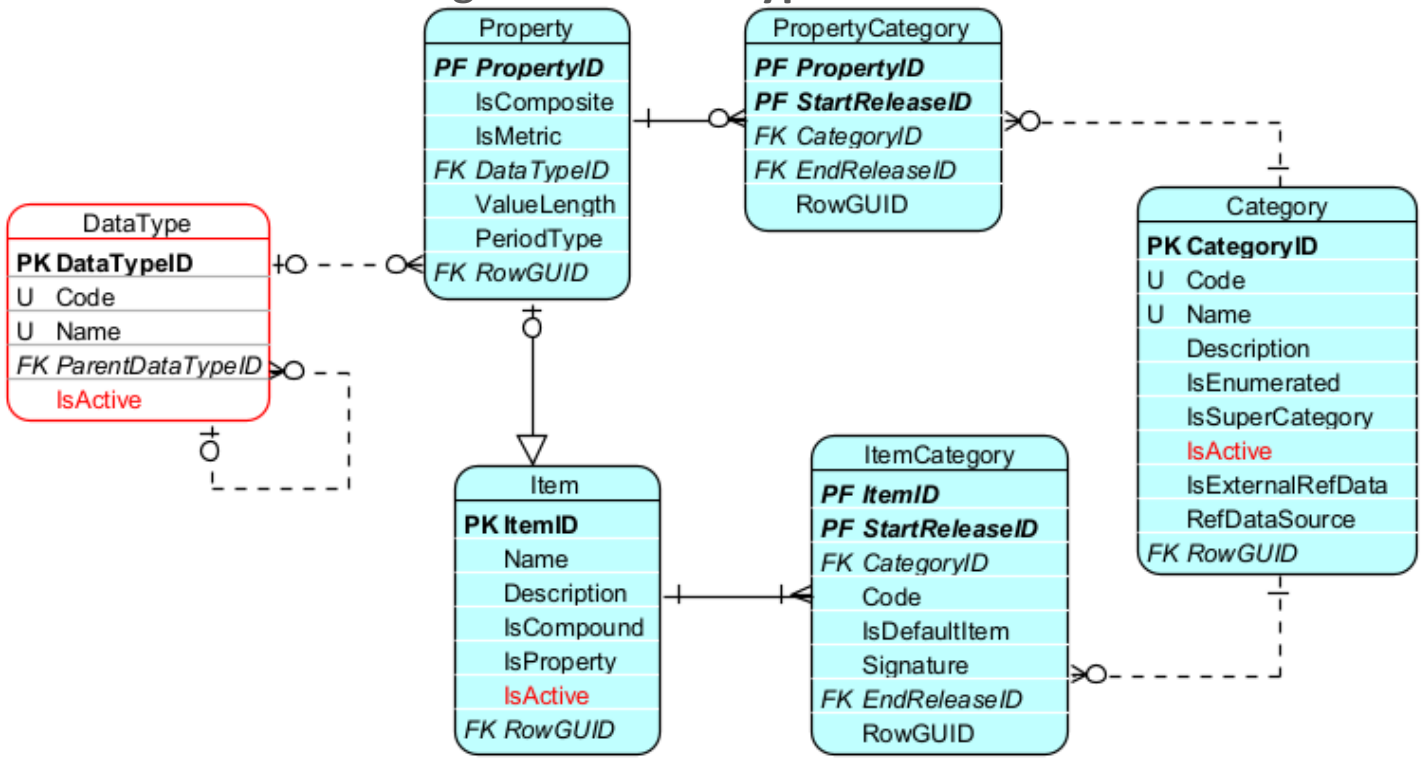
does not address model administration purposes (e.g. creation/modification dates), workflow/development process stages (IWD, PWD, …) neither it helps to handle indication of temporary metadata (work-in-progress for internal comments/feedback)



Historization: Deactivation

Categories and **Items** (therefore also **Properties** for this a counterparty **Item** is created) can be marked as deactivated (e.g. no longer to be used)

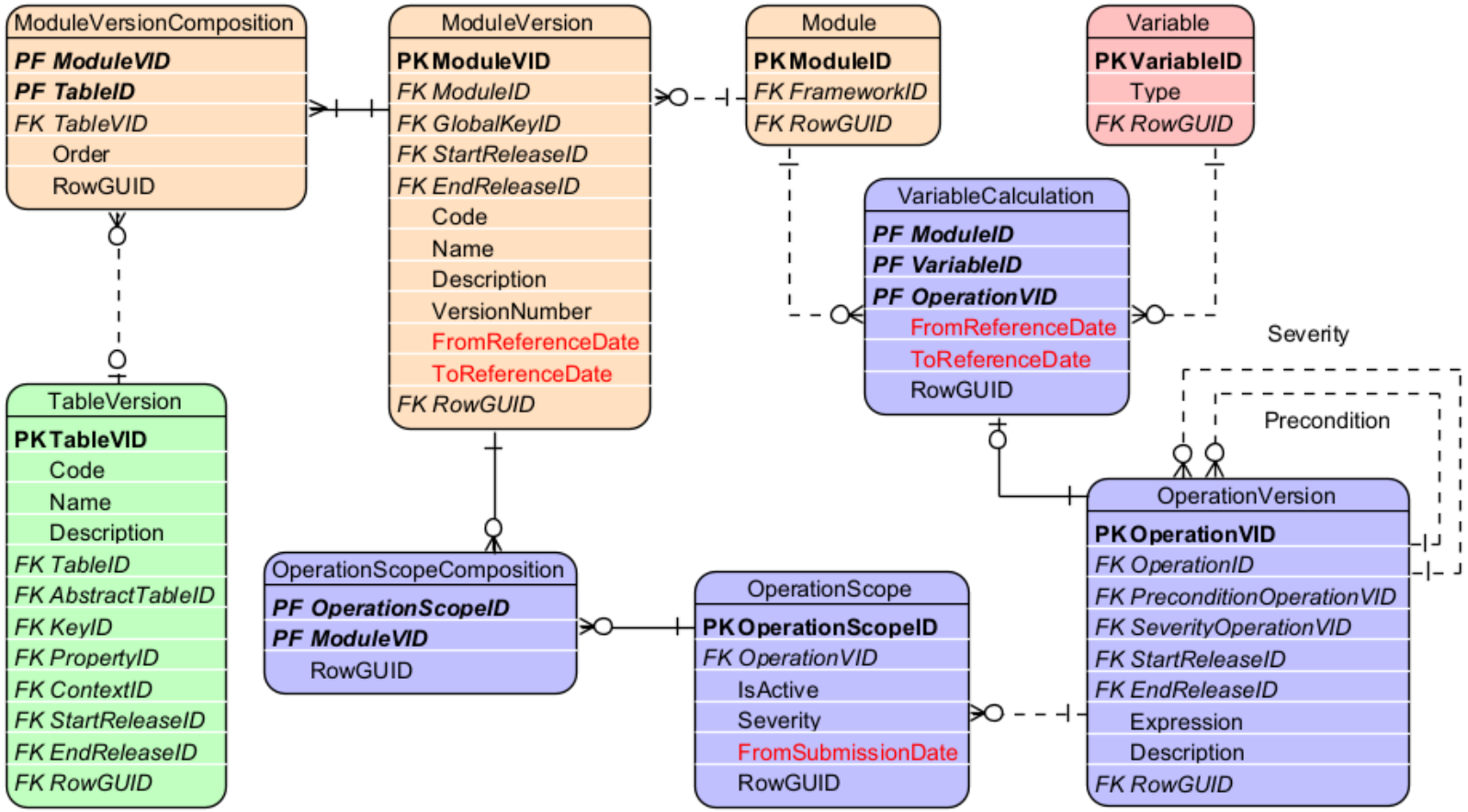
Metamodel authors can decide on deactivating certain **DataTypes**



Historization: Application Dates

ModuleVersions may have their own life cycle with application dates at some point in the future (e.g. module applicable starting from next year)

OperationVersions can be activated or deactivated (or have their severity modified) starting from specified submission date (i.e. applicable to all reports exchanged after certain day)

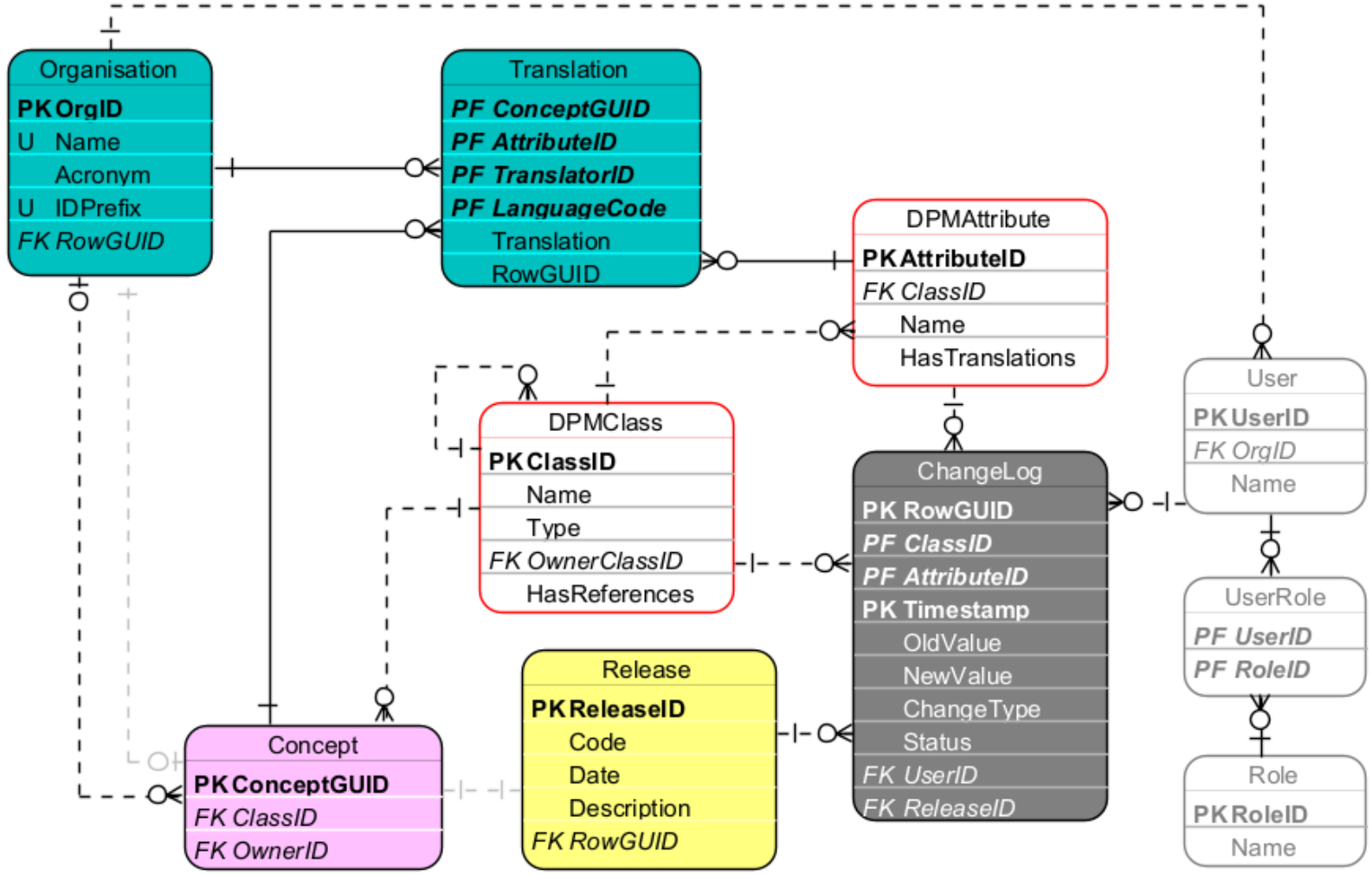


Change Log (non-normative)

non-normative: not aimed to become part of the DPM standard

stores modifications made to the content of model entities or their attributes (when, what, who, why)

implementation aspect: *RowGUID* on every model entity to enable single reference



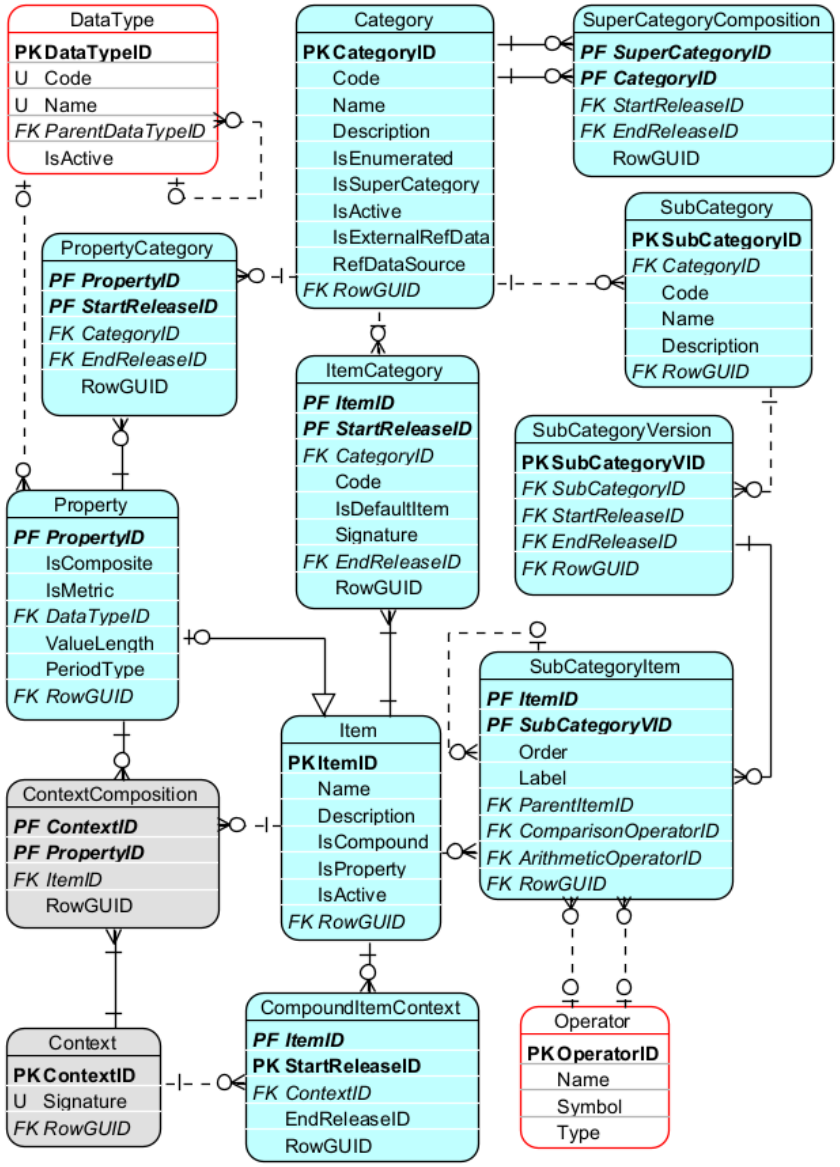
Glossary: Overview

Mapping to current DPM:

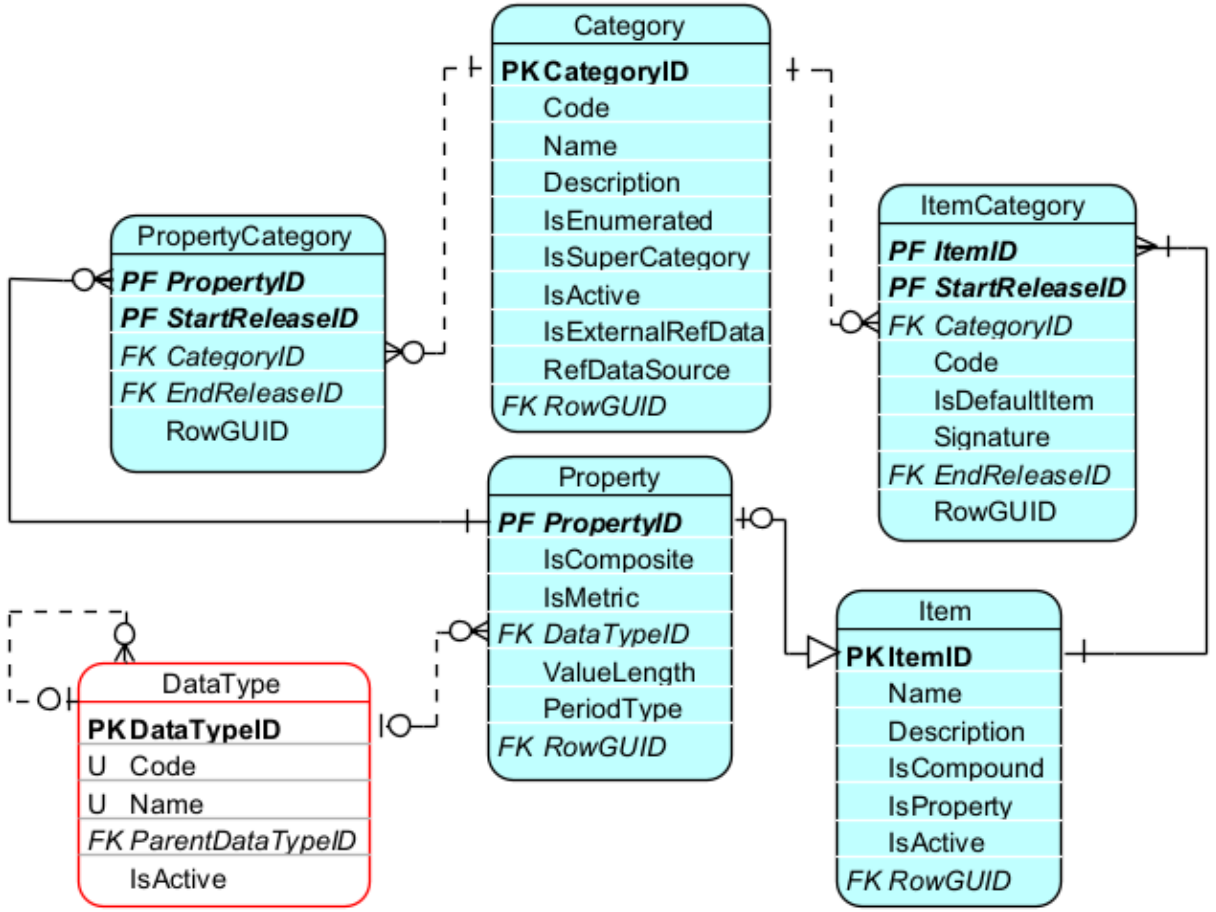
- **Domain ~ Category**
- **Metric ~ Property**
- **Member ~ Item**
- **Dimension ~ Property**
- **Hierarchy ~ SubCategory**

