International
Virtual
Observatory
Alliance

Model Instances in Votables
Version 1.0

IVOA Recommendation 2023-06-08

Working group
DM
This version
http://www.ivoa.net/documents/mivot/20230608
Latest version
http://www.ivoa.net/documents/mivot
Previous versions
This is the first public release
Author(s)
Laurent Michel, Mark Cresitello-Dittmar, François Bonnarel, Gilles Landais, Gerard Lemson, Mireille Louys, Jesus Salgado
Editor(s)
Laurent Michel, Mark Cresitello-Dittmar

Abstract

Model Instances in VOTables (MIVOT) defines a syntax to map VOTable data to any model serizalized in VO-DML. The annotation operates as a bridge between the data and the model. It associates the column/param metadata from the VOTable to the data model elements (class, attributes, types, etc.) of a standardized IVOA data model, expressed in the Virtual Observatory Data Modeling Language (here after VO-DML). It also brings up VOTable data or metadata that were possibly missing in the table metadata.

The data model elements are grouped in an independent annotation block complying with the MIVOT XML syntax. This annotation block is added as an extra resource element at the top of the VOTable result resource. The MIVOT syntax allows to describe a data structure as a hierarchy of classes. It is also able to represent relations and composition between them.It can also build up data model objects by aggregating instances from different tables of the VOTable.

Missing metadata can also be provided using MIVOT, for instance by completing coordinate system description, or by providing curation tracing.

The annotation block is made of re-usable bricks that facilitate the development of tools on both client and server sides. The adopted design does not alter the original VOTable content. The backward compatibility is preserved thanks to the limited impact of the annotations on legacy clients.

Status of this document

This document has been reviewed by IVOA Members and other interested parties, and has been endorsed by the IVOA Executive Committee as an IVOA Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. IVOA's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability inside the Astronomical Community.

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

Contents

Acknowledgments

We would like to thank all the people who have contributed to this standard, starting with the authors of the VO Data Modeling Language who have introduced the concepts developed here. We would also like to thank all contributors of the "Source Data Model" session held at the Paris Interop

Meeting in 2019, as well as all members of the IVOA Time Domain Interest Group and the participants to the Data Model workshop we ran in Spring 2021. All helped us to refine our use-cases. We would like the thank the both funding projects: European Project ESCAPE (Grant Agreement no. 824064), OVFrance and agencies (CNRS-INSU). Finally, we say a big thank you to all the colleagues who spent time for assessing how such an annotation procedure could improve their VO services. This acknowledgement does not mention individuals, as the list would be too long, especially the many interns who contributed to this project.

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 quantities relevant to that domain and how they relate to each other. The second purpose is to facilitate the interoperability between different stakeholders involved in the domain. The interoperability consists in arranging exchanged data so that any client can understand it without being concerned with its origin thus allowing the same code to process and compare data coming from different archives. In other words, the more faithful is the data representation towards the data model, the better is the interoperability. The challenge for the VO ecosystem is to design the best way to map various data on standard models while respecting the existing frameworks, protocols and data formats.

We could imagine to develop a new data container specific for that purpose but any solution that ignores the existing assets would be utterly useless and would have no chance of being accepted. The model mapping solution must be based on VOTables since VOTable (Ochsenbein and Taylor et al., 2019) is the standard data container within the VO. Unfortunately, if the VOTable schema allows a very good description of tabular data, it is not well suited for rendering views of complex data models. In the case of simple DAL protocols, this limitation has been worked around by specifying the metadata that must be present in the query responses; the model mapping is thereby defined by the standard.

This approach is no longer applicable for complex models such as Cube or Provenance for which the data tree may vary from one instance to another.

The basis on which the standard is designed is threefold 1) the data content is apriori unknown (e.g. TAP response), 2) it has to be mapped on models that are not defined yet and 3) the mapping block must be fully integrated within a VOTable context. It must also allow to bind native data with a model in a way that model-aware software can see it as model instances while maintaining the possibility to access them in their original serialization.

The VOTable schema supports an XML attribute for this purpose, a UType, that partially meets this function. UTypes are path-like strings (`m:a.b.c`) identifying model leaves. They are labels attached to a `FIELD` or `PARAM` that tells its role in the model structure, typically a non cyclic graph. As there is no common rule to build UTypes, each data model (e.g. Characterization, Obscore) must come with its specific lists of UTypes.

When used as `FIELD` attributes, UTypes allow one to connect particular columns to model leaves. This mapping method fits very well with the VOTable schema since it does not require any specific XML elements. However, some UTypes features that could complicate the mapping process have been discussed many times:

- There is no possibility for one `FIELD` or `PARAM` to play multiple roles.
- UTypes are not intended to be parsed, so they only identify which role is played by a `FIELD`/`PARAM`, in a restricted context. The same element used in different contexts or models have different UTypes; which hinders interoperability.
- UTypes constrain to single-table serializations. They do no allow to join data from different tables.
- UTypes do not support annotating the VOTable content towards multiple models (as TimeSeries or Catalog/Source for instance)

The UType approach could have been extended to overcome these limitations, but it has been decided to move forward a solution closer to the VO-DML (Lemson and Laurino et al., 2018) concepts. VO-DML is a meta-model that gives a standard way to express VO models as machine-readable XML document. In VO-DML, the role of model leaves are no longer given by simple strings like UTypes, but by the local role they play in the model hierarchy.

As a consequence, an annotation mechanism such as MIVOT, based on VO-DML and preserving the model hierarchy can easily provide data representations faithful to that model.

The basis of MIVOT is to insert into the VOTable, an XML block complying with the model structure and containing references to the actual data. These blocks are designed in such a way that a model-aware client can build model instances by copying that structure and by resolving the references. Other clients can just ignore them and rely as usual on the `FIELD`/`PARAM` and `GROUP` structures.

1.1 Role within the VO Architecture

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

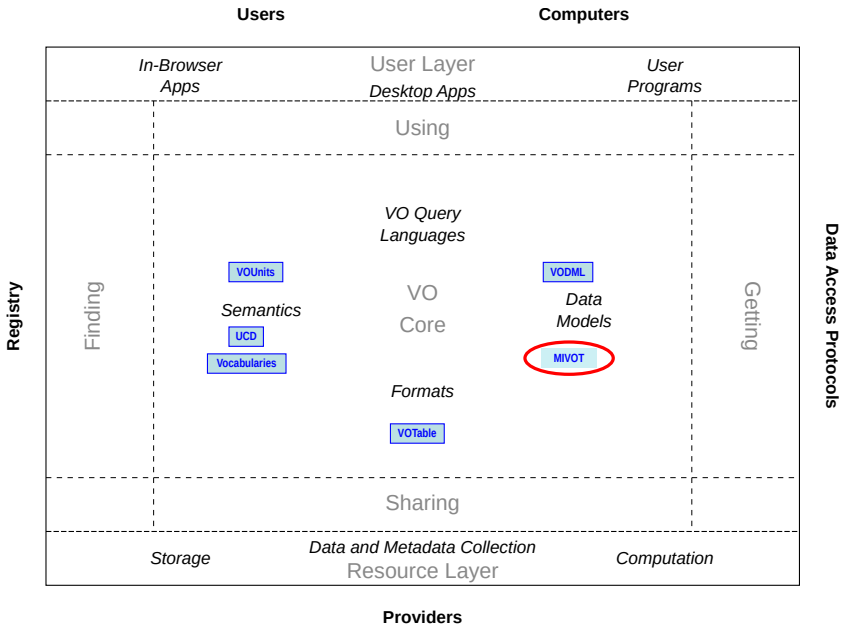

Figure 1: Architecture diagram for this document

2 Use Cases and Requirements

2.1 Use Cases

The ultimate goal of this standard is to allow clients to understand complex data whether it is legacy or set for a particular model. It is however important not to force client developers to choose between a regular table data extraction process and a full object oriented approach. Both ways should be allowed for the same data file.

The design of the mapping syntax addresses the necessity of supporting intermediate levels of use that correspond to concrete use cases.

1. The recent modeling efforts (Meas, Coords and PhotDM) provide very accurate descriptions of coordinate frames or photometric systems that need to be serialised in VOTables:

   - Clients often need to get such coordinate systems descriptions upon different axes to properly display datasets.
   - Data sets can contain different quantities expressed in the same coordinate system (corrected position vs. raw position) or quantities of the same nature but expressed in different systems (sky coordinates in ICRS vs. Gal). This is partially supported by the actual VOTable schema (TIMESYS, COOSYS) but any change

or addition in these ad-hoc elements would require a VOTable version upgrade. Using the mapping syntax disconnects the level of description of the different axes from the VOTable schema.

2. Quantities made with multiple components (spread over multiple table columns) may need to be aggregated by the client to be used correctly:

   - value-error associations
   - error quantities spread over several columns (covariance matrices).
   - quantities with quality flags
   - positions with proper motions and/or parallaxes to compute error ellipses or positions at a given epoch

3. The above cases relate to the description of instances in a single VOTable. Model annotations become even more important in cases where interoperability is required, for example to identify and reconcile quantities from multiple VOTables.

   This can be achieved by giving common data structures for all quantities of interest. This is the purpose of e.g. Measure model which provides classes for most of the physical quantities that can be rendered by the mapping syntax. Measure classes are not meant to be used as standalone elements but as parts of host models (e.g., Cube DM, MANGO DM); however, clients are free to either process those host models as a whole or to chase individual components.

   With more precise information on coordinate systems in the annotation:

   - Cross matching sources in VOTables containing the same quantities, e.g., the sky position, becomes easier. The reliability of the process is improved since the engine does not need to infer information that is not present in the FIELD metadata.
   - Building SEDs from datasets that have the same photometric calibration representation becomes straightforward.

4. In more advanced cases, we need to be able to extract complete model instances from VOTables.

   - Extracting TimeSeries instances to feed into software built upon classes generated from that model.
   - Building model instances that can be serialised in another format (e.g., json) in order to be shared in another context than the VOTable processing (e.g. via SAMP).

5. MIVOT can be used to annotate static science products such as e.g. time-series. It can also be used to annotate TAP responses on the fly, as shown by Louys and Michel et al. (2022). Note that this process relies on the correct execution of the TAP query with respect to the database content and may fail due to missing searched columns, illegal joins, or bad data aggregation, for instance.

2.2 Requirements

1. Shy annotation

   Model annotations enter into a workflow which has been working very well for years. As a matter of fact, the first requirement is to keep legacy services operational without any change. Therefore:

   - Annotation must not alter the VOTable content.
   - Annotation blocks must be located in a way that it can easily be skipped by model-unaware clients.
   - The vocabulary in the annotation name-space must not overlap with the VOTable elements (names or attributes)
   - The annotation must provide an element for communicating success or failure of the annotation and possible diagnostics.

2. Schema and validation:

   - The content of the annotation block must be able to be validated against a specific schema.
   - The annotation schema must be independent from the VOTable schema.
   - The evolution of the annotation schema must not impact the VOTable schema.
   - The evolution of the VOTable schema must not impact the annotation schema.
   - The annotation syntax must be validated using the usual tools, i.e., without using specific compilers.
   - Validators are not meant to check whether references can be resolved. Clients are in charge of handling possible inconsistencies.

3. Model agnostic behavior:

   - The annotation syntax must be able to map data on any VO-DML compliant model
   - The annotation syntax must allow clients to use their own strategy to consume mapped data, so they could:

  - just ignore it
  - pick some elements of interest
  - pick model metadata and process the sequence of data rows as usual
  - pick instances of model components
  - get instances of the whole model
- In order to allow clients to retrieve the VO-DML specifications a annotation block claims to adhere to, it must make it easy to find out data model names and versions in use.

4. On the fly annotation status:
   - Clients connecting TAP services registered as delivering annotated data must get VOTable with annotation blocks in any case.
   - We would like to have a place where we can stick error messages in case the annotation went wrong.
   - Clients must be informed of the data model name and version used for the annotation

3 Relation to VOTable

The data model annotation will reside within the scope of a VOTABLE V1.1+.

Location

The mapping block:

- MUST be contained in a VOTable `RESOURCE` with `type="meta"`. This extra feature is consistent with VOTable xml schema `RESOURCE` type definition and doesn't require any modification in the xml schema.
- MUST appear in the same parent `RESOURCE` as the mapped `TABLE`-s.
- SHOULD appear before the `TABLE` to enable streaming of MIVOT-mapped records.
- MUST be unique in the `RESOURCE` containing the data to be annotated.

The scope of the mapping block is the whole content its parent `RESOURCE`.

Namespace

The mapping element must be isolated from the VOTable elements by a name space set as an attribute of the `VODML` element.

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

Listing 1: Mapping block in a VOTable

4 Syntax

4.1 Overview

The syntax has been designed as a restricted but still complete set of XML elements. It relies on XML attributes to setup the element behavior in a particular context. As shown in table 1 the mapping elements are used in three different scopes. There are only three elements for the models structure itself. We assume that any data hierarchy can be represented by a combination of collections `COLLECTION`, tuples `INSTANCE`, and simple values `ATTRIBUTE`. The other elements are either to set the mapping block structure or to connect data to each other.

Scope	Elements
Data modeling tags	`ATTRIBUTE INSTANCE COLLECTION`
Mapping block structure	`VODML MODEL REPORT TEMPLATES GLOBALS`
Data references and identification	`REFERENCE JOIN FOREIGN_KEY PRIMARY_KEY WHERE`

Table 1: Mapping elements grouped by scopes.

As shown in Table 2 and following the `VODML` pattern, any model node is characterized by a role `@dmrole` and a type `@dmtype`. All of the other attributes are used to bind data with either VOTable elements or other mapping elements.

The values of both `@dmrole` and `@dmtype` are constructed by taking the vodml-id of the respective VO-DML elements and prefixing it with the canonical short name of the model and a colon.

Scope	Attributes
Model related	`@name @url`
Modeled node related	`@dmrole @dmtype`
Related to attribute values	`@value @unit @arrayindex`
Related to VOTable elements	`@tableref @ref`
Mapping element identification	`@dmref @dmid @sourceref @primarykey @foreignkey`

Table 2: Attributes of mapping elements grouped by scopes.

4.2 Reader's Guide

Each element subsection comes with 2 or 3 tables: 1) one giving the role of each attribute, 2) one giving the allowed children elements (when relevant), 3) one telling when each attribute is mandatory (MAND), optional (OPT)

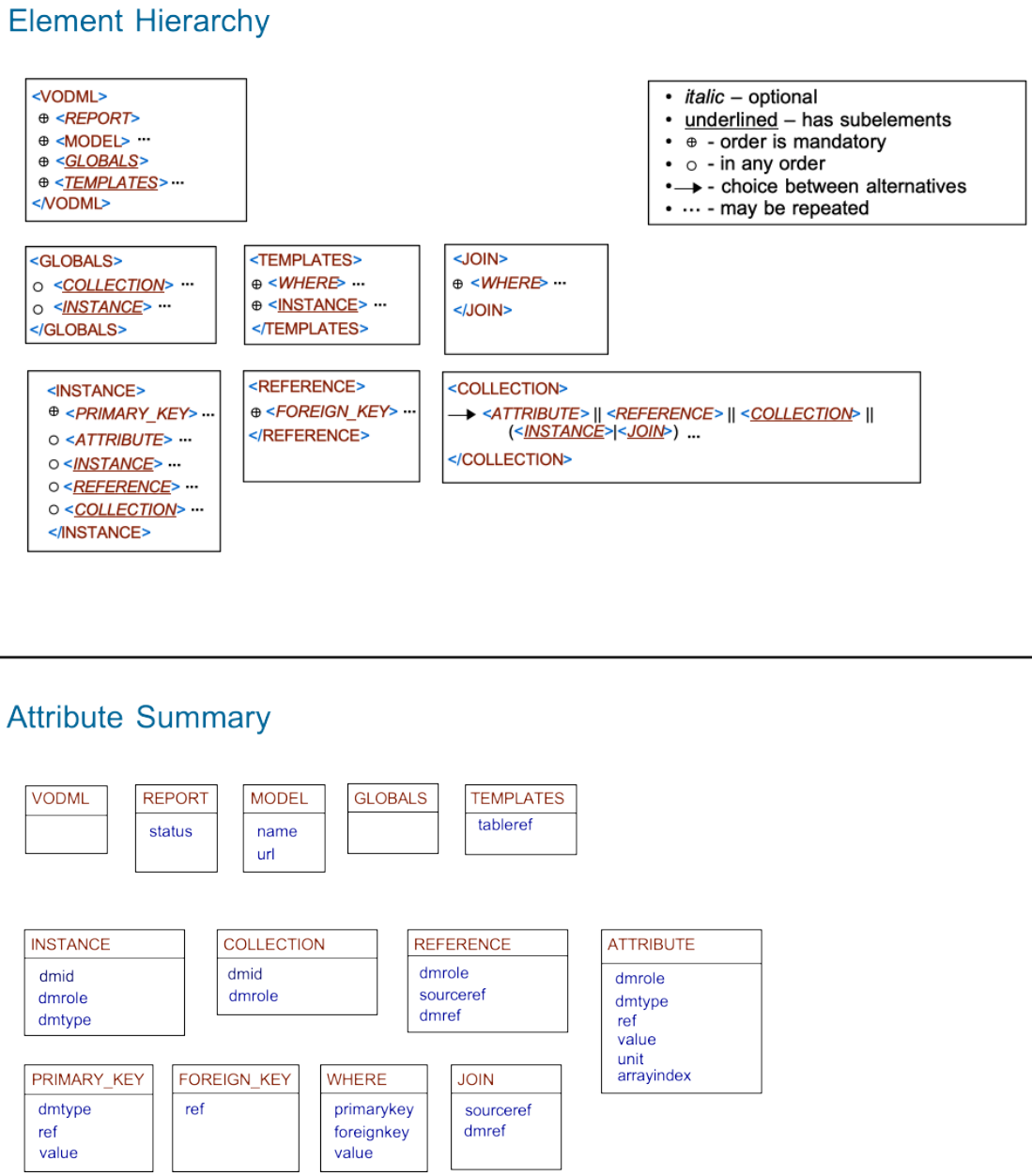

Figure 2: Annotation Syntax Summary.

or prohibited (NO). The content of these tables is a transcription of the XSD assertions.

In the following normative subsections, all mapping patterns are illustrated with XML snippets. Most of these excerpts are taken out of a complete VOTable listed in appendix A. The others are out of the

box, they refer to model elements or VOTable metadata that do not exist but that have been chosen to achieve better readability. Readers can have a look at https://github.com/ivoa-std/ModelInstanceInVot/tree/master/data-sample to inspect most of them in a context of real data.

4.3 VODML

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

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
  <REPORT status="OK"/>
  <MODEL> ... </MODEL>
  <GLOBALS> ... </GLOBALS>
  <TEMPLATES> ... </TEMPLATES>
   ...
</VODML>
```

Listing 2: Example VODML mapping block.

Element	Position	Occurs
REPORT	1	0-1
MODEL	2	0-*
GLOBALS	3	0-*
TEMPLATES	4	0-*

Table 3: Allowed children for VODML.

4.4 REPORT

Services providing annotated responses must use the REPORT element to tell the client whether the annotation process succeeded or not.

- REPORT is not mandatory.
- It must be the first VODML child if present.

```
 <VODML xmlns:dm-mapping="http://www.ivoa.net/xml/mivot" >
    <REPORT status="OK">Mapping compiled by hand</REPORT>
        <!--
           other annotations
         -->
</VODML>
```

Listing 3: REPORT example for an valid annotation (see line 34 in Appendix A).

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">

        <REPORT status="FAILED">
            Missing column for the declination, positions cannot be mapped.
        </REPORT>
        <!--
           No other annotation
         -->
</VODML>
```

Listing 4: REPORT example for an annotation failure.

Attribute	Role
@status	Status of the annotation process; must be either `OK` or `FAILED`

Table 4: `REPORT` attributes.

4.5 MODEL

A VOTable can provide serializations for an arbitrary number of data models. In order to declare which models are represented in the file, data providers must declare them through the `MODEL` elements. All models that define vodml-ids used in the annotation must be declared.

```
<MODEL name="ivoa"
       url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
<MODEL name="mango"
       url="https://github.com/ivoa-std/MANGO/blob/master/vo-dml/mango.vo-dml.xml" />
<MODEL name="cube"
       url="https://github.com/ivoa-std/Cube/vo-dml/Cube-1.0.vo-dml.xml" />
<MODEL name="ds"
       url="https://github.com/ivoa-std/DatasetMetadata/vo-dml/DatasetMetadata-1.0.vo-dml.xml" />
<MODEL name="coords"
       url="https://www.ivoa.net/xml/VODML/Coords-v1.vo-dml.xml" />
<MODEL name="meas"
       url="https://www.ivoa.net/xml/VODML/Meas-v1.vo-dml.xml" />
```

Listing 5: Example `MODEL` mapping block. (see line 36 in Appendix A)

Attribute	Role
`@name`	Name of the mapped model as declared in the VO-DML/XML model serialization. This attribute MUST not be empty. Its value is used to prefix the values of both `@dmrole` and `@dmtype` attributes of elements from that model.
`@url`	URL to the VO-DML/XML serialization of the model. If present, this attribute MUST not be empty.

Table 5: `MODEL` attributes.

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

Table 6: Valid attribute patterns for `MODEL`.

4.6 GLOBALS

The `GLOBALS` block holds model instances or collections of model instances that are not connected with any table column.

The instances or collections contained in `GLOBALS` have a global scope. They can be referenced by other instances anywhere in the mapping block. They must only contain literal values or reference to `PARAM` elements. They

must not refer to VOTable `FIELD` element.

Related instances may be grouped within `COLLECTION` blocks to enable selection via the MIVOT `JOIN` mechanism. See `COLLECTION` subsection for more details.

```
<VODML xmlns:dm-mapping="http://www.ivoa.net/xml/mivot" >
    <REPORT status="OK">Mapping compiled by hand</REPORT>
    <MODEL... />
    <!--
        place holder for singular data model instances not associated with a singular VOTabme TABLE
     -->
    <GLOBALS>
        <!--
          Container for the coordinate systems
         -->
        <COLLECTION dmid="IDCoordinateSystems" dmrole="" >
            ...
        </COLLECTION>
        ...
    </GLOBALS>
    ...
 </VODML>
```

Listing 6: Example `GLOBALS` block (see line 47 in Appendix A) which contains a collection of coordinate systems.

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

Table 7: Allowed children elements for `GLOBALS`.

4.7 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.

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
    ...
    <GLOBALS>
    ...
    </GLOBALS>
    <!--
       Mapping of the rows of the table Results
     -->
    <TEMPLATES tableref="Results">
        <INSTANCE dmid="IDtsIDdata" dmrole="" dmtype="cube:NDPoint">
            <COLLECTION dmrole="cube:NDPoint.observable">
                <INSTANCE dmtype="cube:Observable">
                    <ATTRIBUTE dmrole="cube:DataAxis.dependent" dmtype="ivoa:boolean" value="False"/>
                    ...
                </INSTANCE>
            </COLLECTION>
            ...
        </INSTANCE>
    </TEMPLATES>
  </TEMPLATES>
</VODML>
```

Listing 7: Example of a `TEMPLATES` block mapping the rows of the table `Results` (see line 204 in Appendix A). Each row of the table named `Results` is be mapped on an instance of the VO-DML type `cube:NDPoint`. Instances mapping a table row do not play any particular role.

Attribute	Role
`@tableref`	ID or name of the mapped VOTable `TABLE`. ID match takes precedence over name matches when resolving the reference.

Table 8: `TEMPLATES` attributes.

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

Table 9: Valid attribute patterns for `TEMPLATES`.

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

Table 10: Allowed children for `TEMPLATES`.

4.8 COLLECTION

`COLLECTION` is a container element to be used each time we have to gather items of the same kind. It is used in different contexts, each allowing a limited subset of elements for its content. `COLLECTION` items must all be of the same type. `COLLECTION` can be populated either by a static list of items (`INSTANCE`, `ATTRIBUTE`, ..) or by joining to another `COLLECTION` by using the `JOIN` statement.

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 `INSTANCE` for selection by `PRIMARY_KEY`, `FOREIGN_KEY` pairs.

   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 'id' column.

3. As child of COLLECTION

   The collection contains a matrix of atomic values.

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

Listing 8: Example of a `COLLECTION` child of `INSTANCE`. It must have a role as an enclosing `INSTANCE` component. In this example, the collection plays the role `model:Thing.elems` within the enclosing instance. The role is the actual VODMID of the corresponding VO-DML composition.

```
<GLOBALS>
    <!--
      Container for the datasets (one per timeseries)
      -->
    <COLLECTION dmid="IDDatasets" dmrole="">
        <INSTANCE dmid="IDds1" dmtype="ds:experiment.ObsDataset">
            <PRIMARY_KEY dmtype="ivoa:string" value="5813181197970338560"/>
            <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductType"
                       dmtype="ds:dataset.DataProductType" value="TIMESERIES"/>
            <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductSubtype"
                       dmtype="ivoa:string" value="GAIA Time Series"/>
            <ATTRIBUTE dmrole="ds:experiment.ObsDataset.calibLevel"
                       dmtype="ivoa:integer" value="1"/>
            <REFERENCE dmrole="ds:experiment.ObsDataset.target" dmref="IDtg1"/>
        </INSTANCE>

        <INSTANCE dmid="IDds1" dmtype="ds:experiment.ObsDataset">
            <PRIMARY_KEY dmtype="ivoa:string" value="5813181197970338561"/>
             ...
        </INSTANCE>
    </COLLECTION>
</GLOBALS>
```

Listing 9: Example of `COLLECTION` child of `GLOBALS` (see line 143 in Appendix A). It plays no specific role as being part of the `GLOBALS` elements.

```
<TEMPLATES tableref="IDPKTable">
    <INSTANCE dmid="IDTimeSeries" dmrole="" dmtype="cube:SparseCube">
        ....
        <COLLECTION dmrole="cube:SparseCube.data">
            <JOIN dmref="IDtsIDdata">
                <WHERE foreignkey="IDsrcid" primarykey="IDpksrcid" />
                <WHERE foreignkey="IDband" primarykey="IDpkband" />
            </JOIN>
         </COLLECTION>
```

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

Listing 10: Example of `COLLECTION` populated with a JOIN (see line 190 in Appendix A). The collection will be populated with the rows of another collection of the MIVOT block identified by `@dmid=IDtsIDdata` and that match both `WHERE` conditions.

```
<TEMPLATES>
  <INSTANCE dmtype="model:matrix_22">
    <COLLECTION dmrole="model:matrix">
        <COLLECTION>
                <ATTRIBUTE dmtype="ivoa:real" ref="field_11"/>
                <ATTRIBUTE dmtype="ivoa:real" ref="field_12"/>
        </COLLECTION>
        <COLLECTION>
                <ATTRIBUTE dmtype="ivoa:real" ref="field_21"/>
                <ATTRIBUTE dmtype="ivoa:real" ref="field_22"/>
        </COLLECTION>
    </COLLECTION>
  </INSTANCE>
</TEMPLATES>
```

Listing 11: Example of `COLLECTION` mapping a 2x2 matrix.

Attribute	**Role**
`@dmid`	Datamodel element id, MUST be unique within the mapping block.
`@dmrole`	Role of the `COLLECTION` in the data model.

Table 11: `XML attributes for COLLECTION` .

Context	**@dmid**	**@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. It MUST have non-empty `@dmid` to reference for ORM selection of contained `INSTANCE`. This occurs when e.g. the `COLLECTION` is child of `GLOBALS`
3	OPT	NO	The element maps a collection within another `COLLECTION`. `@dmrole` MUST be empty or not set. If present, `@dmid` MUST NOT be empty.

Table 12: Valid XML attribute patterns for `COLLECTION`.

Context: Child of INSTANCE		
Element	**Occurs**	
ATTRIBUTE	0-*	Collection of attributes.
REFERENCE	0-*	Collection of references.
INSTANCE	0-*	Collection of instances.
JOIN	0-1	Collection populated with a set of joined instances. No other child is supported in this case.
COLLECTION	0-*	Collection of collections.
Context: Child of GLOBALS		
Element	**Occurs**	
INSTANCE	0-*	Collection of related instances.
REFERENCE	0-*	Collection of related references.
JOIN	0-1	Collection populated with a set of joined instances. No other child is supported in this case.

Table 13: Allowed children for COLLECTION.

4.9 INSTANCE

The INSTANCE element defines a complex ObjectType or DataType. 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:

1. Child of GLOBALS

   In this case the INSTANCE is a single stand-alone instance which may or may not be referenced by other INSTANCE. This pattern is typically used for shared object e.g. space frames, or for head object of science product models e.g. Time Series. Here then INSTANCE

   - May have @dmid attribute, as possible target of REFERENCE ref
   - must have no @dmrole attribute or an empty one

2. Child of TEMPLATES

   In this case, the INSTANCE is a template for instances which are generated once per row of the associated table, which:

   - may have @dmid attribute, as target of JOIN @dmref
   - must have no @dmrole attribute or an empty one

3. 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 it:
  - must contain at least one `PRIMARY_KEY` sub-element
  - may have a `@dmid` attribute, as possible target of `REFERENCE`
  - must have no `@dmrole` attribute or an empty one. VO-DML roles are not associated with individual collection items.
- each member `INSTANCE` is a cell of an element with multiplicity > 1. Each one :
  - must have no `@dmrole` attribute or an empty one

4. Child of `INSTANCE`

   In this case, each `INSTANCE` represents a complex ObjectType or DataType playing a role in the parent `INSTANCE`. Then it

   - may have `@dmid` attribute
   - must have non-empty `@dmrole` attribute

5. any `INSTANCE`

   - must have non-empty `@dmtype` attribute
   - if `@dmid` attribute is present, it must not be empty

```
<INSTANCE dmid="IDtimesys" dmrole="" dmtype="coords:TimeSys">
  <PRIMARY_KEY dmtype="ivoa:string" value="TCB"/>
  <INSTANCE dmrole="coords:PhysicalCoordSys.frame"
            dmtype="coords:TimeFrame">
    <ATTRIBUTE dmrole="coords:TimeFrame.timescale"
               dmtype="ivoa:string" value="TCB" />
    <INSTANCE dmrole="coords:TimeFrame.refPosition"
              dmtype="coords:StdRefLocation">
      <ATTRIBUTE dmrole="coords:StdRefLocation.position"
                 dmtype="ivoa:string" value="BARYCENTER"/>
    </INSTANCE>
  </INSTANCE>
</INSTANCE>
```

Listing 12: Example of an `INSTANCE` child of `INSTANCE` (see line 56 in Appendix A). It must have a role as a component of the enclosing `INSTANCE`.

There one very specific case where a complex data type can be mapped as a single `ATTRIBUTE` (see section 4.10). VODML comes with a data model (`ivoa`) providing basic datatypes. 2 of those (`ivoa:IntQuantity` and `ivoa:RealQuantity`) that bind values with units. As VODML encourages to use those types, MIVOT accept them to be serialized an single `ATTRIBUTE`-s instead of `INSTANCE`-s enclosing 2 individual attributes (one for the value and one for the unit).

```
<INSTANCE dmrole="meas:Measure.coord" dmtype="coords:PhysicalCoordinate">
  <ATTRIBUTE dmrole="coords:PhysicalCoordinate.cval" dmtype="ivoa:RealQuantity"
             ref="_mag" unit="mag"/>
  <REFERENCE dmrole="coords:Coordinate.coordSys" sourceref="_CoordinateSystems">
```

```
        <FOREIGN\_KEY ref="_band"/>
    </REFERENCE>
</INSTANCE>
```

Listing 13: Example of complex data type (`ivoa:RealQuantity`) serialized as a single `ATTRIBUTE` (see line 226 in Appendix A). The attribute of type `ivoa:RealQuantity` is a shortcut for an `INSTANCE[@dmtype=ivoa:RealQuantity]` which has 2 attributes (value and unit)

The various XML attributes for an `INSTANCE` element, will have constraints depending on their usage.

Attribute	Role
`@dmid`	Element `@dmid` MUST be unique within the mapping block
`@dmrole`	`INSTANCE` role played in the DM
`@dmtype`	Class name in a model context

Table 14: `INSTANCE` XML attributes.

Element	Position	Occurs	
`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`. The Position column indicates the required position of the child element.

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

Table 16: Valid attribute patterns for `INSTANCE`.

4.10 ATTRIBUTE

The ATTRIBUTE element defines either a class attribute or a collection item, both set with atomic values. The requirements on the content (especially @ref and @dmrole, may differ depending on the usage:

1. Child of INSTANCE

   The ATTRIBUTE can be seen as a class attribute; it must have a @dmrole XML attribute.

   In this case, the ATTRIBUTE must be specified by:

   - @ref - reference to a VOTable PARAM or FIELD
   - @value - a literal
   - if both are provided; @value serves as the default if the reference cannot be resolved
   - if @ref cannot resolved and there is no @value, the client must consider that value as NULL in its software context (language, inner model).

2. Child of COLLECTION

   In this case the host COLLECTION can be seen as a vector and the ATTRIBUTE as one coordinate of the vector. It must have no @dmrole XML attribute or an empty one.

   In this case, the ATTRIBUTE must be specified by:

   - @ref - reference to a VOTable PARAM or FIELD
   - @value - a literal
   - if both are provided, @value serves as the default if the reference cannot be resolved
   - if @ref cannot resolved and there is no @value, the client must consider that value as NULL. The definition of the NULL value depends of the software environment

3. Any case :

   The ATTRIBUTE must always have a non-empty @dmtype XML attribute. This @dmtype may complete missing types in the VOTable. In case of inconsistency with VOTable @dmtype SHOULD supersede VOTable types. An ATTRIBUTE with a @ref pointing on a FIELD must be located in the TEMPLATES element mapping the table containing that field. @ref of ATTRIBUTE-s located in the GLOBALS block can only point at PARAM since the GLOBALS element does not map any data table.

MIVOT does not specify the way to handle NULL values. This remains in charge of the client implementation. If a `@ref` points onto a NULL `FIELD` value (in the VOTable meaning), the client has to set the corresponding model leaf as NULL as well. The way for doing it is very specific to its software context (language, inner model).

```
<INSTANCE dmtype="cube:Observable">
    <ATTRIBUTE dmrole="cube:DataAxis.dependent" dmtype="ivoa:boolean" value="False"/>
    <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:Time"
        <INSTANCE dmrole="meas:Measure.coord" dmtype="coords:MJD"
            <ATTRIBUTE dmrole="coords:MJD.date" dmtype="ivoa:real" ref="IDobstime"/>
            <REFERENCE dmrole="coords:Coordinate.coordSys" dmref="IDtimesys"/>
        </INSTANCE>
    </INSTANCE>
</INSTANCE>
```

Listing 14: Example of `ATTRIBUTE` set with either a column reference or a static value (see line 208 in Appendix A). `ATTRIBUTE[@dmrole=cube:DataAxis.dependent]` will always be set as "False" whereas `ATTRIBUTE[@dmrole=coords:MJD.date]` will be set with the value of the `IDobstime` column.

Attribute	Role
`@dmrole`	Role of the attribute in the DM
`@dmtype`	Type of the attribute in the DM
`@ref`	Reference or name of the `FIELD` or `PARAM` that has to be used to set the `ATTRIBUTE` value. In case of duplicate identifiers, which is possible with reference by name, the `FIELD` reference supercedes the `PARAM` one.
`@value`	Default `ATTRIBUTE` value. This value is taken if there is no `@ref` attribue or if `@ref` cannot be resolved.
`@unit`	`ATTRIBUTE` unit. Unit applicable to the attribute value which is given by either `@value` or `@ref`. `@unit` must always be compliant with VOUnit (Derriere and Gray et al., 2014). