EUROPEAN CENTRAL BANK

EUROSYSTEM

Version 2.0

January 2024

SMCube Metadata Model

Contents

1. Introduction

This document describes the Metadata Model of the Single Multidimensional Metadata Model (SMCube), which was developed by the European Central Bank to produce its Single Data Dictionary (SDD) and the Banks' Integrated Reporting Dictionary (BIRD).

The objective of this Metadata Model is to define an abstract model that can describe the structure of any type of dataset and some of the attached characteristics (such as dataset exchanges or transformation), regardless of the purpose of the collection.

The SMCube methodology serves as the basis for the construction of metadata and provides the structure for metadata-driven systems. The metadata, constructed in line with the SMCube methodology, can serve as parameters for the system, so that the definition and management of new datasets is as parametrised as possible from the start, resulting in enhanced collaboration between datasets and in the minimisation of new IT developments.

1.1 SMCube use cases

Currently, many different modelling methodologies are used for defining datasets (SDMX, DPM/XBRL, etc.), this situation is not expected to change in the medium term. Given that the methodologies provide the basis for describing the datasets, different datasets described based on different methodologies is a clear obstacle to data integration.

SMCube establishes a new level of abstraction in addition to these approaches to facilitate the joint use of different datasets.

1.2 SMCube Metadata Model requirements

The SMCube Metadata Model accounts for requirements on both a technical and business basis. The following requirements have been identified to satisfy the use case described in the previous paragraph:

- **Use as metadata layer for metadata-driven systems:** provides system-compatible structural information, assisting information managers and end users.
- **Compatibility with other standards,** such as DPM and SDMX**:** covers the greatest array of datasets and supports industry sponsored models.
- **Business users-driven:** reflects the business needs of end users.
- **Historization:** follows the changes in time of defined datasets.
- **Complex mappings:** integrate existing dictionaries and create links between similar information.
- **Entity Relationship Models:** provides detailed information for several types of Entity Relationship Models.
- **Extensibility:** gives the possibility of other organisations implementing the SMCube methodology and making use of the definitions provided by the ECB in the SDD.

2. Definition of a data set in SMCube methodology

2.1 Maintenance agency

A Maintenance agency is the agency responsible for the maintenance of associated dictionary elements (e.g. Variables, Members, etc.).

The concept of Maintenance agency allows the dictionary to manage different codification systems maintained by different Maintenance agencies. Examples for Maintenance agencies are the SDD reference dictionary (ECB) and the European Banking Authority (EBA) which is responsible for the maintenance of the DPM content.

We define the Maintenance agency SDD reference dictionary (ECB) to be the reference Maintenance agency and refer to the dictionary elements associated with this Maintenance agency as reference objects/codes/elements.

2.2 Definition of a Cube

SMCube is a methodology for defining datasets based on their metadata. Its pivotal role is in defining Cubes, which define the structure of a dataset or logical entity, intended as a set of data organised as a table with attributes (columns, fields) and records (rows).

The following table provides an example of a dataset comprising information about granular loans:

Granular Loans					
[D] Instrument unique identifier	***Instrument type***	***Inception date***	***Legal final maturity date***	***Currency***	***Carrying amount***
aGranularLoan	Other loans	17/03/2015	17/03/2025	Euro	10,000
anotherGranularLoan	Finance leases	01/01/2016	01/01/2021	United States Dollar	13,000
…	…	…	…	…	…

Table 1: dataset of granular loans

The minimum information required to describe the structure of the dataset can be summarised with the following three questions:

1. **What are the attributes of the dataset?**

In the SMCube methodology, the attributes of a dataset are firstly defined as Variables (of the Cube defining this data set) as an independent, re-usable dictionary concept. The Variables are defined independently of the Cube allowing for reusability of concepts meaning that a Variable may be used in multiple Cubes. A Cube may comprise as many Variables as needed to define the dataset.

Referring to Table 1 the Variables are: Instrument unique identifier, Instrument type, Inception date, Legal final maturity date, Currency and Carrying amount.

2. **What are the restrictions for each attribute?**

Restrictions (of Variables) are organised in so called Domains. Domains can be enumerated, if they provide a finite list of **allowed values**, which we then call Members in SMCube (e.g. countries,

currencies, …), or non-enumerated if they provide a logical **constraint** which, when defined, is called Facet[1] in the SMCube methodology

The ***restriction*** of a variable in the context of a Cube is determined by a subset of its Domain, the so-called Subdomain[2].

In case of the Domain of the Variable Currency that comprises all possible currencies (e.g. EUR, USD) including aggregates like Currencies of the European Union and others it becomes obvious that these possible values need to be restricted to a subset when used in a Cube. The same holds true for non-enumerated Domains, e.g. the Variable *Carrying amount* may be defined on the monetary

Domain (allowing positive and negative values) while in the context of a Cube it may be restricted to a subset of this Domain (e.g. positive values only). Please note that a Subdomain may cover the whole Domain.

In the example dataset, some Variables are defined on non-enumerated Domains, like *Instrument unique identifier* (String Domain), *Inception date* (Date Domain), *Legal final maturity date* (Date Domain), *Carrying amount* (Monetary domain). The other Variables are defined on enumerated Domains: Instrument type, Currency.