New:

- **Super Category**
- **Compound Item**
- Historisation (registering changes in composition of **Categories**, versioning of **SubCategories**, …)



Glossary: Categories, Properties and Items



Categories help organising Glossary:

- *Enumerated* **Categories** group **Items** that share common nature/semantics
  - **Item** can change **Category**
- *Not-enumerated* **Categories** gather data-type-constrained **Properties** whose values are impractical or impossible to enumerate (e.g. volatile)

Properties:

- quantitative metrics (*IsMetric*) - identify what is measured, determine data type and time duration of observation
- qualitative - provide perspective/characteristic; enumerated properties contextualise **Items**
- **Property** can change **Category**

In physical implementation: each ***Property*** has its counterpart **Item** (enables arranging **Properties** in **SubCategories** to be used as dropdowns)

Glossary:

ISIN	CUSIP	SEDOL	Amount
…	…	…	…

vs

Code Type	Code Value	Amount
	…	…

ISIN
CUSIP
SEDOL

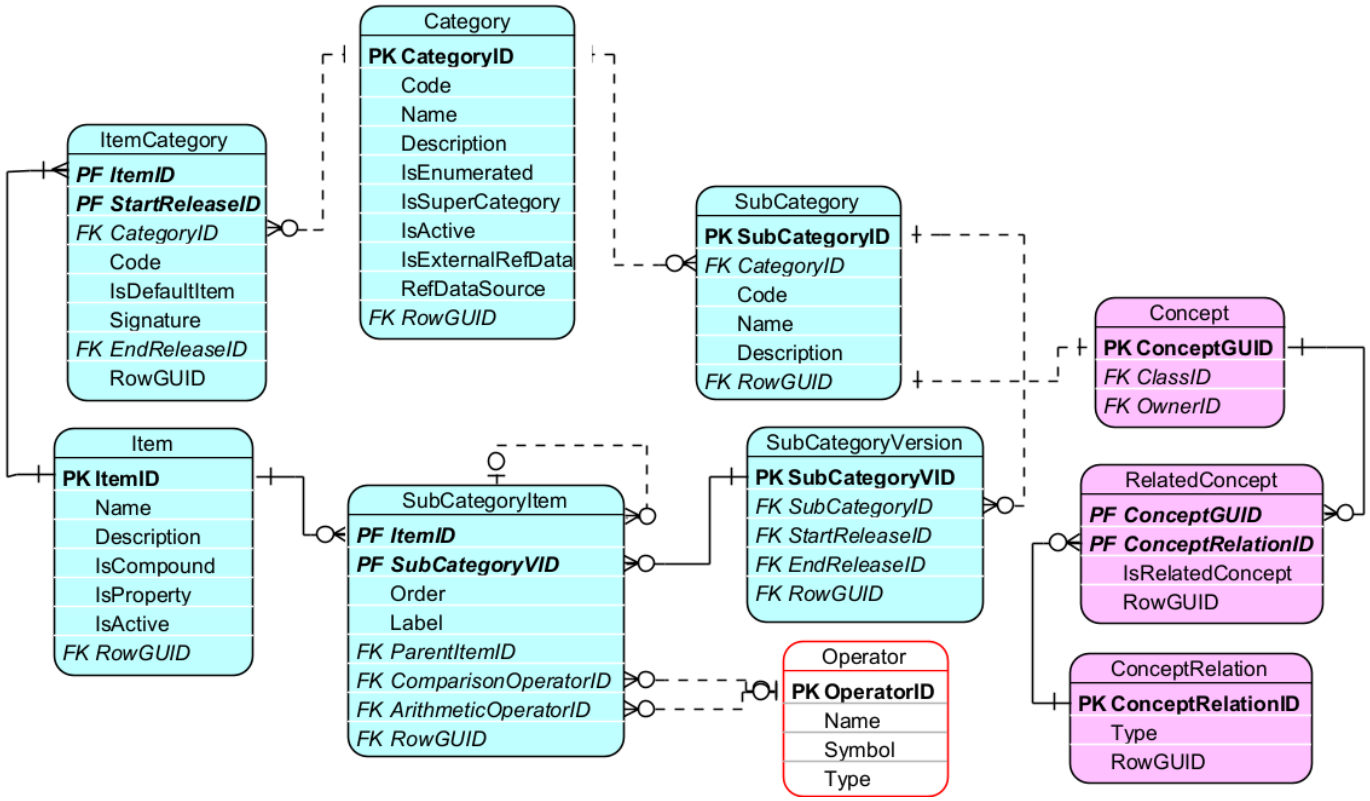
Open and granular data challenges:

How did you learn about this?
(…)
Google
Facebook
In person events
A friend
Newspapers

Property	Data type	Category	Enumerated	Items	SubCategory
Location of activity, Counterparty residence	enumeration	Countries	yes	Poland, Spain, Greece, France, China…	
ISIN, CUSIP, SEDOL, …	string	Instrument codes	no		
		Properties	yes	Location of activity, Counterparty residence, ISIN, CUSIP, SEDOL, Source of information, Name of subsidiary, Carrying amount, Type of instrument code, ...	1: ISIN/CUSIP/SEDOL/…
Source of information, Type of instument code, …	various	Not applicable	no (no meaning)	From Google, From Facebook, From in person events, From a friend, From newspapers, …	1: Google/Facebook/In person event/…
Carrying amount, Name of subsidiary, …		Amount types, Not applicable, none			

Code	Name	Description
_PR	Properties	Contains *Items* which are counterparts of *Properties* in physical implementation of the DPM metamodel.
_NA	Not applicable	Contains *Items* which do not belong to any specified *Category* such as those that are typically used only in dropdowns on *Headers* or *Variables*. It is also linked by *Properties* that do not belong to any real *Category*. Such enumerated *Properties* can refer to *Items* from *Categories*: *"Not applicable"*, *"Properties"* and in such case they can also use *Items* of other *Categories*, in particular by being linked to such mixed *SubCategory*. It is not a semantically meaningful *Category* therefore it is neither enumerated nor not-enumerated.
_TE	Templates	Contains *Items* which represent *Templates* (*TableGroups* or *Tables*) for purposes of resembling Filing indicator *Variables*.

Glossary: Sub Categories



SubCategories are lists of **Items** but enable also their arrangement in hierarchies (if needed, e.g. to reflect nesting or arithmetic relationships – by reference to *Operator*)

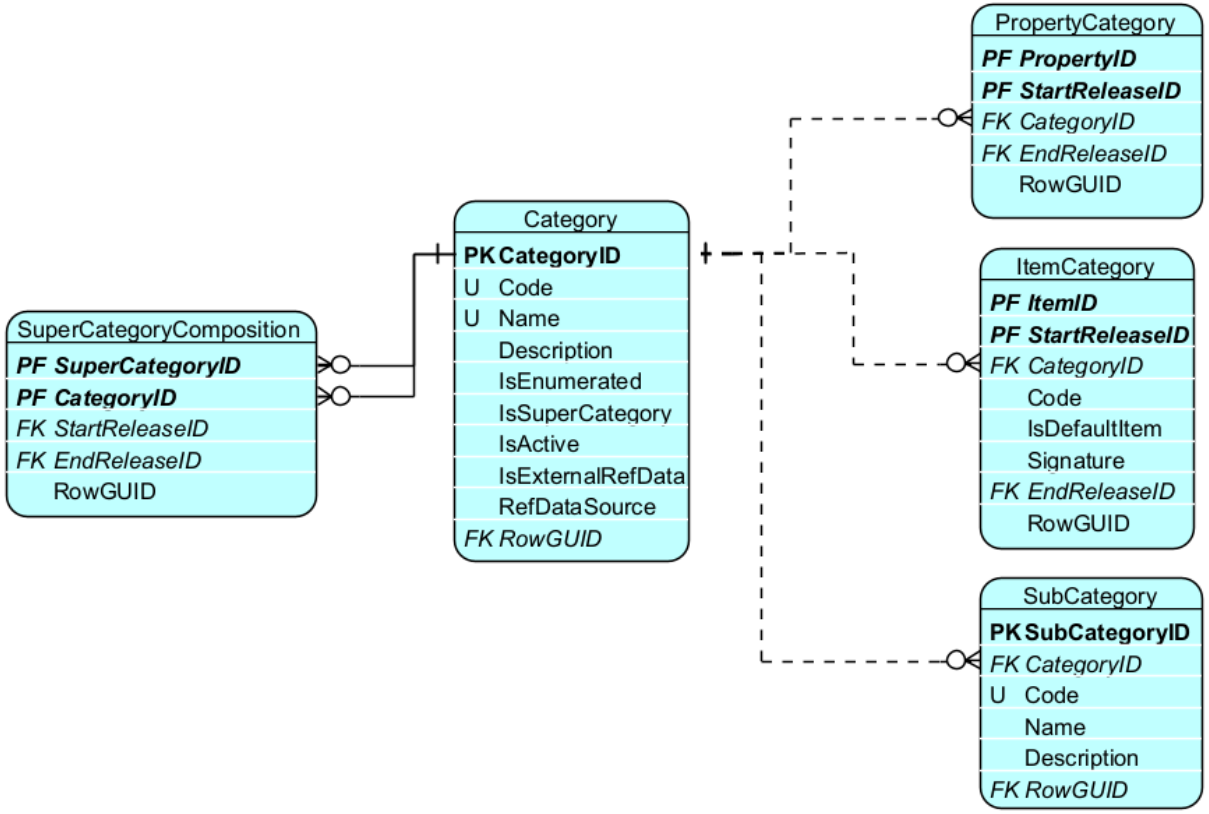
- document and put **Items** in order
- can be used as dropdowns by **Table Headers** (and hence also **Variables**)

SubCategories can change composition (through **SubCategoryVersion**)

SubCategories may be related to one another (via **ConceptRelation**)

- e.g. a *"list of all countries"* can be indicated as a "master"/"full version" for the regional/topic oriented subsets (which may need to be updated once the "master" list is updated)
- For rendering purposes (ability to show a complete breakdown from which only certain options are selectable);

Glossary: Super Categories



SuperCategory is a union of **Categories**

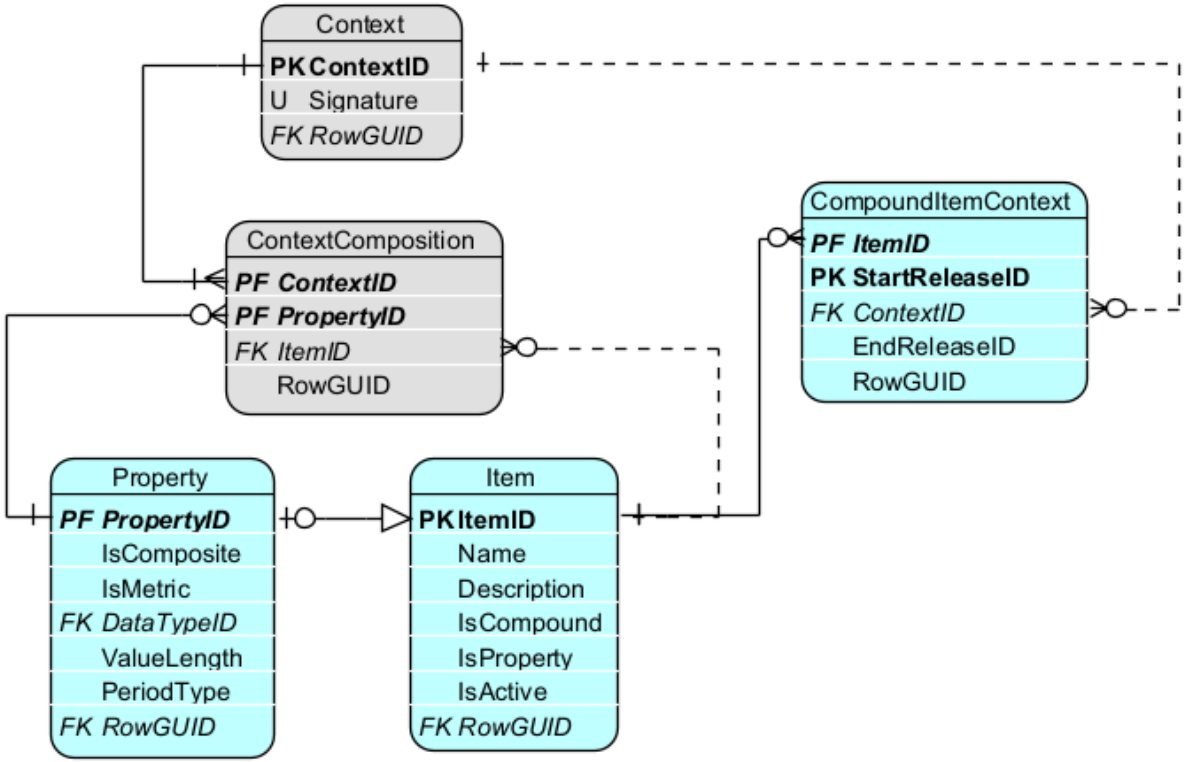
- e.g. frequently requested mix of breakdowns by *"Countries"* and *"International organizations"* (i.e. *IMF*, *BIS*, *ECB*, … along *countries* for exposures/risk classification),
- can combine enumerated and non-enumerated **Categories**

reuses *Items* of other **Categories** (or even **SuperCategories**) and can introduce its own *Items* (e.g. for "total")

may have its own **Properties**

composition of **SuperCategory** is versioned (can change between **Releases**)

Glossary: Compound Items



Compound Items

- are composed of *Property-Item* pairs – simplified representation of complex terms

Instrument type	Treasury bills
	=
Instrument type	Debt securities
Issuer sector	General Government
Original maturity	< 1 year

- they may belong to existing **Categories** (e.g. of one of the contributing **Items**)
- their composition is versioned

implementation reuses **Context** and its **Composition**

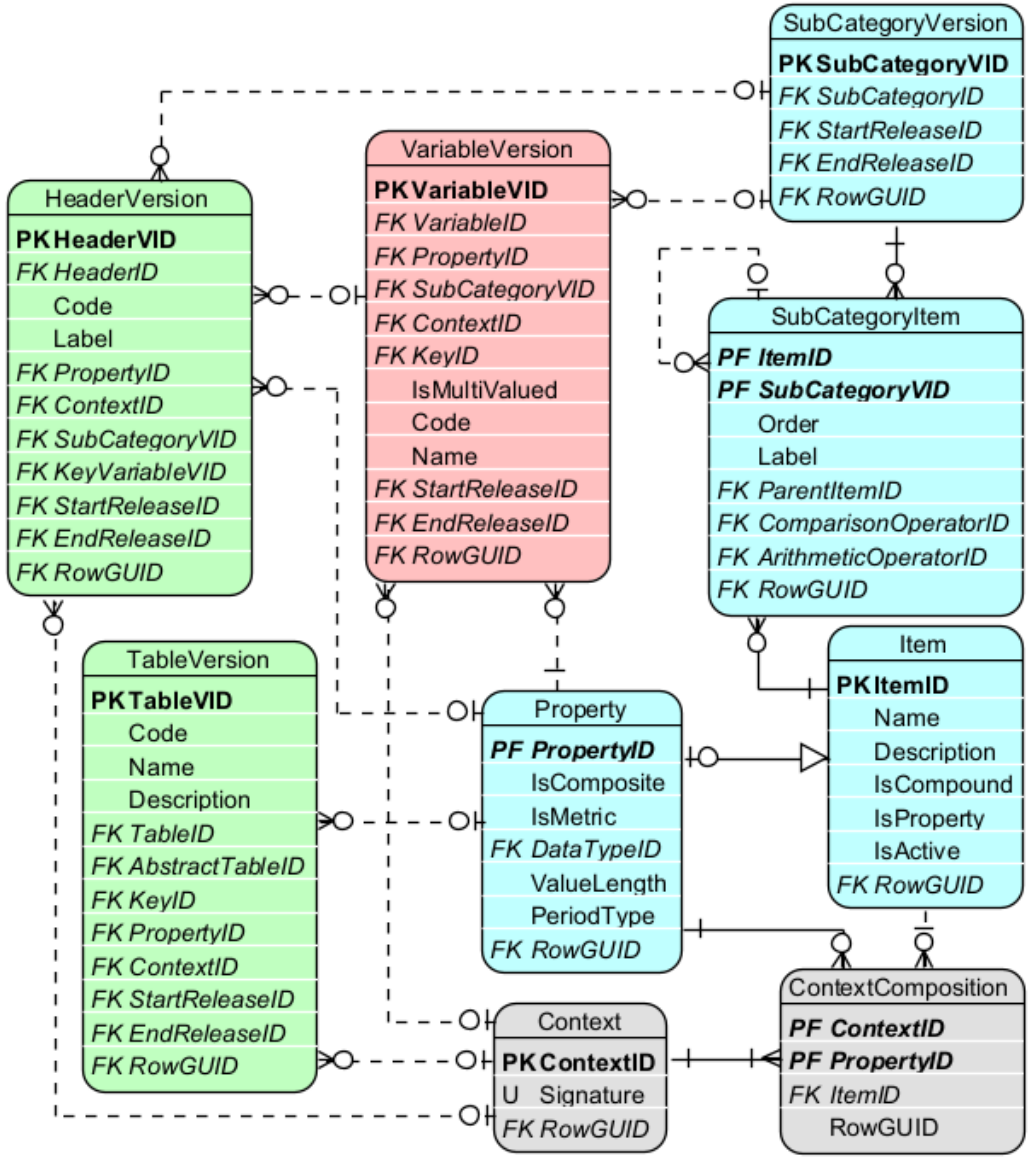
Glossary:

Properties are used on

- **TableVersion** (applicable to entire table)
- **HeaderVersion**
- **Variable** (mandatorily)

For dropdowns the above are associated to **SubCategoryVersion**

Optionally, *Property-Item* pairs gathered in **Contexts** providing additional metadata can be associated to any of the above



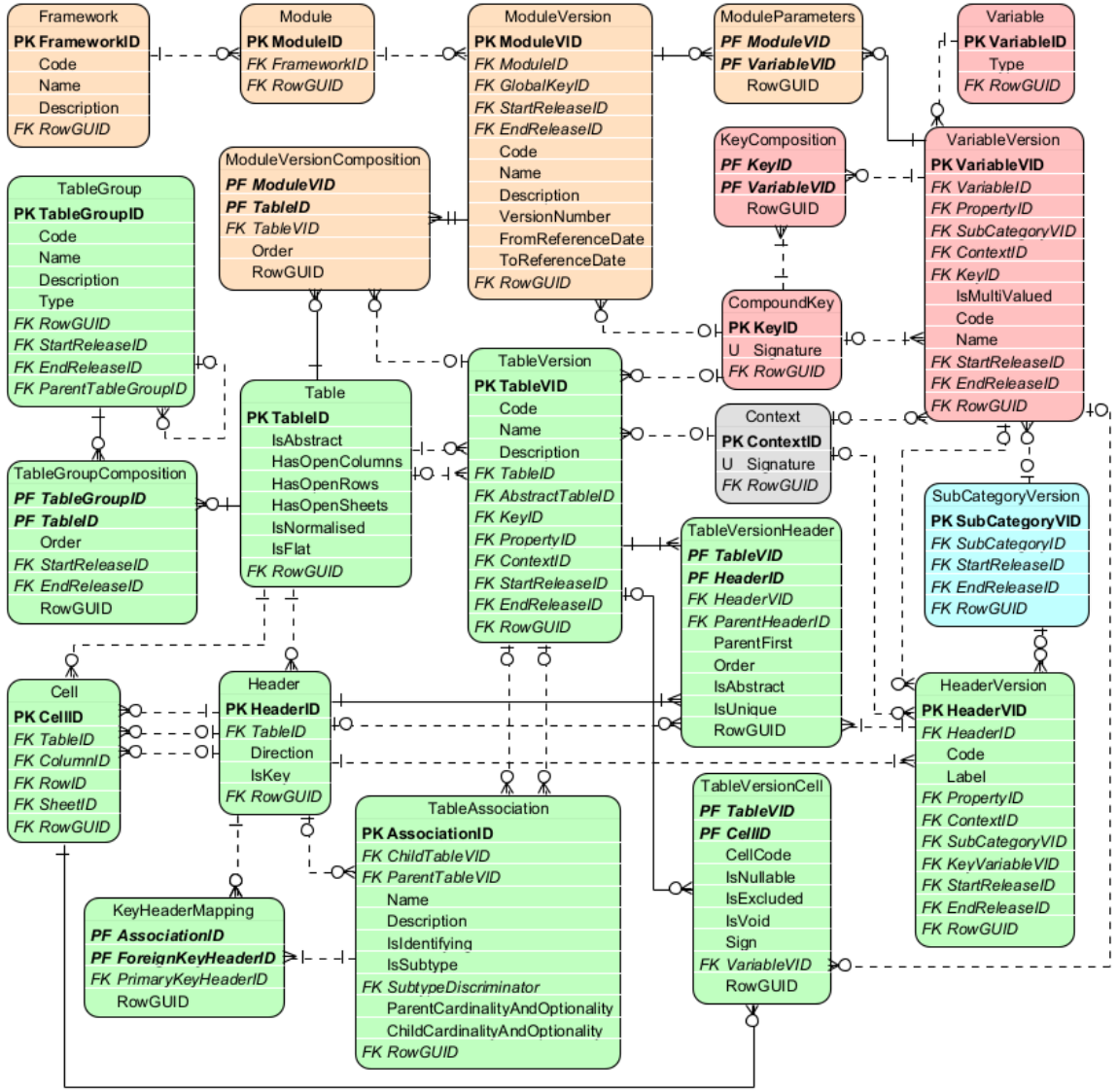
Representation of information requirements

Rendering: **Tables, Headers, Cells, Association** and **KeyHeaders** mapping

Grouping: **TableGroups**

Identification of information requirement: **Variables** (Key, Fact, Attribute, Filing indicator and their linkage)

Packaging: **Frameworks, Modules**

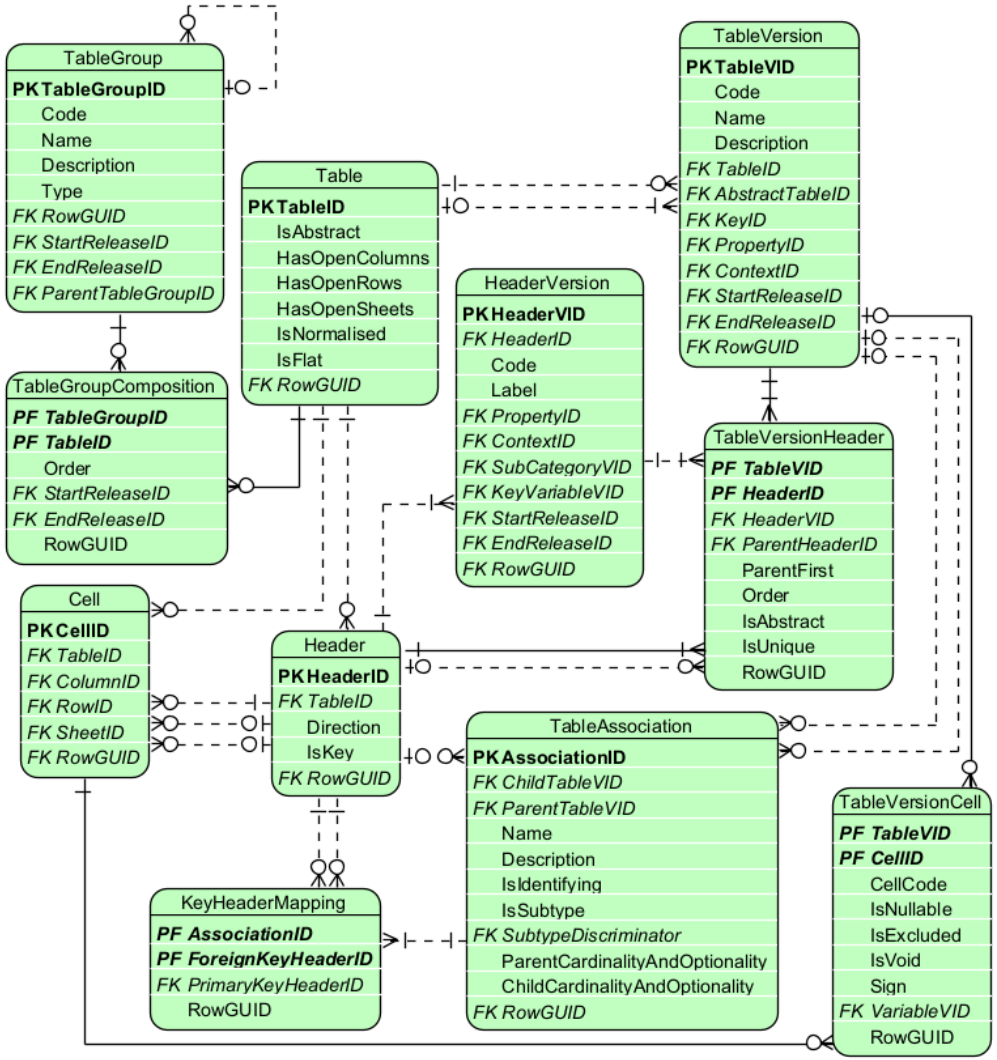


Rendering: overview

Tables have **Headers** (**~Ordinates**) and **Cells**

Tables can be grouped

Tables can be associated to one another



Rendering: table versions and headers

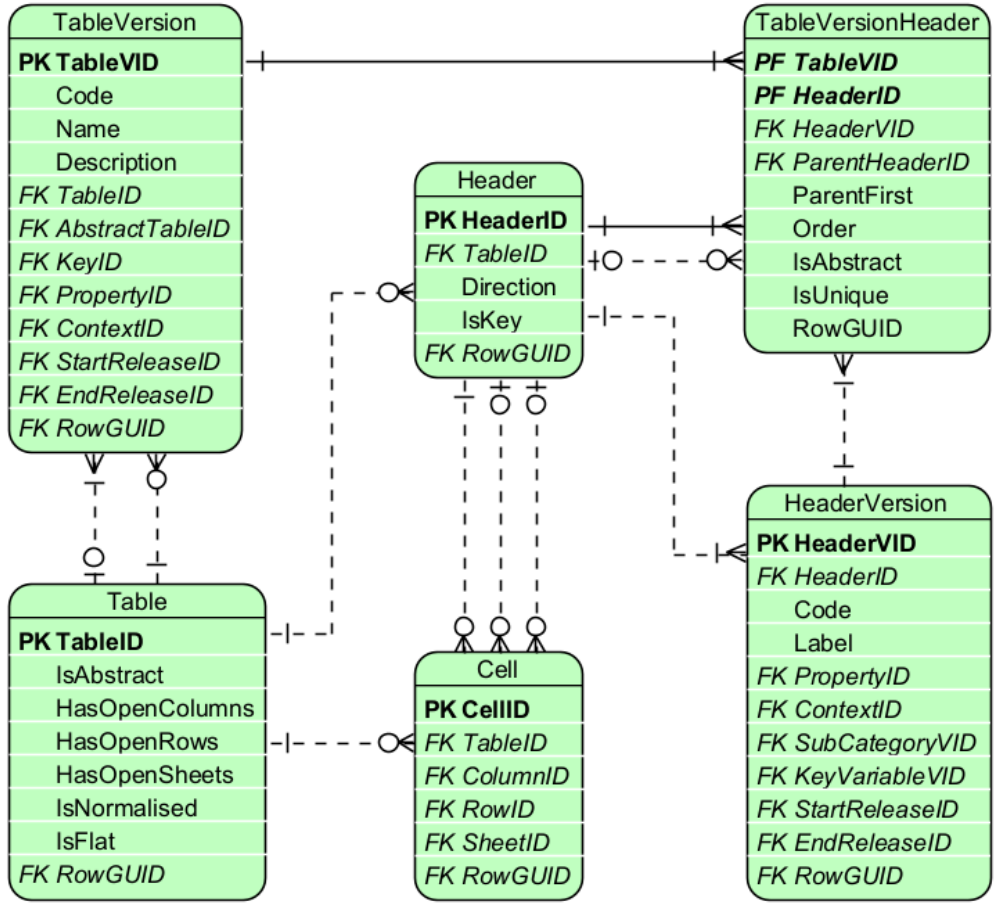
TableVersion may link to *Property* and/or *Context* (*Property-Item* pairs) applicable/common to/shared by all *Table Headers* and *Cells*

Headers Direction can be *Rows/Columns/Sheets* - one set of related headers per disposition (note: typical open tables require only columns)

HeaderVersion:

- contribution to definition of a *Fact Variable*: column/row/sheet *Headers* may point to *Property* and/or *Context*
- definition of a *Key Variable*
  - points to *Property* and may point to *SubCategory*
  - does not link/result in a *Cell*
  - links to *VariableVersion*

Implementation aspect: introduction of **TableVersionHeader** enables modification of headers structure without a need for duplicating objects for the entire table (all its *Headers*, *Cells*, etc)

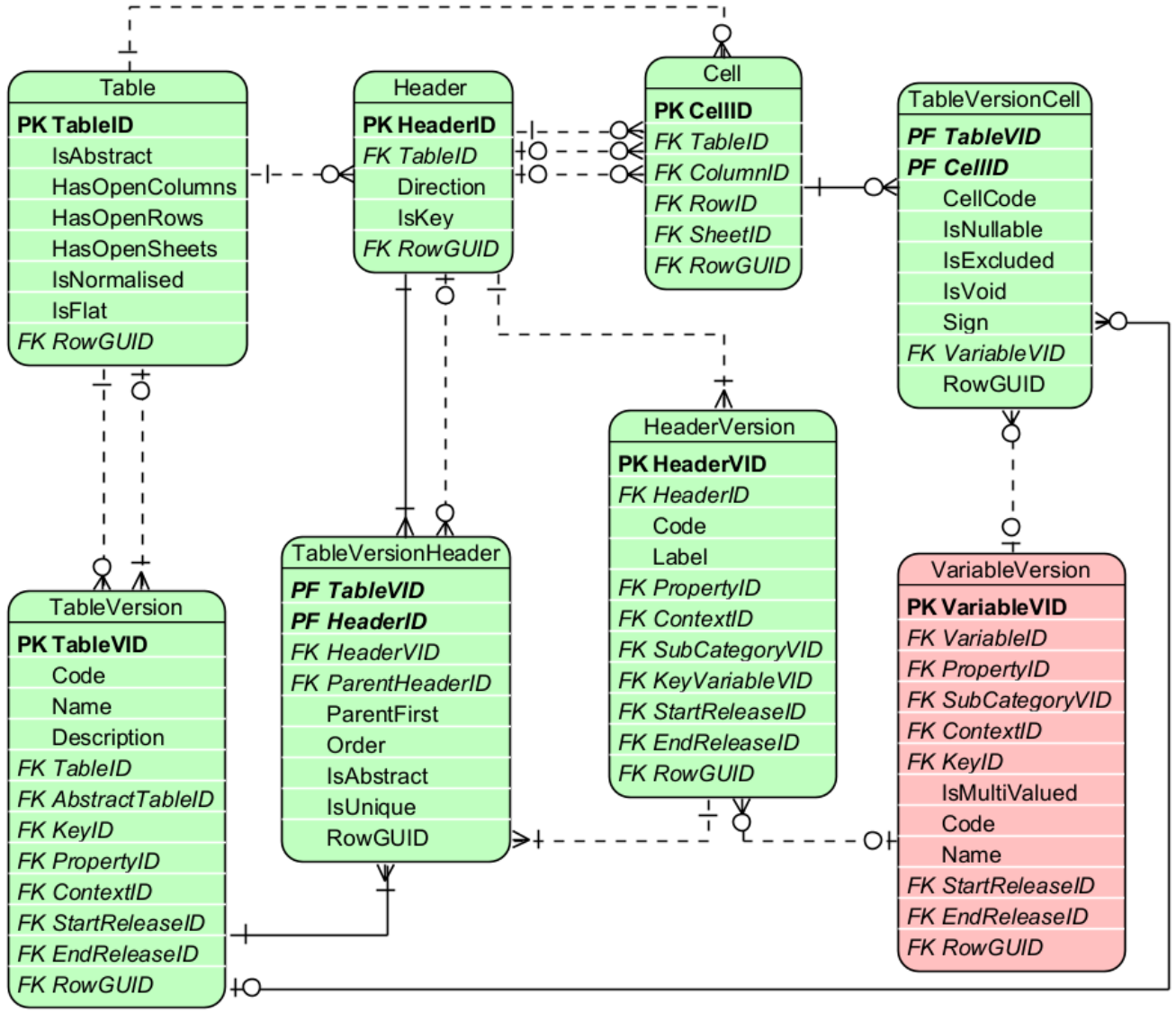


Rendering: cells

Cell points to at least one leaf-level *Header* (e.g. a column in open rows table) but may also point to two or maximum three (one for each: *Column, Row* and *Sheet*)

on **TableVersionCell** it can be indicated if a cell is mandatory or excluded (this information may change for a table between table versions/releases)

TableVersionCell definition results from intersection of *Headers* for this *Cell* (including inheritance) and points to *Fact Variable* unless combination of characteristics from this intersection is illogical (e.g. *"Equity instruments"* issued by *"Government"* with certain *"Maturity period"*)

