SMCube Information Model

Table of contents

1. Introduction

This document describes the Information 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 information model is to define an abstract model that is able to 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 on the basis of different methodologies is a clear obstacle to data integration.

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

1.2 SMCube Information Model requirements

The SMCube Information 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
- **Historisation:** follows the changes in time of defined datasets
- **Complex mappings:** integrate existing dictionaries and create links between similar information
- **Extensibility:** gives the possibility of other organisations implementing the SMCube methodology and making use of the definitions provided by the ECB in the SDD

1.3 Contents of this document

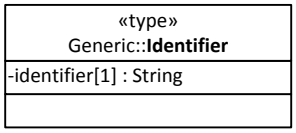
The SMCube Information Model describes artefacts by outlining their properties and purposes, it is defined in terms of UML-class diagrams, structuring the entities and the relationships between them.

Packages are at the basis of the SMCube Information Model, they describe specific aspects of the model, for each of them a class diagram and a definition of each encompassed entity is provided. Finally, a global perspective is given with a full overview of the capabilities and services described in the model.

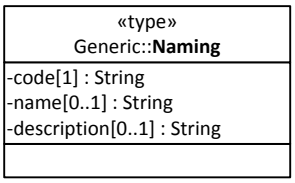
2. Metaclasses

The SMCube contains five custom data types (metaclasses) to define basic characteristics and attributes of its entities:

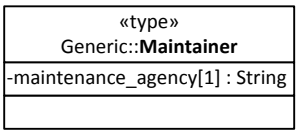
- *Identifier*: contains a unique identifier for each element in the entity.



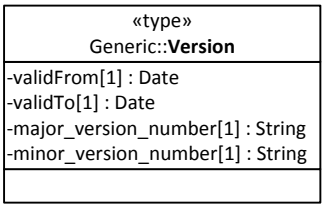
- *Naming*: contains a combination of the mandatory code and name, and optional description.



- *Maintainer*: contains the ID of the maintenance agency in charge of maintaining the entry.



- *Version:* contains mandatory time-based historisation attributes (*valid_from* and *valid_to*) and a versioning attribute (*version*). The version class deals with major and minor changes (see *Versioning concepts*) via these three attributes.



3. Core package

The aim of the core package is to provide the elements that describe the reality of data. Several abstraction layers are used for this purpose. For example, datasets are composed of variables, which are then populated with information. The core package allows such variables to be described, as well as the information that they will contain.