The used Subdomains (for the Variables) in the context of the dataset illustrated in table 1 are as following: *Instrument unique identifier* (String up to 120 characters limited to letters (capital and low cases), numbers, dash and underscore), *Inception date* (All dates), *Legal final maturity date* (All dates), *Carrying amount* (Non-negative monetary amounts), *Instrument type* ({Credit card debt, Current accounts, Factoring, Financial leases, Other loans, Other trade receivables}), *Currency* (ISO4271).

Please also note that for each concept (i.e. Variables, Members) the dictionary shall provide additional information like a description or a legal reference.

3. What is the role of one attribute within one dataset?

One of the most relevant aspects of the structure of the dataset is what the identifier of the record is or, in other words, what combination of attributes makes a record unique. In the example dataset, if nothing is said regarding the structure, applying some business knowledge one may conclude that each record is uniquely identified by its *Instrument unique identifier*. On the other hand, there may be other datasets that may not contain an explicit identifier.

Thus, to get a thorough understanding of the described dataset, the role of the variables needs to be explicit. In the SMCube methodology, attributes that serve as identifiers of the records take the role of a ***Dimension***.

Variables that provide information related to the primary key (i.e. the set of dimensions) take the role of ***Observations*** and attributes that provide additional information related to a dimension or observation take the role of a ***Complementary attribute*****.**

[1] Facets allow us to apply additional constraints onto non-enumerated Domains, e.g. a pattern for the last day of the month.

[2] Please note that the utility of Subdomains for Variables in Cubes allows to organise similar concepts (e.g. countries, regions, etc.) in domains while only allowing a subset (e.g. countries only) of this Domain in a Cube.

It is worth highlighting that one Variable may take different roles in different datasets. For instance, the example dataset is based on granular loans and therefore the Variable Instrument type acts as an observation. In an aggregated dataset (e.g. table 2) this Variable may act as a Dimension.

Aggregated data set			
[D] Instrument type	***[D] Institutional sector***	***[D] Currency***	***Carrying amount***
Loans and advances	Non-financial corporations	Euro	11,000
Equity instruments	Financial corporations	United States Dollar	13,000
…	…	…	…

Table 2: aggregated dataset

In the context of this data set a record is uniquely identified by the Variables *Instrument type*, *Institutional sector,* and *Currency*. Consequently, the Variable *Instrument type* is part of the key of the dataset and therefore acts as a Dimension.

4. Representation using codes

For educational purposes we used the names of the concepts that are used in Table 1 to simplify the understanding of readers regarding the content of the dataset. Normally codes are used as representation of these concepts which makes it more difficult for the user to understand the semantic content of the concepts. Using codes Table 1 may look as following:

GRNLR_LNS					
[D] INSTRMNT_UNQ_ID	***INSTRMNT_TYP***	***DT_INCPTN***	***DT_LGL_FNL_MTRTY***	***CRRNCY***	***CRRYNG_AMNT***
1	1022	17/03/2015	17/03/2025	EUR	10,000
2	80	01/01/2016	01/01/2021	USD	13,000
…	…	…	…	…	…

Table 3: data set of granular loans using codes

Summary

With the SMCube methodology, one cube serves to define the structure of a dataset. One Cube is a set of Variables, for which the allowed values are specified by a Subdomain, and that have a role in the context of the Cube.

The metadata description of the Granular Loans data set (table 1) could be summarised as following:

Role	***Variable***	***Subdomain***
Dimension	Instrument unique identifier	{String up to 120 characters limited to letters (capital and low cases), numbers, dash, and underscore}
Observation	Instrument type	{Other loans, Financial leases, Reverse repurchase agreements, Factoring, Other trade receivables, Current accounts, Credit card debt}
Observation	Inception date	{All dates}
Observation	Legal final maturity date	{All dates}
Observation	Currency	{ISO 4217}
Observation	Carrying amount	{Non-negative monetary amounts}

Table 3: meta data description of Granular Loans data set

The meta data description of the Aggregated data set (table 2) could be summarised as following:

Role	***Variable***	***Subdomain***
Dimension	Instrument type	{Loans and advances, Equity instruments, Debt securities}
Dimension	Institutional sector	{Non-financial corporations, Financial corporations, General government, Households}
Dimension	Currency	{Euro, United States Dollar, Yen, Other currency}
Observation	Carrying amount	{Monetary amounts}

Table 4: meta data description of Aggregated data set

Please note that for educational purposes we used the names of the concepts.

3. The SDD's Metadata Model (MM)

This section is dedicated to the description of the SDD's Metadata Model (MM). It tries to describe the various objects of the MM and their purpose. The MM is separated into multiple packages that serve different purposes. In the following chapters we will describe the individual packages, their purpose, and the involved objects of the MM.

3.1 Core package

The core package comprises the basic elements of the dictionary, e.g. the concepts that are used in the dictionary and which belong to the conceptual/semantic layer. Those basic elements are used by the other packages to:

(i) define Cubes (Data definition),
(ii) indicate the relationship between non-reference and reference concepts (Mappings),
(iii) provide a template representation of the Cube (Rendering),
(iv) provide a description of how on some concepts can be transformed into other concepts.