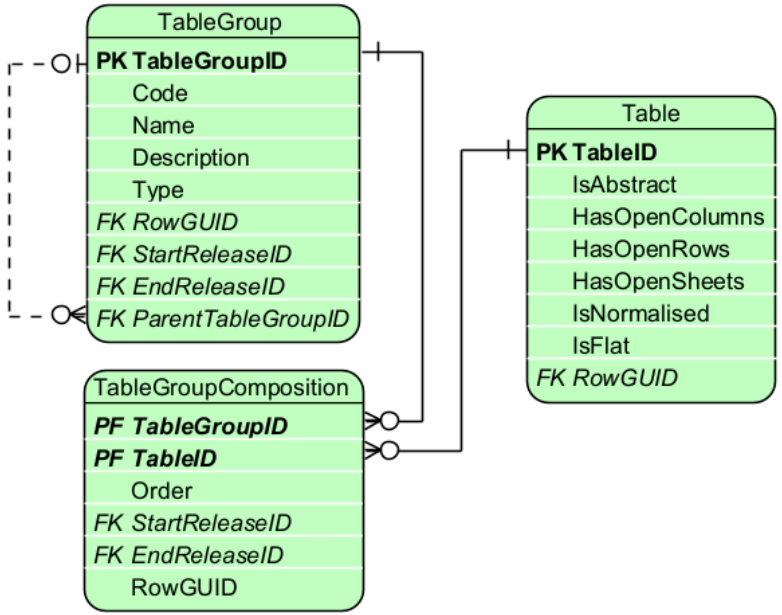


Rendering: Table Groups

TableGroups may be created for various purposes as indicated by *TableGroup.Type*:

- *"templateGroup",*
- *"template",*
- *"templateVariant",*
- *"templateScope".*

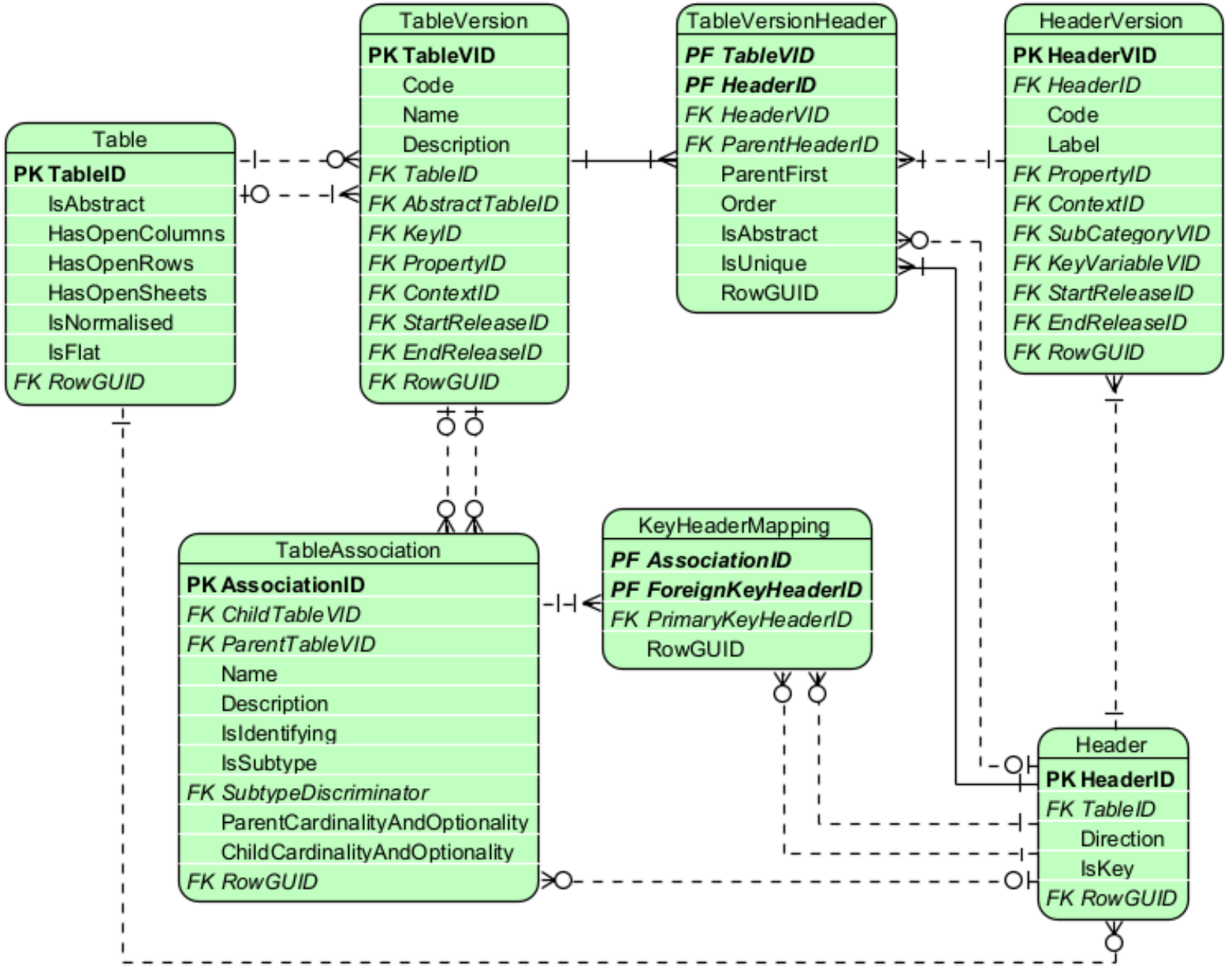
TableGroups of *Type "templateScope"* can also be nested: DPM XL syntax enables using their *Codes* in *Operation.Expression* instead of *Table Codes*



Rendering:

Non-normalized **Tables** do not need to be modelled when they are *IsAbstract*, in which case they are broken down in two or more modelled **Tables** (abstract **Tables**, if modelled, do not contain links to *Glossary* terms, just pure rendering)

Tables can be *Associated* to one another, indicating *FKs* mapping, *Cardinality, Subtyping* (with Discriminator), etc.



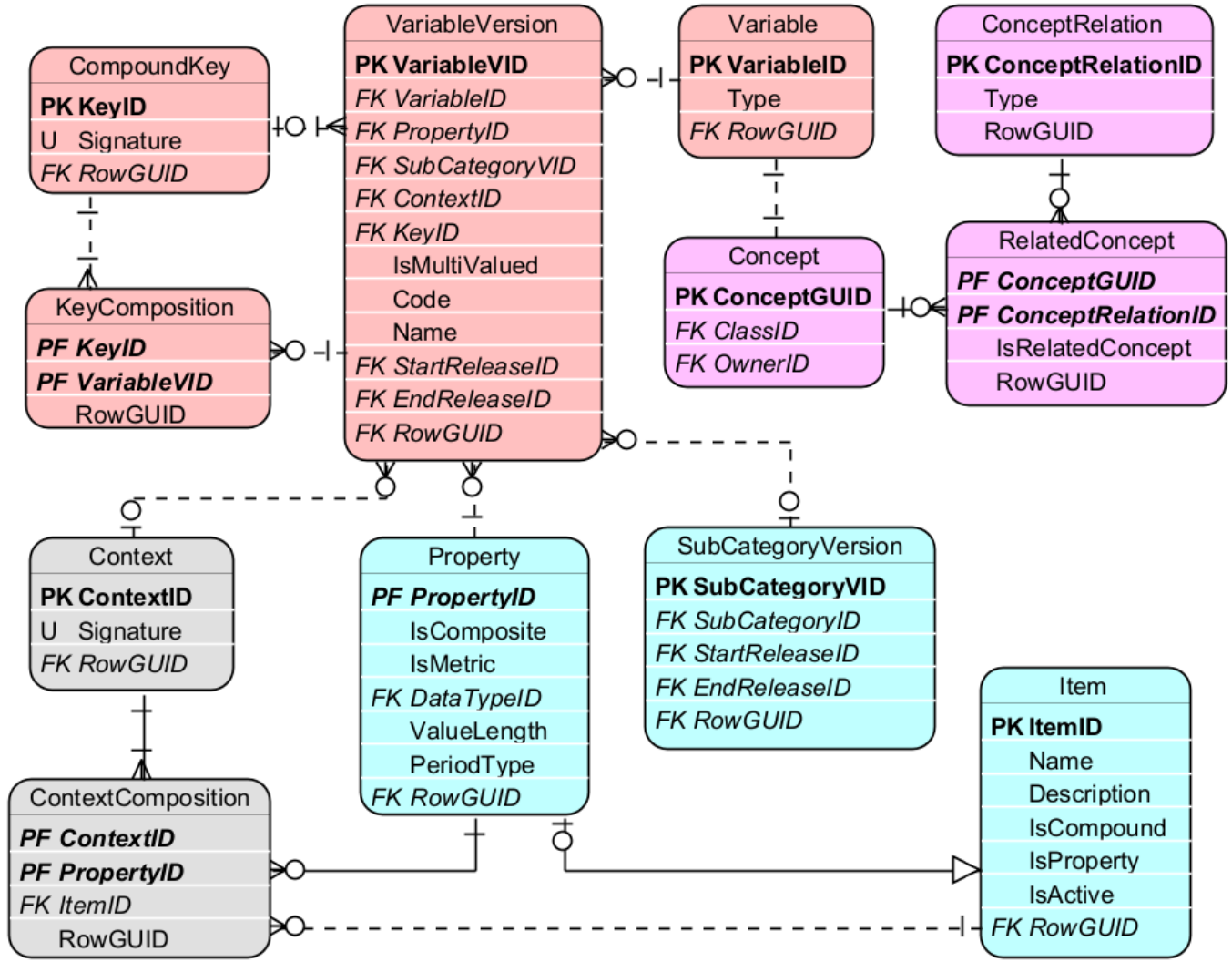
Information requirements: role of variables

Variable represents each distinct reportable value (regardless of its occurrence in rendering) – quantitative or qualitative

- types:
  - **Fact** (monetary amount, text, enumeration, …) ~DataPoint
  - **Key** (e.g. column identifying each row in an open table)
  - **Attribute** (unit of measure, value precision/accuracy, a comment of a Fact or a Key)
  - **Filing indicator** (indicator that certain subset of information, typically a Table or a set of Tables is intentionally contained or not contained in a report)
- *ConceptRelations*: *Variable-Attribute*, *Equivalent-Variable* (same meaning but modelled differently), …

VariableVersion:

- must refer to *Property*
- may refer to *SubCategoryVersion* (list of possible *Items*) or to *Context* (composed of one or many *Property-Item* pairs)
- composition may change between *Releases* (changes in modelling, fixing of bugs, etc)



Information requirements: definition (derivation) of

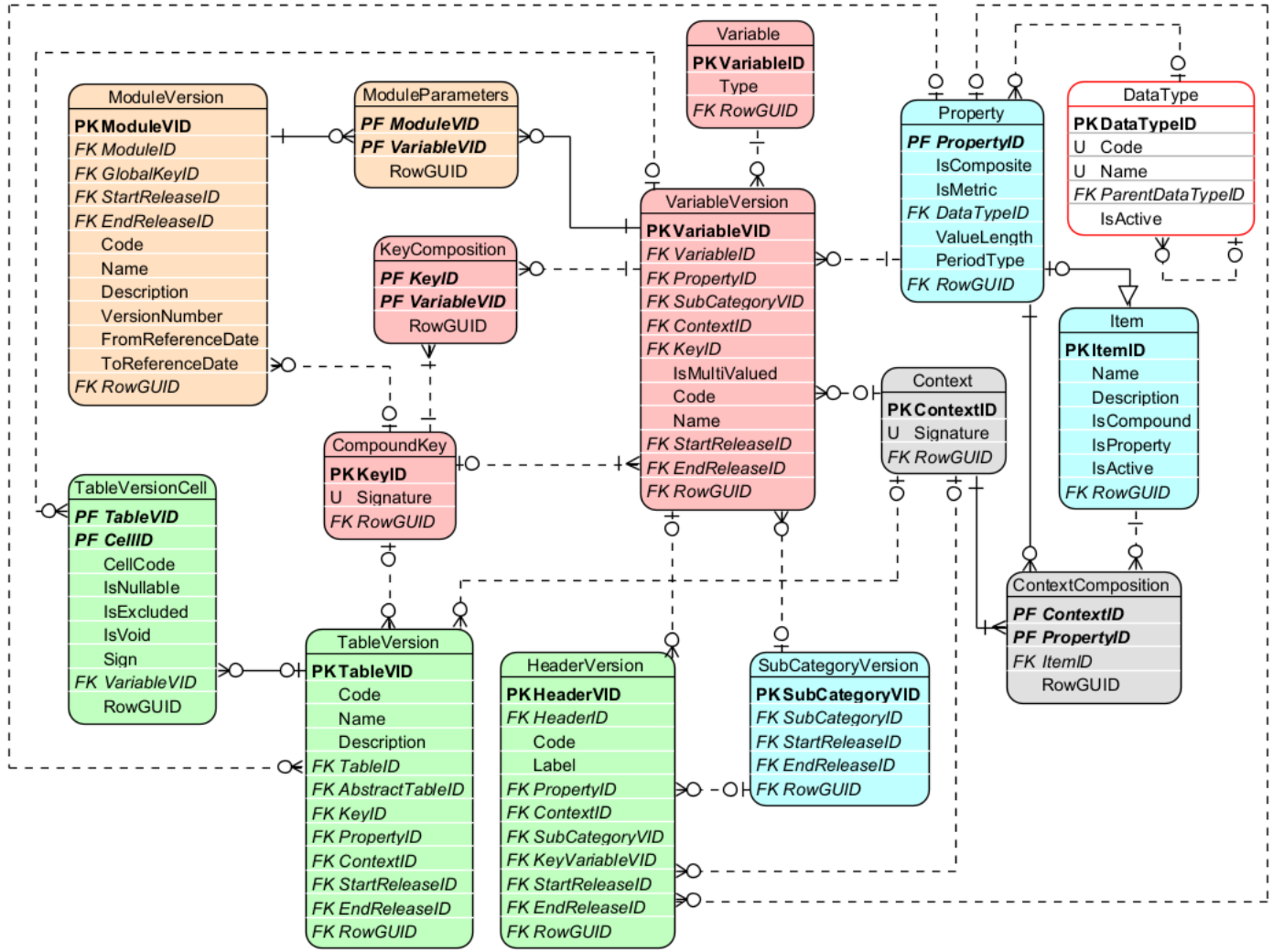
Fact Variables can be derived from *Table Cells*: *Table* level glossary terms contribute to definition of *Table Headers* (where information can be inherited from upper levels) and is further propagated to *Cells* on intersection of these *Headers*

Key variables are derived from *Headers* which have no cells attached (this is to ensure open/semi-open sheets and columns are modelled in the same way)

- *TableVersion* refers to *Key* that though *KeyComposition* identifies *Variables* that are keys in a given table
- *KeyComposition* arranges *Keys* in sets to support rendering-less models

Filing indicator variables are derived from *Cells* of *Tables* using *Items* of dedicated *Category*

ModuleParameters apply globally for *Module*, for example **Attribute Variables** such as report currency and period/date



Packaging: frameworks, modules

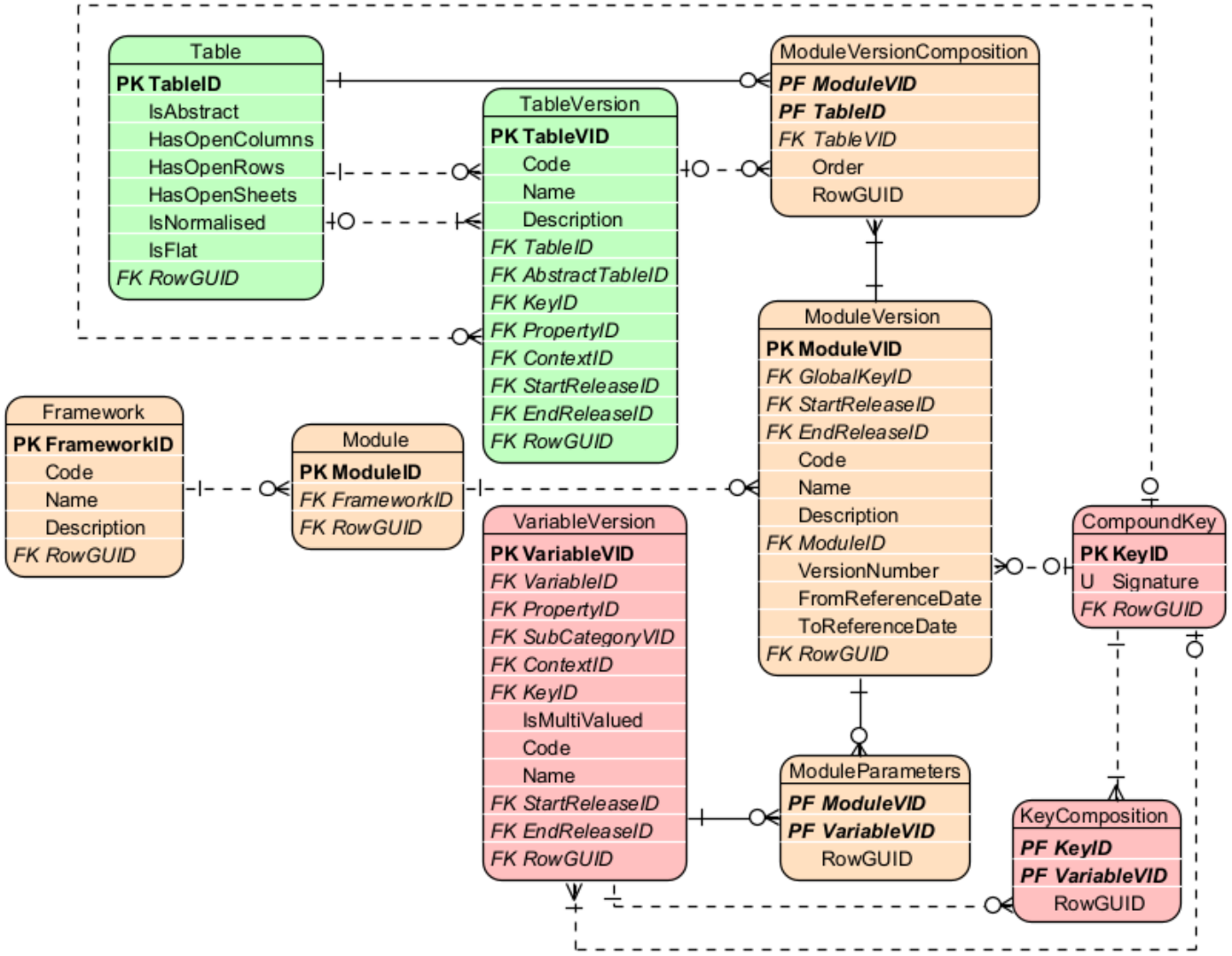
Framework – regulatory reporting requirements organized by subject/thematical area