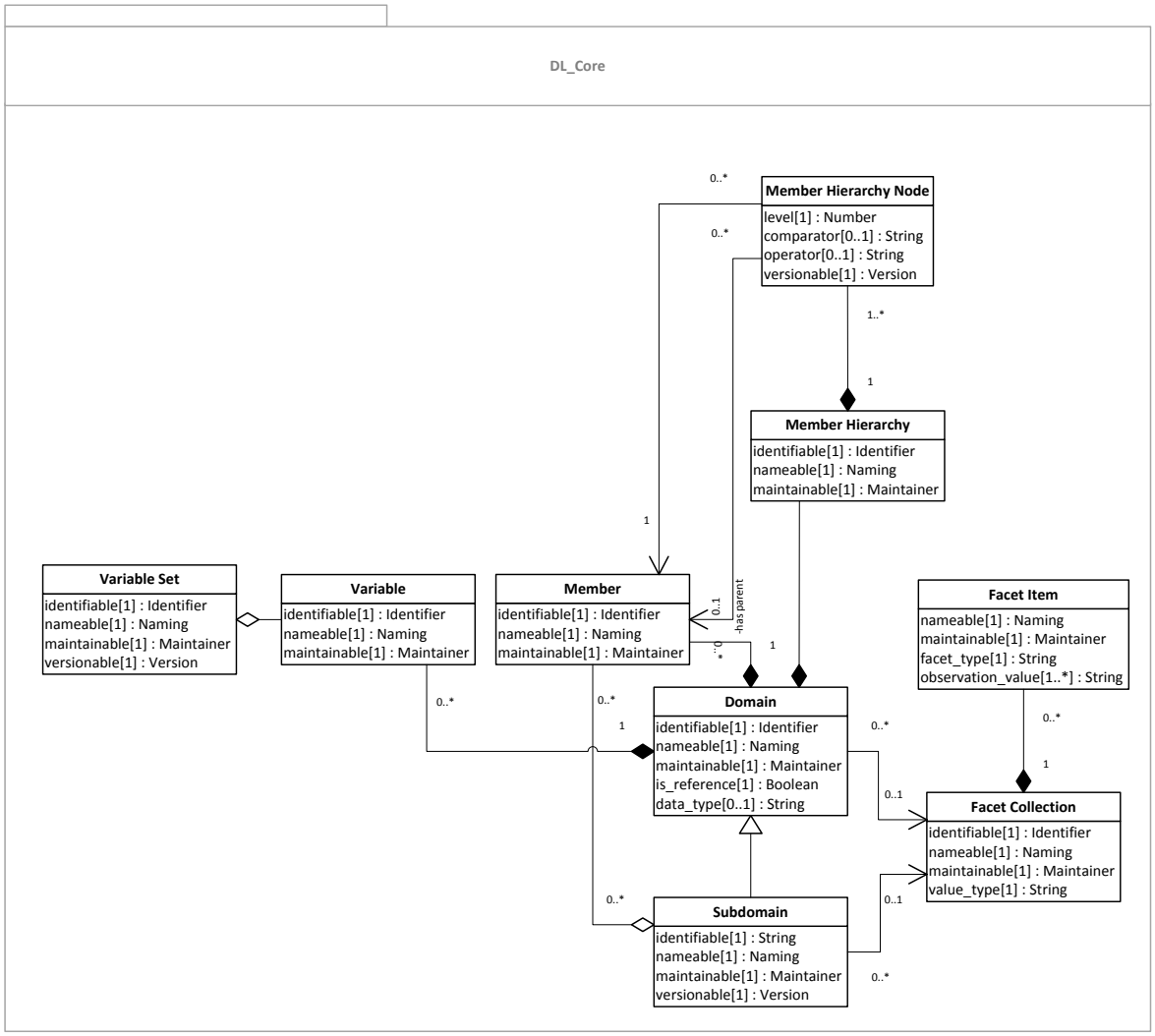


Figure 1: Core package

3.1 Definition of core package elements

3.1.1 Domain

A domain represents the categories of reality in which the information can be organised. Examples of domains are geographical areas or currencies. They can be enumerated (i.e. contain a list of

members or a list of variables for a measure dimension), represent a list of facets (i.e. a list of rules, such as minimum length, data type, or a pattern), or both.

- Must be identifiable, nameable and maintainable.

3.1.2 Member

A member represents a single meaning or a value. For example, for the variable "country", a possible member would be "ES" for Spain. A member could also be a default value (e.g. "-1" for non-reported).

- Must be identifiable, nameable and maintainable.

3.1.3 Subdomain

A subdomain is a restriction of a domain, delimitating the possibilities through facets or enabling the creation of lists of members that can be used in a variable. For example, a list of countries with the euro as a currency is a subdomain of the geographical area domain, thereby confining the list of countries to those which use the euro.

- Must be identifiable, nameable and maintainable.
- It will either contain a list of members or a facet.
- The list of members is historised.

3.1.4 Variable

A variable adds meaning to a domain. For instance, the member "ES" may be used with different meanings, depending on the variables, which can, for example, be the fair value, net value or the *"country of residence of the debtor" or "country of location of activities"*. In terms of implementing a dataset, the variable will typically be a column in a table.

- Must be identifiable, nameable and maintainable.
- May be part of zero or more variable sets (a superset of variables).

3.1.5 Variable set

A variable set is a superset of variables, grouping together variables, where necessary.

- Must be identifiable, nameable and maintainable.

3.1.6 Member hierarchy

Members can be part of a hierarchy. This enables relationships between members to be created and subsequently used for analysis and definition.

It is possible to define several hierarchies for each domain where the rule is that each node can have only one parent member (except the root that does not have a parent member).

- Must be identifiable, nameable and maintainable.

3.1.7 Member hierarchy node

A member hierarchy node is a member within the context of a hierarchy. Each node belongs to a specific hierarchy level, and has a parent member (except for the top level of a hierarchy tree). 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 its children.

- Must be historised.
- Contains the attribute *level* (hierarchy level), *comparator* (symbol marking the interdependency between other elements in the hierarchy), and *operator*.