The Core package comprises the following objects of the MM:

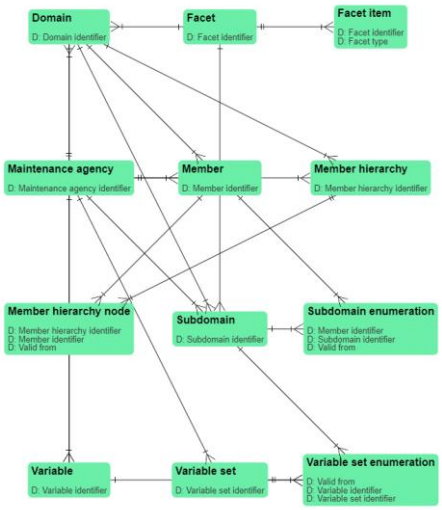

Figure 1: The SDD Core Package

3.1.1 Domain

Domains represent the categories of reality in which information may be organised. Examples of Domains are geographical areas, currencies, or Dates. We distinguish between enumerated and non-enumerated Domains where enumerated Domains are described by (a set of) Members.

3.1.2 Subdomain and its enumerations

A Subdomain is a restriction representing a subset of a Domain. Examples are the countries of the European Union where the Euro is the official currency, or the currencies used in at least one of the countries of the European Union. Similar to Domains we distinguish between listed (or enumerated) and non-listed (or non-enumerated) Subdomains. Please note that the Members of a listed Subdomain do not (necessarily) have to be disjointed, although for the description of normalised data it is recommended to define Subdomains comprising Members that are disjointed. The composition of the Members (of a Subdomain) may change over time (e.g. the composition of a subdomain describing the member states of the European Union).

3.1.3 Member

A Member is a concept in the dictionary which represents allowed values of a Domain. Examples are Austria (AT), Italy (IT), Portugal (PT), Euro (EUR) or not applicable (0). Members are defined on a (enumerated) Domain and may be organised in (listed) Subdomains. A hierarchy of members can be described using Member hierarchies. Members define the allowed values of a specific column (or Variable) in the context of a data set (or Cube).

3.1.4 Variable

A Variable is a concept in the dictionary and is defined on a Domain. E.g. Identifier, Carrying amount, Country, Institutional sector. The Domain specifies the possible values of the Variable. Variables can be represented in logical entities as attributes (which physically can be table columns/fields) or allowed values of another attribute defined on a Variable Set.

3.1.4.1 Variable set & Variable set enumeration

One of the main principles of SMCube methodology is that a concept is only represented once and is univocally identifiable. To comply with this principle, but at the same time allow covering different datasets based on different use cases, we need the functionality provided by so called Variable sets. Imagine for example that the concept of *Carrying amount* is already stored in the dictionary as a Variable (so we can think about it as a column of a data set). At the same time another data set may use the concept of *Carrying amount* but not presented as a column but as the value of a column. To comply with the above stated principle this value must not be stored as a Member but as a Variable. The following tables show the same data represented in different data sets, the first one does not use the concept of Variable sets (i.e. the concepts of *Carrying amount* and *Fair value* are represented as columns) while the second one applies the concept of variable set (i.e. the concepts of *Carrying amount* and *Fair value* are represented as members although they are already defined as variables).

Data set representation without variable set		
[D] Instrument type	***Carrying amount***	***Fair value***
Reverse repurchase loan	31	29
Factoring	19	23
…	…	…

Table 5: Representation of a data set without a variable set (horizontal format)

Data set representation with variable set		
[D] Instrument type	***[D] Type of value***	***Value***
Reverse repurchase loan	Carrying amount	31
Reverse repurchase loan	Fair value	29
Factoring	Carrying amount	19
Factoring	Fair value	23
…	…	…

Table 6: Representation of the same information illustrated in Table 5 using a variable set (vertical format)

Please note that in the second case the variable *Type of value* is part of the primary key.

Please also note that the representation using a variable set (see Table 6) is used, for example, in the EBA ITS templates that are imported from the DPM.

3.1.5 Member Hierarchy and its nodes

Member hierarchies serve to establish a hierarchical relationship between the Members of a Domain (taxonomy), i.e. they give structure to Members in the dictionary. These relationships are independent of datasets, and therefore they are part of the core package. Member hierarchies provide a very powerful tool for data reconciliation and production. Some use cases for hierarchies are:

- to document relationships between the members of a Domain (e.g. Loans and advances = {Loans, Advances})
- to run validations on data based on the comparators defined in the Member hierarchy (e.g. the Carrying amount of Loans and advances needs to be greater or equal to the Carrying amount of Loans),
- to apply Member hierarchies in the aggregation process (e.g. Carrying amount of Loans plus the Carrying amount of Advances)

A Member hierarchy node is a Member within the context of a hierarchy. Each node belongs to a specific hierarchical level, and has a parent member, except for members at the top of the hierarchy. For each node, it is possible to define the operator.

For each node, it is possible to define the operator to be used to perform operations amongst the siblings and the comparator operator to be used on the parent member to compare with its children's operation result. For example, it is possible to define a parent member in the hierarchy with the '>=' comparator for a list of children, whereby the operator attribute is equal to '+'; this expresses a rule that the parent Member has to be greater or equal to the sum of the children.

3.1.6 Facet Collection and Facet Enumeration

A Facet is a constrain which enables the restriction of non-enumerated attributes and the casting of Variables when attributed to a Subdomain or Domain. A Facet can contain zero or more Facet items. Each facet item contains the attributes FACET_TYPE (valid format of the facet) and OBSERVATION_VALUE (valid content for the defined format – one or more).