Module – set of information requirements to be reported together (in one report)

ModuleVersion

- *FromDate* and *ToDate* identify its application dates (may differ from model publication dates i.e. releases)
- Gathers *Tables* (their *Versions*) that are required to be reported together

ModuleParameters are *Variables* (e.g. *Attribute* or *Key*) applied globally to all *Facts* (note: in case these global characteristics are *Property-Item* pairs, *Item* is the only option on **SubCategory**)

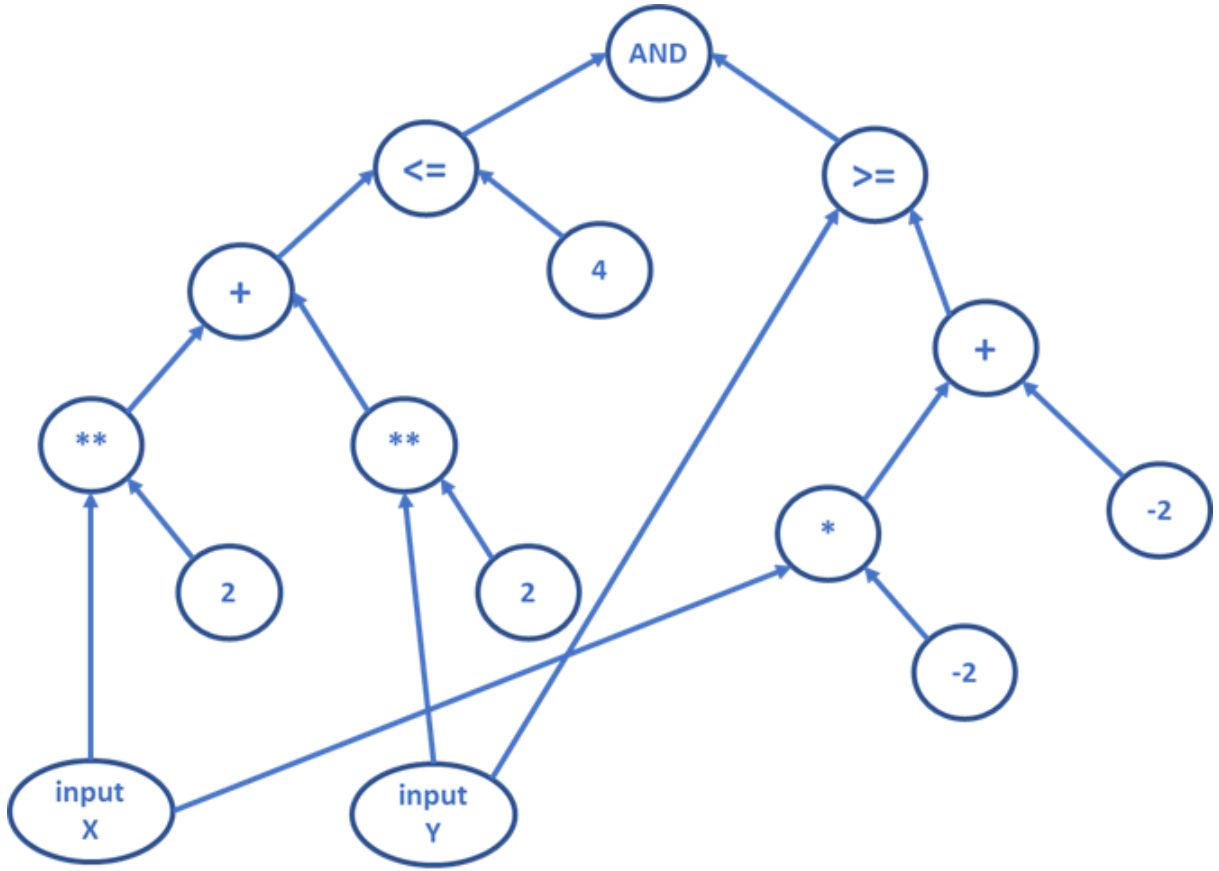


Operations: principles and assumptions

reflected in metamodel using tree structure (AST) comprising of nodes - operators and operands, the latter referring to terms from glossary, rendering, variables (incl. external sources)

syntax agnostic: able to result from and in different syntaxes

automatically populated (parsed from any/some syntax or defined and reviewed in graphical interface by users) and automatically consumed (to produce rules in e.g. XBRL, SQL, VTL, …)



Operations: representation

Operations enable definition of **data quality checks** as well as **data derivation rules** (transformations and computation of Variables)

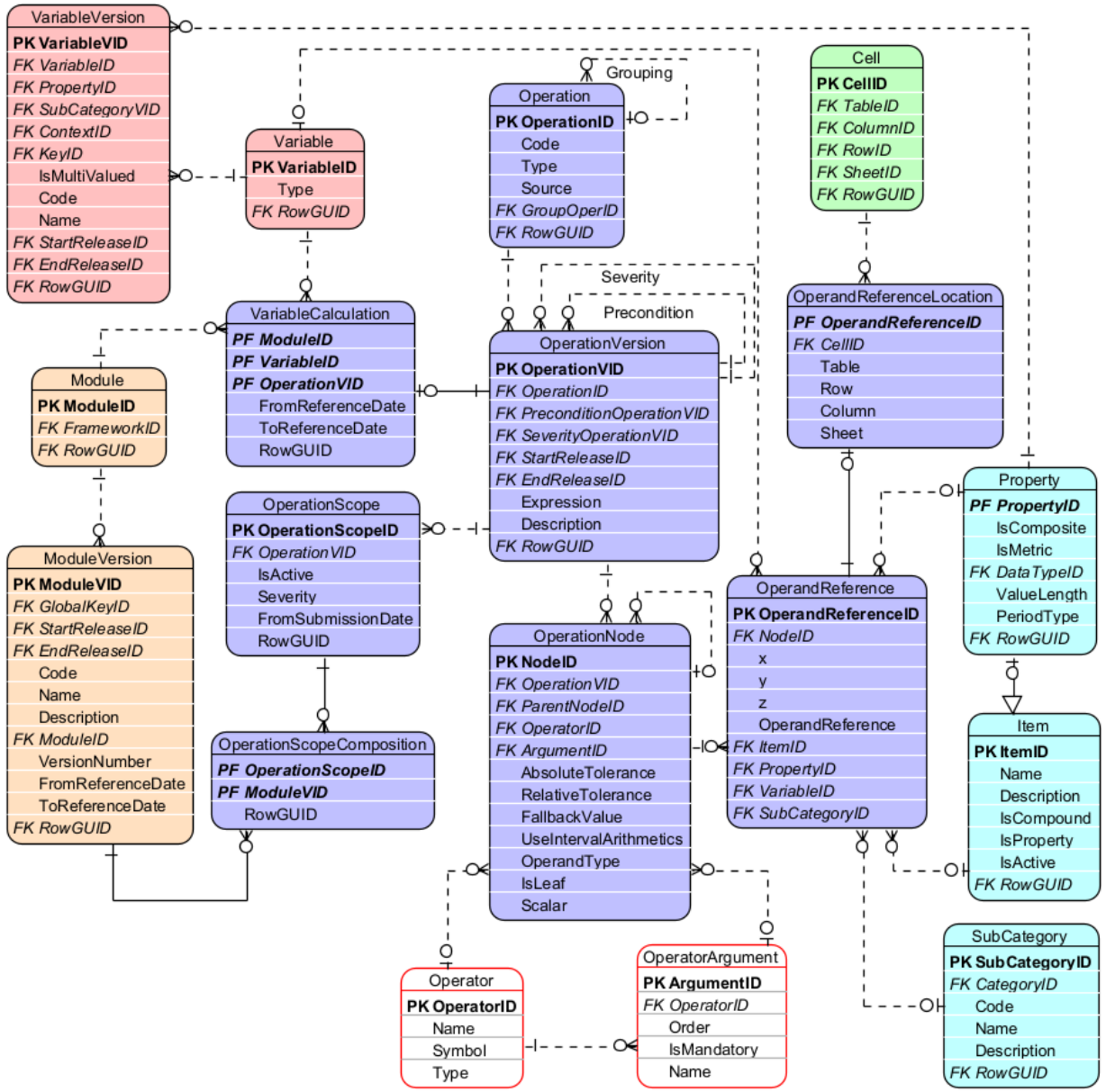
Operations can be sequenced/nested, e.g. serve as precondition, determine dynamic severity, etc

Expression can be translated to natural language, programming language or various syntaxes (e.g. DPM XL)

OperationNode: one of defined **Operators** or an **OperandReference** to any artefact of a model, e.g. Variable, Cell, Item, Property, …)

- Note: *Variables* may indicate external data sources (e.g. master/reference data)

OperationScope and **Composition** provides flexibility in assignment of **Operations** to **Modules**

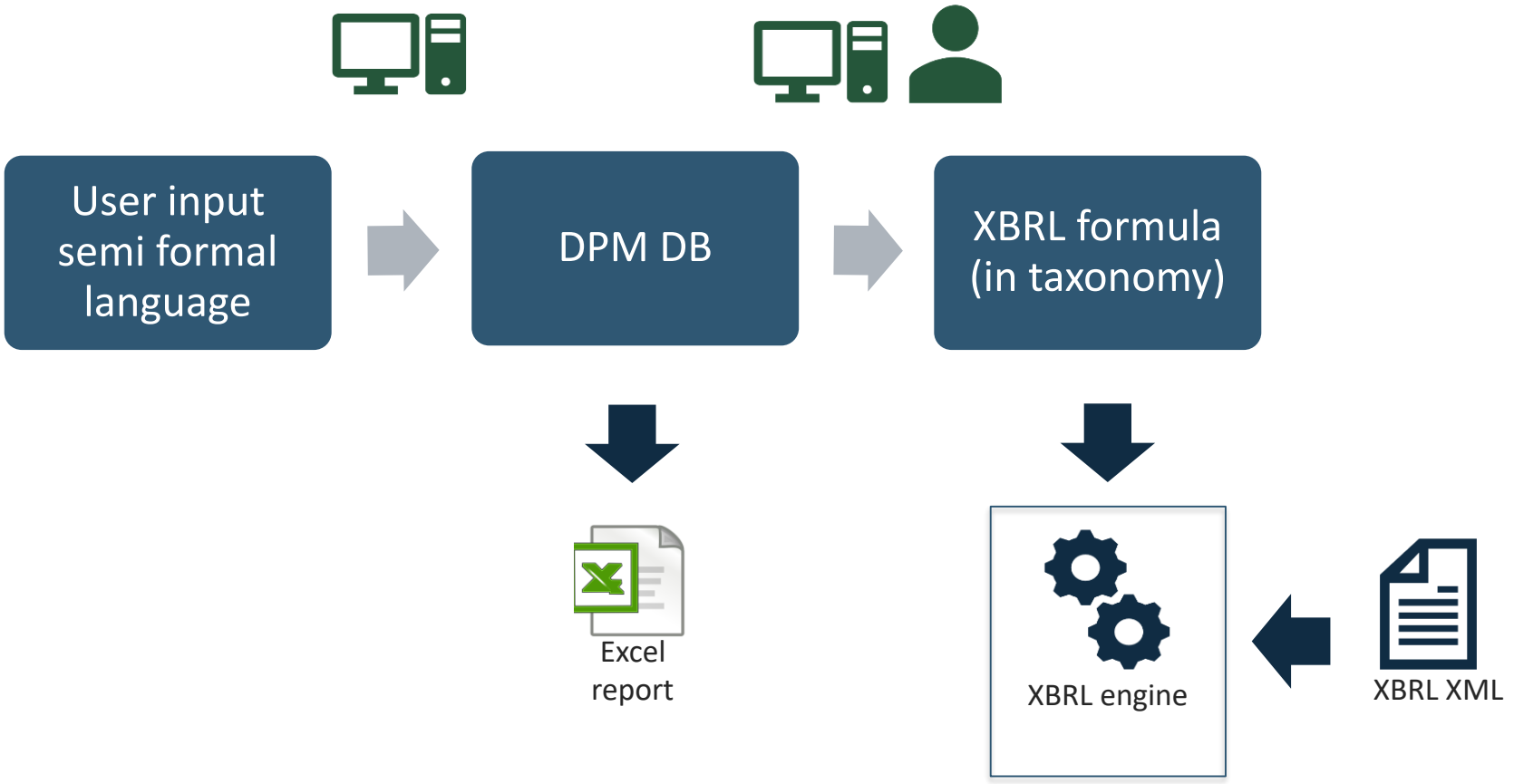


DPM 2.0 Refit project –Validations & Calculation rules - Intro

Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

DPM validations: Current process



Current situation analysis

- EBA and EIOPA have been using a validation language over the years, but without a formal basis and a proper documentation.
- The **lack of formality** causes some challenges:
  - It is not possible to have full automation when translating to other languages (notably, to XBRL Formula).
  - Although EBA and EIOPA are basically using the same language, there are some differences that further difficult automation and common understanding.
  - There may be ambiguity in the meaning of a validation.
- The language is translated **to XBRL formula**, which also adds challenges:
  - Not all validations can be translated to XBRL.
  - XBRL is very difficult to understand.
  - Performance becomes an issue with big instances.

DPM Refit improvements

- The DPM Refit aims to formalize the operations (including validations and other calculations) by having:
  - A formal expression language: **DPM-XL** is a formal language for expressing calculations based on the DPM.
    - Is based on the semi-formal language that the EBA and EIOPA have been using to write and share validation rules for several years.
    - In practice, it is the result of a reverse-engineering process to formalize the language that was already existing, with the minimum changes necessary.
  - A metamodel to represent the tree of operations, as well as the relations of the operands with the core DPM (**DPM-ML**).
- Why DPM-XL and DPM-ML? Because:
  - DPM-ML is based on variables (stable) instead of rendering (unstable).
  - In principle it is possible to translate languages different from DPM-XL into DPM-ML.

DPM Operations specification

The formalisation of the language has three **blocks**:

Information model

Specifies the artefacts that the language is using.

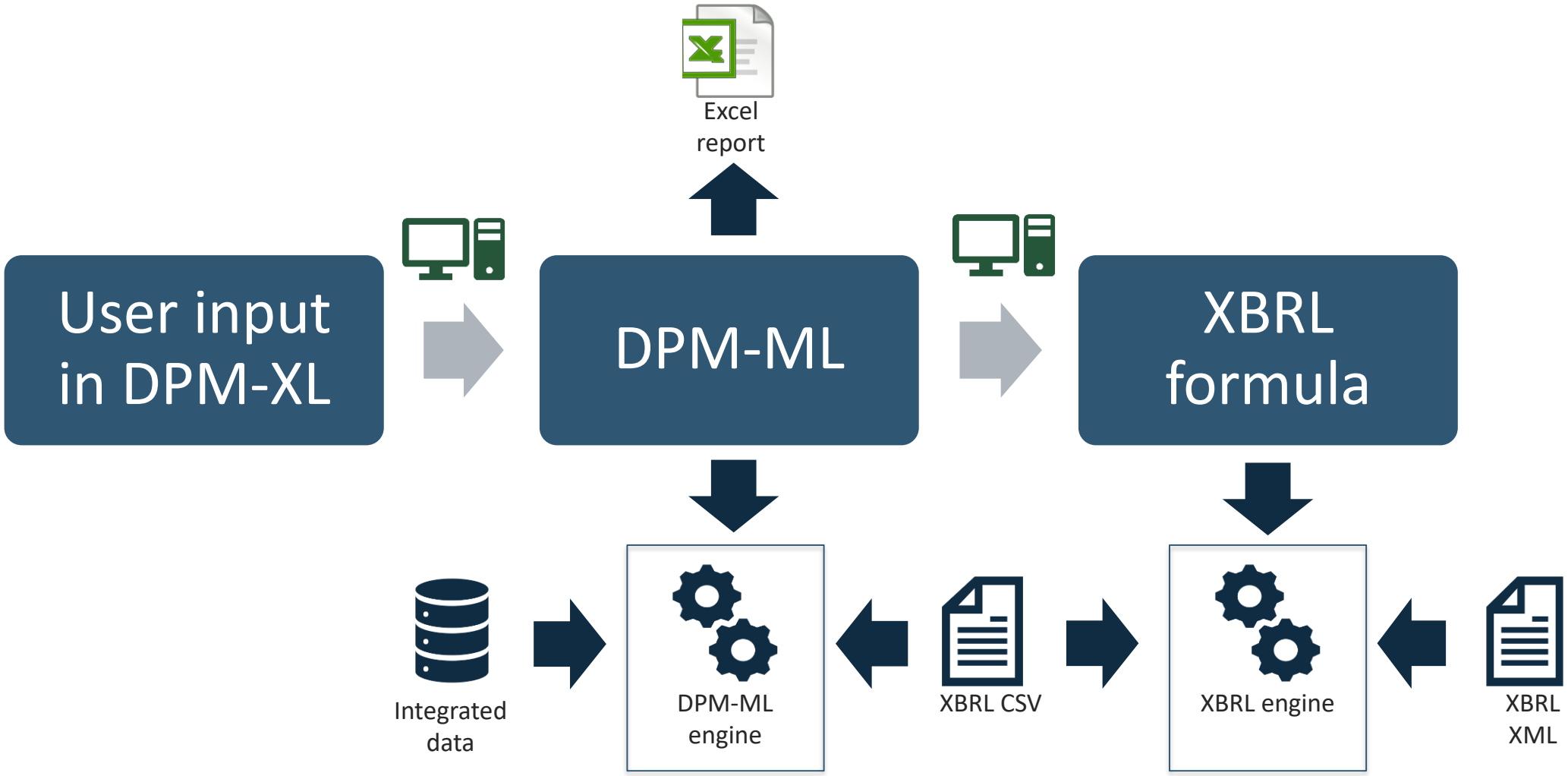
Grammar

Technical definition of the syntax of the language. Allows developers to build parsers for the language (only relevant for DPM-XL).

Semantic specification

Semantics for all operators of the language, specifying formally their constraints and behaviour. For instance, is it possible to use the + operator for two strings?

What are the options the DPM Refit Operations



Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

Validation rule

C 90.00 - Trading book and market risk thresholds (TBT)

			Columns		
			On - and off - balance sheet business subject to market risk		Total assets
				In % of total assets	
			0010	0070	0080
Rows	Month 3	0010	51	5%	1020
	Month 2	0020	42	4%	1010
	Month 1	0030	60	6%	1000

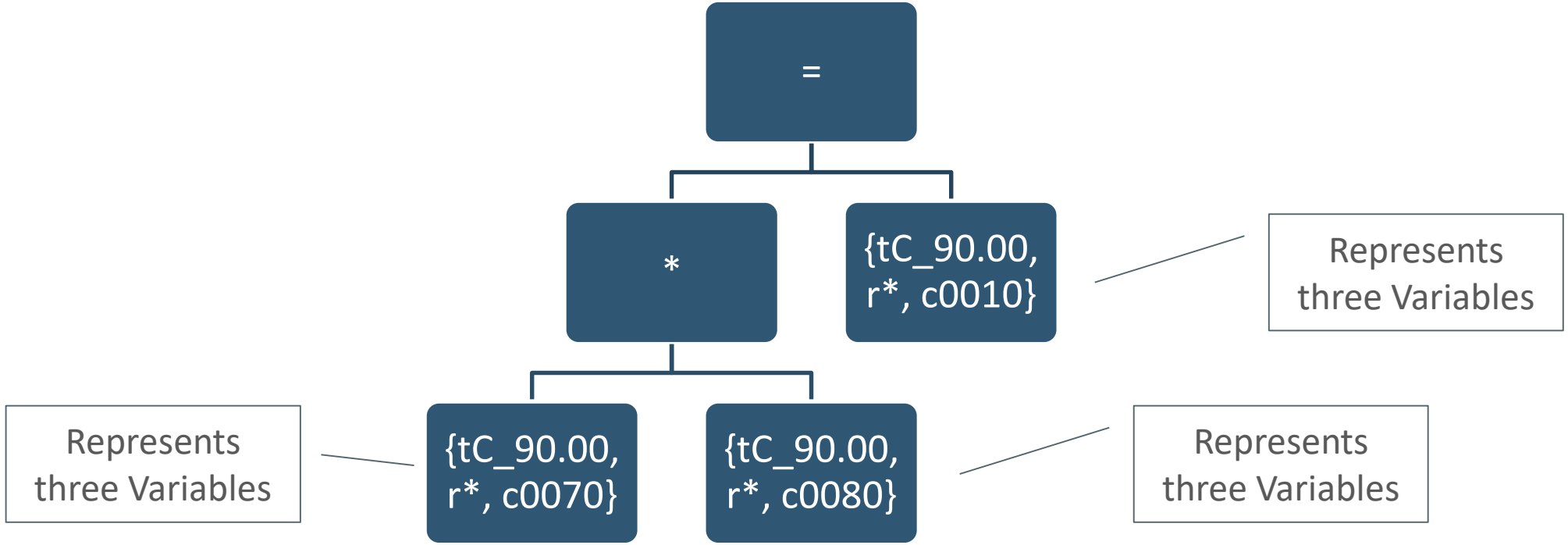
DPM-XL

```
with {tC_90.00, r*}: {c0070} * {c0080} = {c0010}
```

Tree representation of expressions

Any DPM-XL expression can be represented as a tree:

```
with {tC_90.00, r*}: {c0070} * {c0080} = {c0010}
```



DPM-ML

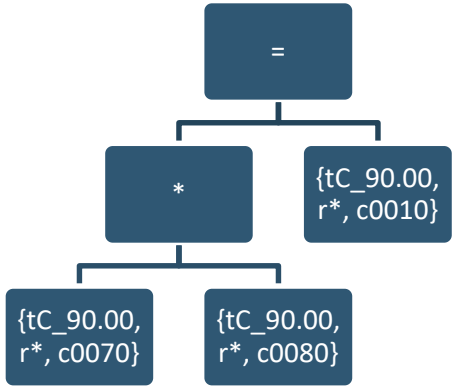
Nodes

Node	ParentNodeID	Operator	Operand
1		=	
2	1	*	
3	2		A
4	2		B
5	1		C

Operands

Operand	Index	Variable
A	1	Dpid1({tC_90.00, r0010, c0070})
A	2	Dpid2({tC_90.00, r0020, c0070})
A	3	Dpid3({tC_90.00, r0030, c0070})
B	1	Dpid4({tC_90.00, r0010, c0080})
B	2	Dpid5({tC_90.00, r0020, c0080})
B	3	Dpid6({tC_90.00, r0030, c0080})
C	1	Dpid7({tC_90.00, r0010, c0010})
C	2	Dpid8({tC_90.00, r0020, c0010})
C	3	Dpid9({tC_90.00, r0030, c0010})

The tree of the expression can be then represented in the DB, with reference to actual DPM variables.



Input data

{tC_90.00, r*, c0010}

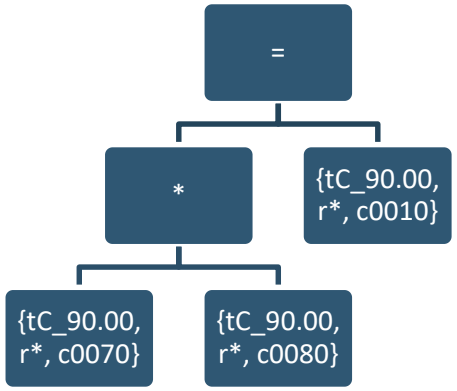
Index	f
1	51
2	42
3	60

{tC_90.00, r*, c0070}

Index	f
1	5%
2	4%
3	6%

{tC_90.00, r*, c0080}

Index	f
1	1020
2	1010
3	1000



Results

```
with {tC_90.00, r*}: {c0070} * {c0080}
```

Index	f
1	51
2	40.4
3	60

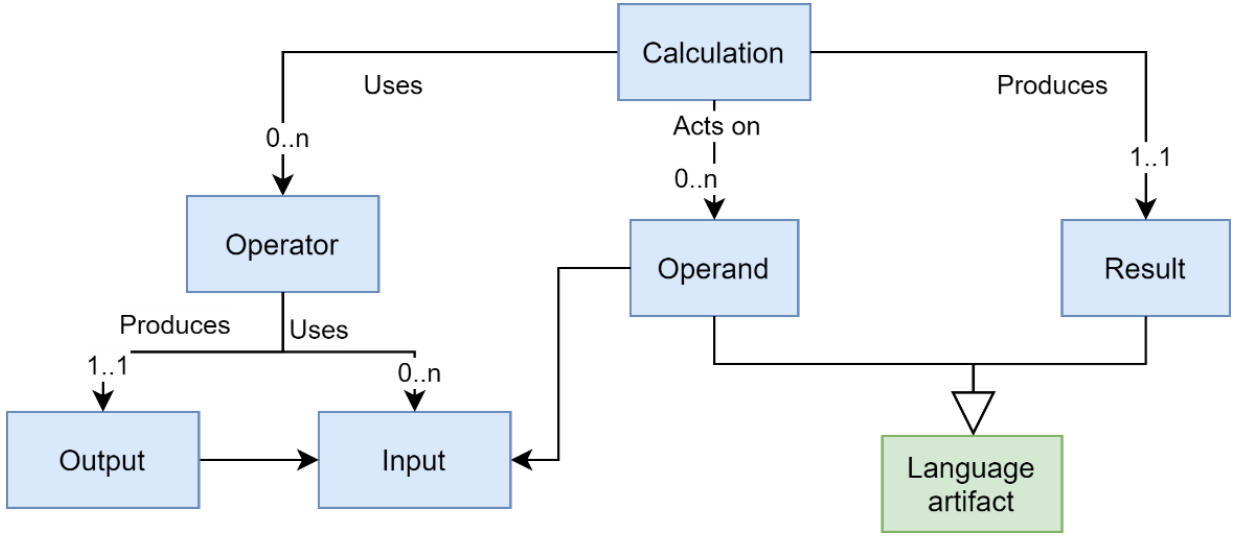
```
with {tC_90.00, r*}: {c0070} * {c0080} = {c0010}
```

Index	f
1	True
2	False
3	True

Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

Information model - Calculations



The DPM Expression Language serves to write **calculations**.

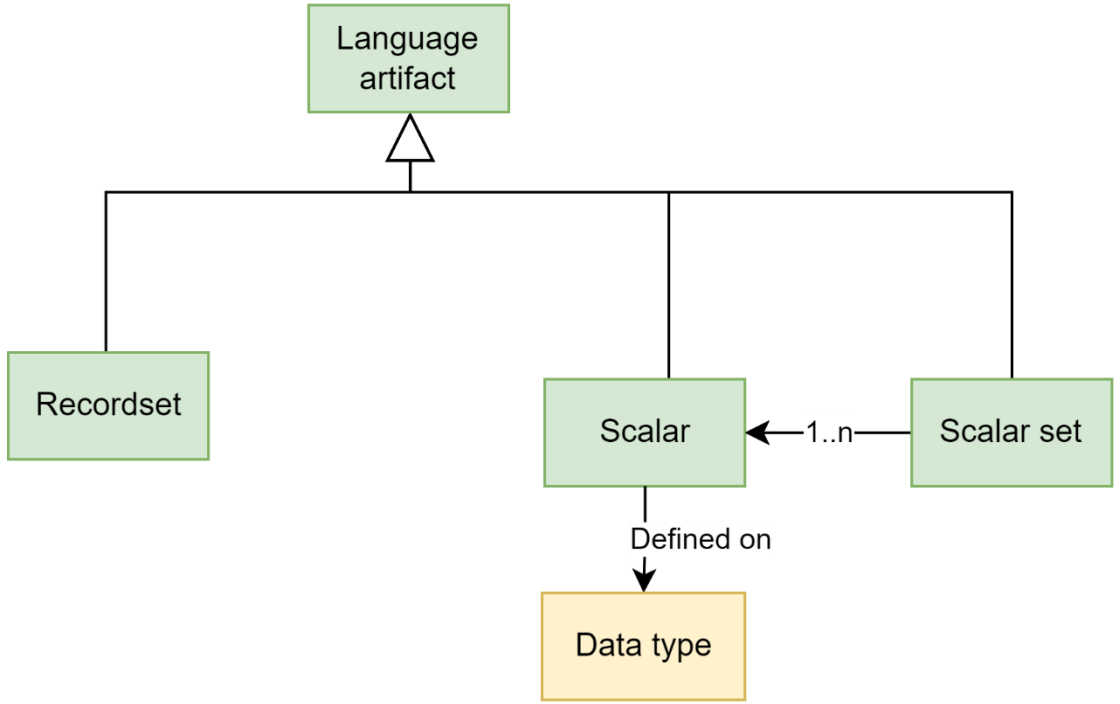
Calculations are expressions that use input operands and/or operators to produce a result. Expressions are finite combinations of symbols that are well-formed according to the syntactical rules of the language. Expressions compose some **operands** in a certain order by means of the **operators** of the language, to obtain the desired **result**. The symbols of the expression designate operators, operands, and the order of application of the operators.

Operators specify a type of operation to be performed on some **input operands** (exceptionally, there may be operators that do not take operands as input, e.g., an operator to get the current time) to generate an **output**. The output produced by one operator may be used as input for another operator (i.e., operators can be nested).

Operands are specific artifacts from the DPM Expression Language referenced in an expression as input.

The **result** produced by a calculation is also a specific artifact from the DPM Expression Language.

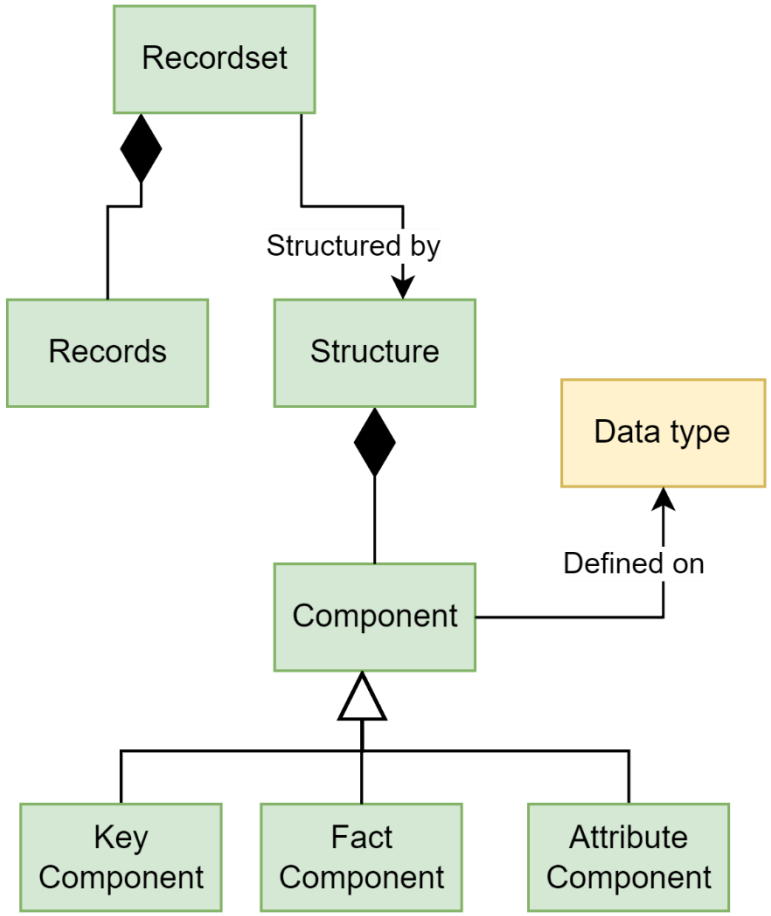
Information model – Language artifacts



Scalars are individual values of a certain *Data Type*.

Scalar Sets *are sets of Scalar values defined on the same Data Type. Scalar Sets are typically used with the in operator.*

Information model – Recordsets



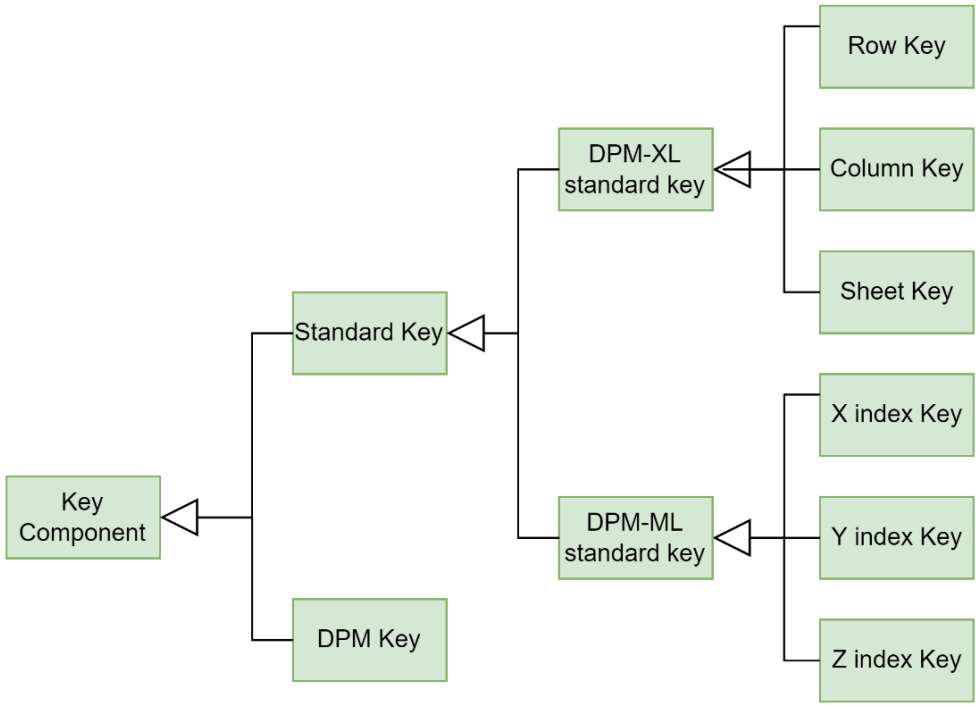
Recordset are collections of *Records* that share a same *Structure*. Technically, *Recordsets* are two-dimensional labelled data structures (tabular), which can be assimilated to Relational Tables or Data Frames. The columns (fields) of the *Recordset* are provided by the ***Components*** of its *Structure*. The rows of the *Recordset* are its composing *Records*.

