

## **Improving transparency in financial and business reporting — Harmonisation topics — Part 5: Mapping between DPM and MDM**

*Einführendes Element — Haupt-Element — Teil 5: Ergänzendes Element*

*Élément introductif — Élément central — Partie 5 : Élément complémentaire*

ICS:

Descriptors:

Document type: CWA  
Document subtype:  
Document stage: Formal Vote  
Document language: E

## Contents

	Page
1 Introduction .....	4
1.1 General.....	4
1.2 Objective.....	4
1.3 Target Audience.....	4
1.4 Relationship to Other Work .....	4
2 Scope .....	5
3 Terms and definitions .....	5
4 Introduction to the Multidimensional Data Model .....	5
5 Preconditions on mapping .....	5
5.1 Types of Database Management Systems (DBMSs).....	5
5.2 Fundamental choices .....	6
5.3 Fact definitions: presentation vs DPM .....	9
5.4 Storing native XBRL facts.....	9
5.5 Dimension/defaultMember .....	11
5.6 Options .....	11
5.7 Versioning .....	11
5.8 Changes on fact values.....	11
6 Definitions .....	12
7 Mapping from Data Point Model to Multidimensional Data Model .....	13
7.1 Introduction .....	13
7.2 Framework.....	14
7.3 Taxonomy .....	16
7.4 Dimensions.....	18
7.5 Context.....	23
7.6 Primary Items .....	24
7.7 Fact table or Data points .....	26
7.8 Summary.....	27
8 Metamodel defined by the EBA (FINREP and COREP) mapped to MDM .....	30
8.1 Introduction .....	30
8.2 Creation of the structure and load of the DPM from the EBA in a RDBMS .....	30
8.3 Loading DPM_ROLAP from DPM_EBA .....	31
9 Implementation of the DPM in the MDM using the design ROLAP .....	37
9.1 Introduction .....	37
9.2 Structure ROLAP .....	38
9.3 Creation of the infrastructure through MS SQL Server .....	39
10 DPM of FINREP 2012 in the MDM using the design ROLAP .....	43
10.1 Introduction .....	43
10.2 DPM of FINREP 2012 .....	43
11 DPM of the first prototype of Solvency II in the MDM using the design ROLAP .....	52
11.1 Introduction .....	52
11.2 DPM of the prototype .....	52

## Foreword

This document has been prepared by CEN/WS XBRL, the secretariat of which is held by NEN.

CWA XBRL 001 consists of the following parts, under the general title *Improving transparency in financial and business reporting — Harmonisation topics*:

- *Part 1: European data point methodology for supervisory reporting.*
- *Part 2: Guidelines for data point modelling*
- *Part 3: European XBRL Taxonomy Architecture*
- *Part 4: European Filing Rules*
- *Part 5: Mapping between DPM and MDM*

This CWA is one of a series of related deliverables. The other deliverables are:

CWA XBRL 002 *Improving transparency in financial and business reporting — Metadata container*

CWA XBRL 003-1 *Improving transparency in financial and business reporting — Standard regulatory roll-out package for better adoption — Part 1: XBRL Supervisory Roll-out Guide*

CWA XBRL 002-2 *Improving transparency in financial and business reporting — Standard regulatory roll-out package for better adoption — Part 2: XBRL Handbook for Declarers*

## 0 Introduction

### 0.1 General

This document aims to provide an introduction to the topic of creating a conceptual model for storing multidimensional data which is received as XBRL instances that follow the rules defined by European taxonomies published by the European Banking Authority (EBA) or by the European Insurance and Occupational Pensions Authority (EIOPA).

**Disclaimer: The Multidimensional Data Model (MDM) presented in this document is intended to be a starting point for a subsequent modelling process to be adjusted and extended to specific analytical or transactional needs. It solely refers to the concepts of Data Point Model (DPM) and European XBRL Taxonomy Architecture (EXTA), which build the basis of European supervisory reporting.**

The structure of the data model is based on meta classes, introduced in part 1 and 4 of the CWA1 document [26]. The data model represents a relational model using Relational Online Analytical Processing (ROLAP). In this document UML data structures of a DPM are used because its comprehension will be easier. With the UML class model representing the description of the European filing rules, the present document visualises the mapping between UML meta classes and their correspondence in the form of database tables in the MDM.

This document consists of eight sections, save the bibliography. Section one explains working with a Multidimensional Data Model as a step towards working with the Relational Data Model. Section two makes a study of the architecture of XBRL, the databases and their aims, requirements and preconditions in catering for XBRL. Section three defines the conditions used for mapping from DPM to MDM. Section four is detailing point by point the mapping. Section five shows the metamodel defined by the European Banking Authority (EBA) through the FINREP (Financial Report) and COREP (Common Solvency Report) taxonomies and its mapping into MDM. Section six displays the MDM implemented in a relational database. Sections seven and eight show two implementation examples.

### 0.2 Objective

The objective of this sample MDM is to provide a starting point into the topic of mapping DPM and XBRL instance structures into a multidimensional database. Based on an easily comprehensible example, more complex issues are addressed that would need to be taken into account by defining an MDM for production use.

### 0.3 Target Audience

This document is aimed at users of European supervisory taxonomies that have the need to store reporting data based on these data definitions and to retrieve them for analytical or transactional purposes. Database experts should get detailed information about the specifics to be taken into account when modelling multidimensional database structures for storing supervisory data based on XBRL. Therefore, the audience of this document might be financial or economic institutions, agencies or universities with the intention to provide micro or macro prudential analysis on supervisory data.

### 0.4 Relationship to Other Work

The reader of this document is expected to be familiar with the principles of data modelling, having a thorough understanding of the concept of DPM as well as basic knowledge of XBRL. The reader is also expected to have knowledge in creating conceptual models for relational and multidimensional databases.

## 1 Scope

This document aims to provide an introduction to the topic of creating a conceptual model for storing multidimensional data which is received as XBRL instances that follow the rules defined by European taxonomies published by the European Banking Authority (EBA) or by the European Insurance and Occupational Pensions Authority (EIOPA).

## 2 Terms and definitions

The terms and definitions used in the mapping with Data Point Model are inspired by vocabulary already known from their use for describing multidimensional databases and Data Warehouses.

## 3 Introduction to the Multidimensional Data Model

The multidimensional database is primarily used to create OLAP (Online Analytical Process) applications and their databases using a fact table and set of dimensions. A multidimensional structure stores multidimensional data, that is to say, cubes. A cell or fact is an intersection consisting of elements that form the dimension(s) which in turn form a cube. A cell can have zero or more measures, but in this document only one measure is taken into account.

The Multidimensional Data Model (MDM) is used instead of the Relational Model, because the European architecture of economic-financial reports is relying on dimensions heavily, which makes implementation in MDM the logical choice. Moreover, the performance of queries is better in this type of database.

The goal of this document is to store the Data Point Model in a database, in an efficient, easy way.

## 4 Preconditions on mapping

### 4.1 Types of Database Management Systems (DBMSs)

In this section some types of DBMS's are analysed that appear suitable for storing DPM and XBRL documents. Only those databases are considered where, in a previous study, it seemed possible to store the DPM and to extend XML or XBRL documents.

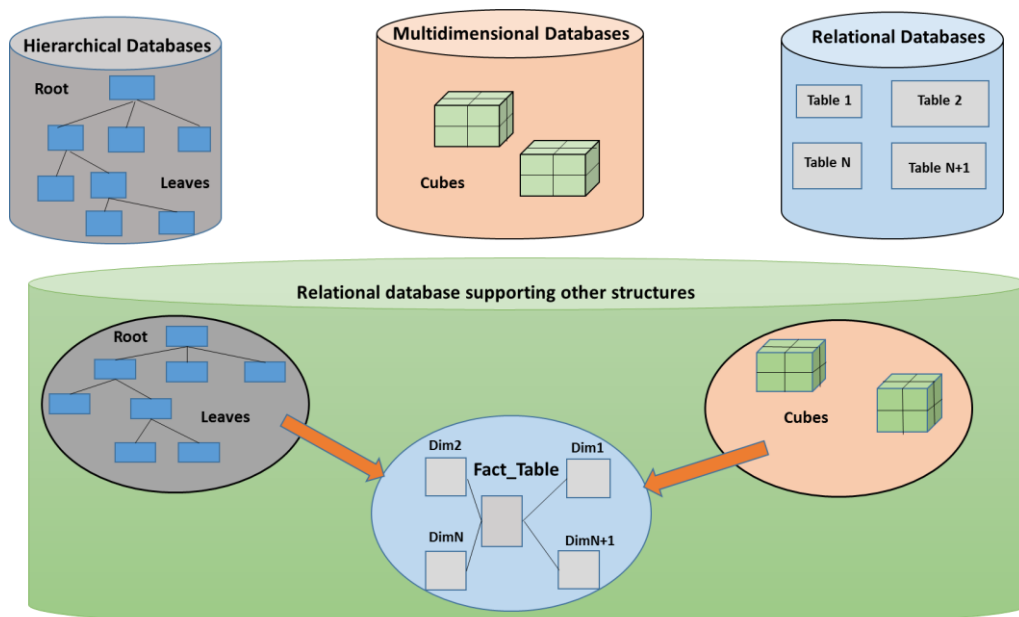


Figure 1 — Different types of DBMS's

The typical solutions are (Figure 1):

- Hierarchical databases;
- Multidimensional databases;
- Relational databases;
- Mixtures, where, normally, the relational database is the base.

Hierarchical databases (e.g. Tamino by Software AG, GT.M, IBM Information Management System (IMS)), which rely on the hierarchical model, that is to say, databases organized into a tree-like structure. In this structure, data uses relationships among their leaves. Each leaf on a superior level has 0..\* relationship with leaves on the inferior level. A leaf on an inferior level only has a 0..1 relationship with a leaf on the superior level.

Multidimensional databases, not being based on relational databases, have the data is stored in an optimized multi-dimensional storage array, and not in a relational format. However, it is necessary to organize the information in a cube beforehand. These databases have very fast response times in queries. Examples of Multidimensional databases are: Essbase, icCube, Infor BI OLAP Server.

In relational databases the information is stored in relational format. But, moreover, in these databases it is possible to store cubes, but in a relational format, changing their internal structures.

In all these solutions, it is necessary to verify that database transactions are processed reliably. For this, a database must fulfil ACID (**A**tomicity, **C**onsistency, **I**solation, and **D**urability) properties. Not all databases fulfil the ACID requirements, this depends on the vendor. These properties are:

- Atomicity: each transaction is "all or nothing";
- Consistency: it ensures that any transaction will bring the database from one valid state to another valid state;
- Isolation: it ensures that the concurrent transactions result in a system state that would be obtained if transactions were executed serially;
- Durability: once a transaction is committed, it will remain so even in the event of power loss, crashes, or errors;

This document will not analyse whether databases carry out the ACID properties. However, the majority of commercial Relational Database Management Systems (RDBMS) achieve these properties. These databases are very common in the Information Systems Departments of this environment. Examples of these RDBMS's, are Oracle, DB2 or MS SQL Server, amongst others.

### 4.2 Fundamental choices

This section will discuss, if the XBRL document instance is stored directly in the database in part or in a relational model.

There are two mainstream solutions for storing XBRL instances and their facts into a relational database system. The question is, when Information Systems (IS) receive a XBRL taxonomy or an instance document, how these XML documents can be stored with the lowest cost in resources in the database. As relational databases can only store relational data and XML documents are not relational, the mapping is not a direct process.

The topic to analyse is:

- Mapping the XBRL instance document to the relational model;
- Storing the XBRL instance document as a blob, or PDF document in the database;

- Storing the XBRL instance as a XML document or as a XBRL document.

Not all XML documents can be mapped into the relational model. However, XBRL instance documents can be mapped to the relational database, as they show many references. The XBRL specification contains a very important aspect: validation by formulae. Formulae are based on XPath 2.0 (XML Path Language), which is based on XML. When the XBRL instance document is transformed into the relational model, the instance document cannot be validated by formulae anymore. Moreover, as these validations are based on the XBRL Formulae and Calculation specifications, the mapping to a RDBMS is not easy nor immediate [19]. As XBRL validation requires the use of XML enabled tools, this cannot be done in the RDBMS. There are many validators, both commercial and open source (Openfiling) in XML. On the other hand, the mapping of instance documents into a relational database is available through different commercial or open source vendors (Openfiling).

An XBRL instance document can be stored in a relational database as an XML document or in a relational format. Analyzing the queries in both solutions resulted in:

- In XML, these queries use XQuery and XPath.
  - The end user has difficulties accessing the language of the queries directly or through tools;
  - The query language is very specific. Experts in this language are necessities;
  - The tuning of XML documents is complex.
- Relational Database use standard SQL.
  - The end user can obtain the data in an easy way through spread sheets, linked tables or other tools;
  - The query language is a standard, and is part of university IT curricula;
  - The performance and tuning of a relational database has been extensively analyzed.
- If the XBRL instance document is stored directly in the database (as a blob), the problems are the same but the RDBMS is an inferior level. Cases are:
  - Storing as a photo (Blog or Clog);
  - Storing as a XML document.

In the first case the database is only used as a storehouse. In the second case, storing as an XML document, with functions embedded in the engine of the database. This means that the database manager has embedded these functions in the engine. Today there are vendors that add the type XML as Oracle, MS SQL Server or DB2. Depending on the vendors the main features are:

- Generating XML Instances;
- Methods or procedures on the XML data type;
- Queries on XML instances;
- Processing namespaces;
- Indexes;
- Navigation through the document;
- ...

XBRL is an extension of XML, but it is not XML, the cost of implementation therefore has to be evaluated, and the performance of the database must be re-tuned for optimization.

MS SQL Server has also utilities for working with XBRL that is necessary to analyse, in the same way.

Oracle 11g release 2 (as from version 11.2.0.3.0) works with XBRL instance documents [Oracle 11g with XBRL](#):

- Manages XBRL content;
- Can create multiple XBRL repositories and project XBRL data relationally or query it in various ways;
- Operations of aggregated business and financial reports such as extraction, transformation, and loading (ETL); business intelligence (BI); and online analytical processing (OLAP);
- The validation is outside to the database [Out oracle](#).

Both the Microsoft and Oracle solutions have to be evaluated in terms of costs, resources, tuning and performance in the engine of the database.

In summary; this document is not considering storing the XML document (instance) as a whole, as it is storing the instance in a native XML database. Only storing the content of the XML document in a RDBMS is discussed. One can either:

- Store almost native facts and their aspects, or
- Convert the facts and the required aspects into a proprietary set of data before storage.

For both scenarios all relevant aspects on the facts will need to be determined from the analyst point of view.

Another consideration for the importance of aspects is to decide if the database will also be the source to generate (the same or new) XBRL instances (more information on Openfiling). More XBRL-specific requirements need to be considered to create a valid instance. When the target is to (re)create instances, special consideration has to be given to any merge processes on fact values. Merged fact values will cause problems for instance creation unless there is a possibility of an 'undo' (split) routine or a structure more complex in the relational model. This can be created as easily as storing both the original fact values and the merged value. However, different instances can coexist because, as it is explained below, each fact is defined in a time period and it belongs to an entity.

Table 1 below shows a summary of the possible advantages and disadvantages of both methods.

**Table 1 — Pros and cons of alternatives**

Proposals	Native store	Convert before store
Quantity of aspects to store (direct from instance)	(+)(-)	(+)(-)
Quantity of aspects to store (indirect from <i>Discoverable Taxonomy Set (DTS)</i> )	(-)	(+)
Speed of storage process	(+)	(-)
Maintenance (mapping table, mapping software)	(+)	(-)
Analyst queries, degree of difficulty	(-)	(+)
Analyst queries, speed	(-)	(+)
Easy handling of new DTS versions	(+)	(-)
Extensibility towards proprietary XBRL reports	(+)	(-)
Extensibility towards proprietary non-XBRL reports	(-)	(+)



### 4.3 Fact definitions: presentation vs DPM

XBRL Taxonomies created with DPM contain two definitions of individual reportable facts:

- Primaries, dimensions and members have readable labels and optional references to external documentation;
- Tables, axes headers and table footers have generic (text) labels and indicators pointing towards an 'RC' (row-column) value that identifies a cell in the templates that form the basis of the DPM

Since there is no guarantee that both definitions will match, a reported fact can rely on either definition. It depends on whether the reporter used a form, based on the table linkbase, or a mapping based on the primaries/dimensions/members combinations. From a theoretical point of view the templates are transformed to DPM and then the DPM into XBRL concepts, i.e., the concepts are leading. This has not been stated explicitly by EBA. In order to stay independent from EBA modelling it is best to store both definitions as relevant aspects. The definition texts as such are the only means for a business analyst to create a query and understand its outcome. Definitions that rely on documentation outside the DTS and is referred to by XLink references, is only available for concepts, not on the presentation of the table. Linking this information into the database (and query) is outside the scope of this document. In theory such external reference pointers could be created on the presentation, EBA has however not used this feature; it would be used in accordance with XBRL specifications.

When using the instance transformation option, the definitions have to be manually mapped to the internal definitions. This only needs to be done once. The maintenance task is to check every new release of the DTS for changes in definitions regardless where they are being used. Every change needs to be re-evaluated and again manually mapped into the internal definitions. Analyst queries work with internal definitions, their meaning should be clear to the users. Another point of consideration is that there is no guarantee that what is dimensionally valid in the DTS will be presented as a cell in any table. The other way around, what is in a table is always dimensionally valid, is guaranteed. There needs to be a process to detect such anomalies, either upon loading a new version of the DTS or upon storage of the facts. There may even be a need for a disclaimer that facts reported without a proper 'cell' in a table are being disregarded. In this sense the table linkbase is forming a third validation mechanism of reportable facts (XSD and XDT being the others).

Lastly the introduction by EBA of a new mechanism called 'filing indicators', needs to be thought through. If instance creation from the database is in order, these XML nodes need to be stored too. They are used to ease the validation process of the XBRL formulae. The mechanism indicates from which tables the instance contains facts. Some facts could be placed in multiple tables (e.g. a total in the total table and in its specification table) and different formulae may need to be executed depending on its usage. There is no mechanism in place that links the *filingIndicator* value to anything in the DTS. Therefore, one could report table 999 that doesn't exist as long as there are no facts reported against it. This makes for little use in back office applications; it only needs to be stored when instance creation is part of the requirements. The table number used stems directly from the templates and the number is accompanied by explanatory texts in the label that is placed on a presentable table. It is not part of any structured part of the taxonomy.