3.2 Data Definition Package

The data definition package defines the structure of the Cubes (datasets) described, and groups them into frameworks. The relationships between the tables in the package are shown in the following diagram:

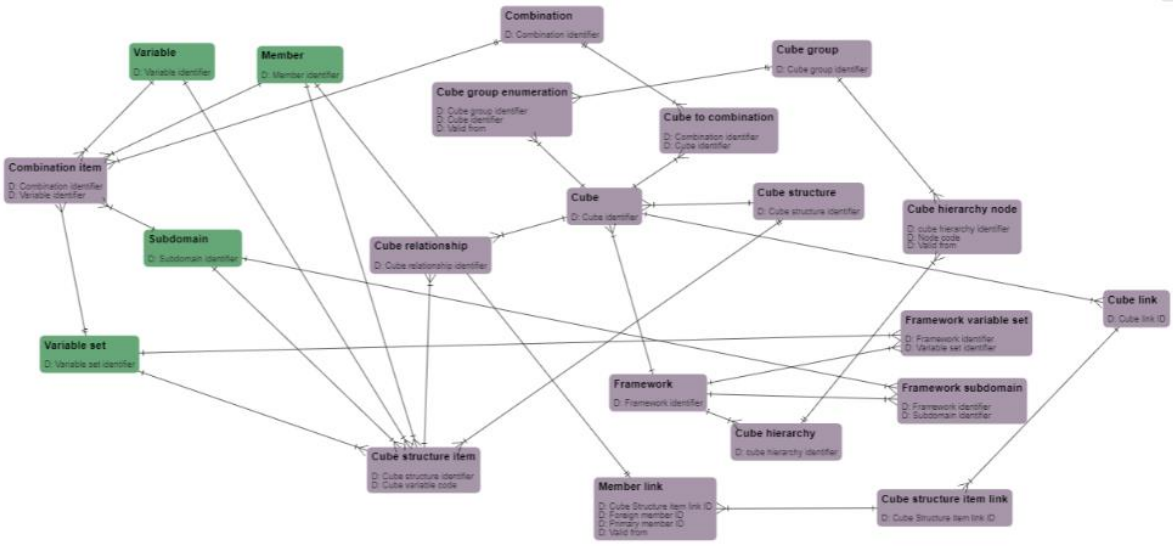

Figure 2: The SDD Data Definition package and its most important related entities

We will now walk through the definition and role of each of the tables presented in the figure 1. All tables must be identifiable, most are nameable and maintainable.

3.2.1 Framework

The framework is normally a regulation or mandate for which the datasets are collected, such as AnaCredit or SHS. A Framework contains Cubes describing the structure of the different datasets which belong to the regulation. In some other cases, a Framework can represent a set of Cubes belonging to a certain business area, when there is no mandate nor regulation defined.

3.2.1.1 Framework Subdomain & Framework Variable Set

A Framework Subdomain and Framework Variable Set can establish the many-to-many relationship between Frameworks and Subdomains, and Framework Variable Sets respectively. The justification of these relationships is that Subdomains/Variable Sets may not only be used directly in the Framework

(e.g., in a Cube structure item associated to the Framework) but also indirectly (e.g. a Subdomain/Variable Set that is used in the Transformations or Validation Rules between Frameworks).

3.2.2 Cube

A Cube is the description of a certain logical entity which can be implemented as a physical table, dataset, or data exchange artefact. It's the central element of the SMCube methodology. It is important to note that Cubes must be versionable and are a part of a Framework.

Each Cube is associated to a particular type. Types include collection, production, dissemination and staging etc. Cubes can be restricted by Combinations which contain one or more Combination items. Cubes metadata also includes the information on whether the combinations associated with the Cube represent the combinations of allowed values (regions or data points) for the Cubes, or combinations of not allowed values.

3.2.3 Cube Structure

A Cube structure is a collection of structural elements (Cube structure items) which depict the multidimensional structure of a Cube. Different Cubes can be based on different subsets of elements of the same Cube structure. This table serves to ensure compatibility with the SDMX standard, where more than one cube can be defined based on the same Cube structure. In practice, in most cases (including the BIRD, AnaCredit, FINREP, and SHS) there is a 1 to 1 equivalence between Cube and Cube structure.

3.2.4 Cube Structure Item

Each Cube structure item represents an attribute by referring to the correspondent generic and reusable Variable; it has a specific role (Observation value, Dimension, or Complementary Attribute), and is associated with a list of possible elements via a Subdomain, an implicit (fixed) Member, and only in case of measure Dimension, a Variable set. The implicit Member is assigned in case the correspondent attribute can only assume one specific allowed value in the context it that Cube.

There are three types of Cube structure items according to their role:

- Dimension: Dimensions represent identifiers of the Cube, similarly to primary keys if the Cube is represented in a database table. If the Cube is conceptualised as a mathematical function, dimensions are the independent variables.
- Observation value: Observation values are items that provide information on the full set of Dimensions. In mathematical terms, they are the dependent variable, which adopt a value for each combination of values for the dimension of the cube.
- Complementary Attribute: Complementary Attributes provide additional information on a single Dimension or Observation value. Attributes are dependent variables, but they depend on a single element (that can be dimension or variable), while the observation values depend on the combination of all values for the dimension.

Additionally, a dimension will be one of four types:

- Time dimension: Time dimensions provide the information about the time in the Cube. Typically, it is a reference period for dynamic data, or valid from and valid to for registries or static data.