The *Structure* of the *Recordset* is a collection of ***Components***, which can have one of three roles: *Key*, *Fact* or *Attribute*. Each *Component* has a name, which must be unique within the Recordset.

Each ***Record*** of the *Recordset* is individually identified by the combination of the values for its *Key Components*.

A *Recordset* having *no Key Components* behaves like a *Scalar*.

Information model – Components



Standard Key Components are common to all the *Recordsets*, independently on how the *Variables* are defined in the DPM. For each Recordset, there may be 0 or 1 occurrence of each subtype of *Standard Key* Component.

- *Row Key*: Identifies the *Row Ordinate* from a *Report Table* where the selected *Variable* is located. Arises in *Variable Set Selections*, when more than one *Row* for one *Report Table* is selected. The name for the component is "r" . It is defined on the *String Data Type*.
- *Column Key*: Identifies the *Column Ordinate* from a *Report Table* where the selected *Variable* is located. Arises in *Variable Set Selections*, when more than one *Colum* for one *Report Table* is selected. The name for the component is "c". It is defined on the *String Data Type*.
- *Sheet Key*: Identifies the *Sheet Ordinate* from a *Report Table* where the selected *Variable* is located. Arises in *Variable Set Selections*, when more than one *Sheet* for one *Report Table* is selected. The name for the component is "s". It is defined on the *String Data Type*.

DPM Key Components are specific to how data is defined in the DPM. Arise when *Open Variables* are selected, and a *Recordset* will have one *DPM Key Component* per each *Key Variable* associated to the selected *Variables*.

The name for the *DPM Key Components* is the *Code* of the *Property* associated to the DPM *Key Variable.*

Information model – Recordset Example I

F 32.01 - ASSETS OF THE REPORTING INSTITUTION (AE-ASS)

		Carrying amount of encumbered assets			
			of which: issued by other entities of the group	of which: central bank's eligible	of which notionally eligible EHQLA and HQLA
		0010	0020	0030	0035
0010	**Assets of the reporting institution**	300		100	100
0015	of which: qualifying fiduciary assets				
0020	Loans on demand	200			100
0030	Equity instruments				
0040	Debt securities	100		100	

```
{tF_32.01, r0020-0040, (c0010, c0030, c0035)}
```

r	c	f
0020	0010	200
0030	0010	
0040	0010	100
0020	0030	
0030	0030	
0040	0030	100
0020	0035	100
0030	0035	
0040	0035	

Information model – Recordset Example II

F 20.05.a - Geographical breakdown of off-balance sheet exposures by residence of the counterparty (a)

ES

			Columns
			Nominal amount
			0010
Rows	Loan commitments given	0010	100
	Financial guarantees given	0020	200
	Other commitments given	0030	300

PT

			Columns
			Nominal amount
			0010
Rows	Loan commitments given	0010	400
	Financial guarantees given	0020	500
	Other commitments given	0030	600

{tF_20.05, r0020-0030, c0010}

RCP	r	f
ES	0020	200
ES	0030	300
PT	0020	500
PT	0030	600

Information model – Recordset Example III

F 40.01 - Scope of the group: “entity-by-entity”

			Columns		
			Code	Type of code	Entity name
			0011	0015	0031
Rows	Investee	999	123456	LEI	Name1
			123456	ISIN	Name2
			1111	LEI	Name3
			LIN <Key value>	TYC <Key value>	

{tF_40.01, c0031}

LIN	TYC	f
123456	LEI	Name 1
123456	ISIN	Name 2
1111	LEI	Name 3

Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

Operators

Selection
- Selection operator
- With

Arithmetic
- Unary plus (+)
- Addition (+)
- Unary minus (-)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Absolute value (abs)
- Exponential (exp)
- Natural logarithm (ln)
- Power (power)
- Logarithm (log)
- Square root (sqrt)

Comparison
- Equal to (=)
- Not equal to (<>)
- Greater than (>, >=)
- Less than (<, <=)
- Element of (in)
- Math characters (match)
- Is null (isnull)

Logical operators
- Conjunction (and)
- Disjunction (or)
- Exclusive disjunction (xor)
- Negation (not)

Aggregate operators
- Sum (sum)
- Count (count)
- Minimum value (min)
- Maximum value (max)
- Average (avg)
- Median value (median)

Conditional operators
- If then else (if)
- Null substitute (nvl)
- Filter (filter)

String operators
- Length (len)
- Concatenate (&)

Time operators
- Time shift (time_shift)

Clause operators
- Where (where)
- Rename (rename)
- Get (get)

The selection operator

- Curly brackets ({}) are used as symbol for the selection operator
- The selection operator has three **parts**:
  - **Recordset selection**: By referencing:
    - Cells (table, rows, columns and/or sheets).
    - Variables: References to variable codes.
    - Operations: References to the results of other operations.
  - **Default value**: Sets a default value in case the selection has missing data or explicit nulls for a data instance.
  - **Interval**: For numeric variables, selects whether the data should be considered as interval or point.
- The *Recordset* resulting from a selection is composed by all possible *Records* in the selection. If one *Variable* is not reported (missing), then the *Record* will have null value.

The selection clause – Example

F 01.01 - Balance Sheet Statement [Statement of Financial Position]: Assets

			Columns
			Carrying amount
			0010
Rows	Cash, cash balances at central banks and other demand deposits	0010	**3000**
	Cash on hand	0020	**1000**
	Cash balances at central banks	0030	**2000**
	Other demand deposits	0040	

```
{tF_01.01, r0010-0040, c0010}
```

r	f
0010	3000
0020	1000
0030	2000
0040	

```
{tF_01.01, r0010-0040,
  c0010, default:0}
```

r	f
0010	3000
0020	1000
0030	2000
0040	0

```
{tF_01.01, r0010-0040, c0010,
  default:0, interval:true}
```

r	f
0010	3000±500
0020	1000±500
0030	2000±500
0040	0±0

The with clause

- The with clause serves to provide a common context to all the selections of an expression.
- In the current Excel files, represented in separate columns.

Validation ID	Template 1	Columns	Validation
BV35	S.16.01	c0020-0080	{r0200}=sum({(r0040-0190)})

- The with clause uses the following syntax:

```
with partial_selection: expression
```

  - **partial_selection**: Is the selection that is completing the selections in the expression , using the selection clause.
  - **expression**: It is an expression containing selection operators.

- The with clause does not produce an output, but modifies the selections in the expression according to some rules. The operator does not produce a node in DPM-ML.
- The selection in the with applies to all selections in the expression unless they are overridden.

The with clause – Examples

```
with {tF_01.01, c0010, default:0, interval:false}:
   {r0010} = {r0020} + {r0030} + {r0040}
```

No operand in the expression overrides the with context.

```
with {tF_01.01, c0010, default:0, interval:false}:
   {r0010} + {r0040} = {tF_04.01, r0010, c0010}
```

The third operand in the expression overrides the table and the column of the with context

```
with {tF_01.01, c0010, default:0, interval:false}:
   {tF_01.01, r0010} + {tF_01.01, r0040} = {tF_04.01, r0010, c0010, default:null}
```

All three operands in the expression override the table in the context. The third operand overrides also the column and the default.

```
with {c0010, default:0, interval:false}:
   {tF_01.01, r0010} + {tF_01.01, r0040} = {tF_04.01, r0010}
```

No operand in the expression overrides the with context

General behaviour for binary operators – Example 1

```
{ts.26.01, r0600, (c0060, c0080)}
```

c	f
0060	100
0080	200

```
0.25 * {ts.26.01, r0600, (c0060,
                          c0080)}
```

c	f
0060	25
0080	50

General behaviour for binary operators – Example 2

{tF_04.02.01, r0120, c0010-0020}

c	f
0010	100
0020	200

{tF_04.02.01, r0140, c0010-0020}

c	f
0010	300
0020	400

with {tF_04.02.01, c0010-0020)}: {r0120} + {r0140}

c	f
0010	400
0020	600

General behaviour for binary operators – Example 3

{tC_28.00, c040}

INC	f
123	1000
456	2000
789	3000

{tC_28.00, c0190}

INC	f
123	-100
456	-200
789	-300

{tC_28.00, c040} + {tC_28.00, c190}

INC	f
123	900
456	1800
789	2700

General behaviour for binary operators – Example 4

{tF_40.01, c0110}

LIN	TYC	f
123	x1	1
456	x1	0.8
789	x1	0.4

{tF_40.02, c0060}

LIN	TYC	STC	LHC	LHO	f
123	x1	111	ABC	x1	0.3
123	x1	111	DEF	x1	0.7
456	x1	222	ABC	x1	0.85

{tF_40.01, c0110} >= {tF_40.02, c0060}

LIN	TYC	STC	LHC	LHO	f
123	x1	111	ABC	x1	true
123	x1	111	DEF	x1	true
456	x1	222	ABC	x1	false

General behaviour for binary operators

- If the two *Operands* of a binary *Operator* are ***Scalars***, the result shall be the *Scalar* resulting of applying the *Operator* to the *Operands*.

- A binary *Operator* applied to a ***Recordset*** *Operand* and a ***Scalar***, will result in a *Recordset* with the same *Structure* as the input *Recordset Operand*. The operator shall be applied to every record of the input *Recordset* and the *Scalar*.

- For **two *Recordsets***:

  - **Constraints**: Binary *Operators* can only be applied to two *Recordsets Operands* if they have:

    - Exactly the same *Key Components;* or

    - the *Key Components* of one *Recordset* (Reference *Recordset*) are a superset of the *Key Components* of the other *Recordset*.

  - **Behaviour**: Performs an inner join and the operator applies to the pairs of values resulting from performing an inner join.

Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

Harmonising existence check

- Currently there are different ways to check existence. Examples:
  - EIOPA BV195: not(empty({S.01.02, c0010}))
  - EBA v6215_m: {P 01.03} != empty
- Proposal to change to isnull() operator

Harmonising operator symbols

- For each operator, only one symbol is allowed. Currently for some operators several symbols are used. Examples:
  - <>, !=
  - *in*, ∈
  - = , ≡, ==
  - *Like, matches*

Default in selection

Currently there are different practices:

EBA

Formula	Prerequisites - apply rule (only) if specified tables are reported (or for module containing the table if written in square brackets)	If value missing (but table prerequisites met)
if {c0080} = false then {c0090} = 0	T 03.02	treat as zero/empty string

EIOPA

Validation	Variables to which the fallback values apply	Validation variable mapping
{er0020,ec0020}+{er0030,ec0020} <= 0.1*({er0010,ec0	a = 0; b = 0; c = 0; f = ();	a = {er0020,ec0020};b = {er0030,ec0020};c = {er0010,ec0020};f = {er0040,ec0020,(sNNN)}

Solution: Parameter in the selection with the default value.

```
{tF_01.01, r0010-0040, c0010, default:0}
```

Interval applicability in selection

Currently there are different practices:

EBA

Formula	Arithmetic approach
{r0020} = +{r0030} + {r0040} + {r0050}	Interval
{N 01.01} >= 0	Point

EIOPA

Validation	Test expression for fallback values and interval arithmetics
{r0200}={r0110}+{r0120}	iaf:numeric-equal($a, iaf:sum(($b, $c)))
{c0010}={c0040}	$a = $b

Solution: Parameter in the selection with the treatment for intervals.

```
{tF_01.01, r0010-0040, c0010, default:0, interval:true}
```

Where/filter – Current practices

Currently there are different practices:

- **EBA**: *where* operator inside *sum* operator

```
sum(where({C 08.02,c0010,s0013}=1) {C 08.02,c0020, s0013})
```

- **EIOPA**: Field in the Excel specifying the filter

Filter	Validation
{c0320} like '##D3'	{c0330} = empty

Where/filter – Solution

Filter operator with two arguments:

- The filtered selection
- The condition for the selection

```
sum(
        where({C
08.02,c0010,s0013}=1)
        C 08.02,c0020, s0013}
)
```

```
sum(
        filter(
            {tC 08.02, c0020, s0013},
            {tC 08.02, c0010, s0013} =
1
        )
)
```

Filter	Validation
{c0320} like '##D3'	{c0330} = empty

```
isnull(
        filter(
                {c0330},
                match({c0320}, '##D3')
        )
)
```

Group by for aggregate operators – Current practices

Currently EBA and EIOPA use columns in the Excel to define the group by:

EBA:

ID	sheets	Formula
v10666_m	(All)	{C 08.01.a, r0070, c0240} * sum({C 08.02, c0140, (rNNN)}) = sum({C 08.02, c0240, (rNNN)} * {C 08.02, c0140, (rNNN)})

EIOPA:

Validation ID	Template 1	Columns	Validation
BV35	S.16.01	c0020-0080	{r0200}=sum({(r0040-0190)})

Group by for aggregate operators – Solution

Aggregate operators have an optional argument which is a list of the Key Components for the **group by**.

```
{C 08.01.a, r0070, c0240} *
        sum({C 08.02,
c0140}) =
sum(
        {C 08.02, c0240} *
        {C 08.02, c0140}
)
```

```
{C 08.01.a, r0070, c0240} *   sum({C
08.02, c0140} group by s)
=
sum(
        {C 08.02, c0240} *
        {C 08.02, c0140}
        group by s
)
```

```
{r0200}=sum({(r0040-0190)})
```

```
with {tS.16.01.01.02, c*:
   {r0200} =
   sum (
      {r0040-0190}
      group by c
    )
```

Other changes

- **with** clause: Added to deal with the scope.
- **xsum** operator: Deleted because it is not necessary under the new approach.
- **string_length** operator: Renamed to len, to make it more consistent with other languages (e.g., Excel).
- **Regex** harmonisation: One Regex flavour selected.

Outline

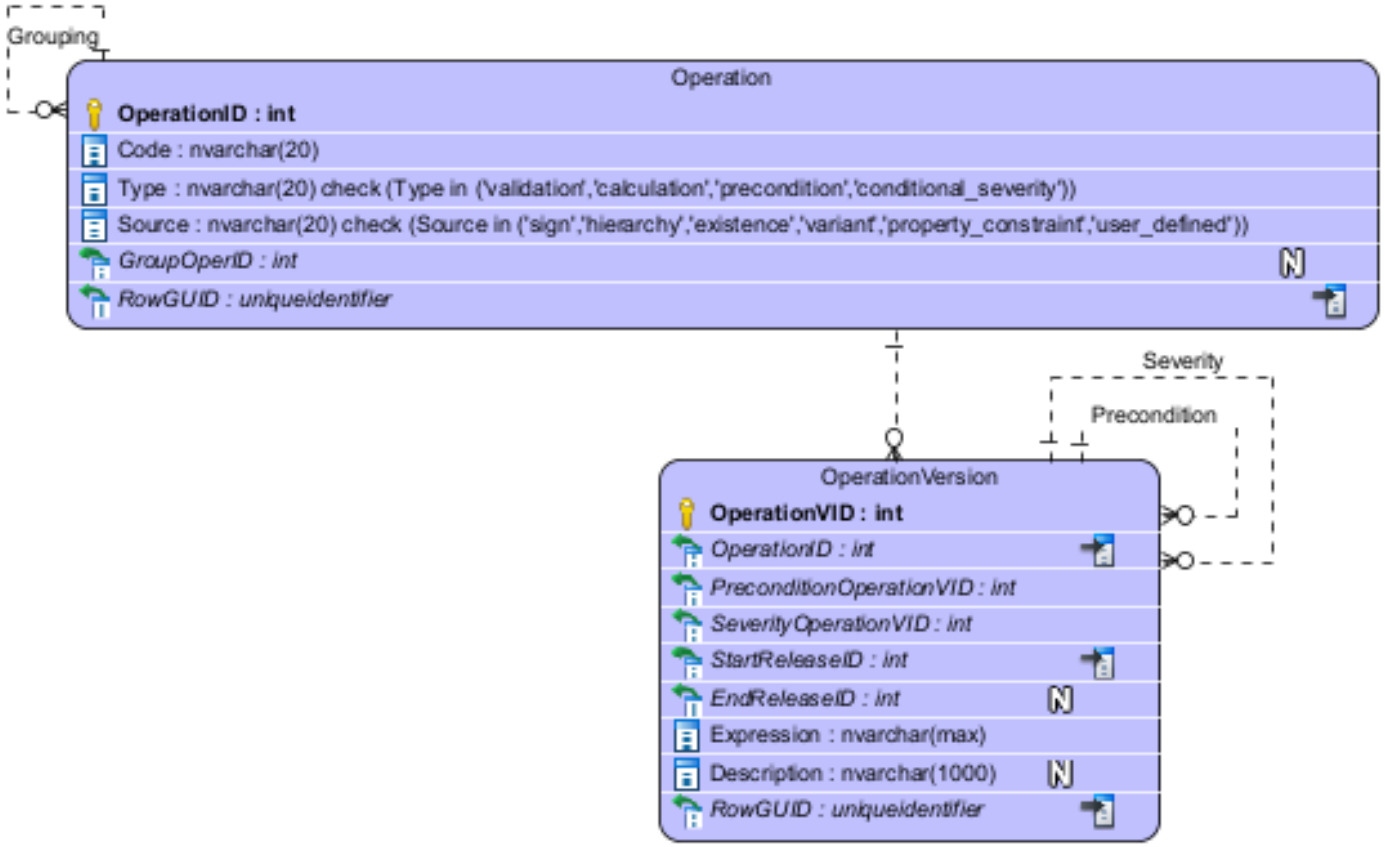
- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