### 4.4 Storing native XBRL facts

Regulators will receive a container file (ZIP) with at least one XBRL instance in it. Depending on internal processes this container needs to be unzipped first and its content evaluated. Validation of the instance is not part of this document, a valid instance is assumed. Instances can represent multiple taxonomies; an assurance statement could be made part of the instance containing the reportable figures. Solutions to prevent or accommodate this are not part of this document. An instance based on a single taxonomy is assumed, referring to a taxonomy that is enabling reportable figures only. An instance can contain Xlink content. This is not discussed in this document. The instance is expected to contain only facts, units, contexts, one schemaRef and filingIndicators. Table 2 shows different aspects of storing native XBRL facts.

Table 2 — Different aspects of storing native XBRL facts

Technical part	Aspects	Comment
Instance file name.	Optional hash code.	For NSA's (National Supervisory Authorities) working with assurance solutions.
Root node <i>xbrli:xbrl</i> .	Character set, and optional language, version and id.	
At least one <i>link:schemaRef</i> .	Contains an URI (Uniform Resource Identifier) and a location.	This is considered to be the entrypoint of the DTS for which this instance is being reported. XBRL allows multiple <i>schemaRef</i> nodes, EBA only one. EBA has determined that the URI represents an absolute location (web address) and the location only the name of the schema file.
Optional multiple <i>link:linkbaseRef</i> .		EBA will not be using these.
At least one <i>find:fIndicators</i> .	This contains multiple <i>find:fIndicator</i> .	The value is string based and represents a table.
Optional multiple contexts using <i>xbrli:context</i> .	Each context must have one ID attribute, one <i>xbrli:entity</i> node and one <i>xbrli:period</i> node. It may contain many <i>xbrli:segment</i> and <i>xbrli:scenario</i> nodes.	
<i>xbrli:entity</i> .	Contains an identifier value and its Scheme URI value.	These represent the reporting entity with its unique identifier within the NSA and the owner of the identifiers (NSA).
<i>xbrli:period</i> .	Contains either an instance date or a <i>periodStart</i> and <i>periodEnd</i> date.	XBRL allows also <i>forever</i> but EBA has prohibited this use.
<i>xbrli:segment</i> and <i>xbrli:scenario</i> container.	Contain dimensional aspects and/or proprietary XML schema based content.	EBA allows only <i>xbrli:scenario</i> to be used and no proprietary content. The dimensional aspects consist of a set of dimension and member <i>QNames</i> and/or a dimension <i>QName</i> with a typed member <i>QName</i> AND its value.
Optional multiple <i>xbrli:unit</i> .	Each unit must have one ID attribute. It can hold either one measure or a set of numerator/denominator.	These are all <i>QNames</i> . Each <i>QName</i> must have a value that goes with it.
Optional multiple facts.	A fact is represented with a <i>QName</i> (a <i>primary</i> concept in the DTS). It holds a <i>contextRef</i> and <i>unitRef</i> attribute (the latter only on numeric typed concepts). It may hold a decimals, language, nilable and ID attribute.	

For the definition of the fact aspects the following may be of interest: Each concept (primary, dimension, member) will have at least one label, the standard label. There may be more types of labels to a concept. A

label is defined by its role (the 'type') and the language it is in. Multiple labels of the same language and role may occur. EBA will provide only the English language and only one occurrence of each role. The label texts may contain special characters. Within a table in the DTS, any cell defined by a set of primary, dimension/member combinations may have multiple labels attached to it. These labels are also represented with a role and language. EBA will again utilize only one occurrence of text in each role per language, the language being English.

#### 4.5 Dimension/defaultMember

Special attention needs to go to default dimension members. All EBA defined dimensions will have a default member. Often the definition of this member reads 'Total/Not applicable'. The XBRL specification describes that any default member that is discovered when starting to discover the DTS from the fact is eligible for the default member. This applies even if that dimension is not used on the fact and even when the fact is not dimensional at all. In theory this means that all defaults apply to all facts since a single entry point will cover the whole of the EBA DTS. With some common sense a limitation can be applied that default members apply only on the facts reported in a certain table, when that table is using the parenting dimension. Logic could even go further stating that individual cells can be evaluated if the default member makes any sense at all. If not, the 'definition' of 'Not applicable' could be read in which case the dimension and member are not appropriate on the fact at all. In all other cases the default member applies to the fact and needs to be stored by an alternative (to storing only data from the instance) process.

Naturally, these default dimension/member combinations must be identified in storage since they are not allowed in the instance.

The XML schema also allows nodes to be identified carrying a default value. In particular, when typed dimensions are being used there could be a typed element that carries a default. The EBA DTS does not use this option.

In the MDM the default member is another normal attribute of dimension. However, it is marked as attribute by default, because it is only relevant for the mapping process and has no a special meaning in the MDM.

#### 4.6 Options

XBRL allows for more presentation texts to be added besides primary, dimension, member, table or axis. These texts could be part of the definition of a fact. Careful evaluation of the taxonomy in an XBRL enabled tool using both XDT and TLB specifications can reveal these texts. If they are part of the definition they need to be stored or used for creating the mapping to local data elements. As, for example, the Linkrole labels, and so on.

#### 4.7 Versioning

When a new version of the DTS is being released, the EBA has chosen to include two special attributes on every concept: *creationDate* and *modificationDate*. Up to the public release DTS of September 18<sup>th</sup>, 2013, there were no *modificationDates* present and the *creationDate* was increased on each new version. In theory these dates could be the trigger to signal any change in definition of the concept but if the mechanism is not used other ways to detect changes must be found. Another matter is that there is no such set of dates on the labels that form the table, which can be equally regarded as representing (a part of) the definition. For this part of the DTS a detailed 'diff' function needs to be designed. It is clear that every definition change breaks the trend on any reported fact. Manual intervention on mapping to local sources must be undertaken.

#### 4.8 Changes on fact values

If the NSA has the authority to change reported fact values, they must be aware that recreating the original instance may be cumbersome, unless appropriate versioning mechanisms have been put in place to conserve the original fact values. Special care has to be taken with business rules that have been defined by the DTS author on such a fact. The change in value may trigger a business rule. These rules can however only be executed on an instance, not the RDBMS.

## 5 Definitions

For the purposes of this document, the following terms and definitions applied are shown. The terms and definitions used in the mapping with Data Point Model are inspired by vocabulary already known from their use for describing multidimensional databases and Data Warehouses [1] [2] [3] [4] [5]. In turn, the DPM is based in the XBRL Meta-metadata Model [27]. IT specialists originally introduced these terms. However, for an understanding and creation of Data Point Models they are established in the language of business specialists as well.

In this section, the set of definitions necessary for mapping the DPM in ROLAP are shown. The majority of the definitions are obtained from [6] [7] [8] [9] [10] [26]. When the definition is in the area of CEN WS XBRL ([CEN WS XBRL Main Page](#)) [11] [22] [26] only the name of the term is shown.

The terms used directly or indirectly in the mapping of DPM in the MDM are:

- concept;
- data point model;
- dimension;
- domain;
- family;
- framework;
- item;
- (domain) member;
- metric;
- namespace;
- owner;
- perspective;
- public elements;
- tablegroup;
- datapoint;
- datacube;
- module;
- hypercube.

A hypercube is an abstract item declaration in the *xbrldt:hypercubeItem substitution group*. A hypercube is an ordered list of dimensions, defined by the set of zero or more dimension declarations linked to the hypercube by hypercube-dimension relationships in a dimensional relationship set, and ordered according to the order of this relationship [10].

In the DPM a hypercube is reflected in the DataCube. A DataCube is a set of DataPoints with its appropriate Dimensions and Members.

A hypercube in the MDM is a set of pairs <dimension, attributes of dimension> and calculated attributes defining one or more facts [19].

- taxonomy;
- context.

The context element contains information about the entity being described, the reporting period and the reporting scenario, all of which are necessary for understanding a business fact captured as an XBRL item [6].

In the MDM, the context is defined as a set of dimension of a fact or group of facts. A context belongs to an entity or financial institution, for a period, a meaning for the business (segment), and a scenario. The scenario shows the specific pairs of dimension and the dimension attribute of business logic [9].

## 6 Mapping from Data Point Model to Multidimensional Data Model

Economic-financial information in the global economy in which we find ourselves is increasingly important. This information has semantic content and must be easy to process, quickly transmittable and reliable. Since the late 90's some specifications for the transmission of economic information have emerged. XBRL represents business information, which is multidimensional. Specific to the European model is that the logical location for its storage is a Data Warehouse (DW) [25].

The Multidimensional Data Model (MDM) is a Conceptual Model and the Relational Model as well as the Data Point Model (DPM) is a Logical Model. The difference is that the Conceptual Model is nearest to the Universe of the Discourse (UD), nearest to the requirement of business user. The Logical Model is nearest to the Physical Model, the implementation in XBRL or in a Database.

This document aims to help to design the Economic-financial information of reports [17]. For this reason, this set of pages is designed to help users of Information Systems create taxonomies using the DPM and in parallel, to map to the Relational Model using the MDM through the Relational Online Analytical Processing (ROLAP) tool.

### 6.1 Introduction

This section presents the mapping between the DPM and the Relational model through ROLAP. In this mapping the transformation from XBRL taxonomies is not handled, however its conversion is possible. [20]. Moreover, in this transformation no process of validation is established, only the DPM structure is mapped into the relational model. However, it is expected that the reader of this document can understand the DPM better when storing the DPM in a RDBMS (Relational Database Management System) using the MDM.

The aim of this document is to obtain a star model representing the DPM. That is to say, the DPM is mapped to the MDM in databases. Figure 2 shows the MDM of the DPM.

The constructor *FactTable* of the figure is equivalent to the set of data points in the DPM. It is a Star model because *BaseDomain* (set of primary items), *Taxonomy* and *Context* are linked to fact tables in three dimensions. The dimension *Taxonomy* is linked with the dimension *Framework*. The *Context* is linked to the dimension *Context\_Dimension\_DimensionAttributes*. The last one is *Dimension* the set of dimension/attributes of a dimension. And, to the set dimension/attributes of dimensions the dimensions end *DimensionAttributes*. It is also possible to add the dimension *Family* to dimensions but these are not drawn, not overcomplicating the drawing.

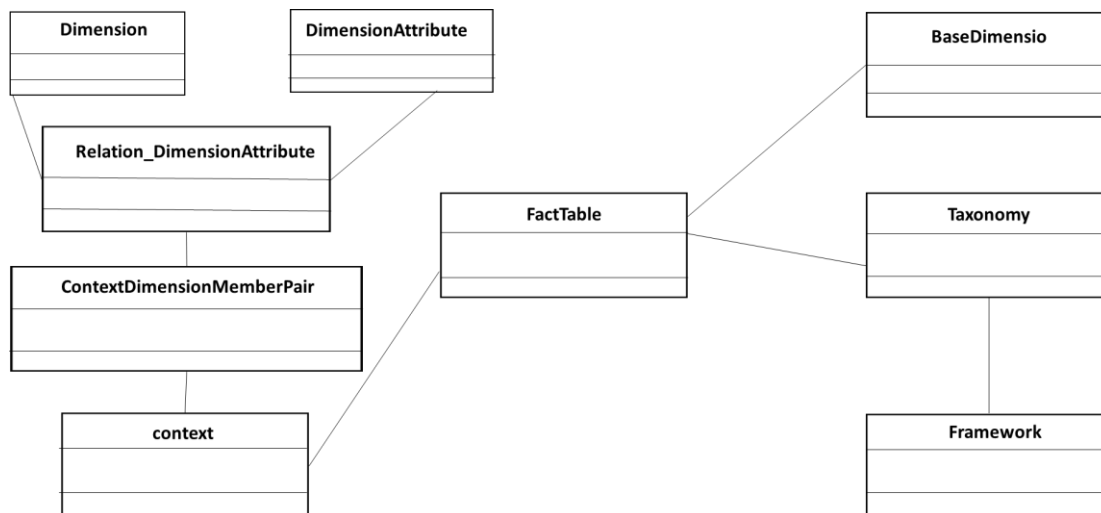


Figure 2 — Star model of the DPM using ROLAP tool.

In section six its implementation in a RDBMS is shown, accompanied by a diagram showing this implementation.

There are various references in the biography that deal with mapping from different sources to a relational database, especially from XML [12] [13] [15] [21] and about query in heterogeneous sources. In particular the paper by Levi et al. [14] is interesting. Nevertheless, the process of transformation of this section is based on Taentzer et al. [15]. This section deals with the different constructors that are corresponding in the DPM, step by step.

In this section, the process of conversion is analysed. Normally, a first step is to study the DPM element or elements to transform. Following this approach, the mapping between the DPM elements and the relational elements are gathered. The transformation process in the figures show the DPM UML graph on the left are the UML class diagrams and to the right the relational model (ROLAP) from the MDM. The black arrows between both are UML syntax but they are customized extensions, which are used to describe the graph transformation. The square between two black arrows contains an abbreviation which is mapped. In this document the following types of mapping rules between the two graphs are distinguished [15]:

- A2C is the automatic transformation between attributes of the DPM and columns of a table;
- MC2T is the automatic transformation between meta class of the DPM and a table;
- NON is the transformation of a comment to the Relational Model.

## 6.2 Framework

Figure 3 shows the structural perspective of the framework. This is an extract of Figure 1 in the DPM, referenced in [22]. The Data Point Model has 1 to N public elements. A public element inherits from different classes, as an element of the dictionary or frameworks [11] [22].

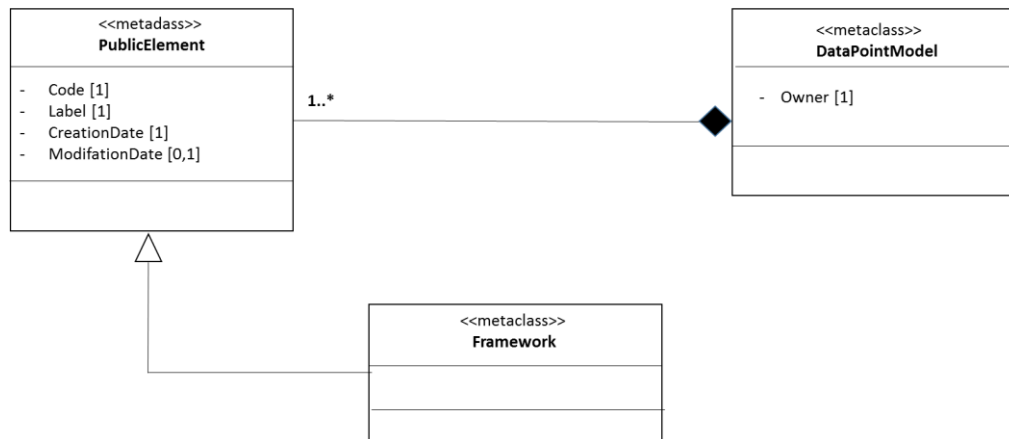


Figure 3 — Structural perspective of the framework

Figure 4 shows the transformation of the class public element and the framework. The aim is to obtain the table framework in ROLAP. For this the attributes of *publicElement* are mapped to the attributes of the table framework in a/the relational model, as the constructor.

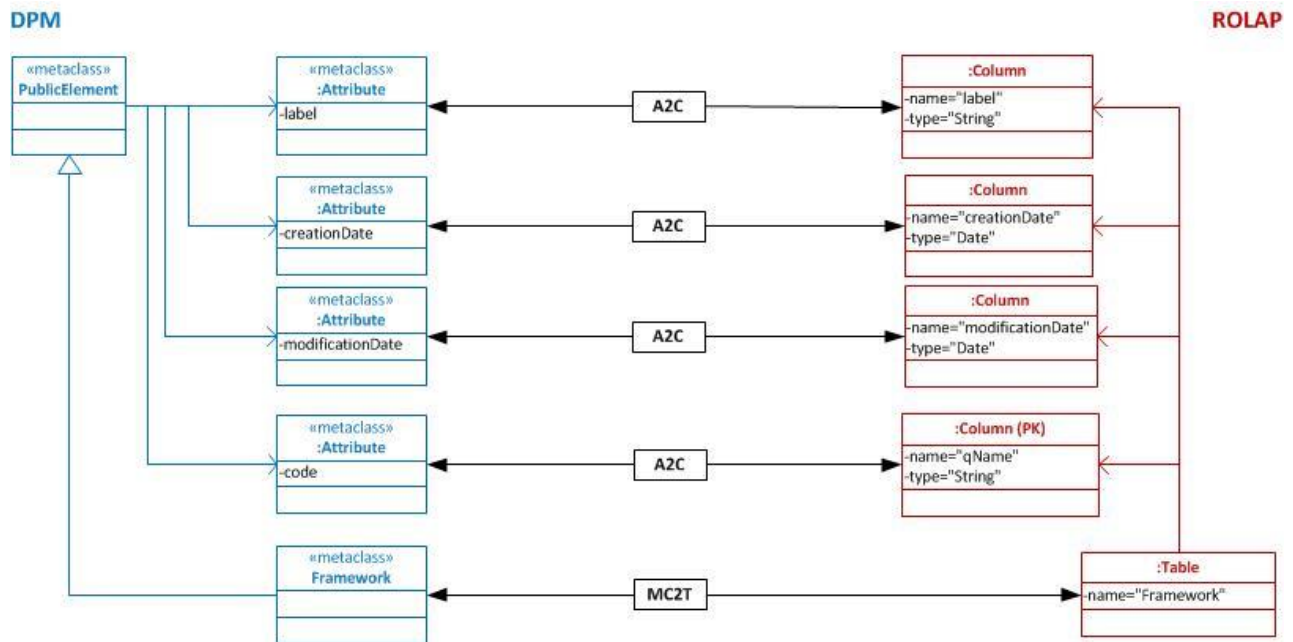


Figure 4 — Mapping for the framework

Table 3 shows the mapping of Figure 3 but in the format of a table. From the attribute label of the metaclass *PublicElement* the label is obtained. The transformation of the constructor Framework (ROLAP) is obtained the name and for deduction that the type of the label is string. The same applies to *Creationdate*, *ModificationDate*, and ID. The acronym 'pk' means primary key.

Table 3 — Mapping DPM vs ROLAP of the constructor *framework*

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constraint
PublicElement	Label	Framework	name	String	
PublicElement	CreationDate	Framework	CreationDate	DateTime	
PublicElement	ModificationDate	Framework	ModificationDate	DateTime	
PublicElement	Code	Framework	ID (Identifier)	String	pk

In the physical implementation of section six the ROLAP is updated. The primary key is another numeric attribute, because it is used to make the independent uniqueness constraint of the name of the business user in label and code. On the other hand, in the implementation information about the business user that has created the Framework is added. Figure 5 shows the implementation. An example of this paragraph is in section five.