- Unit dimension: Unit dimensions represent the statistical unit being analysed, or the subject of the information. In Cubes that represent information about banks, the unit is the dimension that specifies the bank to which the information refers.
- Measure dimension: in some Cubes, the characteristics of the dimensions, normally represented by the observation values, are folded into a single variable, normally called the observation value or fact. In such cases, there is a dimension that specifies the meaning of the observation value, and that is the measure dimension. Measure Dimensions are the only Cube structure items that are associated with a Variable set (or an implicit Variable) instead of a Subdomain (or implicit Member).
- Breakdown dimension: All other Dimensions that are not one of the types above.
- Within the Cube structure item table, the attribute order is mandatory and defines the order of elements within a Cube structure. The attribute CUBE_VARIABLE_CODE represents the code of the Variable in the implementation of the Cube. It is linked to the distribution of the dataset. For observation values and attributes the IS_MANDATORY field indicates whether the element must appear in the Cube. The IS_FLOW field indicates whether a variable contains 'stock' or 'flow' values.

3.2.5 Combination & Combination Item

Combinations restrict the multidimensional space within a Cube, by specifying the combinations that are/are not allowed. When Combinations list the 'allowed' combinations in the form of pairs, such as Dimension/Member, they are comparable to 'time series' in SDMX and 'data point' in the DPM.

Combination must contain one or more Combination items. Each Combination item represents a combination of a Variable, a Member and a Subdomain, or another Member.

3.2.6 Cube to Combination

A Cube to Combination is a combination of a Cube and a Combination. It establishes the many-to-many relationship between Cubes and Combinations.

3.2.7 Cube Group & Cube Group Enumeration

A Cube Group is a collection of Cubes defined by the user to facilitate the navigation of Cubes.

3.2.8 Cube Hierarchy & Cube Hierarchy Node

A Cube hierarchy is a hierarchical structure (tree) of Cubes (tables). Cube hierarchies are also used to classify components of a data model according to business classifications. A Cube hierarchy node is a node in a tree structure of a Cube hierarchy in a specific interval in time.

3.2.9 Cube Relationship

A Cube relationship is a relationship between Cubes in a data model, i.e., the primary and foreign Cubes. They are used (only) to represent relationships of an Entity Relationship Model. The TYPE_OF_RELATIONSHIP field specifies the type of relationship the object refers to. Associative (ASS) relationships associate two Cubes via a one-to-one, one-to-many, many-to-one, or many-to-many relationship and provide information on the PRIMARY_CUBE_VARIABLE_CODE and

FOREIGN_CUBE_VARIABLE_CODE which identify the linked Cube Structure Items. Generalisation (GEN) relationships represent a parent-child relationship between Cubes (logical entities). Within the Cube Relationship table, we can specify both the cardinality (e.g. one-to-one, one-to-many,) and the optionality (mandatory or optional) of the associative relationship. Moreover, this object can contain information on which are the primary and foreign identifiers of the Cubes involved in the relationship.

3.2.10 Cube Link

A Cube link allows to define generic links between different Cubes. For example, Cube links can be used to define the way Cubes of different Frameworks can be joined to integrate their data or the way BIRD Logical Data Model (LDM) is forward engineered in the BIRD Input Layer. It includes a Primary Cube, which represents the base/input Cube of reference and a Foreign Cube, which represents the integrated/output Cube. It should be distinguished from the Cube Relationship object, which scope is defined in the related section of this document.

3.2.11 Cube Structure Item Link & Member Link

A Cube structure item link is a link between two Cube structure items for a certain Cube link. The PRIMARY_CUBE_VARIABLE_CODE represents the linked CUBE_VARIABLE_CODE of the Cube structure item (attribute) of the primary Cube, while FOREIGN_CUBE_VARIABLE_CODE represents the linked CUBE_VARIABLE_CODE of the Cube structure item (attribute) of the foreign Cube. On the other hand, a Member link is a link between two Members in the respective Subdomains of the Cube structure items linked in a Cube structure item link. The linked Variable/Members are either the same Variable/Member or one is a parent (or ancestor) of the other one, in one Primary Concept relationship/Member hierarchy. A Member link object can be linked (IS_LINKED=TRUE) if the foreign Member is available in the foreign Cube structure item or not linked (IS_LINKED=FALSE) if the foreign Member is not available in the foreign Cube structure item. This information is normally important to be considered for Cube links representing data Integration between granular and more aggregated Frameworks as it makes explicit the differences between Primary and Foreign Cubes. For example, AnaCredit Framework does not include advances among its instrument types while FINREP does. Therefore, when linking the respective Members of the Instrument type, the Member *Advances* will be defined with IS_LINKED=FALSE.

3.3 Mapping Package

The mapping package is responsible for linking different codification systems. The links are done at the levels of Variables, Cubes, and Member in a Variable.