3.1.8 Facet collection and facet item

A facet enables the restriction of non-enumerated fields and the casting of variables when attributed to a subdomain or a domain.

- Must be identifiable, nameable and maintainable. A facet is composed of zero or more facet items.
- Each facet item is maintainable and nameable, and contains the attributes *facet_type* (valid format of the facet) and *observation_value* (valid content for the defined format – one or more).

4. Data definition

The data definition package defines the structure of the cubes (datasets) described, and groups them into frameworks.

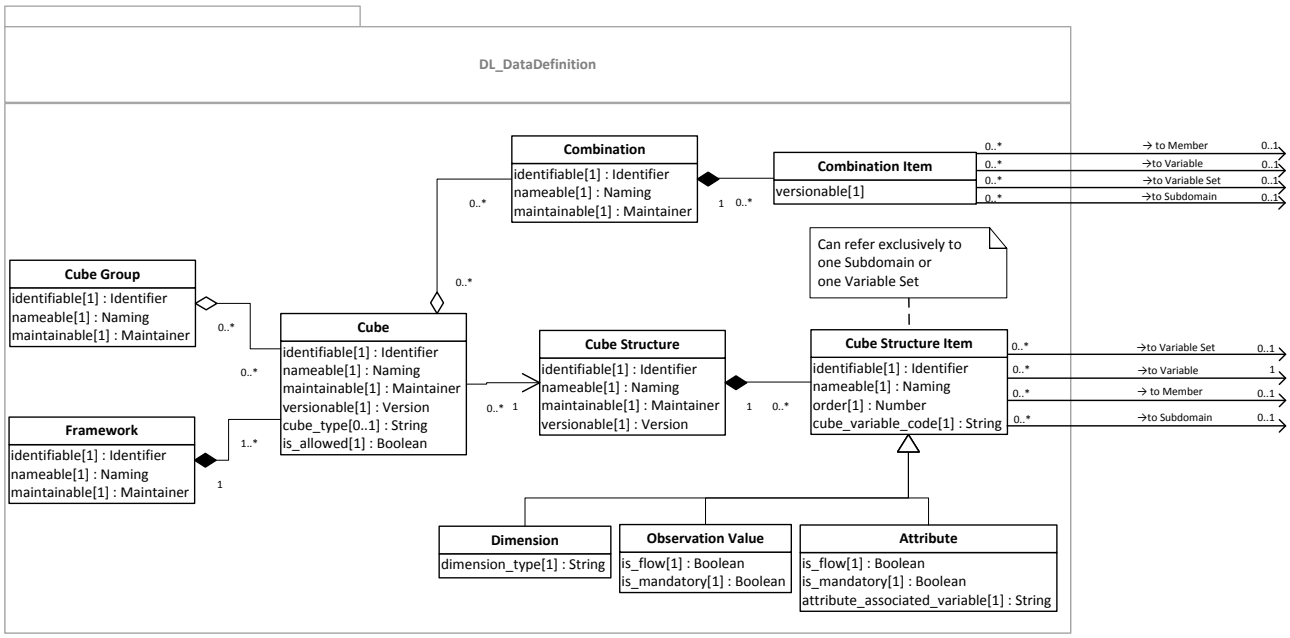


Figure 2: Data definition package

4.1 Definition of the data definition elements

4.1.1 Framework

The framework is typically a regulation or mandate for which datasets are collected, e.g. AnaCredit. It contains cubes describing the structure of the different datasets which belong to the regulation.

- Must be identifiable, nameable and maintainable.

4.1.2 Cube

A cube is the description of a certain dataset. It is the central element of the SMCube. A cube is typically implemented as a table in a database or as a data exchange artefact.

- Must be identifiable, nameable, maintainable and versionable.
- Must be part of a framework.
- Each cube may have a type associated to it: C (collection); P (production); D (dissemination); or S (staging).
- May be restricted by combinations containing one or more combination items.

- The attribute “is allowed” defines whether the combinations associated with the cube represent the allowed values (regions or data points) for the cubes, or the “not allowed” values.[1]

4.1.3 Cube group

A cube group is a collection of cubes defined by the user to facilitate navigation in a variety of cubes. They do not reflect the grouping associated with the framework.

- Must be identifiable, nameable and maintainable.

