New database version

This note aims at giving a short introduction to help understanding the changes included in the new database model as compared to the previously published version.

New tables

Core package

Member hierarchies (and member hierarchies nodes)

Member hierarchies serve to establish a hierarchical relationship between the members of a domain. These relationships are independent of datasets, and therefore they are part of the core package. Member hierarchies are described with the member hierarchy nodes. Such Member hierarchies allow – only to give a simple example – to represent the configuration of the European Union with respect to the members states in the BIRD model.

Variable sets (and variable set enumerations)

In some cases, a dataset is described so that there is only one field for observation values, while there is another field that classifies the meaning of that observation value. This is the case, for instance, for the DPM, where the members of the variable metric (or amount type) are specifying the meaning of the observation value.

As an illustrative example, a FinRep-like table described with the DPM represents a data structure like this:

Option 1 (using a Variable set)

Main Category	Counterparty sector	Metric	Value
Loan	Financial corporations	Carrying amount	100
Loan	Non-financial corporations	Fair value	120
Security	Financial corporations	Carrying amount	300
Security	Non-financial corporations	Fair value	290
…	…	…	…

This information can be alternatively represented in a different manner, like AnaCredit or the BIRD, by having different observation values:

Option 2 (without a Variable set)

Main Category	Counterparty sector	Carrying amount	Fair value
Loan	Financial corporations	100	120
Security	Non-financial corporations	300	290
…	…	…	…

Please note that the information is the same in option 1 and option 2 and the utility of a Variable set allows just for another representation of data.

The SMCube methodology is neutral as regards the preferred option; there may be good reasons for one or the other. What is relevant is that the core package is not affected by the decision on how to structure the dataset, because in both cases *Carrying amount* and *Fair value* are exactly the same concepts.

This means, in practice, that *Carrying amount* and *Fair value* are variables and not members (as they are in the DPM). So, in order to describe a dataset in the option 1 manner, the variable *Metric* needs to be associated to a set of variables, instead of being associated to a subdomain.

Note that this implies that, when importing the DPM, metrics (or amount types) are imported as variables, not as members.

Data definition package

Cube hierarchies, cube hierarchy nodes, cube groups and cube group enumerations

These new tables serve to organise the cubes in a hierarchical manner. This is used, for instance, to organise the BIRD cubes in different groups of cubes (counterparties, protection, instruments…).

Cube structure

This new 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 of the cases (including the BIRD, AnaCredit, FinRep and SHS) there is a 1 to 1 equivalence between cube and cube structure.

Cube relationships

The cube relationships table serves to describe primary/foreign key relationships between cubes and therefore allows for the definition of relations between cubes in the BIRD model.

Rendering package

The rendering package has been taken from the DPM. Note that, in the SMCube methodology, the table cells can be linked to the combinations, which are the equivalent to datapoints.

Mappings package

The mappings package is essential for the development of the BIRD, but also introduces a new degree of complexity. Mappings are needed because different reporting frameworks are developed by different institutions or divisions within the same institution (in SMCube terms, maintenance agencies), which implies that the same concepts use different codes. However, the BIRD itself can

only use one specific set of codes (i.e. the so called reference codes), so mappings from (a different) codification system to a common one is a prerequisite for having a standardised input layer.

But the mappings are complex because different agencies use different codification systems that are not compatible with each other. That's why there are mappings that are not 1 to 1, but even many to many.

Annex III contains a short practical introduction to mappings.

New features

Historisation

Historisation refers to the ability of knowing the structure of the (meta) data at a certain point in time. Note that historisation differs from an audit log, which deals with how the database changed. To illustrate the difference, suppose that one new cube needs to be reported from time t1, and the cube is created in the database at t0. The audit log will deal with the fact that at t0 a certain cube was created, while historisation serves to specify that the new cube is valid from t1.

The SMCube methodology uses two historisation methods.

Versioning

In some cases, historisation is done with versions of elements. One element can have different versions, and each version has a validity range. This is the case for cubes and cube structures.

