International
Virtual
Observatory
Alliance

Model Instances in Votables
Version 1.0

IVOA Working Draft 2020-09-15

Working group
DM
This version
http://www.ivoa.net/documents/model-instance-in-vot/20200915
Latest version
http://www.ivoa.net/documents/model-instance-in-vot
Previous versions
This is the first public release
Author(s)
François Bonnarel, Gilles Landais, Laurent Michel, Jesus Salgado
Editor(s)
Laurent Michel

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 (Bradner, 1997).

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 (Ochsenbein and Taylor et al., 2019). 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 (Lemson and Laurino et al., 2018) 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 Lemson and Laurino et al. (2017), 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 (Michel, 2018). 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

Fig. 1 shows the role this document plays within the IVOA architecture (Arviset and Gaudet et al., 2010).

???? and so on, LaTeX as you know and love it. ????

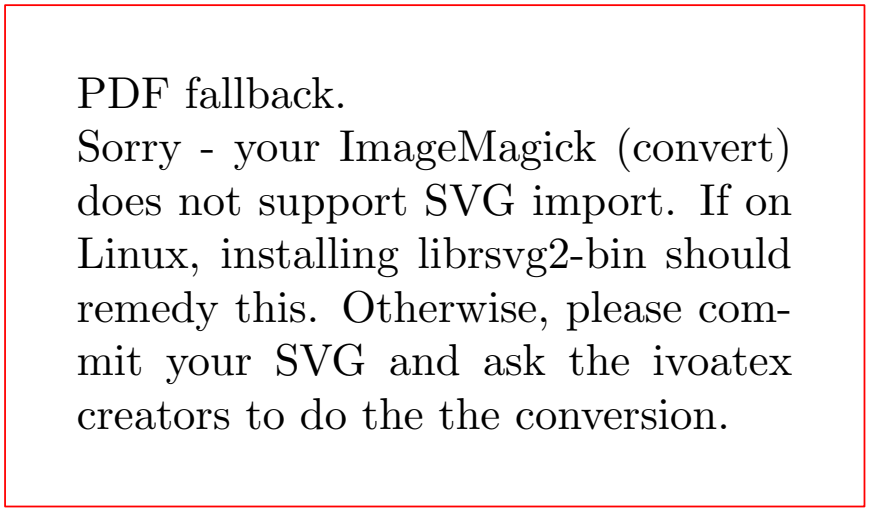

Figure 1: Architecture diagram for this document

2 Use Cases and Requirement

2.1 Use Cases

2.1.1 Client Side

The mapping must be able to provide clients with self-consistant information needed to reconstruct datastructures compliant with the mapped model. A model-aware client must be able to do this without implementing model-specific code.

Identifying the nature of the content of a VOTable: A client can check the annotation block of a VOTable to decide how to process it. For instance, it can detect that the VOTable contains a Provenance instance and then invoke a specific viewer.

Measurement discovery: A client wants to discover whether a VOTable contains some peculiar measurements (position, velocity...). This can be done with a quick parsing of the annotation block.

Data set comparison: A client wants to compare different data set s (e.g. Xmatch, plot). The annotation provides a homogeneous data representation that allows to put them together in a consistent way.

Data set export: A client wants to export (e.g. with SAMP) a model instance in a convenient format (e.g. json). The exportable model instance can be built from the annotation block.

2.1.2 Server Side

The server use cases are to make possible the realization of those of the clients for a reasonable cost. The annotation process can represent a significant extra work for the curator team that must be limited as much as possible. To do so the mapping syntax is designed to facilitate the use of templates and components. The difficulty of annotating data depends on the service type: 3 have been identified and ranked by growing cost:

1. **Mission data provider**: the data annotation can be set once forever for each data product at the design phase.
2. **Archival data provider**: The data annotation must be done for each archived datat set. The curator has a little control on the data format and he/she has to do his best to match data with the model(s). In this case, it must be possible to limit the annotation on a subset of data.
3. **TAP data provider**: In case of TAP services, the annotation process is in charge of the TAP server that must dynamically match queried data with model quantities, and this for each specific query.

The goal of this version of the specification is to support requirements 1) and 2) with a special attention to make 2) easier. Support of requirement 3) is still an experimental feature at the time this specification is written.