Operation and Operation version

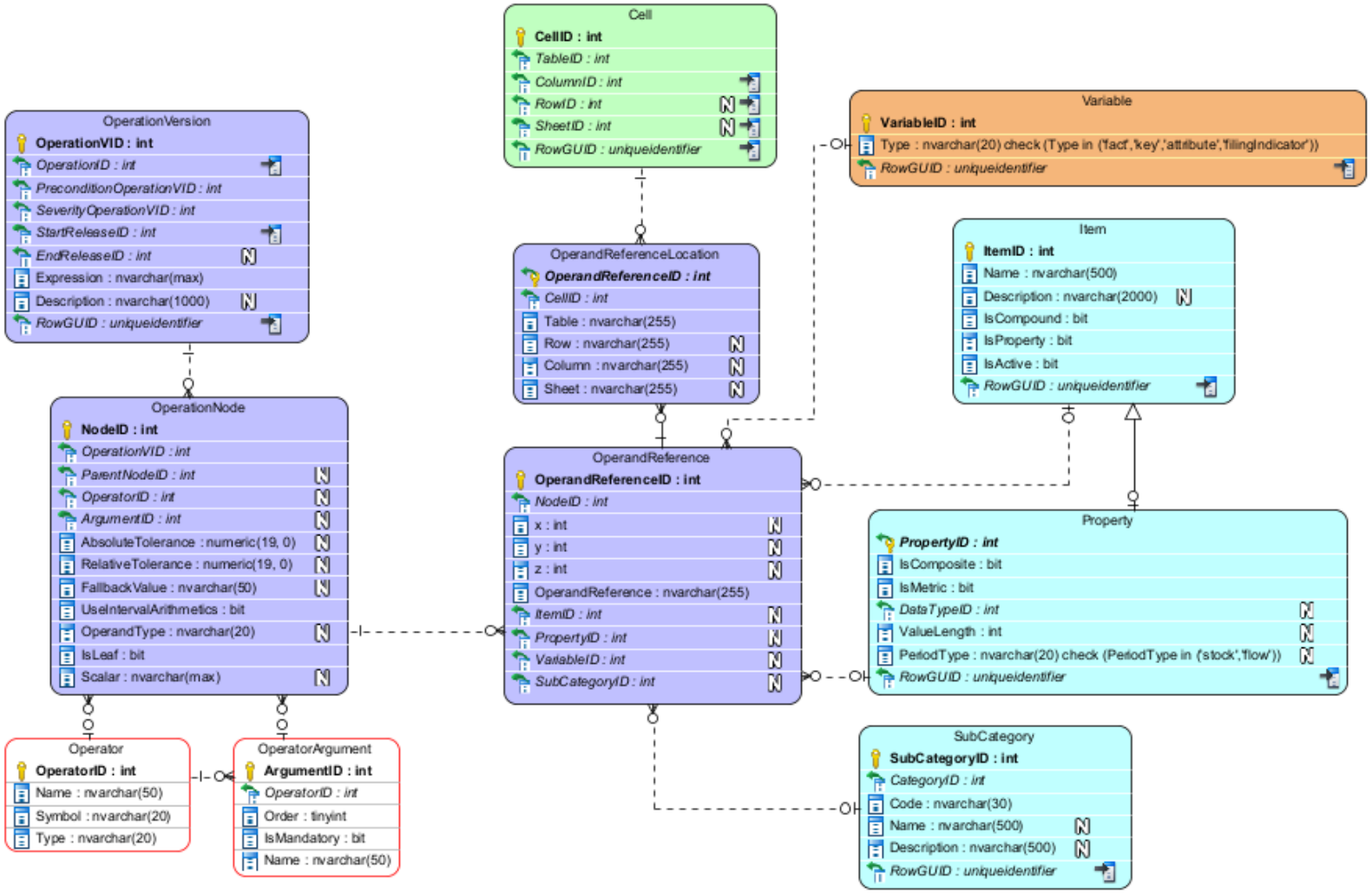
Operations have a code and can be grouped.

Groups of operations refer to the cases where users define a parent validation and the system derives automatically children validations. There can only be two levels of operations (i.e., a child validation cannot be the parent of another validation). The types of validations that generate children are *Variant* and *Property constraint* validations.

Operations can have many versions. A new version of an operation is required whenever any of their attributes change.

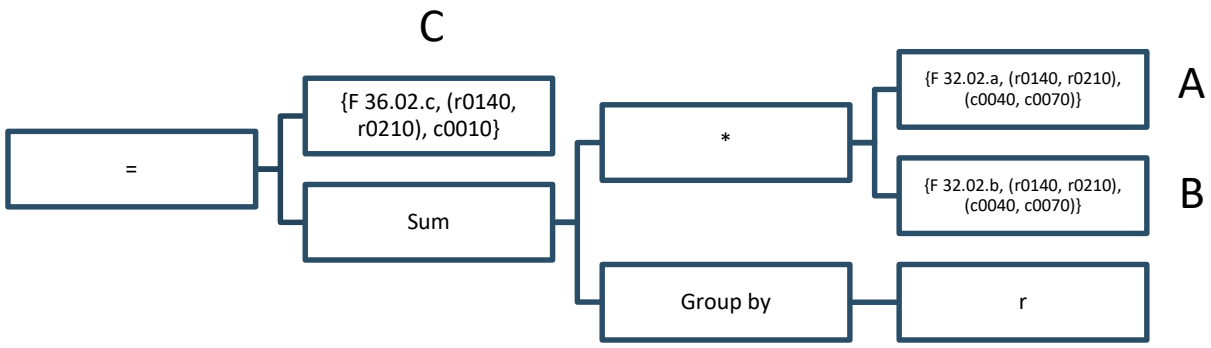


Operation tree



Operation tree: Example xyz index

```
{F 36.02.c, (r0140, r0210), c0010} =
        sum(
                {F 32.02.a, (r0140, r0210), (c0040, c0070)} *
                {F 32.02.b, (r0140, r0210), (c0040, c0070)}
            group by r)
```



Operand	x	y	z	location
A	1	1		{F 32.02.a, r0140, c0040}
A	1	2		{F 32.02.a, r0140, c0070}
A	2	1		{F 32.02.a, r0210, c0040}
A	2	2		{F 32.02.a, r0210, c0070}
B	1	1		{F 32.02.b, r0140, c0040}
B	1	2		{F 32.02.b, r0140, c0070}
B	2	1		{F 32.02.b, r0210, c0040}
B	2	2		{F 32.02.b, r0210, c0070}
C	1			{F 36.02.c, r0140, c0010}
C	2			{F 36.02.c, r0210, c0010}

Operation tree: Example xyz index

```
{tF_32.02.a, (r0140, r0210), (c0040, c0070)} *
{tF_32.02.b, (r0140, r0210), (c0040, c0070)}
```

A

x	y	z	location
1	1		{tF_32.02.a, r0140, c0040}
1	2		{tF_32.02.a, r0140, c0070}
2	1		{tF_32.02.a, r0210, c0040}
2	2		{tF_32.02.a, r0210, c0070}

*

B

x	y	z	location
1	1		{tF_32.02.b, r0140, c0040}
1	2		{tF_32.02.b, r0140, c0070}
2	1		{tF_32.02.b, r0210, c0040}
2	2		{tF_32.02.b, r0210, c0070}

x	y	z	Result
1	1		A{1,1} * B{1,1}
1	2		A{1,2} * B{1,2}
2	1		A{2,1} * B{2,1}
2	2		A{2,2} * B{2,2}

```
sum(
      {tF_32.02.a, (r0140, r0210), (c0040, c0070)} *
      {tF_32.02.b, (r0140, r0210), (c0040, c0070)}
  group by r)
```

x	y	z	Result
1			A{1,1} * B{1,1} + A{1,2} * B{1,2}
2			A{2,1} * B{2,1} + A{2,2} * B{2,2}

```
{tF_36.02.c, (r0140, r0210), c0010} =
  sum(
        {tF_32.02.a, (r0140, r0210), (c0040, c0070)} *
        {tF_32.02.b, (r0140, r0210), (c0040, c0070)}
    group by r)
```

C

x	y	z	Location
1			{tF_36.02.c, r0140, c0010}
2			{tF_36.02.c, r0210, c0010}

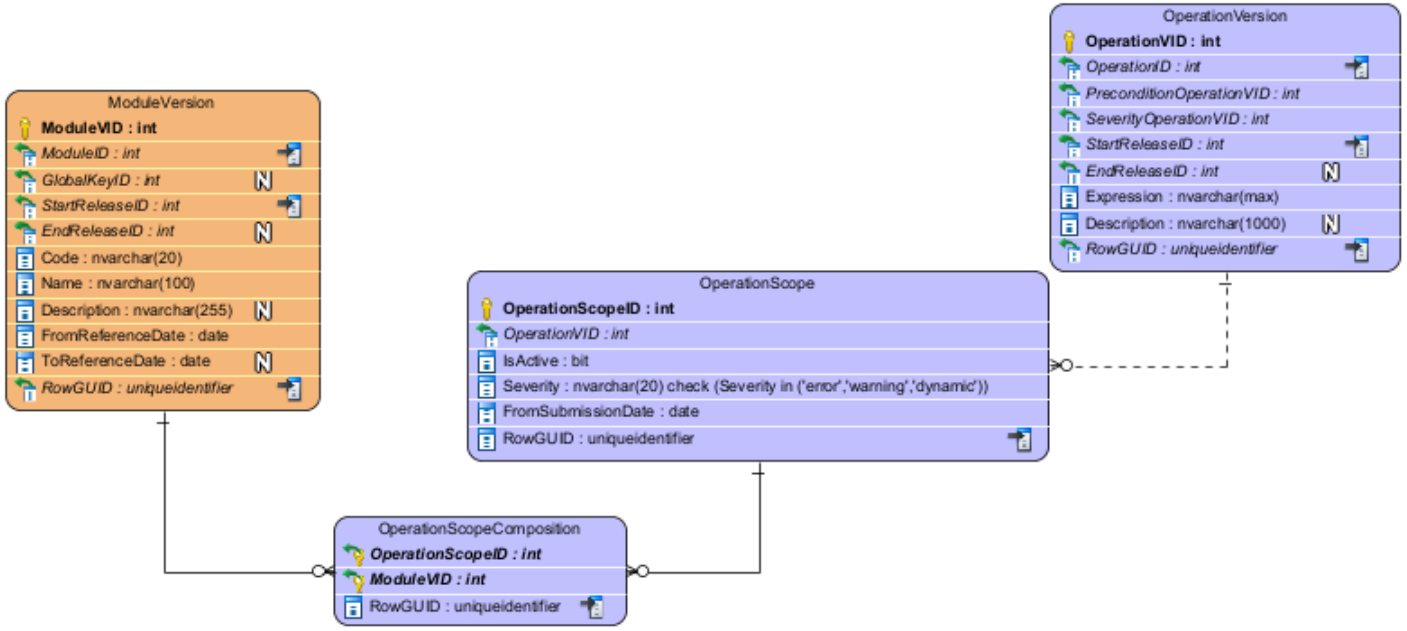
x	y	z	Result
1			A{1,1} * B{1,1} + A{1,2} * B{1,2} = C{1}
2			A{2,1} * B{2,1} + A{2,2} * B{2,2} = C{2}

Operation scope

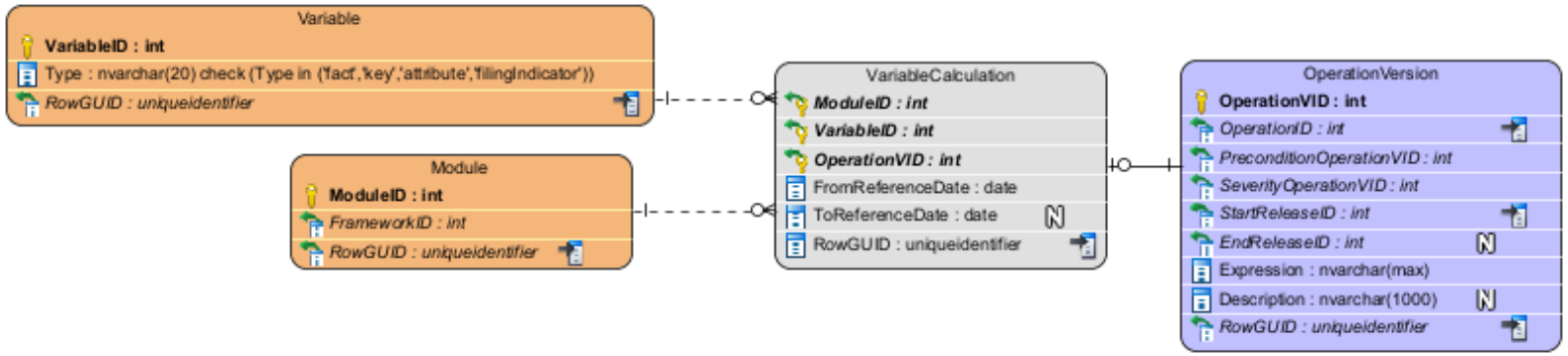
Each operation version can be applied in different scopes. An operation scope refers to the individual module versions or sets of module versions (for cross-module validations) to which the operation logically applies.

For each operation scope, there may be different values for the attributes:

- IsActive: Determines whether the operation shall be run for a certain scope.
- Severity: Determines the severity of the error, if the validation is not passed.
- FromSubmissionDate: Sets a date from which the validation applies to subsequent submissions.



Variable calculation



For calculations (i.e., operations that serve to calculate values for variables), it is necessary to link the operation to the actual variable that is generated. This is done with the *VariableCalculation* table, which links to the *Variable* and the *Module*. The link to the *Module* is necessary due to the fact that a *Variable* can be used in several *Modules*, being calculated in some cases and not calculated in other cases.

Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations

About calculations

- Calculations are very similar to validations: Algorithms to manipulate input data into output data.
- The DPM-XL and DPM-ML for validations can also be used for calculations.
- Only difference is in the treatment of related issues, like triggers to run validations vs calculations, or what to do with the results of execution.

The Transparency exercise case - I

Consider the current transparency exercise table with mapping:

		(mln EUR, %)	As of 30/09/2020	As of 31/12/2020	As of 31/03/2021	As of 30/06/2021	COREP CODE
			1	2	3	4	
A	**1**	**OWN FUNDS**	**#VALUE!**	**#VALUE!**	**#VALUE!**	**#VALUE!**	C 01.00 (r010,c010)
A.1	**2**	**COMMON EQUITY TIER 1 CAPITAL (net of deductions and after applying transitional adjustments)**	**#VALUE!**	**#VALUE!**	**#VALUE!**	**#VALUE!**	C 01.00 (r020,c010)
A.1.1	3	Capital instruments eligible as CET1 Capital (including share premium and net own capital instruments)	C 01.00_030_010	C 01.00_030_010	C 01.00_030_010	C 01.00_0030_0010	C 01.00 (r030,c010)
A.1.2	4	Retained earnings	C 01.00_130_010	C 01.00_130_010	C 01.00_130_010	C 01.00_0130_0010	C 01.00 (r130,c010)
A.1.3	**5**	Accumulated other comprehensive income	C 01.00_180_010	C 01.00_180_010	C 01.00_180_010	C 01.00_0180_0010	C 01.00 (r180,c010)
A.1.4	**6**	Other Reserves	C 01.00_200_010	C 01.00_200_010	C 01.00_200_010	C 01.00_0200_0010	C 01.00 (r200,c010)
A.1.5	7	Funds for general banking risk	C 01.00_210_010	C 01.00_210_010	C 01.00_210_010	C 01.00_0210_0010	C 01.00 (r210,c010)
A.1.6	8	Minority interest given recognition in CET1 capital	C 01.00_230_010	C 01.00_230_010	C 01.00_230_010	C 01.00_0230_0010	C 01.00 (r230,c010)
A.1.7	**9**	Adjustments to CET1 due to prudential filters	C 01.00_250_010	C 01.00_250_010	C 01.00_250_010	C 01.00_0250_0010	C 01.00 (r250,c010)
A.1.8	**10**	(-) Intangible assets (including Goodwill)	C 01.00_300_010+C 01.00_340_010	C 01.00_300_010+C 01.00_340_010	C 01.00_300_010+C 01.00_340_010	C 01.00_0300_0010+C 01.00_0340_0010	C 01.00 (r300,c010) + C 01.00 (r340,c010)

The Transparency exercise case - II

The table could be represented as a DPM table, and each cell would be a variable.

The DPM-XL (and the related metadata) can assign the results of operations to DPM variables.

For calculations, being able to concatenate operations is critical.

Example: calculation of variables in row 10 (DPM-XL):

```
TE_capital_r10 := {tC_01.00, r0300, c0010} + {tC_01.00, r0410, c0010};

tTE_capital_r10_c1 <-
        filter(oTE_capital_r10, date='2020-09-30');

tTE_capital_r10_c2 <-
        filter(oTE_capital_r10, date='2020-12-31');
...
```

DPM standard