As an illustrative example, suppose a cube structure ABC that up to the date t1 has three variables (A, B and C), but from t1 will need to have four (A,B,C,D). This implies two records in the cube structure table, one per version, and the full version of both versions in the cube structure item. The database tables (simplified for illustration purposes) would be:

Cube structure table

CUBE_STRUCTURE_ID	CODE	VERSION	VALID_FROM	VALID_TO
ABC_1	ABC	1	t0	t1
ABC_2	ABC	2	t1	31/12/9999

Note that the code is the same, but there are two different ids.

Cube structure item table

CUBE_STRUCTURE_ID	CUBE_VARIABLE_CODE
ABC_1	A
ABC_1	B
ABC_1	C
ABC_2	A
ABC_2	B
ABC_2	C
ABC_2	D

This approach is used for cubes, cube structures and combinations.

Enumeration validity

In some other cases, the validity is provided with the enumeration of an item. In these cases, the elements that belong to the item evolve over time without creating different versions. This is the case for the hierarchies. Suppose a hierarchy with the composition of the Euro Area. Some members may join the Euro Area over time, but the Euro Area concept is always the same, so there are no different versions. A member hierarchy with a changing composition is illustrated below:

Member hierarchy table

MEMBER_HIERARCHY_ID	NAME
EA	Euro Area

Member hierarchy node table

MEMBER_HIERARCHY_ID	MEMBER_ID	LEVEL	PARENT	VALID_FROM	VALID_TO
EA	EA	0		t0	31/12/9999
EA	A	1	EA	t0	31/12/9999
EA	B	1	EA	t0	t1
EA	C	1	EA	t2	31/12/9999
EA	D	1	EA	t3	31/12/9999

Additional considerations

There are practical reasons for providing both ways of historisation. Creating versions is very useful to stress that there are changes. But, as seen in the example, creating new versions imply more changes and redundancy.

Maintenance agencies

All the elements described with the SMCube methodology belong to one maintenance agency. The reason for this is that the elements are owned by different agencies, and it has to be clear which the owner for each element is.

Annex I: FinRep translation

The content of EBA's DPM is imported into the SMCube methodology automatically from the XBRL taxonomies. All the DPM content remains untouched, only a translation between methodologies is performed. All the items belong to the maintenance agency "EBA", reflecting the fact that the content is left exactly as published by the EBA through the taxonomies.

Core package translation

The core package of the SMCube methodology is very similar to the dictionary layer of the DPM, so in most cases the translation is very straightforward.

The most relevant changes that should be noted are:

- Metrics are members of the domain *Amount type* in the DPM. As explained above, in SMCube metrics are treated as variables. Nevertheless, metrics are also imported as members, because the DPM contains member hierarchies with in which the members are metrics, so they get imported in the SDD as members in order not to lose the hierarchies.
- New domains are added with the metric data types. The reason is the conversion of metrics into variables. Variables in SMCube are defined on a domain, and metrics have a data type, so the data types of the metrics are created as domains. These new domains are:
    - Monetary
    - String
    - Percent
    - Boolean
    - Date
    - Integer
    - Decimal
- New variables are added to make explicit some dimensions that the DPM retains implicit, but that are included in the XBRL instances. These variables deal with the reporting agent, the date and the observation value, and are:
    - Period
    - Entity
    - Unit
    - Observation_Value

Data definition package translation

The translation of the data definition package is not straightforward, since the DPM does not represent multidimensional structures. Therefore, a convention to what is equivalent to a cube is required. The convention followed has been translating one table as one cube.

In the cube structure items, the implicit variables are added to the cube. All the DPM dimensions have the dimension role also under the SMCube methodology.

Data points are translated as combinations.

Annex II: Introduction to mappings

Mappings provide a way to establish that two concepts created by different maintenance agencies are equivalent. The concepts we are interested in are the variables and the members, which are the building blocks for datasets.

In an ideal world mappings would be very simple: One table with two columns could suffice to express mappings.

Table 1: Ideal world mappings

SOURCE MEMBER	DESTINATION MEMBER
a	1
b	2
c	3

SOURCE VARIABLE	DESTINATION VARIABLE
x	7
y	8
x	9

But, unfortunately, the reality is much more complex, and this implies the need for more complex mappings. There are two main sources of complexity: (i) Use of different classification systems, and (ii) errors.