2.2 Requirements

- Shy Annotations: The data mapping must not affect the operation of existing clients.
- Faithful Annotations: The structure of the annotation must be faithful to the mapped model, whatever it is as long as it is VODML compliant.
- Different Usage Levels
  - The data mapping should be easily overlooked by the customer
  - The data mapping must allow clients to easily detect the model on which data are mapped if there is one.
  - The data mapping must allow clients to easily get the general metadata (e.g. coordinate systems).
  - The data mapping must allow clients to get full model instances for each table row.
- Easy to Build
  - The mapping structure must be independent of the data structure.

  - The mapping syntax should make easy the building of both mapping components and templates.

- Complex Data Mapping
  - The mapping syntax must be able to annotate data spread over several tables.
  - The mapping syntax must be able to filter data rows that have to be instanciated.

3 Syntax

The syntax rules specified in this standard allow to build consistent annotations for any model. However, they do not prevent to do foolish things in the same way that a programming language grammar does not protect people against writing irrelevant software. In the following annotation snippets, the values of the XML elements attribute do not refer to any particular model or VOTable. They have been choosen to help readers to figure out their meanings.

3.1 Mapping Block Structure

The mapping block must be the first child of `VOTABLE`. Its scope is the whole VOTable. Its stucture is given below.

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          version="1.4">
  <MODEL_INSTANCE name="MANGO"
            syntax="model-instance-in-vot"
           uri="http://server.mango.vo-dml.xml">
    <GLOBALS> ... </GLOBALS>

     <TABLE\_MAPPING tableref=...> ... </TABLE\_MAPPING tableref=...>
     <TABLE\_MAPPING tableref=...> ... </TABLE\_MAPPING tableref=...>
   ...
  </MODEL_INSTANCE>
  ..
</VOTABLE>
```

Listing 1: Complete mapping block example

The mapping construction rules are independant from any model or data layout either.

- The mapping is located in a `<MODEL_INSTANCE>` block, child of `<VOTABLE>`.

- The mapping elements denote the model structure.
- The first child of <MODEL_INSTANCE> must be the <GLOBALS> block containing data shared by the whole mapping.
- The <GLOBALS> block is followed a sequence of <TABLE_MAPPING>.
- There is one <TABLE_MAPPING> per mapped <TABLE>.

3.2 Mapping Top Level Structure

3.2.1 MODEL_INSTANCE

The MODEL_INSTANCE element encompasses the mapping block. This element can refer or not to a specific model. There no obligation to have a tight coupling between a mapping block and a specific model. The use of the model referenced MODEL_INSTANCE is left at the discretion of the client. The client can just look for interesting quantities without regards on a particular model or on the opposite build a model instance by merging the vo-dml model serialization which URL is given here, with the mapping block.

Attribute	Role
@name	Name of the mapped model (informal). This attribute must be left empty
@syntax	Syntax used for the mapping. The default syntax is this proposed in this document@syntax=model-instance-in-vot . This attribute must be left empty when present.
@uri	Uri of the vo-dml serialization of the mapped model.

Table 1: MODEL_INSTANCE attributes

@name	@syntax	@uri	Pattern
MAND	OPT	OPT	Always mandatory

Table 2: Valid attribute patterns for MODEL_INSTANCE

3.2.2 GLOBALS

Contains INSTANCEs that can be used everywhere in the MODEL_INSTANCE.

- INSTANCEs children of GLOBALS should have an @ID attribute so that they can be referenced from other instances.
- The role of the GLOBALSs children, (INSTANCEs by construction), must be ignored although being mandatory.

- References within GLOBALSs sub-elements to VOTable data (FIELD ot PARAM) must be searched in all tables. They must be resolved by the first occurence matching the reference found.
- GLOBALS has no attributes.

```
<GLOBALS>
  <INSTANCE ID="SpaceCoordFrame" >
    <INSTANCE dmrole="coords:SpaceFrame.refPosition"
              dmtype="coords:StdRefLocation">
        <ATTRIBUTE dmrole="coords:StdRefLocation.position"
                   dmtype="ivoa:string" value="NoSet"/>
    </INSTANCE>
    <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame"
               dmtype="ivoa:string" value="ICRS"/>
    <ATTRIBUTE dmrole="coords:SpaceFrame.equinox"
               dmtype="coords:Epoch" value="NoSet"/>
  </INSTANCE>
</GLOBALS>
```

Listing 2: GLOBALS block example

Child	Role
INSTANCE	Model instances with a scope covering the whole VOTable .

Table 3: Allowed GLOBALS children

3.2.3 TABLE_MAPPING

TABLE_MAPPING blocks contain the mapping statements of the data contained in one TABLE .

- There is one TABLE_MAPPING block for each mapped TABLE in the VOTAble.
- A TABLE cannot be referenced by more than one TABLE_MAPPING element.
- The table related to a TABLE_MAPPING is identified by the @tableref attribute. It must be first resolved against the TABLE identifier (@ID) and then against the table name.

```
<TABLE_MAPPING tableref="OtherResults">
    <COLLECTION dmrole="test:detections">
        <TABLE_ROW_TEMPLATE>
            <INSTANCE dmrole="" dmtype="test:Detection">
```

```
            <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                       ref="_num_148" />
            <ATTRIBUTE dmrole="test:detection.id" dmtype="ivoa:real"
                       ref="_foreign" />
        </INSTANCE>
    </TABLE_ROW_TEMPLATE>
</COLLECTION>
</TABLE_MAPPING>
```

Listing 3: TABLE_MAPPING block example

Child	Role
INSTANCE	Mapping of an object or a data type.
COLLECTION	Mapping of an object collection
GROUPBY	One object collection for each group of rows.

Table 4: Valid TABLE_MAPPING children

Attribute	Role
@tableref	The @ID or the @name of the mapped table

Table 5: TABLE_MAPPING attributes

@tableref	Pattern
MAND	Always mandatory

Table 6: Valid attribute patterns for TABLE_MAPPING

3.3 Data Hierarchy

3.3.1 INSTANCE

Mapping for either object types or non primitive datatypes instances.

```
<INSTANCE dmrole="ds:Dataset.dataID" dmtype="ds:DataID">
    <ATTRIBUTE dmrole="ds:DataID.title" value="Gaia TS Mapping Test" />
    <ATTRIBUTE dmrole="ds:DataID.datasetID" value="ivoa://gaia/ts/12345" />
    <ATTRIBUTE dmrole="ds:DataID.creatorDID" value="ivoa://esa/gaia/" />
    <ATTRIBUTE dmrole="ds:DataID.version" value="0.0" />
    <ATTRIBUTE dmrole="ds:DataID.date" value="2018:11:11" />
    <ATTRIBUTE dmrole="ds:DataID.creationType" value="LiteMappingTest" />
    <INSTANCE dmrole="ds:DataID.creator" dmtype="ds:Creator">
        <INSTANCE dmrole="ds:Role.party" dmtype="ds:party.Individual">
            <ATTRIBUTE dmrole="ds:Party.name" value="MODEL_INSTANCE-Team" />
      </INSTANCE>
```

```
    </INSTANCE>
</INSTANCE>
```

Listing 4: INSTANCE block example

Child	Role
INSTANCE	Another embedded instance .
ATTRIBUTE	Primitive attribute .
COLLECTION	Set of items

Table 7: Valid INSTANCE children

Attribute	Role
@dmrole	VODML role. Can be empty if located in GLOBALS or in TABLE_ROW_INSTANCE INSTANCE being child of open ended COLLECTIONs take the role of the parent collection. This is due to the fact no specific role can be assigned to open-ended collection items. The role dmrol=root can be used to tell the parser whic class has to be instanciated first.
@dmtype	VODML type of the instance. Must never be empty
@dmref	Reference to another instance in the mapping block. Must never be empty
@ID	Unique identifier of the instance. Must never be empty

Table 8: INSTANCE attributes

@dmrole	@dmref	@dmtype	@ID	Pattern
MAND		MAND	OPT	Instance of a certain type playing a certain role.
MAND	MAND			Reference to another instance. No allowed children in this case.

Table 9: Valid attribute patterns for INSTANCE

3.3.2 ATTRIBUTE

Mapping statement for primitive attributes.

- ATTRIBUTEs are the model leaves.

- `ATTRIBUTE` values can either be set by reference on tabke data or by literal values.
- `ATTRIBUTE`s have no children.

```
<INSTANCE dmrole="model:value.example" dmtype="model:value.Example">
    <ATTRIBUTE dmrole="model:preset.value" value="Preset Value" />
    <ATTRIBUTE dmrole="model:ref.value" ref="fieldID" />
    <ATTRIBUTE dmrole="model:reforpreset.value"
               value="Preset Value" ref="fieldID" />
</INSTANCE>
```

Listing 5: ATTRIBUTE examples

Attribute	**Role**
@dmrole	VODML role of the attribute. Must never be empty
@dmtype	VODML type of the instance attribute. Must never be empty
@value	Literal value of the instance attribute. If `ATTRIBUTE` has also a `@ref`, `@ref` MUST be resolved first. `ATTRIBUTE` `@val` must be taken when `@ref` cannot be resolved
@ref	Reference of the data element (`FIELD` or `PARAM`). Must refer to an element of the `TABLE` referenced by the current `TABLE_MAPPING` The client MUST first look for a `FIELD` matching `@ref`. If no `FIELD` is found, it must look for a `PARAM`

Table 10: `ATTRIBUTE` attributes

@dmrole	@dmtype	@ref	@value	Pattern
MAND	MAND	MAND	OPT	The instance attribute must take the value pointed by `@ref`. If the reference cannot be resolved, the attribute takes the value of `@val` if present. It is considered as not set otherwise.
MAND	MAND		MAND	The attribute takes the value of `@val`.

Table 11: Valid attribute patterns for `ATTRIBUTE`

3.3.3 COLLECTION

Mapping statement fort sets of either instances or collections.

- A `COLLECTION` can contain a fixed set of instances or collections. In this case, each element must be mapped individually. Elements can be either `INSTANCE`s or `COLLECTION`s.
- A `COLLECTION` can contain an unbounded set of instances, one per selected table row. In this case, all items have the same type and thus the same mapping. They can be set with local table data or with data from a joint table.

The example below show up a a fixed length `COLLECTION`.

```
<TABLE\_MAPPING tableref="Results">
    <COLLECTION dmrole="meas:Measure.errors" size="2">
        <INSTANCE dmref="globals_stat_error" />
        <INSTANCE dmref="globals_sys_error" />
    </COLLECTION>
</TABLE\_MAPPING>
```

Listing 6: `COLLECTION` example

‘

Child	Role
INSTANCE	Mapping of one collection item. A collection can embed multiple INSTANCEs
COLLECTION	Mapping of one collection item. A collection can embed multiple COLLECTIONs
TABLE_ROW_TEMPLATE	The collection is populated with with one instance per row of the current table. When present, this element must be the only child.
JOIN	The collection is populated with data read from another table. When present, this element must be the only child.

Table 12: Valid COLLECTION children

Attribute	Role
@dmrole	Role played by the collection (VODML relation name usually). Cannot be empty.
@size	Collection size. This attribute is not necessary to parse the mapping block.

Table 13: Valid attributes for COLLECTION

@dmrole	@size	Role
MAND	OPT	Role played by the collection (VODML relation name usually). Cannot be empty

Table 14: Valid attribute patterns for COLLECTION

3.4 Parsing Statement

3.4.1 TABLE_ROW_TEMPLATE

This element indicates that one element must be added to the host COLLECTION for each table row.

- The row mapping is given by the INSTANCE child.
- One and only one INSTANCE can be mapped per row. This makes senses since collection elements cannot be made with more than one instance.
- TABLE_ROW_TEMPLATE has no attributes.

```
<TABLE\_MAPPING tableref="Results">
    <COLLECTION dmrole="test:detections">
        <TABLE_ROW_TEMPLATE>
            <INSTANCE dmtype="test:Detection">
```

```
            <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                       ref="_num_148" />
            <ATTRIBUTE dmrole="test:detection.id" dmtype="ivoa:real"
                       ref="_num_149" />
        </INSTANCE>
    </TABLE_ROW_TEMPLATE>
</COLLECTION>
</TABLE\_MAPPING>
```

Listing 7: TABLE_ROW_TEMPLATE examples

Child	Role
INSTANCE	Mapping to be applied to table row

Table 15: Supported TABLE_ROW_TEMPLATE children

3.4.2 FILTER

This element filters the table rows that are to be mapped.

- The filtering condition is based on the equality of a column value with the filter value.
- The mapping syntax does not specify the way to deal with data types.

In the example below::

- The light curve will be populated with table rows mapped by the INSTANCE of type test:photometric.point
- Each of these rows must have the value of the column phot_filter_name equals to G.

```
<COLLECTION dmrole="test.lightcurve">
    <TABLE_ROW_TEMPLATE>
        <FILTER ref="phot_filter_name" value="G"/>
            <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                             dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                             dmtype="ivoa:real" ref="_num_149" />
            </INSTANCE>
      </FILTER>
    </TABLE_ROW_TEMPLATE>
</COLLECTION>
```

Listing 8: FILTER examples

Child	Role
INSTANCE	Mapping to be applied to table rows matching the filter

Table 16: Valid FILTER children

Attribute	Role
@ref	Identifier of the column on which the filtering criteria must be applied
@value	Literal value that is used as filtering criteria

Table 17: FILTER attribute

@ref	@value	Role
MAND	MAND	All attributes must be set in any case

Table 18: Valid FILTER attribute pattern

3.4.3 JOIN

This element populates the host collection with data taken out from a foreign table and matching the join criteria.

- Each matching row of the foreign table is mapped as one INSTANCE of type test:Detection .
- Self-joins on the local table are allowed.
- The join criteria is based on the equality of the column values. The mapping specification does not specify the way to deal with data types.

```
<TABLE_ROW_TEMPLATE>
    <INSTANCE dmrole="primary:point" dmtype="Point">
        <ATTRIBUTE dmrole="test:detection.num" dmtype="ivoa:real"
                   ref="_poserr_148" />
        <COLLECTION dmrole="test.detections">
            <JOIN tableref="OtherResults" primary="_poserr_148"
                  foreign="_foreign">
                <INSTANCE dmtype="test:Detection">
                    <ATTRIBUTE dmrole="test:detection.num"
                               dmtype="ivoa:real" ref="_num_148" />
                    <ATTRIBUTE dmrole="test:detection.id"
                               dmtype="ivoa:real" ref="_foreign" />
                </INSTANCE>
            </JOIN>
        </COLLECTION>
    </INSTANCE>
```

```
</TABLE_ROW_TEMPLATE>
```

Listing 9: JOIN example

Child	Role
INSTANCE	Mapping to be applied to the matching rows.

Table 19: Supported JOIN children

Attribute	Role
@primary	Column identifier of the primary table used by the join
@foreign	Column identifier of the foreign table used by the join
@tableref	ID or name of the foreign table

Table 20: JOIN attributes

@primary	@foreign	@tableref	Role
MAND	MAND	MAND	All attributes must be set in any case

Table 21: Valid JOIN attribute pattern

3.4.4 GROUPBY

This element aggregates host table rows in groups which elements have all the same value for a given column.

- Each matching row is mapped as one instance of the INSTANCE child.

In the example below:

- The collection with @dmrole=test.lightcurves will be populated with a set of collections.
- Each of these sub-collections is populated with set of instances mapped by the INSTANCE of test:photometric.point type.
- All INSTANCEs are built wiith rows having all the same values for the column source_name

```
<TABLE_MAPPING tableref="Results">
    <GROUPBY ref="filter_name" dmrole="test.lightcurve">
        <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
```

```
                        dmtype="ivoa:real" ref="_num_148" />
                <ATTRIBUTE dmrole="test:photometric.point.mag"
                        dmtype="ivoa:real" ref="_num_149" />
        </INSTANCE>
    </GROUPBY>
</TABLE_MAPPING>
```

Listing 10: GROUPBY examples

If we suppose that the data table have photometric points for 3 filters (R,G,V), the above statement is equivalent to the following one.

```
<TABLE_MAPPING tableref="Results">
    <COLLECTION dmrole="test.lightcurve">
          <FILTER ref="filter_name" value="R"/>
          <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                          dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                          dmtype="ivoa:real" ref="_num_149" />
          </INSTANCE>
   </COLLECTION>
    <COLLECTION dmrole="test.lightcurve">
          <FILTER ref="filter_name" value="G"/>
          <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                          dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                          dmtype="ivoa:real" ref="_num_149" />
          </INSTANCE>
   </COLLECTION>
    <COLLECTION dmrole="test.lightcurve">
          <FILTER ref="filter_name" value="V"/>
          <INSTANCE dmtype="test:photometric.point">
                  <ATTRIBUTE dmrole="test:photometric.point.time"
                          dmtype="ivoa:real" ref="_num_148" />
                  <ATTRIBUTE dmrole="test:photometric.point.mag"
                          dmtype="ivoa:real" ref="_num_149" />
          </INSTANCE>
   </COLLECTION>
</TABLE_MAPPING>
```

Listing 11: GROUPBY examples

Child	Role
INSTANCE	Mapping to be applied to the matching rows.

Table 22: Valid GROUPBY children

Attribute	Role
`@ref`	Identifier of the column used for the grouping
`@dmrole`	Role of the grouped sub-collections

Table 23: `GROUPBY` attributes

@ref	@dmrole	Role
MAND	MAND	Must be set with non empty values

Table 24: Valid `GROUPBY` attribute pattern

3.5 Shortcuts

VODML encourgages people to use the `ivoa` model for the primitive types. Some of these types have a complex structures that associate units with values. This is the case for the types derived from `ivoa:Quantity` (`ivoa:RealQuantity` and `ivoa:IntegerQuantity`). The XML snippet below shows the regular mapping for a real quantity..

```
<INSTANCE dmrole="coords:PhysicalCoordinate.cval"
dmtype="ivoa:RealQuantity">
    <ATTRIBUTE dmrole="ivoa:RealQuantity.value" dmtype="ivoa:real"
                    ref="col_id" />
    <ATTRIBUTE dmrole="ivoa:RealQuantity.unit" dmtype="ivoa:Unit"
                    value="m/sec" />
</INSTANCE>
```

Listing 12: ivoa:RealQuantity example

This element being part of the VODML standard, it canm be replaced with a compact element named shortcut.

3.5.1 SC_REALQUANTITY

Shortcut for `ivoa:RealQuantity` class.

- Can only be used within an INSTANCE
- Using shorcuts requires units to be literals
- Both `@ref` and `@value` attributes work the same way as with `ATTRIBUTE`
- No `@dmtype`, it is set as ivoa:RealInteger by construction

```
<SC_REALQUANTITY dmrole="coords:PhysicalCoordinate.cval"
          ref="col_id" value="0.0" unit="m/sec" />
```

Listing 13: `ivoa:RealQuantity` example

3.5.2 SC_INTQUANTITY

Shortcut for `ivoa:IntegerQuantity` class.

- Can only be used within an INSTANCE
- Using shorcuts requires units to be literals
- Both `@ref` and `@value` attributes work the same way as with `ATTRIBUTE`.
- No `@dmtype`, it is set as ivoa:RealInteger by construction

```
<SC_INTQUANTITY dmrole="coords:PhysicalCoordinate.cval"
                ref="col_id" value="0" unit="m/sec" />
```

Listing 14: `ivoa:IntegerQuantity` example

3.5.3 SC_FIELD

Equivalent to `FIELDREF`.

- Can only be used within an INSTANCE
- Using shorcuts requires units to be literals
- Both `@ref` and `@value` attribute work the same way as with `ATTRIBUTE`.
- No `@dmtype`, it is set as ivoa:RealInteger by construction

```
<SC_FIELD dmrole="coords:PhysicalCoordinate.cval" ref="col_id"/>
```

Listing 15: `ivoa:IntegerQuantity` example

A Changes from Previous Versions

No previous versions yet.

References

Arviset, C., Gaudet, S. and the IVOA Technical Coordination Group (2010), 'IVOA architecture', IVOA Note.
http://www.ivoa.net/documents/Notes/IVOAArchitecture

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.
http://www.ietf.org/rfc/rfc2119.txt

Lemson, G., Laurino, O., Bourges, L., Cresitello-Dittmar, M., Demleitner, M., Donaldson, T., Dowler, P., Graham, M., Gray, N., Michel, L. and Salgado, J. (2018), 'VO-DML: a consistent modeling language for IVOA data models Version 1.0', IVOA Recommendation 10 September 2018.
http://doi.org/10.5479/ADS/bib/2018ivoa.spec.0910L

Lemson, G., Laurino, O., Donaldson, T., Cresitello-Dittmar, M., Michel, L., Dowler, P., Demleitner, M., Graham, M. and Salgado, J. (2017), 'Mapping data models to votable', WD 20170323.
https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml-mapping/doc/VO-DML_mapping_WD.pdf

Michel, L. (2018), 'Serializing time domain data a (simplified) vo-dml approach', Talk 2018-05-10.
https://wiki.ivoa.net/internal/IVOA/InterOpMay2018TDIG/Talk_VODML_Lite_at_a_glance_en.pdf

Ochsenbein, F., Taylor, M., Donaldson, T., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A. and Wicenec, A. (2019), 'VOTable Format Definition Version 1.4', IVOA Recommendation 21 October 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.1021O