4.1.4 Cube structure

A cube structure is the collection of structural elements (cube structure items) defining the multidimensional structure of a cube. Different cubes can be based on different subsets of elements of the same cube structure.

- Must be identifiable, nameable, maintainable and versionable.

4.1.5 Cube structure item

Each item represents a variable (field), which has a specific role (observation value, dimension or attribute), and is associated with a list of possible elements via a subdomain, an implicit member and, only in the case of measure dimensions, a variable set or an implicit variable.

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

- Dimension: these 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, the dimensions are the independent variable.
- Observation value: these are the items that provide information on the full set of dimensions. In mathematical terms, they are the dependent variable, which adopts a value for each combination of values for the dimensions of the cube.
- Attribute: these provide additional information on a single dimension or observation value. So they are dependent variables, but they depend on a single element (that can be a dimension or variable), while the observation values depend on the combination of all the values for the dimensions.

Additionally, a dimension will be one of four types:

[1] See SDMX similar concept in https://sdmx.org/wp-content/uploads/SDMX_2-1-1_SECTION_2_InformationModel_201108.pdf (line 2008).

- Time dimension: this is the dimension or dimensions that 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: this is the dimension that represents the statistical unit being analysed, or the subject of the information. For instance, 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: dimensions that are not one of the types above.

The characteristics of the cube structure item are:

- Must be identifiable and nameable.
- 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” indicates whether the element must appear in the cube. The “is_flow” indicates whether a variable contains “stock” or “flow” values.

4.1.6 Combination and combination item

Cubes define, through the cube structure items, a multidimensional space where, in principle, all combinations of “allowed” values are possible. Combinations serve to restrict this space by listing the “allowed” or the “not allowed” combinations of a cube.

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.

- Must contain one or more combination items.
- Each combination item represents a combination of a variable, a member and a subdomain, or another member. It inherits the maintenance agency of the combination.

5. Mapping package

The mapping package is responsible for linking different codification systems. The links are realised at the levels of variables, cubes and members in a variable.