As an illustrative example, let's take the European System of Accounts (ESA) classification of financial instruments. This classification is done with a specific purpose, and mixes different concepts within the same classification. For instance, in the ESA classification of instruments there are two values for *log-term debt securities* and *short-term debt securities*. So the data frameworks that follow ESA classification tend to have one variable where two possible values are *long-term debt securities* and *short-term debt securities*. But in other frameworks, like FinRep, this classification is not followed, and therefore there are two separate variables: The *type of instrument* and the *original maturity*.

Table 2: Need for complex mappings

ESA_ INSTR_CLASS		TYP_INSTRMNT	ORGNL_MTRTY
F32 - Long-term debt security	➔	1 - Debt security	1 - Long-term
F31 - Short-term debt security	➔	1 - Debt security	2 - Short-term

The SMCube methodology provides a model able to address complex (n to m) mappings. In the SMCube, one full mapping points to one mapping of variables and, eventually, one mapping of members.

One full mapping points only to a variable when the variable is not enumerated. For example, if we want to map the variable *Carrying amount*, with code mi53 in the DPM, to the same concept with code CRRYNG_AMNT in the reference dictionary.

If the mapping is for enumerated variables, then it needs also to point to the member mappings. The MAPPING_DEFINITION table contains the full mappings. It includes one field with the mapping type. The most relevant types of mappings have the value 'E' and 'A'. 'E' mappings imply that a member mapping is required, while 'A' mappings imply that an algorithm is required. The algorithm in the latter case serves to add operations, if needed to the values in the non-enumerated variables. In most cases it will not have any value, meaning that no operation has to be done.

VARIABLE_MAPPING and VARIABLE_MAPPING_ITEM tables provide the variable mappings, while MEMBER_MAPPING and MEMBER_MAPPING_ITEM provide the member mappings.

The two previous examples would be described (for illustrative purposes, the tables are simplified and only the enumeration tables are shown):

MAPPING_DEFINITION

MAPPING_ID	MAPPING_TYPE	ALGORITHM	VARIABLE_MAPPING_ID	MEMBER_MAPPING_ID
ie1	E		vm1	mm1
ie2	A		vm2	

VARIABLE_MAPPING_ITEM

VARIABLE_MAPPING_ID	VARIABLE_ID	IS_SOURCE
vm1	ESA_ INSTR_CLASS	TRUE
vm1	TYP_INSTRMNT	FALSE
vm1	ORGNL_MTRTY	FALSE
vm2	mi53	TRUE
vm2	CRRYNG_AMNT	FALSE

MEMBER_MAPPING_ITEM

MEMBER_MAPPING_ID	MEMBER_MAPPING_ROW	VARIABLE_ID	IS_SOURCE	MEMBER_ID
mm1	1		TRUE	F32
mm1	1	TYP_INSTRMNT	FALSE	1
mm1	1	ORGNL_MTRTY	FALSE	1
mm1	2		TRUE	F31
mm1	2	TYP_INSTRMNT	FALSE	1
mm1	2	ORGNL_MTRTY	FALSE	2

Note that:

- Given that mappings are n to m, the number of source and destination elements is unknown. This is the reason for having one record per element, and not per mapping.
- Each mapping maps 1 set of variables, but several sets of members. That is why the MEMBER_MAPPING_ITEM table has the field MEMBER_MAPPING_ROW.
- The first mapping is 1 to 2, so when coming to the mappings, there is no need to specify the variable for the source member (it can only be one: ESA_ INSTR_CLASS), but the specification of the destination variable is required, because if, for instance, only '1' was specified, there would be doubts on whether that 1 would apply to TYP_INSTRMNT or ORGNL_MTRTY.

SHS reference cubes

The database contains two sets of SHS cubes: The original cubes and the translation to the reference codes. The original cubes describe the information to be transmitted, as described in the original documentation. The translated cubes are created automatically from the original cubes, by applying the mappings. This way, the SHS reference cubes can be compared to the BIRD, because the codification used is the same.