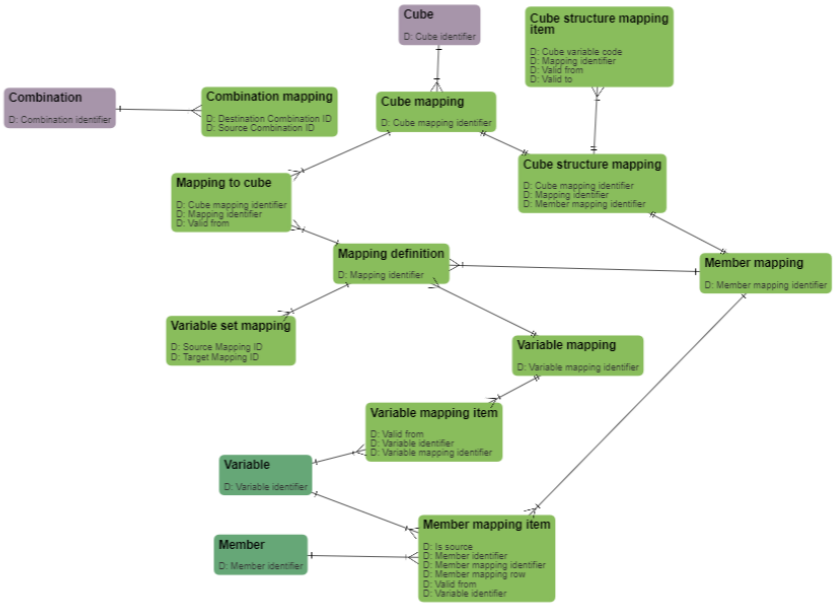

Figure 3: The SDD Mapping package and its most important related entities

3.3.1 Mapping Definition

Each Mapping definition describes the details of the mapping of a set of source Variables to a set of output/destination Variables. Mapping definitions are defined firstly on a conceptual level but can be applied on a logical or physical level if they are used in a Cube mapping process. A Mapping definition contains one or more Variable mapping items (linked via Variable Mapping) and can refer to one Cube mapping (combination of source and destination Cubes). There are four types of mappings: D (Deletion), A (Algorithm), E (Equivalence), G (Generation), and V (Variable Set). They are subtypes of the Mapping definition.

- Deletion mapping serves to eliminate one Variable from the source cube in the destination one. Yet the destination structure is consistent with the source one.
- Algorithm mappings can be used to map non-listed Variables. The algorithm needs to be defined within the mapping.
- Equivalence mappings map listed variables, according to the content of an equivalence table. Equivalence tables provide the mapping from source to destination Members.
- Generation mappings (G) can be used to include some additional and implicit information in the target Cube. They are subtypes of Equivalence mappings, as they are in general zero-to-n mappings.
- Variable set mappings (V) are used to map Variable sets.
- Mapping definitions are normally used to map non-reference Variables to their correspondent reference version. Please note that normally it is assumed that for Mapping Definitions to be unambiguous, the involved input Variables should be included only once in the input Cubes.

3.3.2 Variable Mapping & Variable Mapping Item

Each Variable mapping item defines which Variables are involved in a mapping (source and destination). This element is used for all types of Mapping definition, but Deletion mappings.

3.3.3 Member Mapping & Member Mapping Item

Each Member mapping item defines which Members within a Variable are involved in a mapping (source and destination). This element is used only for equivalence (enumerated) mappings, i.e. the ones including enumerated Variables. Member Mapping Items are organised in rows. Every row maps a defined combination of Members for all the involved source Variables into a combination of Members for all the output Variables of the Mapping Definition.

3.3.4 Variable Set Mapping

Variable set mapping is an object representing the link between source and target Mapping definitions. It is used to map measure dimensions, which possible values are sets of Variables instead of sets of Members. The source Mapping Definition refers to the mapping responsible to map the Variable defined over the Variable Set (in a Cube) while each related Target Mapping Definition refers to the mapping responsible to map a Variable contained in the Variable set.

3.3.5 Cube Mapping

Cube mapping is an object that allows to describe the mapping of a source non-reference Cube to a target (normally reference or partially reference) Cube.

3.3.6 Mapping to Cube

A Mapping to Cube is a combination of a Cube mapping and a Mapping definition in a specific time interval. It establishes the many-to-many relationship between Cube mappings and Mapping definitions. The latter are used to map the Cube structure items of the source Cube identified in the related Cube mapping Such Cube structure items are divided in groups of interdependent elements. Every group of Cube structure items is mapped by an individual Mapping definition.

3.3.7 Combination Mapping

Combination mapping is an object that allows to describe association between a target Combination and a source Combination.

3.3.8 Cube Structure Mapping and Cube Structure Mapping Item

A Cube structure mapping is a mapping object responsible to map a subset of Cube Structure Items for a specific Cube (structure). While Mapping Definitions are re-usable generic mappings focusing on Variables and the related Members, Cube structure Mappings and related items focus on mapping Cube structure items directly. Such mappings can be used to cover mapping uses cases which are not allowed/recommended to be covered using Mapping definitions.

For example, this object can be used to map output Cube structure items where the CUBE_VARIABLE_CODE is different than the CODE of the Cube structure item-related Variable, which is normally the case for Cube structure items having the role of Complementary attributes (“A”). It can also be used to map structure to a non-reference codification system or even the same codification system as the input one.

Like Mapping definitions, Cube structure mappings can also be linked to Member Mappings to express the mappings of Cube structure items allowed values (Members).

3.4 Rendering Package

The Rendering package allows stakeholders to represent data combinations in predefined multidimensional table formats. It therefore provides the possibility of creating predefined formatted templates to present data and linking the template structure with the dataset/cube items (variables, member hierarchies and members) it belongs to.

The rendering package is based on the rendering used by XBRL and the DPM 1.0.

