International
Virtual
Observatory
Alliance

Model Instances in Votables
Version 1.0

IVOA Working Draft 2021-10-12

Working group
DM
This version
http://www.ivoa.net/documents/merged-syntax/20211012
Latest version
http://www.ivoa.net/documents/merged-syntax
Previous versions
This is the first public release
Author(s)
François Bonnarel, Gilles Landais, Laurent Michel, Jesus Salgado, Gerard Lemson
Editor(s)
Laurent Michel, Mark Cresitello Dittmar

Abstract

Vodml-instance-vot proposes a syntax to map VOTable data on any model serialized in VO-DML. Vodml-instance-vot annotations are grouped in a single XML block located in the VOTable head. The annotation block allows to easily reconstruct the model structure. It designed in a way that the block can be reused on different data sets in order to facilitate the annotation process. Vodml-instance-vot is enable to join data from different tables

Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

Acknowledgments

CDS/TDIG/SourceDM contributors

Conformance-related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (**?**).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

The first purpose of a model is to provide, for a particular domain, a formal description of the relevant quantities and of the way they are connected together . This documentary role facilitates the communication between the stack-holders and thus the design of interoperability protocols.

At data level, interoperability consists in arranging searched data in a way that a client can understand them without taking care of their origin. So that, the same code can process and compare data coming from different sources. That way to arrange data is given by the model.

This is not done by default with VOtables because VOTables are containers (**?**). The VOTable schema cannot say how data are mapped on a given model or whether they match any model at all. This is not an issue for simple protocol responses (ref) because the VOTable structure is defined by the protocol itself. This is however a big issue for VOTables containing native data such as Vizier or TAP query responses.

The challenge here is to bind native data with a given model in a way that a model aware software can see them as model instances while maintaining the possibility to access them in their original forms.

This is partially done with UTypes which may connect FIELDs or PARAMs with model leaves in the case of simple tree-views of the model. Unfortunately, there is nos unique way to build and parse UTypes in the context of complex models. This occurs when e.g the same class is used in different location of the model or when the model contains loops. It is also not possible to refer data from different tables with UTypes.

The landscape has dramatically changed in 2016 when VODML (**?**) became a recommendation. VODML is a meta-model that gives a standard way to express VO models and to make them machine-readable. In VODML, model leaves are no longer identified by a simple string like UTypes do but by a certain role played in a given location in the model hierarchy. The consequence is that any annotation mechanism based on VODML will preserve the model hierarchy to save the role played by any components. In this context, it might be easy to re-construct model instances from the annotations.

The main concept of this standard is to insert on the top of the VOTable an XML block compliying with the model structures and containing references to the actual data. In such a way that a model-aware client only has to make a copy of that structure and to resolve the references to build an instance. More generic model-unaware clients can just ignore the mapping block. This approach, proposed by **?**, allows a perfect restitution of the model from the annotation, a round-trip validation. It follows a real ORM schema actually. Our approach is a bit different. From our use case perspective (see below), clients do not need to care about the difference between data types and object types or between relations and compositions or some

other finesses. They need to be able to reconstruct a browable data hierarchy. The can be done by assembling key/vakues pairs, tuples and arrays. This way to serialize complex data is used with a great success by most of the Web applications working with JSON/YAML messages. Our bet is that the loss of certain features of the model will allow significant gain in readability, and thus in reliability, while facilitating the work of annotation. The proposed syntax renders the data hierarchy with three elements sibling to the JSON concepts (ATTRIBUTE as key/vakue pair, INSANCE as tuple and COLLECTIONS as arrays). In addition to this, some other elements have been added to guide the parse. The connection with the data is made with element attributes in order to keep the structure of the XML elements independant from the data layout.

These ideas were first tested first in the framework of the TDIG on VOTABLEs containing time series provided by different missions such as Gaia or ZWICKI (**?**). Then, the syntax has been refined to be used to validate the Mango (in press) model on real data.

1.1 Role within the VO Architecture

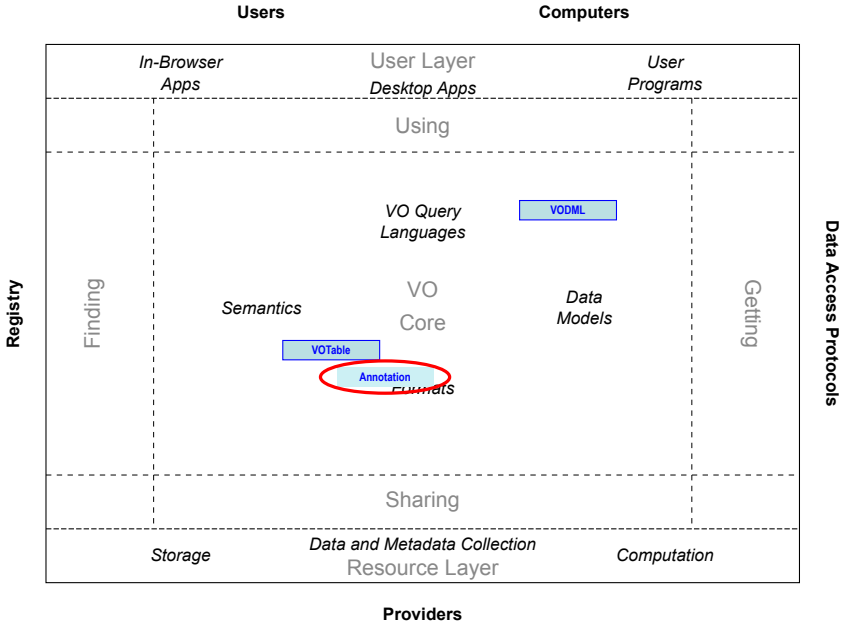


Figure 1: Architecture diagram for this document

Fig. 1 shows the role this document plays within the IVOA architecture (**?**).

2 Use Cases and Requirements

2.1 Use Cases

2.2 Requirements

3 Relation to VOTable

The data model annotation will reside within the scope of a VOTABLE. *Describe which version(s) of VOTable should this be expected to work with? V1.3+*

Location

The mapping block:

- MUST be contained in a VOTable RESOURCE with `type="meta"`.
- which MUST be the first child of a RESOURCE with `type="results"`.
- there MUST be no more than one mapping block per 'results' RESOURCE.

The scope of the mapping block is the whole content of the 'results' RESOURCE.

Namespace

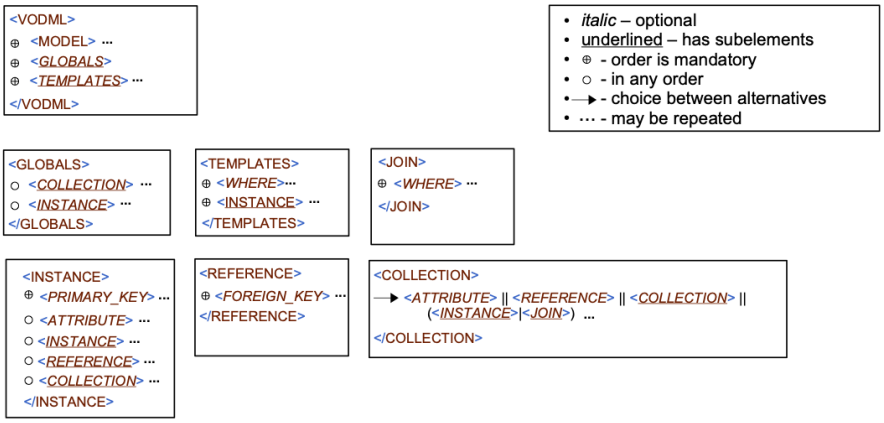
The `dm-mapping` name space isolates VOTable elements from mapping elements, and MUST be defined on the mapping block.

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3">
  <RESOURCE type="results">
    <RESOURCE type="meta">
      <dm-mapping:VODML xmlns:dm-mapping="http://www.ivoa.net/xml/merged-syntax">
        ...
      </dm-mapping:VODML>
    </RESOURCE>
    <TABLE name="myDataTable">
      ....
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

Listing 1: Mapping block in a VOTable

4 Syntax

Element Hierarchy



Attribute Summary

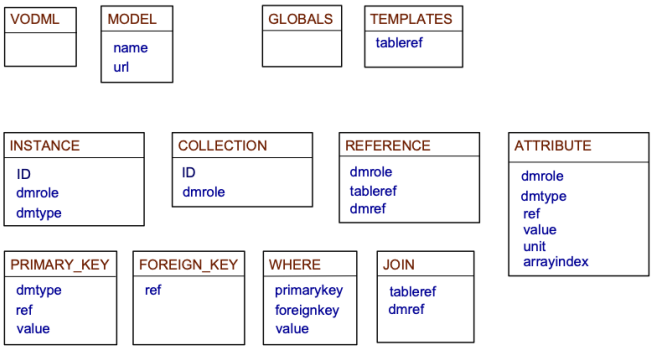


Figure 2: Annotation Syntax Summary

4.1 VODML

The VODML element is the top level container for the mapping eleements for a single VOTable RESOURCE.

```
<dm-mapping:VODML>
  <dm-mapping:MODEL> ... </dm-mapping:MODEL>
  <dm-mapping:GLOBALS> ... </dm-mapping:GLOBALS>
  <dm-mapping:TEMPLATES> ... </dm-mapping:TEMPLATES>
   ...
</dm-mapping:VODML>
```

Listing 2: Example VODML mapping block

Element	Position	Cardinality
MODEL	1	1-*
GLOBALS	2	0-*
TEMPLATES	3	0-*

Table 1: Allowed children for VODML

4.2 MODEL

A VOTable can provide serializations for an arbitrary number of data model types. In order to declare which models are represented in the file, data providers must declare them through the MODEL elements. Only models that are used in the file must be declared. A model is used if at least one element in the mapping block refer to it. In other terms, only models that define vodml-ids used in the annotation must be declared.

```
<dm-mapping:VODML>
  <dm-mapping:MODEL name="sample-ext"
                    url="https://www.myorg.net/models/SampleExt-v1.0.vo-dml.xml" />
  <dm-mapping:MODEL name="sample" url="https://www.ivoa.net/xml/DNE/Sample-v1.0.vo-dml.xml" />
  <dm-mapping:MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
</dm-mapping:VODML>
```

Listing 3: Example MODEL mapping block

Attribute	Role
@name	Name of the mapped model as declared in the VODML/XML model serialization. This attribute MUST not be empty and forms the prefix used in dmrole/dmtype tags of elements from that model.
@url	URL to the vo-dml serialization of the model. If present, this attribute MUST not be empty.

Table 2: MODEL attributes

@name	@url	Pattern
MAND	OPT	Unique attribute pattern supported by `MODEL`

Table 3: Valid attribute patterns for `MODEL`

4.3 GLOBALS

The `GLOBALS` block holds singular data model instances, and is not associated with a VOTable `TABLE`.

The contained instances have a global scope, and may be referenced by other instances anywhere in the mapping block. `INSTANCE` attributes within the `GLOBALS` scope may only refer to VOTable `PARAMS` or contain explicit values, they MUST NOT refer to VOTable `FIELDS`. Note: This rule is not enforced via the XSD schema which is restricted to the mapping block only.

Related instances may be grouped within a `COLLECTION` block to enable selection via the ORM elements provided in this syntax. See `COLLECTION` for more details.

```
<dm-mapping:VODML>
  <dm-mapping:MODEL name="sample" url="https://www.ivoa.net/xml/DNE/Sample-v1.0.vo-dml.xml" />
  <dm-mapping:MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
  <dm-mapping:GLOBALS>
    <!-- a collection of coordinate systems -->
    <dm-mapping:COLLECTION dmid="\_CoordinateSystems" >
      <dm-mapping:INSTANCE dmid="\_timesys" dmtype="sample:TimeSys">
        ...
      </dm-mapping:INSTANCE>
      <dm-mapping:INSTANCE dmid="\_spacesys" dmtype="sample:SpaceSys">
        ...
      </dm-mapping:INSTANCE>
    </dm-mapping:COLLECTION>

    <!-- a singular, stand-alone instance -->
    <dm-mapping:INSTANCE dmtype="sample:Thing">
      <dm-mapping:ATTRIBUTE dmrole="sample:Thing.name" dmtype="ivoa:string" value="MyThing"/>
      <dm-mapping:ATTRIBUTE dmrole="sample:Thing.date" dmtype="ivoa:string" ref="\_date"/>
      ...
    </dm-mapping:INSTANCE>
  </dm-mapping:GLOBALS>
</dm-mapping:VODML>
```

Listing 4: Example `GLOBALS` block

Element	Position	Cardinality
`INSTANCE`	Any	0-*
`COLLECTION`	Any	0-*

Table 4: Allowed children for `GLOBALS`

4.4 TEMPLATES

The `TEMPLATES` block defines a template for deriving multiple data model instances, one for each row of the associated VOTable `TABLE`. A subset of the associated `TABLE` rows may be selected using the `WHERE` syntax element.

```
<dm-mapping:VODML>
  <dm-mapping:MODEL name="sample" url="https://www.ivoa.net/xml/DNE/Sample-v1.0.vo-dml.xml" />
  <dm-mapping:MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
  <dm-mapping:TEMPLATES tableref='Results'>
    <!-- instance template -->
    <dm-mapping:INSTANCE dmtype="sample:Thing">
      <dm-mapping:ATTRIBUTE dmrole="sample:Thing.name" dmtype="ivoa:string" ref="\_field1"/>
      <dm-mapping:ATTRIBUTE dmrole="sample:Thing.date" dmtype="ivoa:string" ref="\_field2"/>
      ...
    </dm-mapping:INSTANCE>
  </dm-mapping:TEMPLATES>
</dm-mapping:VODML>
```

Listing 5: Example `TEMPLATES` block

Attribute	Role
@tableref	ID of the mapped VOTable `TABLE`.

Table 5: `TEMPLATES` attributes

@tableref	Pattern
OPT	If @tableref is not present, `TEMPLATES` maps the first `TABLE` of the `RESOURCE`

Table 6: Valid attribute patterns for `TEMPLATES`

Element	Position	Cardinality	
`WHERE`	1	0-*	The mapping applies to rows matching the `WHERE` condition only
`INSTANCE`	2	0-*	Mapped instance templates

Table 7: Allowed children for `TEMPLATES`

4.5 COLLECTION

COLLECTION is a container element. It is used in different contexts, each allowing a limited subset of elements for its content.

1. As child of INSTANCE

   The COLLECTION serves as a container for elements with multiplicity > 1.
   Examples of usage in this context would be:

   - an array attribute
   - a reference relation with multiplicity > 1
   - a composition relation with multiplicity > 1

2. As child of GLOBALS

   The COLLECTION serves as a proxy for TABLE, grouping common INSTANCES for selection by PRIMARY/FOREIGN_KEY. Examples of usage in this context would be:

   - a set of photometry filters, which apply to various rows of a photometric data table, based on the value of the 'band' column.
   - a set of Dataset metadata instances, which apply to various rows of a photometric data table, based on the value of the 'band' column.

3. As child of COLLECTION

   The use-case for this is unclear

```
<dm-mapping:INSTANCE dmtype="model:Thing">
    <dm-mapping:COLLECTION dmrole="model:Thing.elems">
        <dm-mapping:ATTRIBUTE dmtype="model:Foo" value="100" />
        <dm-mapping:ATTRIBUTE dmtype="model:Foo" value="110" />
    </dm-mapping:COLLECTION>
</dm-mapping:INSTANCE>
```

Listing 6: Example of COLLECTION child of INSTANCE

```
<dm-mapping:GLOBALS>
    <dm-mapping:COLLECTION dmid="_filters" >
        <dm-mapping:INSTANCE dmtype="model:PhotometryFilter" >
            <dm-mapping:PRIMARY_KEY dmtype="ivoa:string" value="RP"/>
            <dm-mapping:ATTRIBUTE dmrole="model:PhotometryFilter.name" dmtype="ivoa:string"
                                  value="GAIA/GAIA2r.Grp"/>
        </dm-mapping:INSTANCE>
        <dm-mapping:INSTANCE dmtype="model:PhotometryFilter" >
            <dm-mapping:PRIMARY_KEY dmtype="ivoa:string" value="BP"/>
            <dm-mapping:ATTRIBUTE dmrole="model:PhotometryFilter.name" dmtype="ivoa:string"
                                  value="GAIA/GAIA2r.Gbp"/>
        </dm-mapping:INSTANCE>
    </dm-mapping:COLLECTION>
<dm-mapping:GLOBALS>
```

Listing 7: Example of COLLECTION child of GLOBALS

Attribute	**Role**
@dmid	Element dmid, MUST be unique within the document.
@dmrole	Role of the `COLLECTION` in the data model.

Table 8: `COLLECTION` attributes

Context	@ID	@dmrole	Pattern
1	OPT	MAND	The element maps a collection playing a role in a modeled `INSTANCE`. @dmrole MUST not be empty. If present, @dmid MUST not be empty.
2	MAND	NO	The collection, has no role. MUST have non-empty dmid to reference for ORM selection of contained `INSTANCE`.

Table 9: Valid attribute patterns for `COLLECTION`

Context: Child of INSTANCE			
Element	**Position**	**Cardinality**	
`ATTRIBUTE`	Only	0-*	Collection of attributes.
`REFERENCE`	Only	0-*	Collection of references.
`INSTANCE` and/or `JOIN`	Any	0-*	Collection of instances.
`COLLECTION`	Only	0-*	Collection of collections.
Context: Child of GLOBALS			
Element	**Position**	**Cardinality**	
`INSTANCE`	Only	0-*	Collection of related instances.

Table 10: Allowed children for `COLLECTION`

4.6 INSTANCE

Mark proposal (as interpreted by LM

The INSTANCE element defines a complex ObjectType or DataType.

```
<dm-mapping:INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceFrame">
        <dm-mapping:INSTANCE dmrole="coords:SpaceFrame.refPosition"
                             dmtype="coords:StdRefLocation">
                <dm-mapping:ATTRIBUTE dmrole="coords:StdRefLocation.position"
                                      dmtype="ivoa:string" value="NoSet" />
        </dm-mapping:INSTANCE>
        <dm-mapping:ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame"
                              dmtype="ivoa:string" value="ICRS" />
        <dm-mapping:ATTRIBUTE dmrole="coords:SpaceFrame.equinox"
                              dmtype="coords:Epoch" value="2015" />
</dm-mapping:INSTANCE>
```

Listing 8: Example of INSTANCE child of GLOBALS

Attribute	Role
@dmid	Element dmid, MUST be unique within the mapping block
@dmrole	INSTANCE role in the DM
@dmtype	Class name

Table 11: `INSTANCE` attributes

It may be a child of several other elements, and the requirements on the content (especially dmid and dmrole), may differ depending on the usage:

- Child of GLOBALS: In this case the INSTANCE is a single stand-alone instance which may or may not be referenced by other INSTANCEs.
  - must have dmid, as possible target of REFERENCE.ref
  - must have no or empty `dmrole`
- Child of TEMPLATES: In this case, the INSTANCE is a template for instances which are generated once per row of the associated table.
  - may have dmid, as target of JOIN.dmref
  - must have no or empty dmrole `dmrole`
- Child of COLLECTION: There are 2 uses for this pattern.
  - each member INSTANCE is a target for selection using the PRIMARY/FOREIGN_KEY elements. This pattern is only allowed within the GLOBALS environment. In this case:
    - must contain at least one PRIMARY_KEY sub-element
    - must have dmid, as possible target of REFERENCE.ref
    - must have no or empty dmrole
  - Elements INSTANCE are collection cells with multiplicity > 1 Each one has:
    - must have dmid, as possible target of REFERENCE.ref. this pattern is only allowed if within the GLOBALS environment
    - must have no or empty dmrole
  - Child of INSTANCE: In this case, each INSTANCE represents a complex ObjectType or DataType playing a role in the parent INSTANCE.
    - must not have dmid (may not be referenced) ??
    - must have non-empty dmrole
  - any INSTANCE:
    - if dmid is present, it must not be empty
    - must have non-empty dmtype

Element	Position	Cardinality	
PRIMARY_KEY	First	0-*	Primary key to be used to in a JOIN context.
REFERENCE	Any	0-*	Object attribute as a reference to either another INSTANCE or a COLLECTION.
INSTANCE	Any	0-*	Object attribute as a class instance.
ATTRIBUTE	Any	0-*	Object attribute as a simple attribute.
COLLECTION	Any	0-*	Object attribute as a collection.

Table 12: Allowed children for INSTANCE

Original

VO-DML structured types are annotated by using the INSTANCE element. Note that there is no difference, from a schema point of view, between ObjectTypes and DataType.

```
<dm-mapping:INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceFrame">
        <dm-mapping:INSTANCE dmrole="coords:SpaceFrame.refPosition"
                             dmtype="coords:StdRefLocation">
                <dm-mapping:ATTRIBUTE dmrole="coords:StdRefLocation.position"
                                      dmtype="ivoa:string" value="NoSet" />
        </dm-mapping:INSTANCE>
        <dm-mapping:ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame"
                              dmtype="ivoa:string" value="ICRS" />
        <dm-mapping:ATTRIBUTE dmrole="coords:SpaceFrame.equinox"
                              dmtype="coords:Epoch" value="2015" />
</dm-mapping:INSTANCE>
```

Listing 9: INSTANCE child of GLOBALS

Attribute	Role
@dmid	Element dmid, MUST be unique within the mapping block
@dmrole	INSTANCE role in the DM
@dmtype	Class name

Table 13: INSTANCE attributes

@dmid	@dmrole	@dmtype	Pattern
MAND	NO or EMPTY	MAND	MUST be applied when the `INSTANCE` is child of `GLOBALS`. The element has no role because it is not embedded in a model mapping block. It must be referable by a `REFERENCE`
OPT	MAND	MAND	MUST be applied in any other location. It may be referable a `REFERENCE` .

Table 14: Valid attribute patterns for `INSTANCE`

Element	Position	Cardinality	
`PRIMARY_KEY`	First	0-*	Primary key to be used to in a JOIN context.
`REFERENCE`	Any	0-*	Object attribute as a reference to either another INSTANCE or a COLLECTION.
`INSTANCE`	Any	0-*	Object attribute as a class instance.
`ATTRIBUTE`	Any	0-*	Object attribute as a simple attribute.
`COLLECTION`	Any	0-*	Object attribute as a collection.

Table 15: Allowed children for `INSTANCE`

4.7 ATTRIBUTE

Attribute	Role
@dmrole	Role of the attribute in the DM
@dmtype	Type of the attribute in the DM
@ref	Reference of the `FIELD` or `PARAM` that has to be sued to set the `ATTRIBUTE` value.
@value	Default `ATTRIBUTE` value. This value is taken if there is no @ref attribue or if @ref cannot be resolved.
@unit	`ATTRIBUTE` unit. This is the unit in which the native value must be converted to be complient with the model. This attribute is always optional.
@arrayindex	Index of the native value to be taken to set the `ATTRIBUTE`. Must be ignored if the native value is a single value. An error must be risen if @arrayindex is out of range.This attribute is always optional.

Table 16: `ATTRIBUTE` attributes

@dmrole	@dmtype	@ref	@value	Pattern
MAND	MAND	MAND	OPT	The `ATTRIBUTE` value must be set with the value of the element referenced by @ref. The @ref can not be resolved and @value is present, @value must taken as `ATTRIBUTE` value
MAND	MAND	NO	MAND	The `ATTRIBUTE` value must be set with @value

Table 17: Valid attribute patterns for `ATTRIBUTE`

4.8 REFERENCE

INSTANCE reference that can be used with the patterns as for INSTANCEs. There different uses for the REFERENCEs

- Static reference: the element has a @dmref attribute with the identifier (@dmid attribute) of the referenced INSTANCE.
- Dynamic reference: The element has a @tableref attribute identifying the TEMPLATES or the GLOBALS/COLLECTION where to fetch the referenced object. In this case, REFERENCE must have one or more FOREIGN_KEY children. In this case, the REFERENCE must be within a TEMPLATES.

```
<dm-mapping:REFERENCE dmrole="_role" dmref="_target1" />
```

Listing 10: Simple `REFERENCE`, to be replaced with the INSTANCE with @dmid=_target1

```
<dm-mapping:REFERENCE dmrole="_role" tableref="_target1">
    <dm-mapping:FOREIGN_KEY ref="_col1" />
</dm-mapping:REFERENCE>
```

Listing 11: Dynamic `REFERENCE`, to be replaced with the INSTANCE of the table of collection _target1 and having a PRIMARY_KEY matching the value of column _col1

Attribute	Role
@dmrole	Role of the referenced instance or collection in the DM
@tableref	dmid of the `COLLECTION` to be joined with in case of using a `FOREIGN_KEY`
@dmref	dmid of the referenced instance or collection

Table 18: `REFERENCE` attributes

@dmrole	@tableref	@dmref	Pattern
MAND	MAND	NO	This is the `FOREIGN_KEY` pattern. @tableref gives the dmid of the `COLLECTION` to be joined with. In this case `REFERENCE` must have one `FOREIGN_KEY` child and the joined `COLLECTION` must have a `PRIMARY_KEY`
MAND	NO	MAND	Simple reference to either an `INSTANCE` or `COLLECTION`, usually searched in the `GLOBALS`

Table 19: Valid attribute patterns for `REFERENCE`

4.9 JOIN

The JOIN element allows to populate a COLLECTION with INSTANCEs from another collection, namely the foreign collection. The foreign collection can either be a static element (GLOBALS/COLLECTION) or a collection of INSTANCES resulting from the iteration over a TEMPLATES.

The JOIN element must contain attributes identifying the foreign collection (@tableref and/or @dmref). It can have WHERE children stating the join condition.

It must be child of a COLLECTION that has not other children element that INSTANCE. A COLLECTION cannot host more that 1 JOIN.

JOIN may have 2 uses:

- Join with TEMPLATES data:
  - must have a @tableref attribute identifying the foreign TEMPLATES
  - may have a @dmref attribute the identify an INSTANCE within the foreign TEMPLATES.

  The mapping of the enclosing COLLECTION items can be either implicit or explicit:
  - If it has no other children, it must be populated with the whole table rows.

  - Otherwise the item content is given by the enclosed INSTANCE. The reference (@ref) contained in those INSTANCEs must refer to te field of the foreign table.

- Join with COLLECTION data:
  - must have no @tableref attribute, referencing a table in this case is irrelevant.
  - must have a @dmref attribute the identify the COLLECTION to be joined with. This COLLECTION must be GLOBALS child.

  The mapping of the enclosing COLLECTION items can be either implicit or explicit:
  - If it has no other children, it must be populated with the foreign collection items.
  - Otherwise the item content is given by the enclosed INSTANCEs. The reference (@ref) contained in those INSTANCEs must refer to te field of the foreign table, we must not have such @ref if the foreign collection is static

Listing 12: `JOIN`

Attribute	**Role**
@tableref	Reference of the table to be joined with.
@dmref	Reference of the `COLLECTION` (in `GLOBALS` to be joined with.

Table 20: `JOIN` attributes

@tableref	**@dmref**	**Pattern**
MAND	NO	The join is done against the table identified by @tableref
NO	MAND	The join is done against the `COLLECTION` identified by @rmref

Table 21: Valid attribute patterns for `JOIN`

Element	**Position**	**Cardinality**	
`WHERE`	1	0-*	Join condition

Table 22: Allowed children for `JOIN`

4.10 WHERE

The WHERE element is used to filter iteration outputs. Value are accepted when the key equals to the value. The mapping syntax does not specify the data types to be used to evaluate the expression. There are 2 different uses for this element:

- As a child of TEMPLATES. Only the table rows satisfying the WHERE conditions will be mapped. With this pattern WHERE must have one @primarykey attribute and one @value attribute. @primarykey references the column (FIELD) to be checked. The WHERE condition is satisfied for the rows having @primarykey equals to @value
- As a child of JOIN. Only the joined data items satisfying the WHERE conditions will be taken. With this pattern WHERE must have one @foreignkey attribute and one of @value either @primarykey attribute. @foreignkey references the column of the foreign collection to be checked. The WHERE condition is satisfied for the rows having @primarykey equals to either @value or @primarykey value.

```
<dm-mapping:TEMPLATES tableref="table">
  <dm-mapping:WHERE primarykey="col1" value="val1" />
  <dm-mapping:WHERE primarykey="col2" value="val2" />
  <dm-mapping:INSTANCE dmtype="type" />
</dm-mapping:TEMPLATES>
```

Listing 13: `WHERE` Example: only rows having val1 as col1 value and val2 as col2 value are mapped

```
<dm-mapping:JOIN tableref="ft" >
        <dm-mapping:WHERE foreignkey="ftc" primarykey="ptc" />
</dm-mapping:JOIN>
```

Listing 14: `WHERE` Example: the join is satisfied when the value of the ptc column is equals to the ftc column of the foreign table

Attribute	Role
@primarykey	FIELD identifier of the primary key column
@foreignkey	FIELD identifier of the foreign key column
@value	Literal value the @primarykey cell must match with

Table 23: `WHERE` attributes

@primarykey	@foreignkey	@value	Pattern
MAND	MAND	NO	2 tables join criteria: @primarykey = @foreignkey
MAND	NO	MAND	Simple join criteria: @primarykey = @value

Table 24: Valid attribute patterns for `WHERE`

4.11 PRIMARY_KEY

The PRIMARY_KEY element allows to set an identification key to an INSTANCE. The primary keys are only used in the context of REFERENCEs using FOREIGN_KEY. A primary key can be either static or dynamic.

- Static: the key value is given by the @value attribute.
- Dynamic: the key value is given by the value of the field referenced by @ref. This pattern is only valid if the INSTANCE is within a TEMPLATES.

the type of te key must always be specified by the @dmtype attribute.

Attribute	Role
@ref	ID of the FIELD used as primary key
@dmtype	Type of the key
@value	Literal key value. Used when the key relates to a `COLLECTION in the GLOBALS`

Table 25: `PRIMARY_KEY` attributes

@ref	@dmtype	@value	Pattern
MAND	MAND	NO	The FIELD referenced by @ref is a primary key. This pattern is used within a `TEMPLATES`
NO	MAND	MAND	@value gives the key value. This pattern is used to set a primary key to a `COLLECTION`

Table 26: Valid attribute patterns for `PRIMARY_KEY`

4.12 FOREIGN_KEY

FOREIGN_KEY is only used within a REFERENCE located within a TEMPLATE. It identifies the FIELD that must match the primary key of the referenced collection.

```
<dm-mapping:REFERENCE dmrole="coords:Coordinate.coordSys" tableref="_CoordinateSystems">
      <dm-mapping:FOREIGN_KEY ref="_band"/>
</dm-mapping:REFERENCE>
```

Listing 15: The REFERENCE is resolved by the INSTANCE of table _CoordinateSystems that has a primary key equals to the value of the column _band

Attribute	**Role**
@ref	Identifier of the FIELD that must match the primary key of the referenced collection

Table 27: `FOREIGN_KEY` attributes

5 Changes from Previous Versions

No previous versions yet.