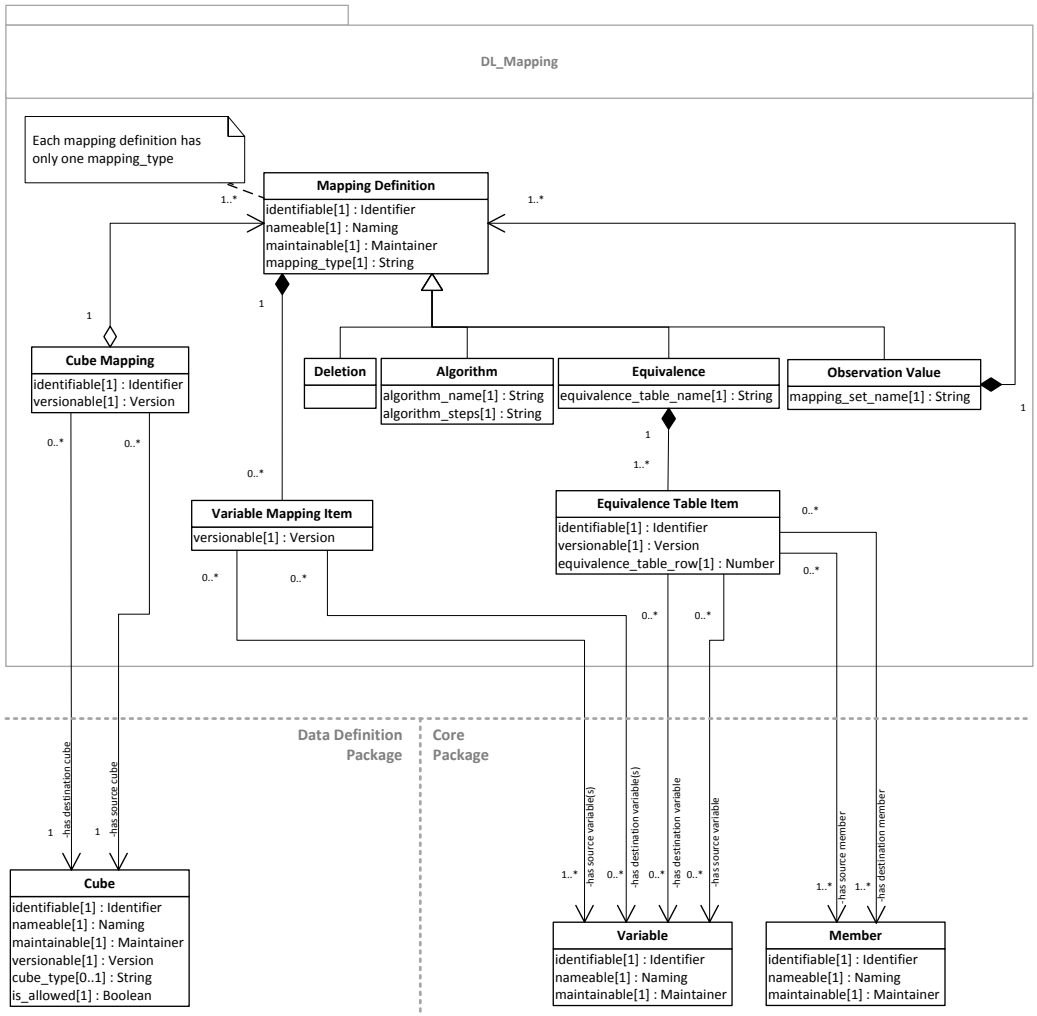


Figure 3: Structural metadata - mapping package

5.1 Definition of mapping package elements

5.1.1 Mapping definition

Each mapping definition describes the existence of a full equivalence, from variable(s) in a source to variable(s) in a destination cube (cube mapping). There are four types of mappings: D (deletion), A (algorithm), E (equivalence), and O (observation value). They are subtypes of the mapping definition.

- *Deletion mapping:* serves to eliminate one variable from the source cube in the destination cube.
- *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.
- *Observation value mappings:* are used to map measure dimensions. As explained in the data definition package, the possible values associated with measure dimensions are sets of variables instead of members. Therefore, equivalence tables are not applicable in this case, since they map members.

Characteristics of mapping definitions are:

- Must be identifiable, nameable and maintainable.
- May contain one or more variable mapping items.
- Must refer to one cube mapping (combination of source and destination cubes).

5.1.2 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.

- Must be versionable.

5.1.3 Equivalence table item

An equivalence table item describes how to apply the necessary transformation between members. It defines the source and destination variable/member pairs for the mapping.

- Must be identifiable.

5.1.4 Mapping set

A mapping set is a collection of mapping definitions to be used in the context of observation value mappings.

5.1.5 Associations

5.1.5.1. Cube mapping to cube

The entity cube mapping represents the source cube/destination cube pair for the mapping. The objective of mappings is to guide the transformation from an initial cube (source) into a new cube (destination) through one of the four instruments available in the mapping package. These are the “deletion”, “algorithm”, “equivalence”, or “observation value” mappings.

5.1.5.2. Variable mapping item to variable

Each mapping describes the data movement "as-is" or data transformation between a variable(s) (source) into one or more (destination). A mapping definition is composed of several variable mapping items, each of which is related to one of the parts of the correspondence: source or target. In other words, the variable mapping item lists the source and destination variables involved in a mapping. One mapping can map a combination of "n" source variables to a combination of "m" destination variables.

5.1.5.3. Equivalence table item to variable or member

Mappings of the type "equivalence" will make use of an equivalence table. This table will define the mappings of members. Thus, the equivalence table provides the list of member mappings, with the possibility of mapping a combination of "n" source members to a combination of "m" members. Each "equivalence table item" will define one part of a row mapping.

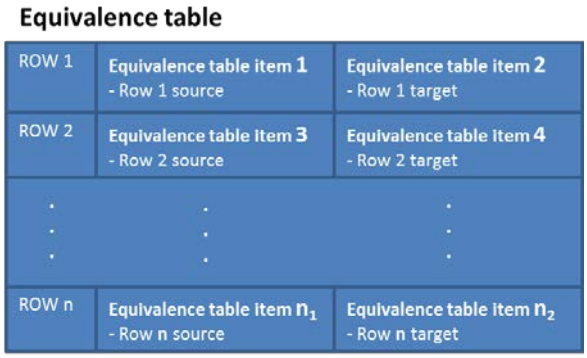


Figure 41: Equivalence table structure

Whenever one mapping of members contains, either at the source or the destination, a combination of members, then it is necessary to specify the variable to which the member refers. In other cases, the variable is not necessary, and the equivalence table will not refer to them.