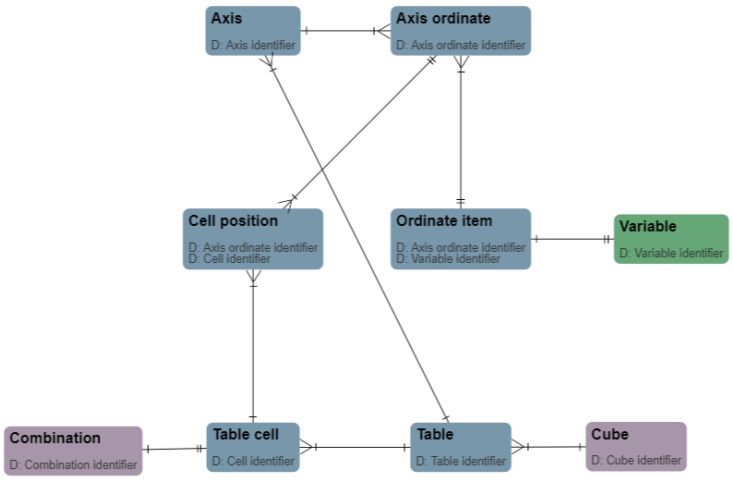

Figure 4: The SDD Rendering package and its most important related entities

3.4.1 Table

The Table object represents the template content structure for the representation of data. For example, a balance sheet table report with a format. Tables are composed of one or many table cells.

3.4.2 Axis

The object Axis contains information about one dimension of a multidimensional table. Tables contain at least two axes, x (columns) and y (rows). For example, balance sheets may have a list of items by rows and two time periods (current and previous) by columns. An axis must be identifiable and nameable. A Table can have two or more Axis. Each Axis will have an orientation (e.g. x, y, z) and an order.

3.4.3 Axis Ordinate

An Axis ordinate object represents an item of a dimension in a Table. Each row and each column of the balance sheet constitute an Axis ordinate. Each ordinate may have an associated dimensional description, i.e. a set of pairs, such as Variable/Member or Variable/Subdomain from the core package.

An Axis ordinate must be identifiable and nameable. Each Axis ordinate contains the attributes order (order of the presentation of the element), and level (the level of hierarchy of ordinates).

3.4.4 Table Cell

A Table Cell represents the cell of a Table. It is used to link the Cell to a Combination. Table Cell contains the attribute IS_SHADED for those cells that should not have a value (i.e. they are greyed-out).

3.4.5 Cell Position

A Cell position describes the combination of a Table Cell and an Axis ordinate; it provides the coordinates of a Table Cell.

3.4.6 Ordinate Item

An Ordinate item provides information about the Variables and their related allowed value (Member) assigned to a certain row or column of a certain Table.

3.5 Transformation Package

The Transformation package allows stakeholders to represent information on physical, logical and semantic Transformation rules in the dictionary. Objects in this package are divided in two main groups:

- Transformation scheme, Transformation and Transformation node objects are used to store information/metadata on physical/technical transformations which can refer, for example, to SQL or VTL statements. These transformations provide a fully detailed representation on how to elaborate data through the different steps of a data process (e.g. ETL). They provide information on calculations/formulas, filtering and joins between different tables of a data model. Transformations can be de-composed into different nodes, using the abstract syntax tree of the respective language (e.g. SQL).
- Semantic transformation rule, Transformation to Variable and Transformation to Cube are used to store information on semantic transformations. Such transformations focus on providing more high-level and business-friendly information than the physical/technical transformation. They hide several details such as join conditions between input table to produce output information or “group by” statements. Such information is already provided by other dictionary components such as data models, which can contain information on relationships between entities, allowing to reproduce needed join conditions; it also provides information on dimensions/keys which allows systems to understand the different level of granularity between input and output Cubes. Whenever an output Cube includes a subset of the combined input Cubes dimensions, the transformation rule implies that a “group by” condition is needed to reduce the level of granularity of the input information (i.e. aggregation). Furthermore, previously defined Semantic transformation rules information can be enriched with the full logical links between input and output Cubes by providing Transformation to Cube link objects information.

Please note that, as for several other SMCube packages, the level of richness of metadata description can vary and depend on business needs. Stakeholders of a certain Framework/dataset can document information both on a semantic/logical and physical/technical level, the former deemed mostly for

business users while the latter for technical users/systems. Stakeholders can however also provide information for only one of the two types or none.

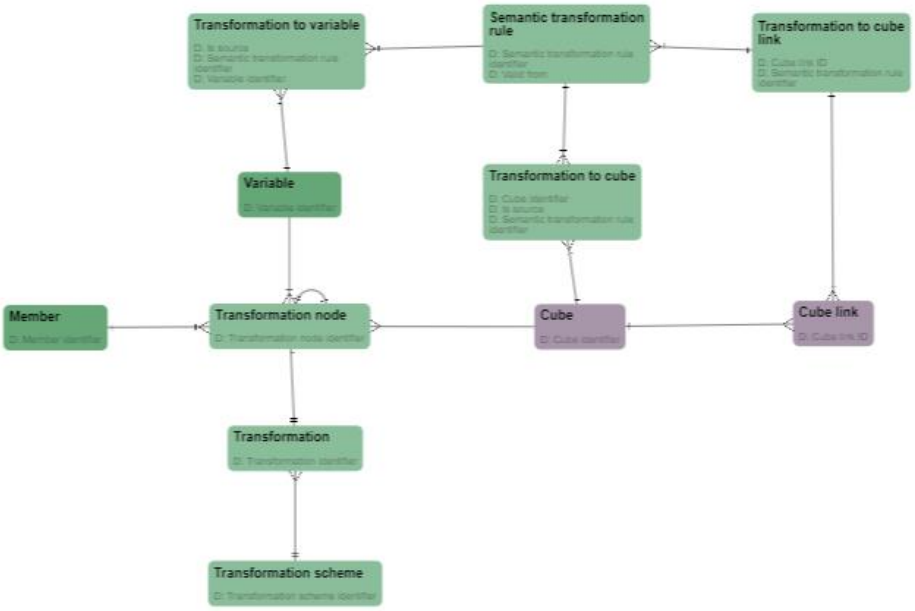

Figure 5: The SDD Transformation package and its most important related entities

3.5.1 Transformation Scheme

A Transformation scheme is an object that allows to organise Transformations in schemes (blocks). For example, all Transformations describing operations on the same initial data set.

3.5.2 Transformation

A Transformation is an expression that describes the manipulation of a Cube (entity/data set) where the result is still a Cube..

3.5.3 Transformation Node

A Transformation node is a component of a Transformation obtained by de-composing a Transformation based on its abstract syntax tree. It can include operands, operators, parameters etc.

3.5.4 Semantic Transformation Rule

A Semantic transformation rule is a Transformation rule that describes in words, or semantically, how data can be transformed to produce other data needed for any business need. A Semantic transformation rule can be either a derivation or a generation Rule. Derivations have as a purpose to derive one Variable based on a set of other Variables. Generations are used to generate one Cube based on a set of other Cubes given a possible filtering condition. A Semantic transformation rule can be defined once for the same output structure and linked to more than one set of Cubes, each set

belonging to a different model, given that they include the necessary input information. In the ALGORITHM field the formal part of the transformation rule can be expressed using standard SQL language focusing on formulas/calculations and control structures such as IF, ELSE etc. ALGORITHM in this object normally includes information on generic Variables and Members with no reference to specific Cubes and related attributes, to allow the Semantic transformation rules to be re-used in different contexts/input models.

3.5.5 Transformation To Variable

Transformation to Variable represents the combination of a Semantic transformation rule with one of its related Variables. It establishes which are the input and output Variables of a Semantic transformation rule and therefore provides the Transformation links on a purely Semantic Level. It is normally defined only for derivation Semantic Transformation rules.

3.5.6 Transformation To Cube

A Transformation to Cube is the combination of a Semantic Transformation Rule with one of its related Cube. It links a Transformation Rule with the generated output Cube and, if necessary, with a set of input Cubes. It is normally defined only for Generation Semantic Transformation Rules and only in case specific information on the logical links between input and output Cubes cannot be provided, for example when the Semantic Transformation rules describe only high-level instructions on how to generate the Cube.

3.5.7 Transformation To Cube Link

A Transformation to Cube is the combination of a Semantic transformation rule with one of its related Cube links. It associates any type of Semantic transformation rule with the related links between input and output Cubes which, in turn, refer to Cube structure items and Members links. It can be used for example to apply the Semantic transformation rules in the context of a specific data model or across different data models. In the ALGORITHM field the formal part of the transformation rule can be expressed using standard SQL language focusing on formulas/calculations and control structures such as IF, ELSE etc. ALGORITHM in this object includes information on a specific Cubes and attributes (Cube Structure Items). This field provides more detailed information than the one of the related Semantic transformation rules since it related to a specific layer/data model. Consequently, it is easier for users to automatically translate the content of this field (when available) into physical/technical transformation rules which can be applied on the real data.

3.6 Legal Reference and Classification Package

The Legal reference and Classification package allows stakeholders to associate legal references to SMCube objects. For example, it is possible to associate a specific Variable or Member to the regulations which required information on such Variable/Member to be reported to authorities (e.g. CRR, AnaCredit regulation etc.).

This package allows also to provide a generic classification (“tag”) to SMCube objects. For example, it could be used to classify the Risk Weight Variable as a “risk-related” information.

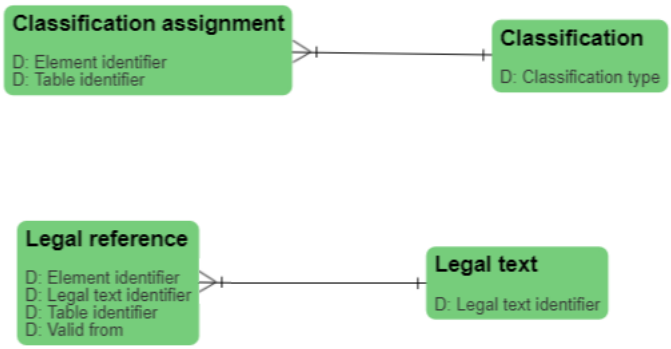

Figure 5: The SDD Legal Reference and Classification package

3.6.1 Legal Text

A Legal text is a reference to a document.

3.6.2 Legal Reference

A Legal reference is the combination of a dictionary element/object and a Legal text. Please note that the dictionary element/object is referred to via the combination of {Table identifier, Element identifier} and is therefore restricted to dictionary elements/objects having a primary key in the form of one field only.

3.6.3 Classification

A Classification is a category of information. It is used to classify an object in the dictionary.

3.6.4 Classification Assignment

A Classification Assignment is a combination of a dictionary object (element) and the related Classification. It represents the many to many connection between dictionary objects and Classifications. Please note that the dictionary element/object is referred to via the combination of {Table identifier, Element identifier} and is therefore restricted to dictionary elements/objects having a primary key in the form of one field only.

Annex 1: The SDD Metadata model