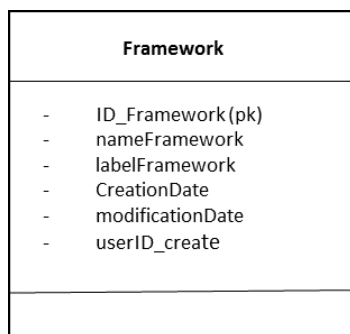


Figure 5 — Framework in the Relational Model

### 6.3 Taxonomy

In the same way the class taxonomy inherits the public element [11] [22], as the figure 6 shows.

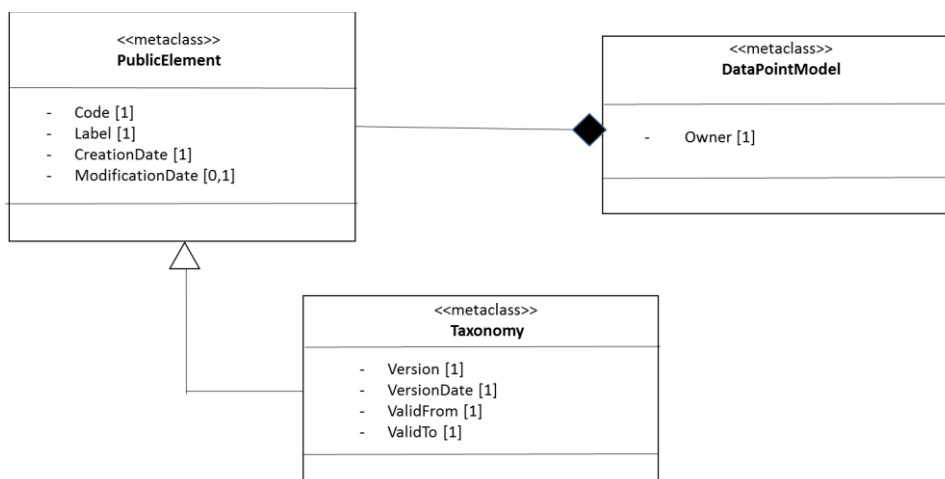


Figure 6 — Structural perspective of the taxonomy

In figure 7 the mapping between the meta classes *PublicElement* and *Taxonomy* of the DPM and the constructor Taxonomy and the RM (Relational Model) is shown. The official location of the taxonomy is added (comment in the UML).



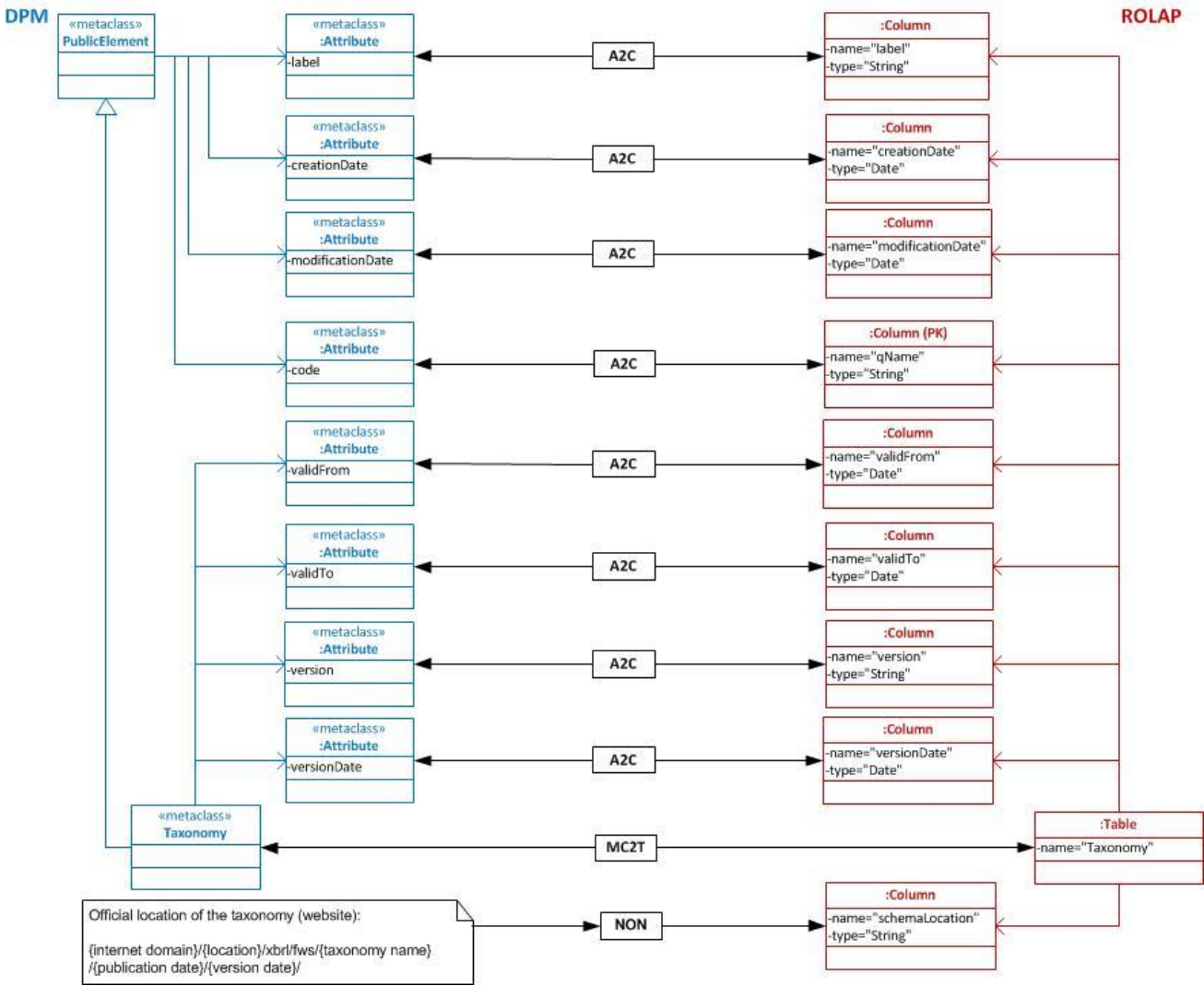


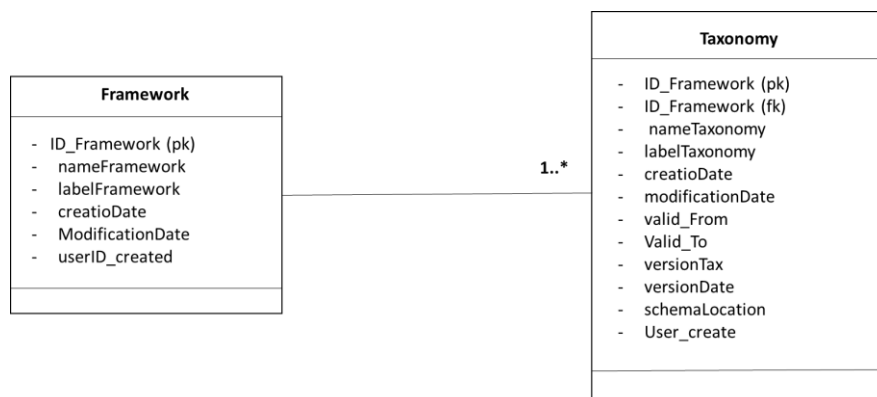
Figure 7 — Mapping for the constructor taxonomy

Next table 4 shows the mapping of figure 7 but in the format of a table. From the attribute label of the metaclass *PublicElement* the label is obtained and in the transformation of the constructor *Taxonomy* (ROLAP) the name is obtained and the type of the label is string. The same goes for *Creationdate*, *ModificationDate* and so on.

Table 4 — Mapping DPM vs ROLAP of the constructor *taxonomy*

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constrainst
PublicElement	Label	Taxonomy	name	string	
PublicElement	CreationDate	Taxonomy	CreationDate	DateTime	
PublicElement	ModificationDate	Taxonomy	ModificationDate	DateTime	
PublicElement	code	Taxonomy	ID (Identifier)	String	pk
PublicElement	ValidFrom	Taxonomy	ValidFrom	DateTime	
PublicElement	ValidTo	Taxonomy	ValidTo	String	
PublicElement	version	Taxonomy	version	String	
PublicElement	versionDate	Taxonomy	versionDate	DateTime	
Taxonomy		Taxonomy	nameTaxonomy	String	
		Taxonomy	schemaLocation	String	

In the physical implementation, in section six, the table is updated in the ROLAP design. The primary key is another numeric attribute, because it makes the independent uniqueness constraint of the name of business user in label and code. On the other hand, in the implementation the business user that has created the Taxonomy is also added. Moreover, the referential constraint is defined. Figure 8 shows the implementation of both constructors: Framework and Taxonomy. An example of this paragraph is shown in the section five. The acronym 'fk' refers to foreign key.

Figure 8 — Relationship between *framework* and *taxonomy* in the relational model

## 6.4 Dimensions

This section defines the mapping of the constructor *Dimension*. Figure 9 shows a perspective of the structure of the dimension and this is an extract of figure 1 in the DPM, referenced in [22].

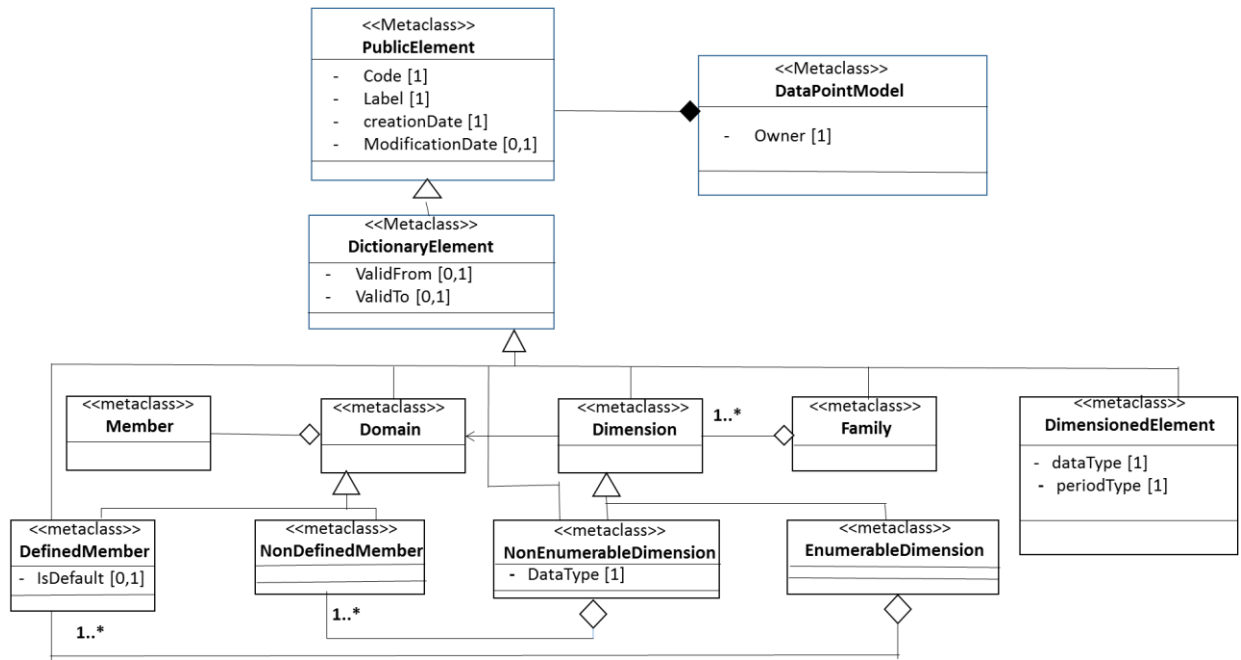
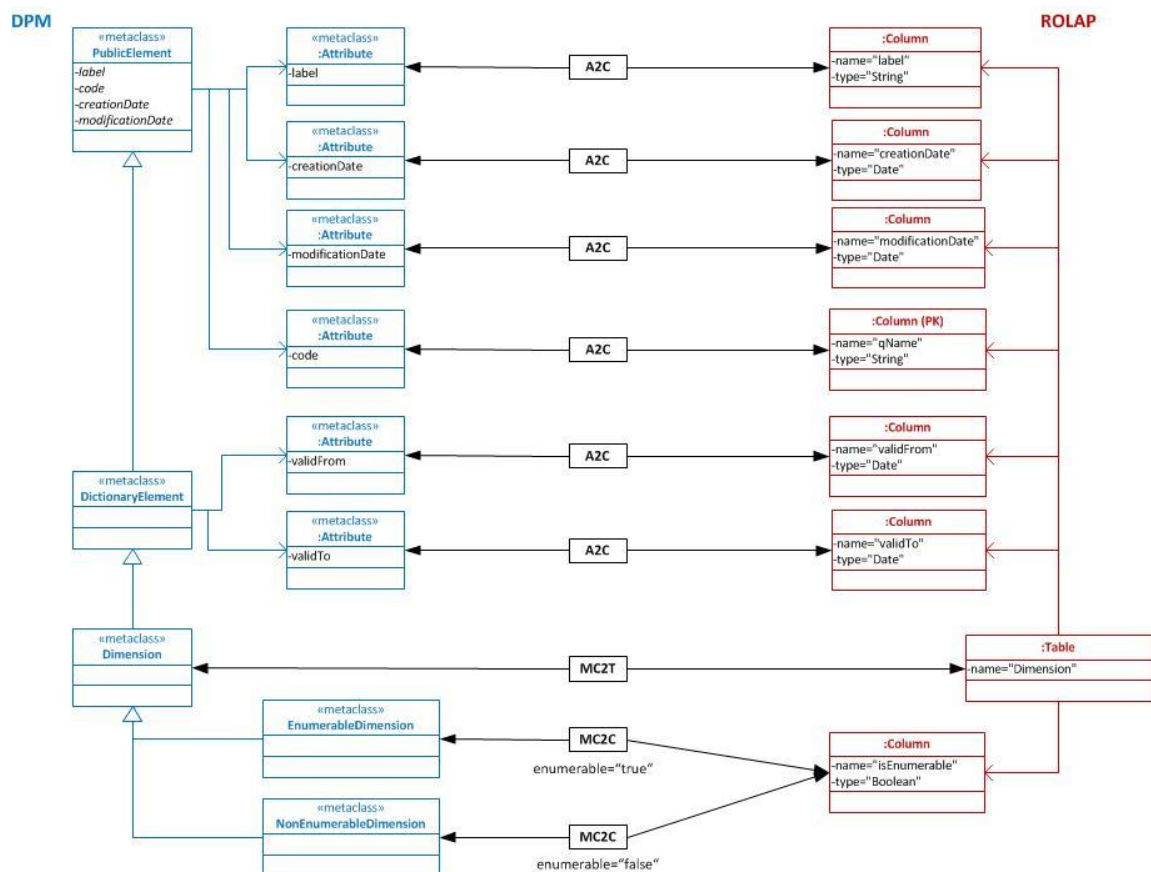


Figure 9 — Structural perspective of the dimension

This figure shows two types of dimensions [9] [10], the *enumerable* and the *non-enumerable* dimensions. But in an upper level is the *family*. However the family is not mapped to MDM in this document [20]. On the other hand, in a *non-enumerable* dimension, their *domain-members* are not known in advance, ergo in the Relational model (RM) it is not shown until the document instance is obtained, but they have a specific XSD type.



**Figure 10 — Mapping for dimensions**

Figure 10 shows the mapping of the enumerable and the non-enumerable dimensions to the ROLAP. The transformations among *PublicElement*, *DictionaryElement* and *EnumerableDimension* are detailed to help the reader understand.

Table 5 shows the mapping of figure 10 but in the format of a table. The label is obtained from the attribute label of the meta class *PublicElement* and the name is obtained in the transformation of the constructor Dimension (ROLAP). Assumed is the type of the label or name is a string. The same goes for *Creationdate*, *ModificationDate*, and so on.

**Table 5 — Mapping DPM vs ROLAP of the constructor *dimension***

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constrainst
PublicElement	Label	Dimension	name	String	
PublicElement	CreationDate	Dimension	CreationDate	DateTime	
PublicElement	ModificationDate	Dimension	ModificationDate	DateTime	
PublicElement	code	Dimension	ID (Identifier)	String	pk
DictionaryElement	ValidFrom	Dimension	ValidFrom	DateTime	
DictionaryElement	ValidTo	Dimension	ValidTo	String	
Dimension	EnumerableDimension	Dimension	isEnumerable	Boolean	
Dimension	NonEnumerabledimension	Dimension	isEnumerable	Boolean	

Figure 11 depicts the constructor Dimension in the ROLAP design in the physical implementation (Annex B). The primary key is another attribute of numeric type, because its purpose is to make the independent uniqueness constraint of the name of business user in label and code. The attribute *typeData* shows the data type of the members that are not defined in the non-enumerable dimensions. In addition, an attribute to this constructor is added that serves as a reference to the domain. An example of this paragraph is in section five.

Dimension
<ul style="list-style-type: none"> <li>- DimensionID (pk)</li> <li>- DimensionCode</li> <li>- DimensionLabel</li> <li>- CreationDate</li> <li>- ModificationDate</li> <li>- DomainID</li> <li>- isEnumerable</li> <li>- typeData</li> <li>- Valid_from</li> <li>- Valid_to</li> </ul>

**Figure 11 — Constructor Dimension in ROLAP**

In the Relational model, the constructors *enumerable* and *non-enumerable* are the constructor dimension (figure 12). The entity *Dimension* will have an attribute for showing if the dimension is *non-enumerable* or *enumerable* and another attribute with the data type of the *domain-members* of the constructor *non-enumerable dimension*. Moreover, in this implementation is added the name of the domain that belongs to the dimension.