For instance, a mapping between ISO 3166 alpha 2 and 3166 alpha 3 is one to one (for instance AT = AUT), thus there is no need to specify the variables and the equivalence table is reusable. If there is a member which refers to "original maturity over two years and remaining maturity over two years", it can be mapped to a combination of pairs, i.e. variable/member: "Original maturity"/"Over two years" and "Remaining maturity"/"Over two years".

6. 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.

The new elements which belong to this package appear in black in the figure below. The elements in grey are part of other packages (data definition and core package).

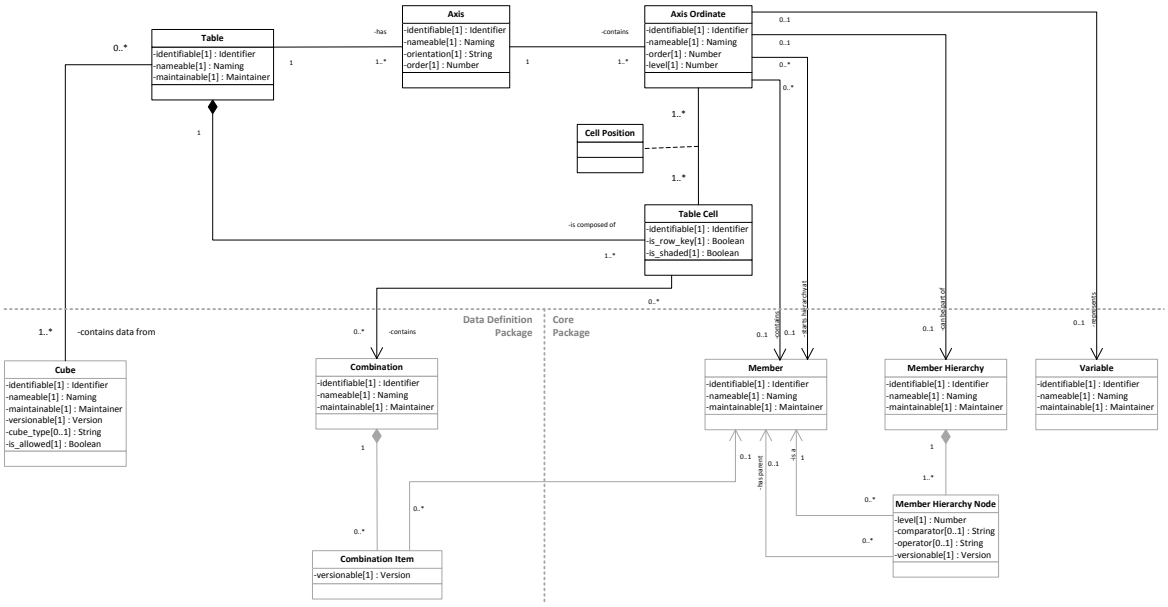


Figure 5: Rendering package

6.1 Definition of rendering package elements

6.1.1 Table

The table entity represents the content structure for the representation of data. For example, a balance sheet table report with a format.

- Must be identifiable, nameable and maintainable.
- It is composed of one or many table cells (see 5.10.4).
- A table may contain data items from one or more cube definitions.

6.1.2 Axis

The entity “axis” contains information about one dimension of a multidimensional table. Tables contain at least two axes, one x (columns) and one y (rows). A balance sheet 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 axes.
- Each axis will have an orientation (e.g. x, y, z), and an order.

6.1.3 Axis ordinate

An axis ordinate entity represents an item of a dimension of a table. Each row and each column of the balance sheet constitutes 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 should be identifiable and nameable.
- Each axis ordinate contains the attributes order (order of presentation of the element), and level (the level in the hierarchy of ordinates).

6.1.4 Table cell

A table cell is the list of combinations of x and y ordinates (i.e. combinations of rows and columns).

- It should be identifiable.

It contains attributes, the “is_row_key” (which defines, in the case of tables, those cells with an “open axis”, i.e. an axis for which the ordinates are not enumerated, which is the key enumerated ordinate) and the “is_shaded” (those cells that should not have a value).