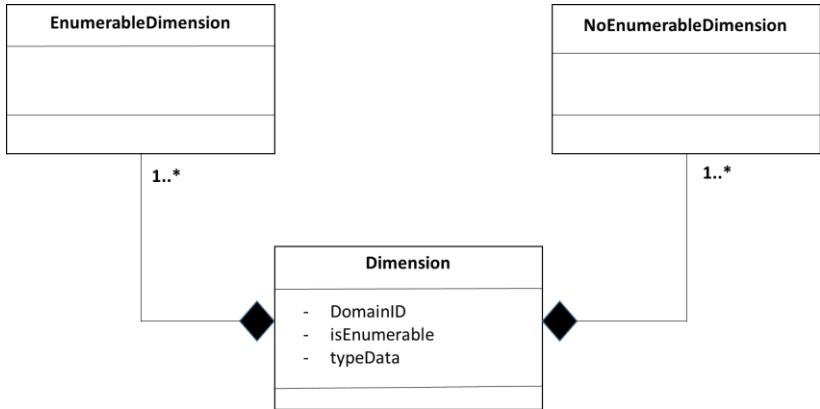


Figure 12 — Constructors enumerable, non-enumerable and dimension in ROLAP

If the dimensions are defined, as a next step the domain-members are defined. Figure 13 shows the mapping of the members in the ROLAP design. However, the name is changed to *DimensionAttribute*.

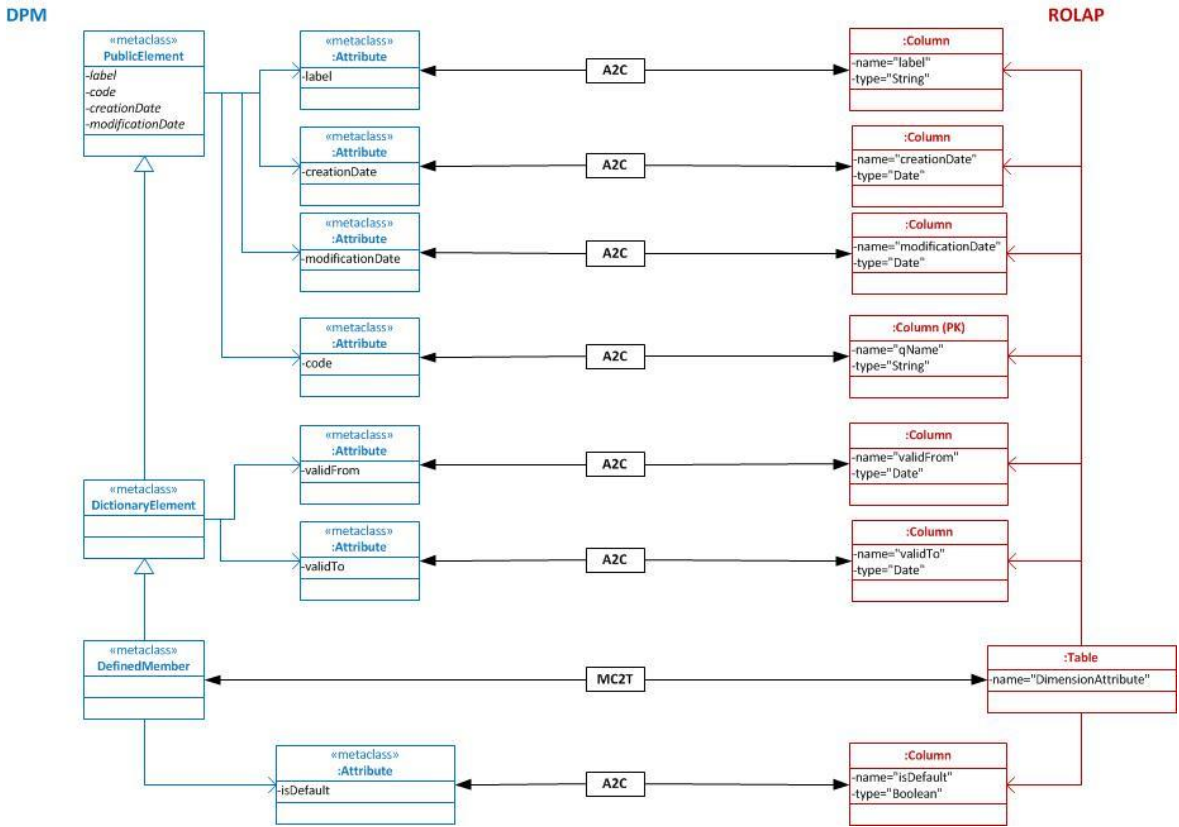


Figure 13 — Mapping of members in the ROLAP

Table 6 shows the mapping of figure 13, but in the format of a table. From the attribute label of the meta class *PublicElement* the label is obtained and in the transformation of the constructor *DimensionAttribute* (ROLAP) the name is obtained. Assumed is that the type of the label or name is a string. The same goes for Creationdate, ModificationDate, and so on.

Table 6 — Mapping DPM vs ROLAP of the constructor *DimensionAttribute*

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constrai nst
-----	-----------------------	------------------	-----------	------	-----------------

PublicElement	Label	DimensionAttribute	name	string	
PublicElement	CreationDate	DimensionAttribute	CreationDate	DateTime	
PublicElement	ModificationDate	DimensionAttribute	ModificationDate	DateTime	
PublicElement	code	DimensionAttribute	ID (Identifier)	String	pk
DictionaryElement	ValidFrom	DimensionAttribute	ValidFrom	DateTime	
DictionaryElement	ValidTo	DimensionAttribute	ValidTo	String	
DefinedMember	isDefault	DimensionAttribute	isDefault	boolean	

Figure 14 depicts the constructor *DimensionAttribute* in the ROLAP, in the physical implementation (section six). The primary key is another numeric attribute, because it makes the independent uniqueness constraint of the name of business user in label and code. This table in the Relational Model is filled out with the concepts of the taxonomy, but also during run-time, because the attributes of the dimension are for enumerable and non-enumerable dimensions. Moreover, an attribute is added in this constructor that has a reference to the domain. An example of this paragraph is referenced in section five.

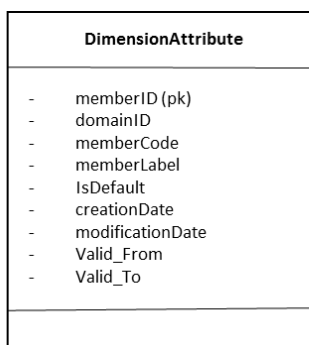


Figure 14 — DimensionAttribute in the ROLAP

Figure 15 shows the mapping of Dimensions and domain-members in the DPM and Dimensions/Dimension attributes in the Relational data model (ROLAP). This constructor, named *Dimension\_DimensionAttribute*, really is an artifact, is not defined in the DPM. However, this constructor is important, because the model claims that the combinations between dimensions and attributes of dimensions in the relational Model are precise.

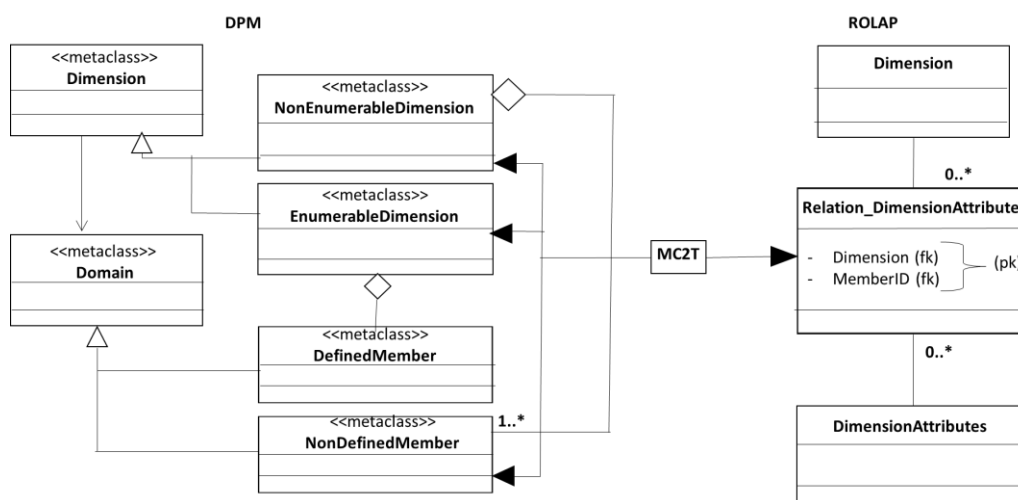
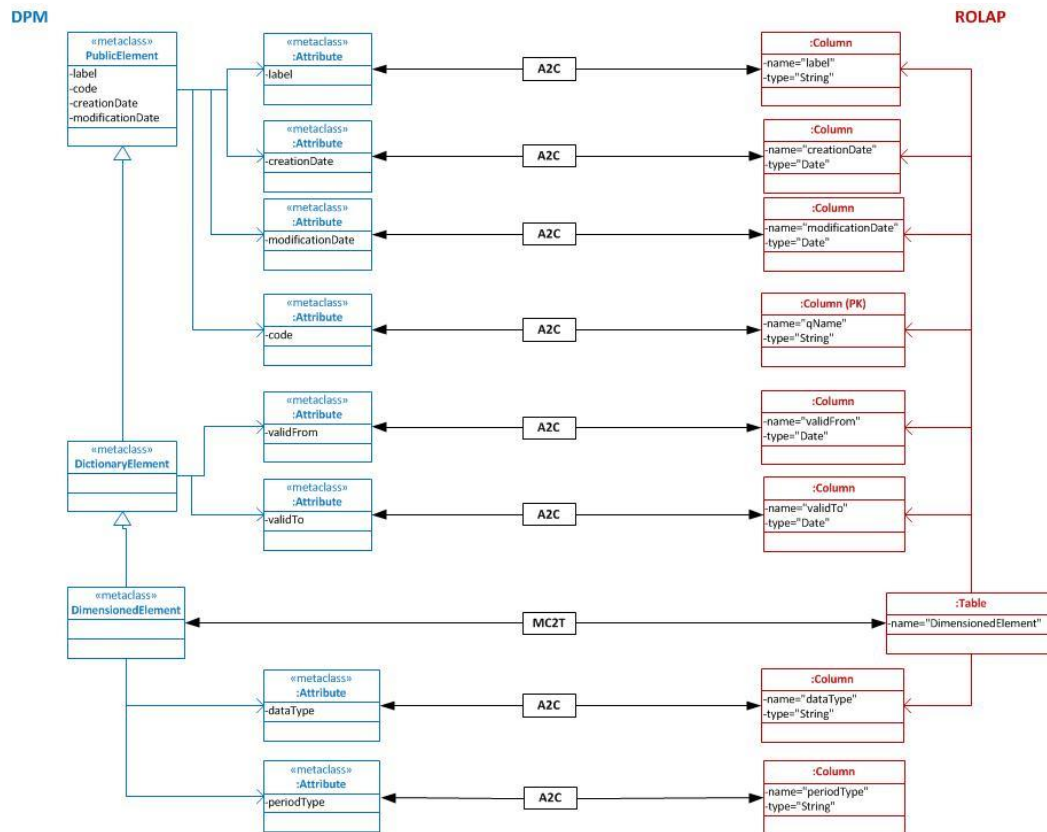


Figure 15 — Mapping of *Dimensions* and *Members*

## 6.5 Context

The *context* is not part of the DPM. The *context* is defined in the *instance* (XBRL document instance or XBRL report). The corresponding UML model is included in the filing rules document of CWA1 [23].

The figure 16 shows the mapping of the *context* and the pairs *dimension/member* belong to the instance.



**Figure 16 — Mapping of the *context* and the pairs *dimension/member***

In the mapping to the ROLAP design there are two necessary constructors. These constructors are *context* and *contextDimensionMemberPar*. The mapping is shown in tabulated format in tables 7 and 8. In the transformation of table 8, the three columns have the acronym 'pk' (primary key), because the primary key is the set of the three attributes.

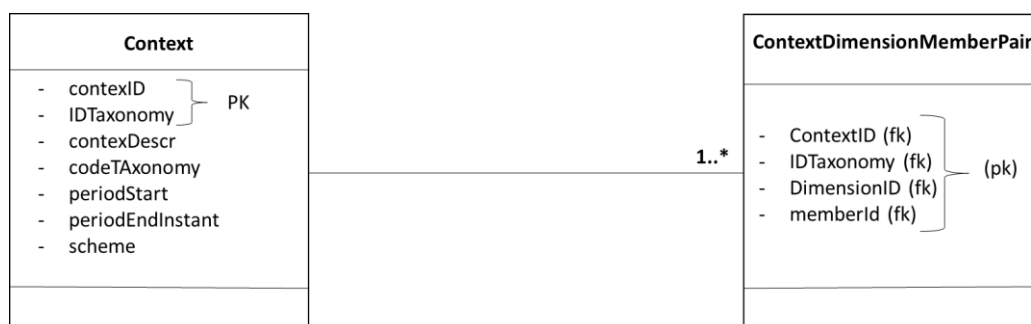
**Table 7 — Mapping DPM vs ROLAP of the constructor *context***

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constrainst
PublicElement	idContext	Context	contextDescr	String	
PublicElement	id	Context	IDContext	String	pk
PublicElement	periodStart	Context	periodStart	DateTime	
PublicElement	periodEndIntant	Context	periodEndIntant	DateTime	
PublicElement	ValidFrom	Context	ValidFrom	DateTime	
PublicElement	scheme	Context	scheme	String	

**Table 8 — Mapping DPM vs ROLAP of the constructor *contextDimensionMemberPair***

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constraint
PublicElement	id	contextDimensionMemberPair	IDContext	String	pk
PublicElement	qNameDimension	contextDimensionMemberPair	dimensionID	String	pk
PublicElement	qNameMeber	contextDimensionMemberPair	memberID	string	pk

In the physical implementation, the table *context* consists of the name of the context and in this approach the *taxonomy*, because, in theory, it could have different taxonomies with the same *context*, but with different semantics. The table 7 shows this mapping with the *context*.

**Figure 17 — Relational model of the *context* and *contextDimensionMemberPair***

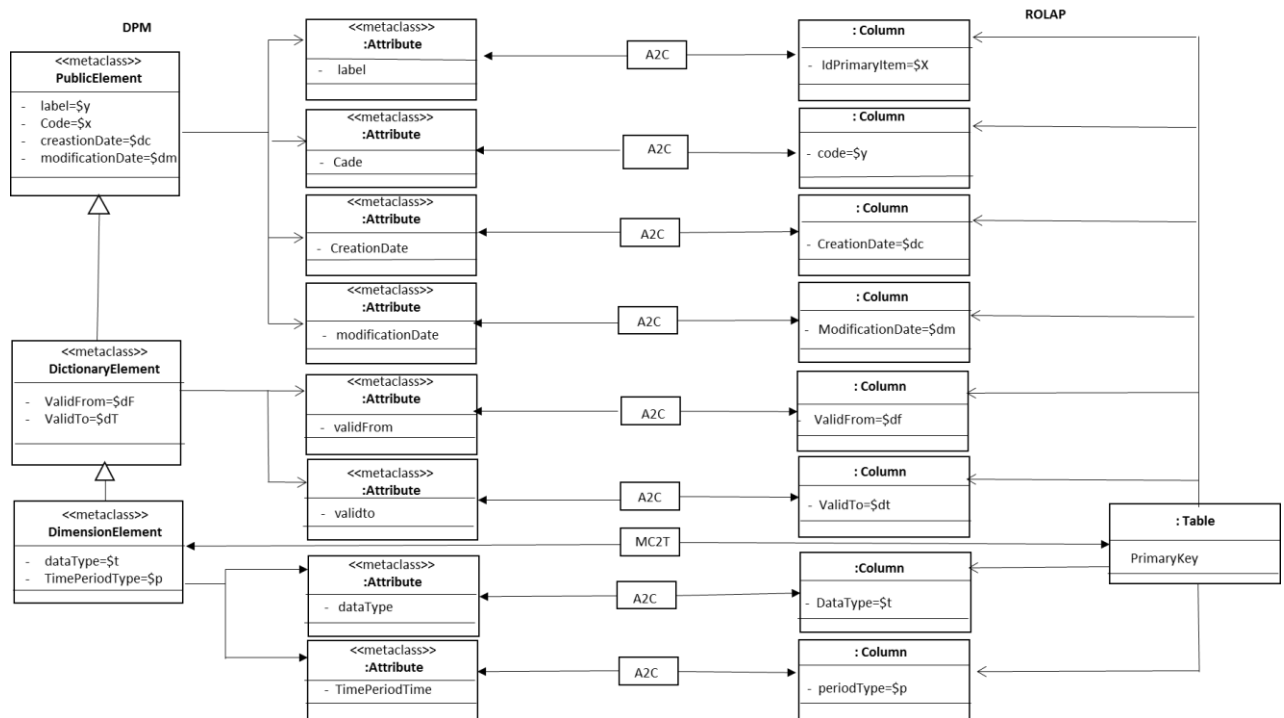
## 6.6 Primary Items

The primary item could be a domain-member of a dimension, however, it is somewhat special, because two attributes are associated with this concept: the type of the data and the time period type. It therefore has an important semantic content. Figure 18 shows the mapping with the relational model. The set of primary items are grouped in the base dimension. In this figure, this is called the constructor *PrimaryItem*. The EBA (section five) considers the base domain as a normal dimension.

This specific dimension, called *primary item* or *base domain* has the next features that it holds more semantics contents [9] [10]:

- It has a basic data type. This characteristic specifies the kind of data to be reported: a number, a date, a text, a monetary amount (a number plus a currency). This information is also used by IT applications to determine the way data is represented in electronic files. If the type is *monetary* there is an attribute named *balance*, with two values: *[credit|debit]*.
- Time period type to which the data refers: does it refer to a specific point in time (*instant*) or to an interval of time (*duration*).



Figure 18 — Mapping for the *Base Dimension* (set of *primary items*)

The table 9 shows in tabulate format this mapping of the figure 18.

Table 9 — Mapping DPM vs ROLAP of the constructor *Base\_Dimension*

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constraint
PublicElement	code	Base_Dimension	IDPrimaryItem	String	pk
PublicElement	label	Base_Dimension	descrPrimItem	String	
PublicElement	CreationDate	Base_Dimension	CreationDate	DateTime	
PublicElement	ModificationDate	Base_Dimension	ModificationDate	DateTime	
DictionaryElement	ValidFrom	Base_Dimension	ValidFrom	DateTime	
DictionaryElement	ValidTo	Base_Dimension	ValidTo	DateTime	
DimensionElement	dataType	Base_Dimension	dataType	String	
DimensionElement	TimePeriodTime	Base_Dimension	periodTime	String	

The figure 19 shows the constructor *Base\_Dimension* in the design ROLAP. In this implementation is added the balance with its operation of check, and the time period type with its check, although in this document is not dealt with the validation. The user that has created the primary item is added.

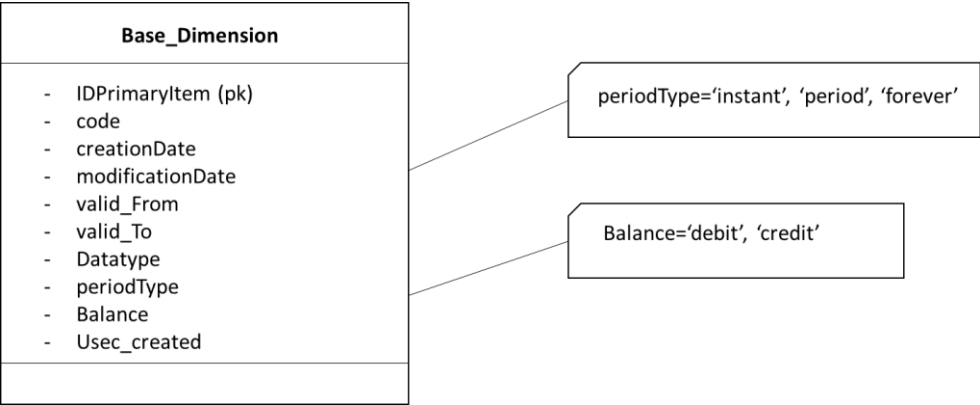


Figure 19 — Base\_dimension (set of primary items) in the Relational Model

6.7 Fact table or Data points

The *Datapoint* in the DPM is equivalent to the fact table in the MDM, and it is the union of the table context, set of *primary items* or base dimension and taxonomy. The figure 20 shows the mapping.

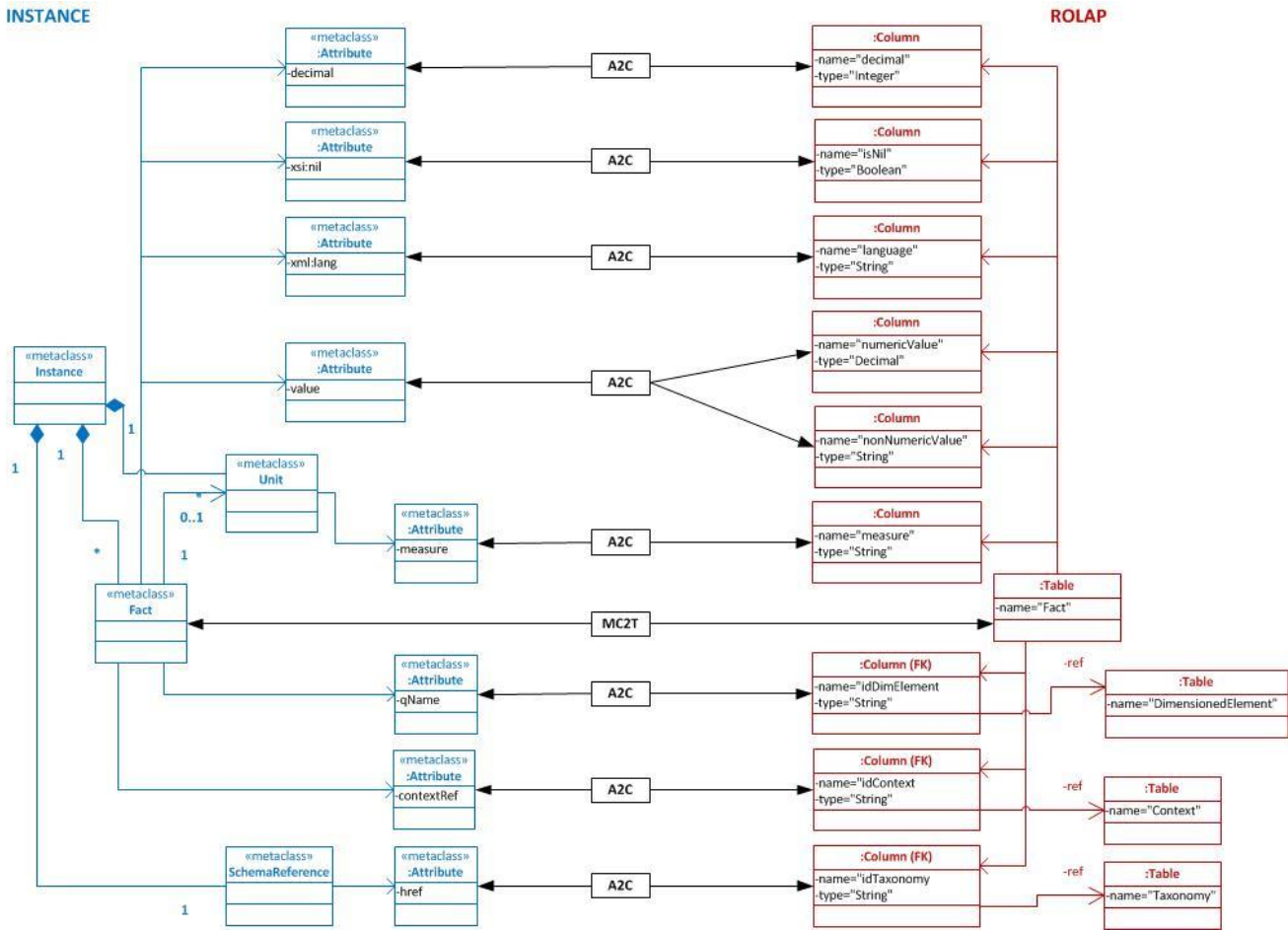


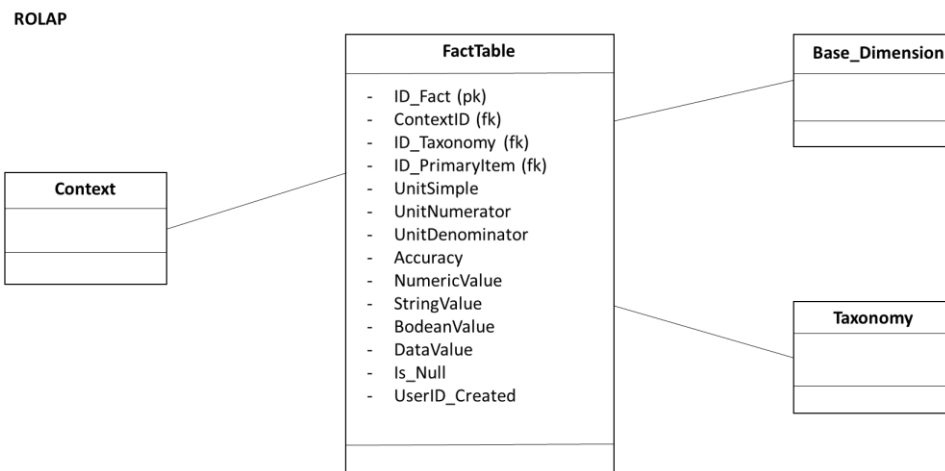
Figure 20 — Mapping of the *data point* and the *fact table*

The table 10 shows in tabulate format this mapping of the figure 20.

**Table 10 — Mapping DPM vs ROLAP of the constructor *Fact table***

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constrainst
Fact	qName	FactTable	IDFact	String	pk
SchemaRef	href	FactTable	IDTaxonomy	String	
Fact	contextRef	FactTable	contextID	String	
PublicElement	code	FactTable	IDPrimaryItem	String	
Instance/Fact	Unit	FactTable	Unit	String	
Instance	value	FactTable	Value	String	
Instance	language	FactTable	lang	String	
Instance	Isnil	FactTable	Is_Null	Booleanan	
Instance	decimal	FactTable	decimal	Number	

Figure 21 shows the constructor fact table in the relational model. However, in this model the type of unit, the accuracy, the value is added. Depending on the type it will be string, numeric or Boolean. The name of the user that created the fact is also added.

**Figure 21 — Diagram ROLAP of the *Fact table* of the DPM**

## 6.8 Summary

The figure 22 shows with more detail the figure 2 of the Star model. In this figure is possible to see constructors with their columns and the relationships through the foreign keys.

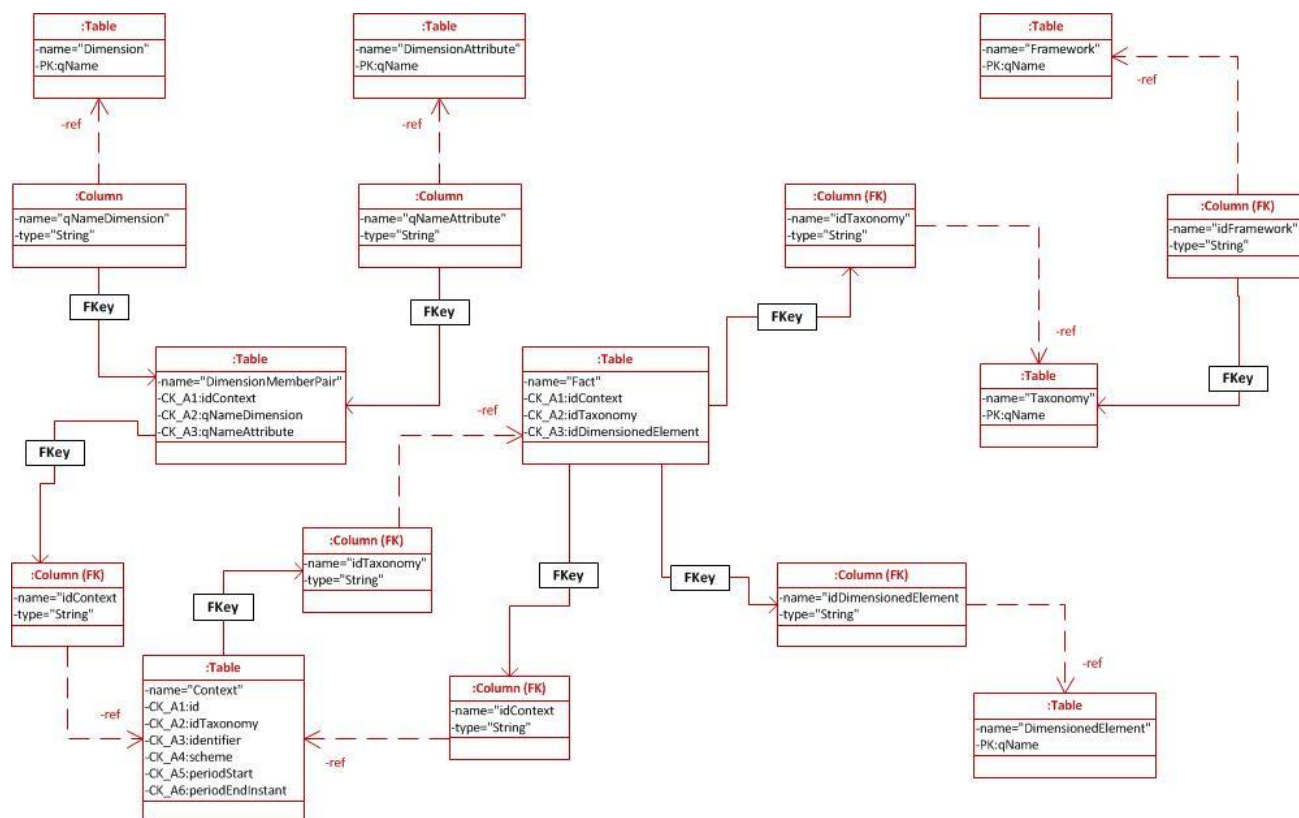


Figure 22 — Diagram ROLAP, summary

Next table 11 shows a summary in tabulating format of the total mapping DPM versus design ROLAP.

Table 11 — Mapping DPM vs ROLAP of the set of constructors

DPM	Attribute/constructor	Costructor ROLAP	Attribute	Type	Constraints
PublicElement	Label	Framework	name	String	
PublicElement	CreationDate	Framework	CreationDate	DateTime	
PublicElement	ModificationDate	Framework	ModificationDate	DateTime	
PublicElement	code	Framework	ID (Identifier)	String	pk
PublicElement	Label	Taxonomy	name	String	
PublicElement	CreationDate	Taxonomy	CreationDate	DateTime	
PublicElement	ModificationDate	Taxonomy	ModificationDate	DateTime	
PublicElement	code	Taxonomy	ID (Identifier)	String	pk
PublicElement	ValidFrom	Taxonomy	ValidFrom	DateTime	
PublicElement	ValidTo	Taxonomy	ValidTo	String	
PublicElement	version	Taxonomy	version	String	
PublicElement	versionDate	Taxonomy	versionDate	DateTime	
Taxonomy		Taxonomy	nameTaxonomy	String	
		Taxonomy	schemaLocation	String	
PublicElement	Label	Dimension	name	String	
PublicElement	CreationDate	Dimension	CreationDate	DateTime	

PublicElement	ModificationDate	Dimension	ModificationDate	DateTime	
PublicElement	code	Dimension	ID (Identifier)	String	pk
DictionaryElement	ValidFrom	Dimension	ValidFrom	DateTime	
DictionaryElement	ValidTo	Dimension	ValidTo	String	
Dimension	EnumerableDimension	Dimension	isEnumerable	boolean	
Dimension	NonEnumerabledimension	Dimension	isEnumerable	Boolean	
PublicElement	Label	DimensionAttribute	name	String	
PublicElement	CreationDate	DimensionAttribute	CreationDate	DateTime	
PublicElement	ModificationDate	DimensionAttribute	ModificationDate	DateTime	
PublicElement	code	DimensionAttribute	ID (Identifier)	String	pk
DictionaryElement	ValidFrom	DimensionAttribute	ValidFrom	DateTime	
DictionaryElement	ValidTo	DimensionAttribute	ValidTo	String	
DefinedMember	isDefault	DimensionAttribute	isDefault	Boolean	
PublicElement	idContext	Context	contextDescr	String	
PublicElement	id	Context	IDContext	String	pk
PublicElement	periodStart	Context	periodStart	DateTime	
PublicElement	periodEndIntant	Context	periodEndIntant	DateTime	
PublicElement	ValidFrom	Context	ValidFrom	DateTime	
PublicElement	scheme	Context	scheme	String	
PublicElement	id	contextDimensionMemberPair	IDContext	String	pk
PublicElement	qNameDimension	contextDimensionMemberPair	dimensionID	String	pk
PublicElement	qNemeMeber	contextDimensionMemberPair	memberID	String	pk
PublicElement	code	Base_Dimension	IDPrimaryItem	String	pk
PublicElement	label	Base_Dimension	descrPrimItem	String	
PublicElement	CreationDate	Base_Dimension	CreationDate	DateTime	
PublicElement	ModificationDate	Base_Dimension	ModificationDate	DateTime	
DictionaryElement	ValidFrom	Base_Dimension	ValidFrom	DateTime	
DictionaryElement	ValidTo	Base_Dimension	ValidTo	DateTime	
DimensionElement	dataType	Base_Dimension	dataType	String	
DimensionElement	TimePeriodTime	Base_Dimension	periodTime	String	

Fact	qName	FactTable	IDFact	String	pk
SchemaRef	href	FactTable	IDTaxonomy	String	
Fact	contextRef	FactTable	contextID	String	
PublicElement	code	FactTable	IDPrimaryItem	String	
Instance/Fact	Unit	FactTable	Unit	String	
Instance	value	FactTable	Value	String	
Instance	language	FactTable	lang	String	
Instance	Isnil	FactTable	Is_Null	Booleanan	
Instance	decimal	FactTable	decimal	Number	

## 7 Metamodel defined by the EBA (FINREP and COREP) mapped to MDM

### 7.1 Introduction

This section maps the relational model of the DPM supplied by the EBA in the MDM, using the design ROLAP.

The EBA published its meta model on March 15, 2013 and after several modifications the final version on November 29, 2013 for the reporting year 2014. The publication contained an updated version of the templates, instructions, validation rules and data point model for Implementing Technical Standards (ITS) on supervisory reporting, COREP and FINREP [16]. At that same time EBA published the DPM Database 0.1.1 as a Meta model structure used as the repository for all the metadata defined in the DPM from which the XBRL taxonomies will be derived. This section will map this structure of the EBA to the relational data model [18]. The database is built from this document and with the help of a paper under review [19]. For a better understanding the implementation is done in MS SQL Server, version 2012, Sp1. However, the move to another RDBMS is easy, because ANSI-SQL is a standard. In the first step, the structure of the DPM is created in the RDBMS, in this case MS SQL Server. The second step is to populate the DPM in the database with the datamodel of the EBA (DPM Database 0.1.1) through an ETL (Extract, Transform and Load) tool.

The EBA version for this example did not contain any XBRL Instance documents, which made it impossible to fill out the fact table with an example, but the structure of the DPM is complete. This database model found however a difference with the DPM data model, the 'base dimension' is a normal explicit dimension but was unused therefore the table base dimension is empty.

### 7.2 Creation of the structure and load of the DPM from the EBA in a RDBMS

Section six is displaying the creation of the structure of the DPM in a RDBMS using the MDM, hosted by CWA1.

The zip file with the meta data model structure, DPM Database 0.1.1 is available for download from the EBA webpage [16]. When this file is obtained, its structure and data has to be moved to RDBMS. In this document MS SQL Server (the free edition, Microsoft® SQL Server®2012 Express) is used. However, it is possible to use other RDBMS's, such as Oracle, DB2, etc. For the purpose of this document, Integration Services (IS) of MS SQL Server (there is a free edition) is used to move from Access to SQL Server. In this tool, the data source is in Access (The driver used is Microsoft Access (Microsoft Set Database Engine), the target the client is SQL Server Native client 11.0 and the database is called DPM\_EBA (for the purpose of this document). Next, all tables have to be selected and the packet is submitted. Figure 23 shows a general view of the load of Access in a RDBMS and the mapping to DPM in a relational database.

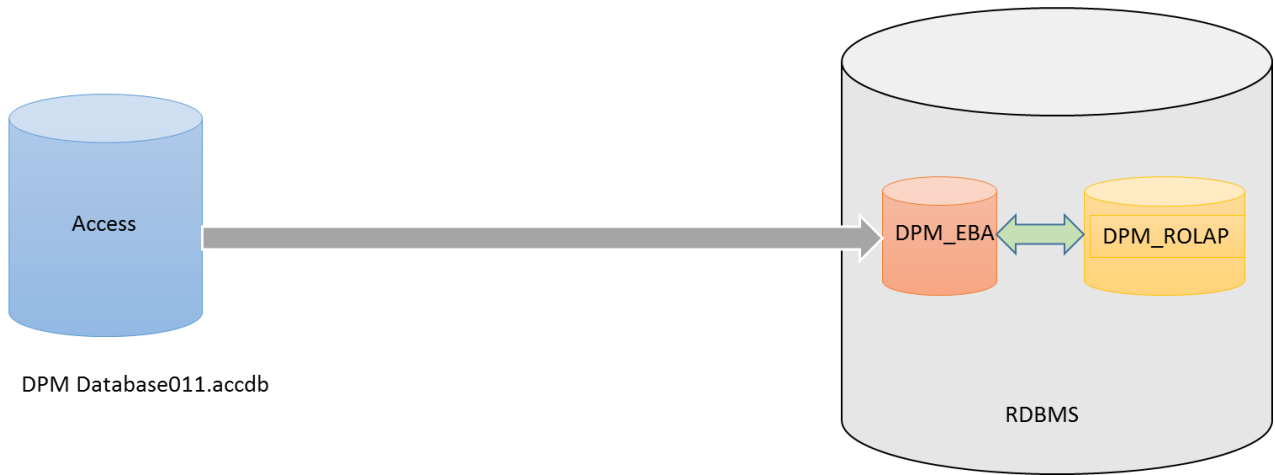


Figure 23 — General view of the mapping from Access to the RDBMS of the EBA and after the DPM in the design ROLAP

7.3 Loading DPM\_ROLAP from DPM\_EBA

This section contains a mapping from the database DPM\_EBA to the database DPM\_ROLAP, but with different models. DPM\_EBA is loaded in the above section and the DPM\_ROLAP database is created according to the process described in Annex B of this document.

As the first step, the table Framework is loaded from *ReportingFramework*. This load is shown in Figure 24, through its design and after the code. In the code in this document the dates are simulated.

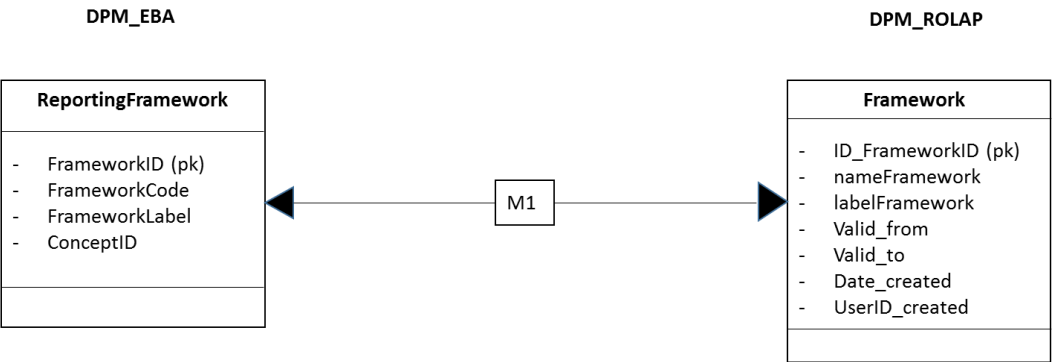


Figure 24 — Mapping of the framework

The code of M1 is:

```
use DPM_ROLAP
--
-- M1 CODE
--
delete from Framework
go
insert into Framework (ID_Framework, nameFramework, creationDate, userID_created)
select FrameworkID as ID_Framework,
       FrameworkCode as nameFramework,
       convert(datetime, '20130327', 112),
       'EBA'
FROM DPM_EBA..ReportingFramework
go
select * from Framework
go
```

If the *framework* is loaded, the next table is *Taxonomy*, which is loaded from the database *DPM\_EBA..Taxonomy*.

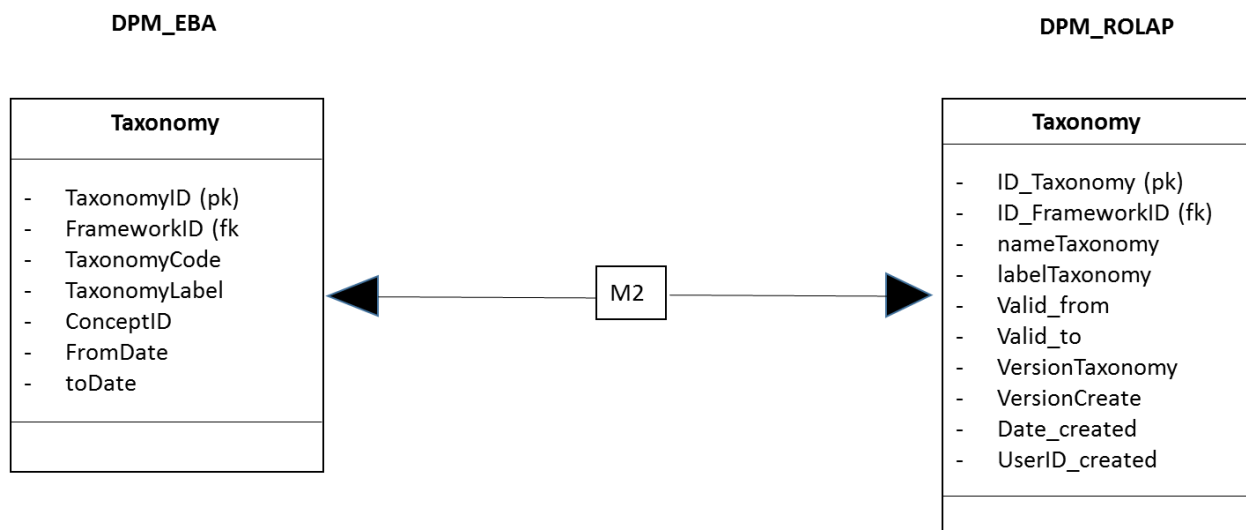


Figure 25 — Mapping of the taxonomy



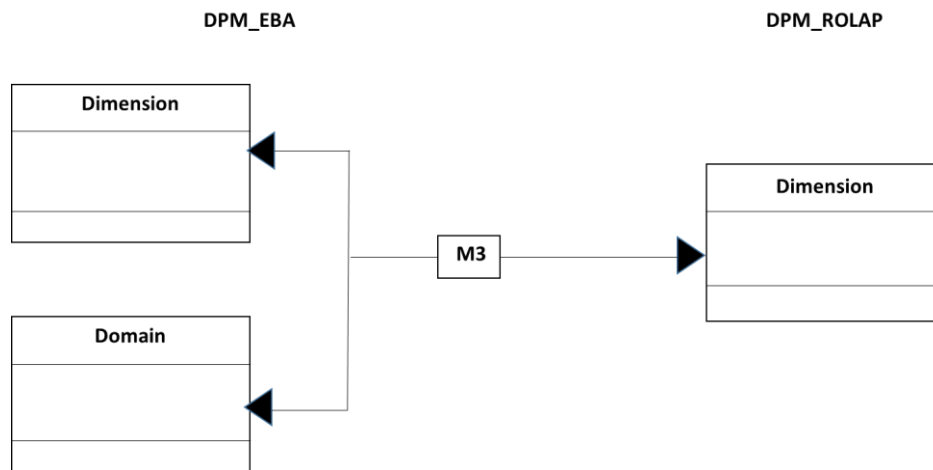
The code of the mapping M2 is:

```

use DPM_ROLAP
--
-- code M2
--
--truncate table taxonomy
delete from Taxonomy
go
insert into Taxonomy(ID_Taxonomy, ID_Framework, nameTaxonomy,
                    labelTaxonomy, valid_from, versionTax,
                    creationDate, userid_created)
select TaxonomyID as ID_Taxonomy, FrameworkID, TaxonomyCode,
       TaxonomyLabel, convert(datetime, '20130327', 112), '0',
       convert(datetime, '20130327', 112), 'EBA'
from [DPM_EBA].[dbo].[Taxonomy]
go
select * from Taxonomy
go

```

The next step is to obtain dimensions from the EBA, and it is shown in the figure 26.

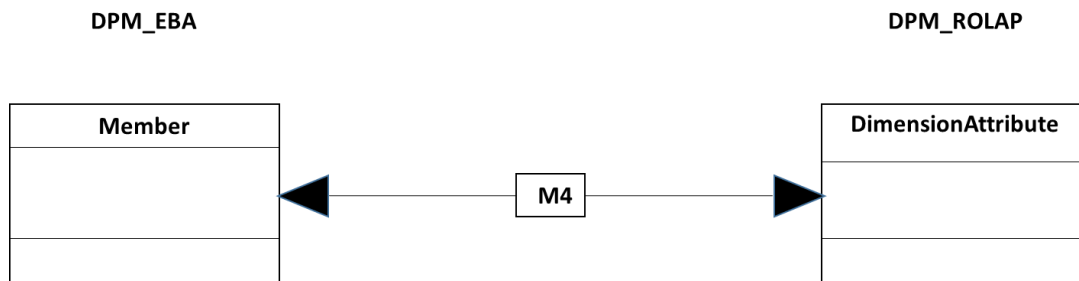


**Figure 26 — The mapping of dimensions**

The code of the mapping M3 is:

```
--
-- code M3
--
go
delete from Dimension
go
insert into Dimension (dimensionID, dimensionCode,
    dimensionLabel, domainID,
    isEnumerable,
    typeData, creationDate,
    valid_from)
select a.DimensionID, a.DimensionCode,
    a.DimensionLabel as dimensiondescr, a.DomainID,
    a.IsTyped as typedDim,
    cast(b.DataTypeID as nvarchar(30)) as typeData,
    convert(datetime, '20130327', 112) as creationDate,
    convert(datetime, '20130327', 112) as valid_from
from DPM_EBA.dbo.Dimension a inner join DPM_EBA.dbo.Domain b
    on a.DomainID=b.DomainID
go
select dimensionID, dimensionCode,
    dimensionLabel, domainID,
    isEnumerable,
    typeData, creationDate,
    valid_from
from Dimension
go
```

As a next step, the dimension attributes are obtained, as it is shown in the figure 27.



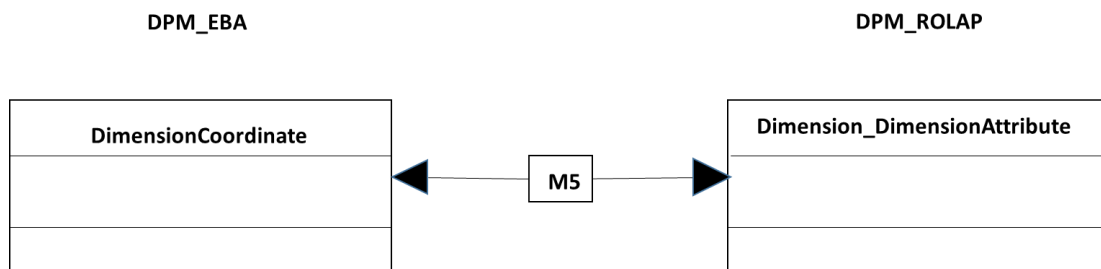
**Figure 27 — Mapping of the attributes of dimensión (ROLAP)**

The code of the mapping M4 is:

```
--
--- Code M4
---
delete from DimensionAttribute
go
insert into DimensionAttribute(memberID, domainID, memberCode,
    memberLabel, isDefault, creationDate,
    valid_from)
Select MemberID, DomainID, MemberCode,
    MemberLabel as memberLabel, IsDefaultMember as isDefault,
    convert(datetime, '20130327', 112) as creationDate,
    convert(datetime, '20130327', 112) as valid_from
from DPM_EBA.dbo.Member
go

select memberID, domainID, memberCode, memberLabel,
    isDefault, creationDate, valid_from,
    valid_to
from DimensionAttribute
go
```

The relations between dimensions and attributes of dimension are shown in figure 28.



**Figure 28 — Relationship between dimensions and attributes of dimension**

The code of the mapping M5 is:

```
---
--- Code M5
---
go

delete from Dimension_DimensionAttribute
go

insert into Dimension_DimensionAttribute(dimensionID, memberID)
select DimensionID, MemberID
from DPM_EBA.dbo.DimensionCoordinate
go

select dimensionID, memberID
from Dimension_DimensionAttribute
go
```

The next table shows the *context* and Figure 29 shows the mapping. As a data point (a fact) can be referenced by a context, but this *context* belongs to a taxonomy, the *context* needs the taxonomy (module is named by the EBA).

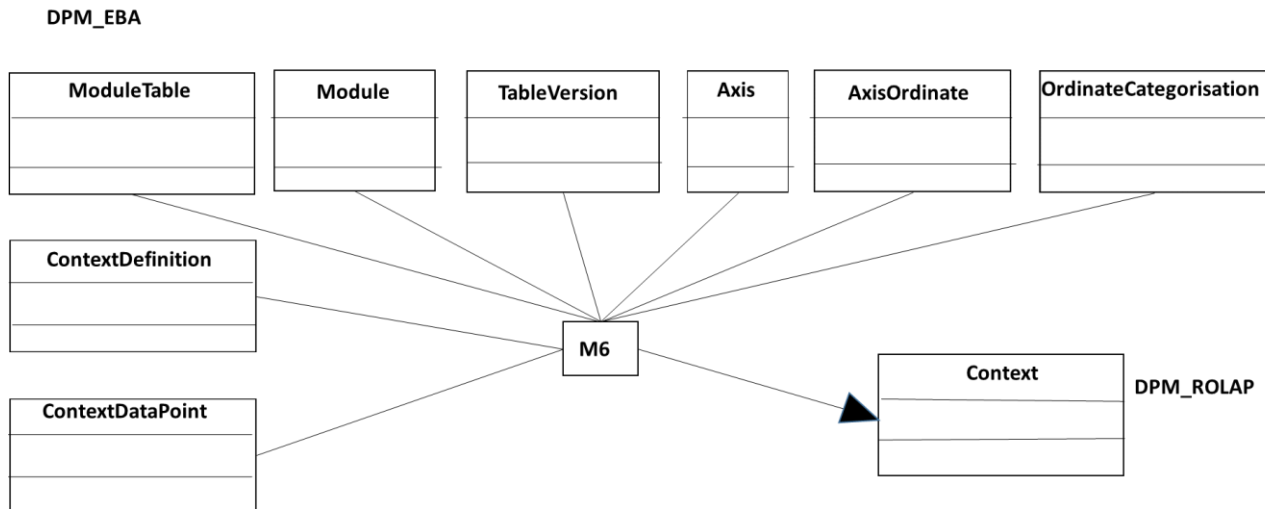


Figure 29 — Mapping of the context from DPM\_EBA

The code of the transformation M6:

```

---
--- Code M6
---
go

delete from Context
go

insert into Context (contextID, ID_Taxonomy, contextDescr, codeTaxonomy)
select g.ContextID, b.ModuleID as ID_Taxonomy,
       h.ContextKey as contextDescr, b.ModuleCode as codeTaxonomy
from DPM_EBA.dbo.ModuleTable a inner join DPM_EBA.dbo.Module b
    on a.ModuleID=b.ModuleID
    inner join DPM_EBA.dbo.TableVersion c
        on a.TableVID=c.TableVID
    inner join DPM_EBA.dbo.Axis d
        on a.TableVID=d.TableVID
    inner join DPM_EBA.dbo.AxisOrdinate e
        on d.AxisID=e.AxisID
    inner join DPM_EBA.dbo.OrdinateCategorisation f
        on e.OrdinateID=f.OrdinateID
    inner join DPM_EBA.dbo.ContextDefinition g
        on (f.DimensionID=g.DimensionID and f.MemberID=g.MemberID)
    inner join DPM_EBA.dbo.ContextOfDataPoints h
        on g.ContextID=h.ContextID
group by g.ContextID, b.ModuleID, b.ModuleCode, h.ContextKey
go
  
```

With regards to the *context* and the dimensions and attributes of dimension, the transformation can be analysed in the figure 30.

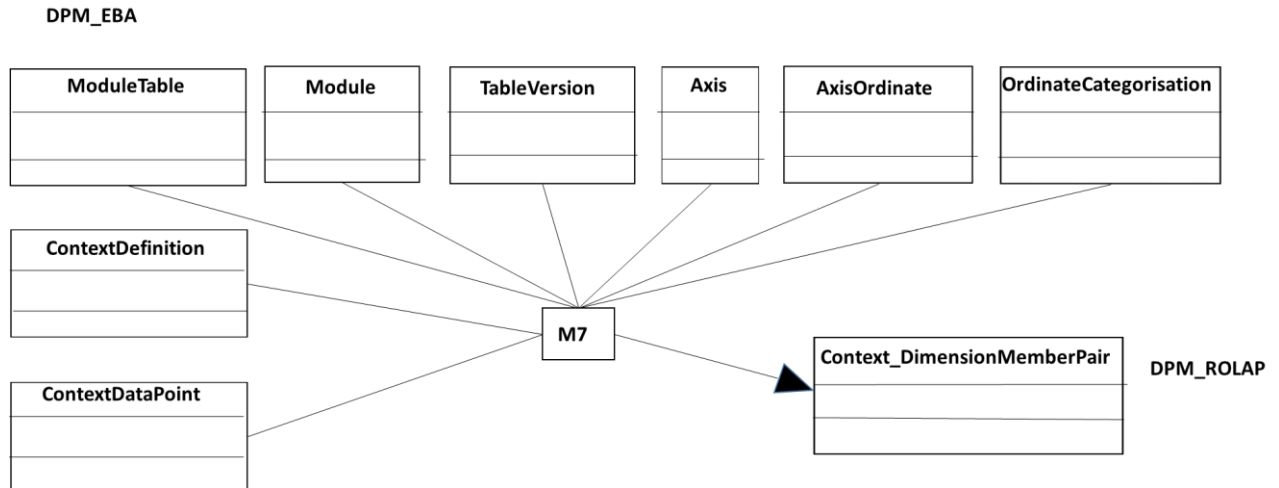


Figure 30 — Mapping of the Context\_DimensionMemberPair

And the transformation code M7:

```

---
--- Code M7
---
delete from contextDimensionMemberPair
go
insert into contextDimensionMemberPair(contextID, ID_Taxonomy, dimensionID,
memberID)
select g.ContextID, b.ModuleID as ID_Taxonomy, f.DimensionID, f.MemberID
from DPM_EBA.dbo.ModuleTable a inner join DPM_EBA.dbo.Module b
on a.ModuleID=b.ModuleID
inner join DPM_EBA.dbo.TableVersion c
on a.TableVID=c.TableVID
inner join DPM_EBA.dbo.Axis d
on a.TableVID=d.TableVID
inner join DPM_EBA.dbo.AxisOrdinate e
on d.AxisID=e.AxisID
inner join DPM_EBA.dbo.OrdinateCategorisation f
on e.OrdinateID=f.OrdinateID
inner join DPM_EBA.dbo.ContextDefinition g
on (f.DimensionID=g.DimensionID and f.MemberID=g.MemberID)
inner join DPM_EBA.dbo.ContextOfDataPoints h
on g.ContextID=h.ContextID
group by g.ContextID, b.ModuleID, b.ModuleCode, f.DimensionID, f.MemberID
order by b.ModuleCode, g.ContextID
go

select contextID, ID_Taxonomy, dimensionID, memberID
from contextDimensionMemberPair
go

select contextID, ID_Taxonomy, contextDescr, codeTaxonomy
from Context
go
  
```

## 8 Implementation of the DPM in the MDM using the design ROLAP

### 8.1 Introduction

This section is divided in two sections, relational model and the creation of the tables.

## 8.2 Structure ROLAP

Figure 31 shows the relational model of the DPM, through a relational diagram obtained from Management Studio from MS SQL Server.



Figure 31 — Structure of the MDM of the DPM

### 8.3 Creation of the infrastructure through MS SQL Server

This section shows the script of creation of the tables. The first part of this script deletes the tables (all) and then the tables and some object more are created.

```

use DPM_ROLAP
go

IF OBJECT_ID(N'FactTable', N'U') IS NOT NULL
DROP TABLE FactTable;
go

IF OBJECT_ID(N'Period_DPM', N'U') IS NOT NULL
DROP TABLE Period_DPM;
go

IF OBJECT_ID(N'TR_Base_Dimension_Balance_DPM', N'TR') IS NOT NULL
DROP TRIGGER TR_Base_Dimension_Balance_DPM;
go

IF OBJECT_ID(N'Base_Dimension', N'U') IS NOT NULL
DROP TABLE Base_Dimension;
go

IF OBJECT_ID(N'contextDimensionMemberPair', N'U') IS NOT NULL
DROP TABLE contextDimensionMemberPair;
go

IF OBJECT_ID(N'Context', N'U') IS NOT NULL
DROP TABLE Context;
go

IF OBJECT_ID(N'Dimension_DimensionAttribute', N'U') IS NOT NULL
DROP TABLE Dimension_DimensionAttribute;
go

IF OBJECT_ID(N'DimensionAttribute', N'U') IS NOT NULL
DROP TABLE DimensionAttribute;
go

IF OBJECT_ID(N'Dimension', N'U') IS NOT NULL
DROP TABLE Dimension;
go

IF OBJECT_ID(N'Taxonomy', N'U') IS NOT NULL
DROP TABLE Taxonomy;
go

IF OBJECT_ID(N'Framework', N'U') IS NOT NULL
DROP TABLE Framework;
go

create table Framework (
    ID_Framework          int primary key,
    nameFramework          nvarchar(255) not null,
    labelFramework         nvarchar(255) null,
    creationDate            datetime      not null default getdate(),
    modificationDate        datetime      null,
    userID_created          nvarchar(30)  not null default current_user)

```

```

go

create table Taxonomy (
    ID_Taxonomy          int primary key,
    ID_Framework         int          not null references Framework,
    nameTaxonomy         nvarchar(255) not null,
    labelTaxonomy        nvarchar(255) not null,
    creationDate         datetime     not null default getdate(),
    modificationDate     datetime     null,
    valid_from           datetime     not null,
    valid_to             datetime     null,
    versionTax           nvarchar(10) not null,
    versionDate          datetime     null,
    schemaLocation       nvarchar(255) null,
    userid_created       nvarchar(30) not null default current_user)

go

create table Dimension (
    dimensionID          int          not null primary key,
    dimensionCode        nvarchar(10) not null, --Code of approach dimension
    dimensionLabel       nvarchar(255) not null,
    creationDate         datetime     not null default getdate(),
    modificationDate     datetime     null,
    domainID            int          not null,
    isEnumerable         bit          not null default(0), -- by default is enumerable (0),
if not is non-enumerable (1).
    typeData            nvarchar(30),
    valid_from           datetime     not null,
    valid_to            datetime     null
)

go

create table DimensionAttribute(
    memberID            int primary key,
    domainID            int not null,
    memberCode          nvarchar(50) not null,
    memberLabel         nvarchar(255) not null,
    isDefault           bit default(0), -- By default a domain-member is not the default
    creationDate        datetime     not null default getdate(),
    modificationDate     datetime     null,
    valid_from           datetime     not null,
    valid_to            datetime     null
);

go

create table Dimension_DimensionAttribute(
    dimensionID          int not null,
    memberID            int not null,
    constraint PK_Dimension_DimensionAttribute
        primary key (dimensionID, memberID),
    constraint FK_dimensionID foreign key (dimensionID)
        references Dimension,
    constraint FK_memberID foreign key (memberID)
        references DimensionAttribute
);

go

```



```

create table Context (
    contextID int not null,
    ID_Taxonomy int not null,
    contextDescr nvarchar(255) not null,
    codeTaxonomy nvarchar(255) null,
    periodStart datetime null,
    periodEndIntant datetime null,
    scheme nvarchar(255) null,
    constraint PK_Context primary key (contextID, ID_Taxonomy)--,
    --constraint FK_taxonomyID foreign key(ID_Taxonomy)
    -- references Taxonomy
);

create table contextDimensionMemberPair(
    contextID int not null,
    ID_Taxonomy int not null,
    dimensionID int not null,
    memberID int not null,
    constraint PK_contextDimensionMemberPair
        primary key (contextID, ID_Taxonomy, dimensionID, memberID),
    constraint FK_contextDimensionMemberPair_ContextID_ID_Taxonomy
        foreign key (contextID, ID_Taxonomy)
        references Context(contextID, ID_Taxonomy),
    constraint FK_contextDimensionMemberPair_dimensionID
        foreign key (dimensionID, memberID)
        references Dimension_DimensionAttribute(dimensionID, memberID)
)

go

create table Base_Dimension (
    IDprimaryItem int identity(1,1) primary key,
    code nvarchar(10) not null,
    creationDate datetime not null default getdate(),
    modificationDate datetime null,
    valid_from datetime not null default getdate(),
    valid_to datetime null,
    datatype nvarchar(20) not null
        check (DataType in ('String','Monetary','Integer','Numeric')),
    periodType nvarchar(10) not null
        CHECK (PeriodType in ('Instant','Period','Forever')),
    balance nchar(10) null
        check (Balance in ('Credit','Debit')),
    userid_creatednvarchar(30) not null default current_user
)

go

create trigger TR_Base_Dimension_Balance_DPM ON Base_Dimension
after insert, update
as

declare @Balance nchar(10),
        @DataType nvarchar(20),
        @code nvarchar(10)
select @code =code,
       @Balance =balance,
       @DataType =datatype

```

```

from inserted
if @Balance is null and @DataType='Monetary'
begin
    raiserror ('If the DataType is Monetary the Balance attribute can not be NULL
ATTENTION: The PrimaryItem with name: %s is not inserted.', 16, 1, @code)
    rollback transaction
end
go

go
create table Period_DPM(
    IDPeriod int identity (1,1) primary key,
    start_date          datetime    null,
    end_date_Instant    datetime    not null,
    instant_Year         nvarchar(4)  not null,
    instant_month        nvarchar(2)  not null,
    instant_day          nvarchar(2)  not null,
    date_created         datetime    not null          default getdate())

go

create table FactTable(
    IDFact              int          primary key, -- Identification of the DPM or the Fact
    ID_Taxonomy         int not null,
    contextID           int not null,
    IDprimaryItem       int not null,
    unit_simple         nvarchar(10) null, --EUR, PURE, ETC.
    unit_numerator       nvarchar(10) null,
    unit_denominator    nvarchar(10) null,
    accuracy            dec(1)       null, --Decimals value
    numeric_value       dec(17,4)    null,
    string_value        nvarchar(4000) null,
    boolean_value       bit          null,
    date_value          datetime     null,
    is_Null             nchar(1)     null, --CHECK: Y ODER N
    language            nvarchar(40) null,
    userid_created      nvarchar(30) null,
    CONSTRAINT CK_boolean_value_DPM CHECK (boolean_value in (1,0)),--CHECK: Y
ODER N
    CONSTRAINT CK_nil_value_DPM CHECK (is_Null in ('Y','N','y','n')),--CHECK: Y
ODER N
    constraint FK_FactTable_Context_Taxonomy
        foreign Key (contextID, ID_Taxonomy)
        references Context(contextID, ID_Taxonomy),
    constraint FK_FactTable_Taxonomy
        foreign Key (ID_Taxonomy)
        references Taxonomy(ID_Taxonomy),
    constraint FK_FactTable_primaryItem
        foreign Key (IDprimaryItem)
        references Base_Dimension(IDprimaryItem)

```

## 9 DPM of FINREP 2012 in the MDM using the design ROLAP

### 9.1 Introduction

This section is based on the [New 2012 FINREP taxonomy](#), and in research works referenced in [20] [24].

From the referenced page in the Eurofiling website the file 'DataPointsModel.xls' containing a version of the DPM in an excel spreadsheet, can be downloaded. The DPM is obtained from the taxonomy and an example of an XBRL instance document is available, as shown in Figure 32 [20]. The DPM is obtained from the taxonomy, meta data and the Fact Table from an XBRL instance document, Data Points.

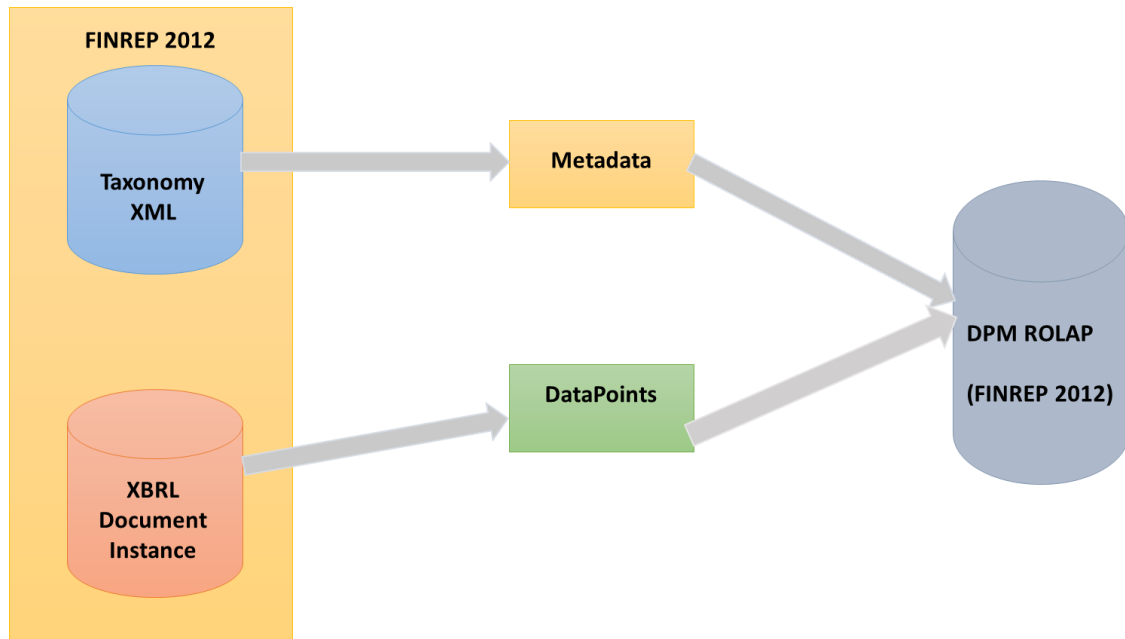


Figure 32 — Process of creation of the *DPM* from the taxonomy *FINREP 2012* and an example of XBRL Document Instance of this taxonomy

### 9.2 DPM of FINREP 2012

The first sheet shows the set of families, including the base dimension, figure 33. However, the *families* are outside of the scope of this document, because this document is more readable and less complex without them.

#	Code	Name	Prefix
0	base	<a href="#">Base items</a>	base
1	CT	<a href="#">Categories</a>	dCT
2	CI	<a href="#">Comprehensive income</a>	dCI
3	AT	<a href="#">Amount types</a>	dAT
4	PL	<a href="#">Portfolios</a>	dPL
5	SE	<a href="#">Sectors</a>	dSE
6	GA	<a href="#">Geographical and political areas</a>	dGA
7	CU	<a href="#">Currencies</a>	dCU
8	TI	<a href="#">Time intervals</a>	dTI
9	CD	<a href="#">Continuing/Discontinued</a>	dCD
10	BA	<a href="#">Before/After</a>	dBA
11	CL	<a href="#">Control</a>	dCL
12	RP	<a href="#">Related parties</a>	dRP
13	RT	<a href="#">Risk types</a>	dRT
14	MA	<a href="#">Markets</a>	dMA
15	RS	<a href="#">Reporting scope</a>	dRS
16	EC	<a href="#">Entity codes</a>	dEC

**Figure 33 — The *families***

The table 12 shows only the *families*.

**Table 12 — DPM in format ROLAP for the constructor *Family***

Code Family
CT
CI
AT
PL
SE
GA
CU
TI
CD
BA
CL
RP
RT
MA
CU
TI
RS
EC

The next sheet, figure 34, shows the set of *base dimension*.

Navi	Base Items	Item type	Name	ID
1	Abstract	Primary item	ad1	base_ad1
1	Assets	Primary item	mi1	base_mi1
2	Assets and liabilities	Primary item	mi2	base_mi2
3	Change in equity	Primary item	md3	base_md3
4	Collateral or other credit enhancement received as security	Primary item	mi4	base_mi4
5	Commitments and financial guarantees given	Primary item	mi5	base_mi5
3	Commitments and financial guarantees received	Primary item	mi3	base_mi3
6	Entity name	Primary item	sd6	base_sd6
7	Entry date	Primary item	dd7	base_dd7
8	Equity	Primary item	mi8	base_mi8
9	Equity and liabilities	Primary item	mi9	base_mi9
10	Expense	Primary item	md10	base_md10
11	Income	Primary item	md11	base_md11
12	Income/(Expense)	Primary item	md12	base_md12
13	Liabilities	Primary item	mi13	base_mi13
14	Removal date	Primary item	dd14	base_dd14
15	Voting rights	Primary item	pi15	base_pi15

Figure 34 — Base dimension

The table 13 shows the constructor *BaseDimension*.

Table 13 — DPM in format ROLAP for the constructor *BaseDimension*

Code BaseDimension
ad1
dd14
dd7
md10
md11
md12
md3
mi1
mi13
mi2
mi3
mi4
mi5
mi8
mi9
pi15
sd6

The next sheet, figure 35, shows only a sheet of a *dimension* and theirs *domain-members*.

Nav	Categories	Item type	Name	ID	Period	Data type
I	Category of assets	Explicit dimension	AS	dim_AS	duration	string
II	Category of assets and liabilities	Explicit dimension	AL	dim_AL	duration	string
III	Category of equity	Explicit dimension	EQ	dim_EQ	duration	string
IV	Category of equity and liabilities	Explicit dimension	EL	dim_EL	duration	string
V	Category of liabilities	Explicit dimension	LI	dim_LI	duration	string
1	Total	Member	x1	dCT_x1	duration	domain
2	Cash	Member	x2	dCT_x2	duration	domain
3	Debt instruments	Member	x3	dCT_x3	duration	domain
4	Debt instruments and other financial	Member	x4	dCT_x4	duration	domain
5	Debt instruments, short positions and other financial	Member	x5	dCT_x5	duration	domain
6	Debt instruments, short positions, derivatives and other financial	Member	x6	dCT_x6	duration	domain
7	Debt securities	Member	x7	dCT_x7	duration	domain
8	Deposits other than redeemable at notice and repurchase agreements	Member	x8	dCT_x8	duration	domain
9	Derivatives	Member	x9	dCT_x9	duration	domain
10	Equity and debt instruments	Member	x10	dCT_x10	duration	domain
11	Equity and debt instruments, and derivatives	Member	x11	dCT_x11	duration	domain
12	Equity and debt instruments, short positions, derivatives and other financial	Member	x12	dCT_x12	duration	domain
13	Equity instruments	Member	x13	dCT_x13	duration	domain
14	Goodwill	Member	x14	dCT_x14	duration	domain
15	Intangible	Member	x15	dCT_x15	duration	domain
16	Intangible other than goodwill	Member	x16	dCT_x16	duration	domain
17	Intangible other than goodwill subject to operating lease	Member	x17	dCT_x17	duration	domain
18	Investment property	Member	x18	dCT_x18	duration	domain
19	Investment property subject to operating lease	Member	x19	dCT_x19	duration	domain
20	Investments in entities accounted for using the equity method	Member	x20	dCT_x20	duration	domain
21	Issued capital	Member	x21	dCT_x21	duration	domain
22	Loans/Deposits	Member	x22	dCT_x22	duration	domain
23	Other financial (different than debt instruments, short positions and derivatives)	Member	x23	dCT_x23	duration	domain

Figure 35 — Sheet of one *dimension* and theirs *domain-members*

The table 14 shows the constructor *Dimension*.

Table 14 — DPM in format ROLAP for the constructor *Dimension*

Code Dimension
AL
AS
AT
BT
CD
CI
CL
CR
CS
DL
EL
EQ
JI
LI
MA
OC
OM
PL

RI
RM
RP
RS
RT

Table 15 shows the constructor *DimensionAttribute*.

**Table 15 — DPM in ROLAP format for the constructor *DimensionAttribute***

Code <i>DimensionAttribute</i>
dAT:x1
dAT:x10
dAT:x11
dAT:x12
dAT:x13
dAT:x14
dAT:x15
dAT:x16
dAT:x17
dAT:x18
dAT:x19
dAT:x2
dAT:x20
dAT:x3
dAT:x4
dAT:x5
dAT:x6
dAT:x7
dAT:x8
dAT:x9
dBA:x1
dBA:x2
... ..

NOTE Only a subset is shown as there are 171 tuples in this constructor.

The table 16 shows the constructor *Relation\_DimensionAttribute*.

Table 16 — DPM in format ROLAP for the constructor Relation\_DimensionAttribute.

dimensionID	memberID
AL	dCT:x1
AL	dCT:x12
AL	dCT:x13
AL	dCT:x22
AL	dCT:x23
AL	dCT:x28
AL	dCT:x38
AL	dCT:x4
AL	dCT:x44
AL	dCT:x7
AL	dCT:x9
AS	dCT:x1
AS	dCT:x10
AS	dCT:x11
AS	dCT:x13
AS	dCT:x14
AS	dCT:x15
AS	dCT:x16
AS	dCT:x17
AS	dCT:x18
AS	dCT:x19
AS	dCT:x2
AS	dCT:x20
AS	dCT:x22
AS	dCT:x26
AS	dCT:x27
AS	dCT:x29
AS	dCT:x3
AS	dCT:x30
AS	dCT:x39
AS	dCT:x40
AS	dCT:x41
AS	dCT:x42
AS	dCT:x44
AS	dCT:x7
AS	dCT:x9



AT	dAT:x1
AT	dAT:x10
AT	dAT:x11
AT	dAT:x12
AT	dAT:x13
DL	dTI:gt180d_le1y
DL	dTI:gt1y
DL	dTI:gt90d_le180d
DL	dTI:x1
EL	dCT:x1
EQ	dCT:x1
EQ	dCT:x21
EQ	dCT:x34
EQ	dCT:x35
EQ	dCT:x36
EQ	dCT:x37
EQ	dCT:x43
JI	dGA:emu
JI	dGA:x2
JI	dGA:x4
LI	dCT:x1
LI	dCT:x22
LI	dCT:x23
LI	dCT:x24
LI	dCT:x25
LI	dCT:x31
LI	dCT:x32
...	... ..

NOTE Only a subset is shown as there are 203 tuples in this constructor.

Table 17 shows the constructor *ContextDimensionMemberPair*. In this case the attribute 'taxonomy' is taken out because in this example there is only FINREP.

**Table 17 — DPM in format ROLAP for the constructor *ContextDimensionMemberPair***

contextID	dimensionID	memberID
e_x11_x3_emu_eur_x10_x1	AS	dCT:x11
e_x11_x3_emu_x2_x10_x1	AS	dCT:x11

e_x11_x3_eu_x10_x1	AS	dCT:x11
e_x11_x3_x10_x1	AS	dCT:x11
e_x11_x3_x2_eur_x10_x1	AS	dCT:x11
e_x11_x3_x2_x2_x10_x1	AS	dCT:x11
e_x11_x3_x4_x10_x1	AS	dCT:x11
e_x13_x3_emu_x14_eur_x16_x1	AS	dCT:x13
e_x13_x3_emu_x14_x2_x16_x1	AS	dCT:x13
e_x13_x3_eu_x14_x16_x1	AS	dCT:x13
e_x13_x3_x14_x16_x1	AS	dCT:x13
e_x13_x3_x2_x14_eur_x16_x1	AS	dCT:x13
e_x13_x3_x2_x14_x2_x16_x1	AS	dCT:x13
e_x13_x3_x4_x14_x16_x1	AS	dCT:x13
e_x17_x3_x21_x1	AS	dCT:x17
e_x17_x3_x3_x1	AS	dCT:x17
e_x17_x3_x6_x1	AS	dCT:x17
e_x19_x3_x21_x1	AS	dCT:x19
... ..	...	... ..

NOTE Only a subset is shown as there are 1278 tuples in this constructor.

The table 18 shows the constructor *Context*.

**Table 18 — DPM in format ROLAP for the constructor *Context*.**

contextID	periodEndIntant	entity
e_x11_x3_emu_eur_x10_x1	2011-06-12	abc
e_x11_x3_emu_x2_x10_x1	2011-06-12	abc
e_x11_x3_eu_x10_x1	2011-06-12	abc
e_x11_x3_x10_x1	2011-06-12	abc
e_x11_x3_x2_eur_x10_x1	2011-06-12	abc
e_x11_x3_x2_x2_x10_x1	2011-06-12	abc
e_x11_x3_x4_x10_x1	2011-06-12	abc
e_x13_x3_emu_x14_eur_x16_x1	2011-06-12	abc
e_x13_x3_emu_x14_x2_x16_x1	2011-06-12	abc
e_x13_x3_eu_x14_x16_x1	2011-06-12	abc
... ..	... ..	...

NOTE Only a subset is shown as there are 237 tuples in this constructor.

The table 19 shows the constructor *FactTable*.

Table 19 — DPM in format ROLAP for the constructor *FactTable*.

IDFact	ID_Taxonomy	contextID	IDprimaryItem	unit_simple	accuracy	numeric_value
1	FINREP	e_x7_x20_x14_eq0d_x11_x1	mi1	EUR	0	5
2	FINREP	e_x7_x20_x14_gt0d_le90d_x11_x1	mi1	EUR	0	5
3	FINREP	e_x7_x20_x14_gt90d_le180d_x11_x1	mi1	EUR	0	5
4	FINREP	e_x7_x20_x14_gt180d_le1y_x11_x1	mi1	EUR	0	5
5	FINREP	e_x7_x20_x14_gt1y_x11_x1	mi1	EUR	0	5
6	FINREP	e_x7_x20_x2_eq0d_x11_x1	mi1	EUR	0	1
7	FINREP	e_x7_x20_x2_gt0d_le90d_x11_x1	mi1	EUR	0	1
8	FINREP	e_x7_x20_x2_gt90d_le180d_x11_x1	mi1	EUR	0	1
9	FINREP	e_x7_x20_x2_gt180d_le1y_x11_x1	mi1	EUR	0	1
10	FINREP	e_x7_x20_x2_gt1y_x11_x1	mi1	EUR	0	1
11	FINREP	e_x7_x20_x5_eq0d_x11_x1	mi1	EUR	0	1
12	FINREP	e_x7_x20_x5_gt0d_le90d_x11_x1	mi1	EUR	0	1
13	FINREP	e_x7_x20_x5_gt90d_le180d_x11_x1	mi1	EUR	0	1
14	FINREP	e_x7_x20_x5_gt180d_le1y_x11_x1	mi1	EUR	0	1
15	FINREP	e_x7_x20_x5_gt1y_x11_x1	mi1	EUR	0	1
16	FINREP	e_x7_x20_x4_eq0d_x11_x1	mi1	EUR	0	1
17	FINREP	e_x7_x20_x4_gt0d_le90d_x11_x1	mi1	EUR	0	1
18	FINREP	e_x7_x20_x4_gt90d_le180d_x11_x1	mi1	EUR	0	1
19	FINREP	e_x7_x20_x4_gt180d_le1y_x11_x1	mi1	EUR	0	1
20	FINREP	e_x7_x20_x4_gt1y_x11_x1	mi1	EUR	0	1
21	FINREP	e_x7_x20_x12_eq0d_x11_x1	mi1	EUR	0	1
22	FINREP	e_x7_x20_x12_gt0d_le90d_x11_x1	mi1	EUR	0	1
--	FINREP	--- --- --- ---	---	---	-	---

NOTE Only a subset is shown as there are 237 tuples in this constructor.

## 10 DPM of the first prototype of Solvency II in the MDM using the design ROLAP

### 10.1 Introduction

This section is based on an idea or concept of the Solvency II taxonomy ('2012-07-01-mdt.rar'). As this taxonomy is very simple at the time of writing this document, it is possible that it can help the reader of this document to understand the structure of the DPM better [24] [20].

The DPM in this case [24] is obtained from the taxonomy and an example of XBRL instance document is shown in Figure 36. The DPM is obtained from the *taxonomy*, *Metadata*, and the *Fact Table* from an *XBRL instance document*, *Data Points*.

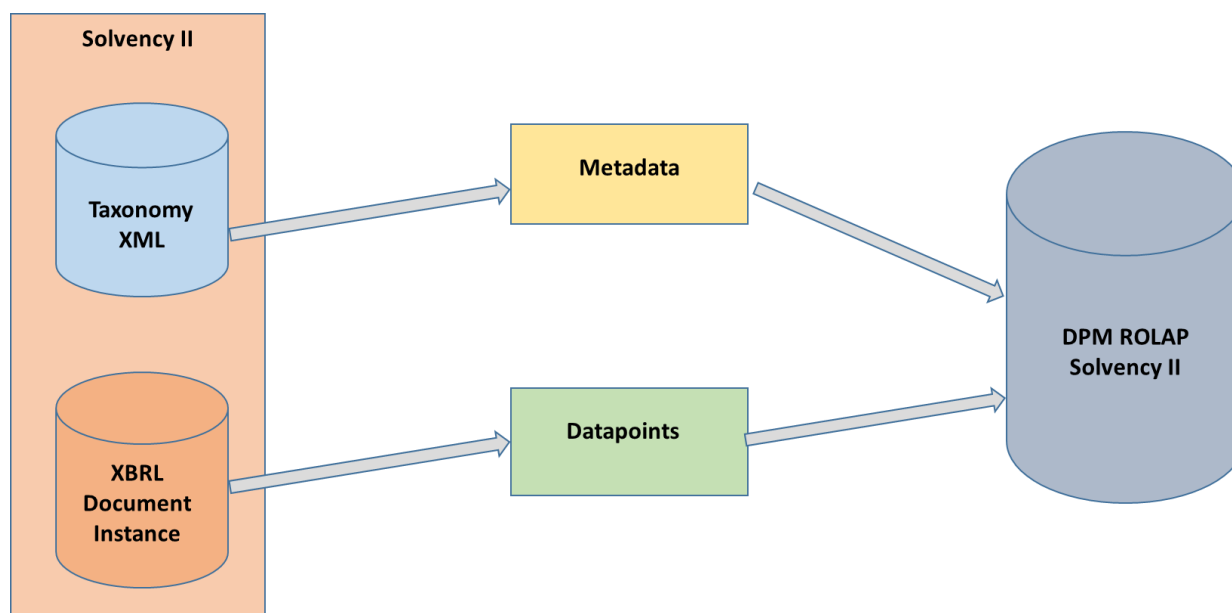


Figure 36 — Process of creation of the *DPM* from the taxonomy *Solvency II* and an example of XBRL Document Instance of this taxonomy

### 10.2 DPM of the prototype

Table 20 shows the constructor *BaseDimension*.

Table 20 — DPM in format ROLAP for the constructor *BaseDimension*

Code BaseDimension
a1
A10A
A10B
A11
A12
A13
A14
A14A
A16
A17

A18
A18A
A19
A19A
A2
A20
A21
A23
A25B
A26
A27
A29
A3
A30
A4
A5
A6
A7
A7A
A8
A8A
A8C
A8D
A9
AS1
AS10A
AS10B
AS11
AS12
AS13
AS14
---

NOTE Only subset is shown since the number of tuples of the *BaseDimension* is 140 (*primary items*).

Table 21 shows the constructor *Dimension*.

**Table 21 — DPM in format ROLAP for the constructor *Dimension***

Code Dimension
PeriodicityDimension
SoloOrGroupDimension

Table 22 shows the constructor *DimensionAttribute*.

**Table 22 — DPM in format ROLAP for the constructor *DimensionAttribute***

Code DimensionAttribute
per:AdHoc
per:Quarterly
per:Yearly
soc:Group
soc:Solo

Table 23 shows the constructor *Relation\_DimensionAttribute*.

**Table 23 — DPM in format ROLAP for the constructor *Relation\_DimensionAttribute***

dimensionID	memberID
PeriodicityDimension	per:AdHoc
PeriodicityDimension	per:Quarterly
PeriodicityDimension	per:Yearly
SoloOrGroupDimension	soc:Group
SoloOrGroupDimension	soc:Solo

Table 24 shows the constructor *ContextDimensionMemberPair*. In this case the attribute *taxonomy* is taken out because there is only one taxonomy: Solvency II.

**Table 24 — DPM in format ROLAP for the constructor *ContextDimensionMemberPair***

contextID	dimensionID	memberID
Context_Instant_Quarterly_Solo	PeriodicityDimension	per:Quarterly
Context_Instant_Yearly_Solo	PeriodicityDimension	per:Yearly
Context_Instant_Quarterly_Solo	SoloOrGroupDimension	soc:Solo
Context_Instant_Yearly_Solo	SoloOrGroupDimension	soc:Solo

Table 25 shows the constructor *Context*.

**Table 25 — DPM in format ROLAP for the constructor *Context***

contextID	periodEndInstant	entity
Context_Instant_Quarterly_Solo	2012-06-30	123456
Context_Instant_Yearly_Solo	2012-06-30	123456

Table 26 shows the constructor *FactTable*.

**Table 26 — DPM in format ROLAP for the constructor *FactTable***

IDFact t	ID_Taxonomy	contextID	IDprimary type	unit_simpl e	accuracy	numeric_value
1	Solvency II	Context_Instant_Quarterly_Solo	AS17A	EURO	0	42000
2	Solvency II	Context_Instant_Quarterly_Solo	AS18	EURO	0	29655
3	Solvency II	Context_Instant_Quarterly_Solo	AS17	EURO	0	12345
4	Solvency II	Context_Instant_Yearly_Solo	AS18	EURO	0	69000
5	Solvency II	Context_Instant_Yearly_Solo	AS17	EURO	0	666
6	Solvency II	Context_Instant_Yearly_Solo	AS17A	EURO	0	100000

## Bibliography

- [1] BUILDW. Building the Data Warehouse. Inmon W. H, 4th Edition. John Wiley & Sons 2005.
- [2] DWTOOLKIT. The Data Warehouse Toolkit series. Kimball R. 2004. John Wiley & Sons 1996-2004.
- [3] FUNDAMENTALSDW. Fundamentals of Data Warehouses. Jarke M., Lenzerini M., Vassiliou Y. and Vassiliadis P. 2nd edition, 2003S, Springer.
- [4] KIMBALLGROUP. Kimball Group, 2013. [Kimball Group](#)
- [5] DWINSTITUTE. Data Warehouse Institute, 2013, [TDWI](#).
- [6] XBRL21. Extensible Business Reporting Language (XBRL) 2.1. Engel P, Hamscher W., Shuetrim G., Vun Kannon D., Wallis H. July 2nd, 2008. XBRL International. [Extensible Business Reporting Language \(XBRL\) 2.1](#).
- [7] DMMATRIXSCHEMA. Data Model and Matrix Schemas. Schmehl K. November 16th, 2009. XI European Banking Supervisor, XBRL Workshop hosted by the Oesterreichische Nationalbank, Vienna. [XI European Banking Supervisors XBRL Workshop](#)
- [8] XBRL\_MDM. XBRL and the Multidimensional Data Model. Santos I, Castro E. In Proceedings of the 7th International Conference on Web Information Systems and Technologies, WEBIST 2011, pages 161-164, Noordwijkerhout. The Netherlands, May 6th-9th, 2011.
- [9] XBRLINTEROPERABILITY. XBRL Interoperability through a Multidimensional Data Model. Santos I, Castro E. IADIS International Conference on Internet Technologies & Society (ITS2011), Shanghai, China. December 8th-10th, 2011.
- [10] XBRLDIM. XBRL Dimensions 1.0 XBRL International. Hernandez-Ros I, Wallis H. April 26th, 2006. <http://www.xbrl.org/Specification/XDT-CR3-2006-04-26.rtf>.
- [11] EURXBRLTAXONARCHIT. European XBRL Taxonomy Architecture V2.0. Declerck T, Homes R, Heinze K, 2013. CEN Workshop Agreement. [European XBRL Taxonomy Architecture V3.0](#).
- [12] COMPLEXMODELS. A vision for management of complex models. Bernstein P A, Halevy A Y, Pottinger RA. SIGMOD Record 29 (4), 2000, 55-63.
- [13] TSIMMIS. The TSIMMIS Project: Integration of heterogeneous information sources. Chewathe S, García-Molina H, Hammer J, Ireland K, Papakonstantinou Y, Ullman J, and Widom J. In Proc. 10th Meeting of the Information Processing Society of Japan, pages 7-18, 1994.
- [14] QUERING. Querying heterogeneous information sources using source descriptions. Levi A, Rajaraman A, and Ordille J. VLDB'96, Proceedings of Twenty-second International Conference on Very Large Data Bases.
- [15] MODELTRANSFORMATION. Model Transformation by Graph Transformation: A comparative Study. Taentzer G, Ehrig K, Guerra E, Lara J, Lengyel L, Levendovszky T, Prange U, Varro D, and Varro-Gaypay S. Model Transformation in Practice Workshop 2005 (MIIP2005).
- [16] REQUIEBA. Update on the technical standards on supervisory reporting requirements. EBA. 2013. <http://www.eba.europa.eu/-/update-on-the-technical-standards-on-supervisory-reporting-requirements>.
- [17] DPMVERSUSMDM. Data Point Model (DPM) versus Multidimensional Data Model (MDM), Santos I. Contribution for DPM chapter in CEN WS XBRL Plenary Session, Dublin, April 19th 2013. Hosted by Central Bank of Ireland.



- [18] ACADEMY. Academy works. 2013. Openfiling. [Openfiling Academy](#).
- [19] CMLM. Conceptual and Logical Models in the Design of Economic and Financial Reports Using the XBRL Specification. Santos I, Castro E, Velasco M. 2013. In the journal Business & Information Systems Engineering (BISE), under review.
- [20] POC1. Proof of concept of mapping a XBRL report versus a RDBMS. Santos I, Castro E. Openfiling 1st General Assembly, organized by XBRL Operational Network of the European Banking Authority, and hosted by Bank of Italy. September 5th, 2011. Banca d'Italia, Rome, Italy. [http://www.openfiling.info/?page\\_id=286](http://www.openfiling.info/?page_id=286).
- [21] XERE. Xere: Towards a Natural Interoperability between XML and ER Diagrams. Della Penna G, Di Marco A, Intrigila B, Melatti I, and Pierantonio A- Lecture Notes in computer Science, volume 2621 2003, pp 356-371. Book: Fundamental Approaches to software Engineering.
- [22] EEURDPM. European Data Point Methodology V2.0. Declerk T, Hommes R, Heinze K. 2013. CEN Workshop Agreement. [European Data Point Methodology V2.0](#).
- [23] EURFILNRULES. European Filing Rules. Declerk T, Hommes R, Heinze K. 2013. CEN Workshop Agreement. [European Filing Rules](#).
- [24] AUTOMATION. Automation and mapping from the data model of the XBRL specification in Database. León Y. 2012. Final Project of the Polytechnic School of the Carlos III University of Madrid, Spain. Date: October 4th, 2012. Tutors: Santos I, Castro E. [Automation and mapping](#)
- [25] MULTIDIMENSIONALCOREP. New Technical and Normative Challenges for XBRL: Multidimensional in the COREP Taxonomy. Boixo I, Flores F. July 18th, 2005. The International Journal of Digital Accounting Research, Vol. 5, N. 9, 2005, pages 79-104. ISSN:1577-8517.
- [26] CEN WS XBRL: Open Working Area. 2013. [XBRL Spain Main page](#)
- [27] XBRL Meta-metadata Model. Valencia J. 2011. Final project of Computer Engineering Technology Management, Carlos III University of Madrid. Date September 27th, 2011. Tutors: Santos I, Castro E. [XBRL Meta-metadata Model](#).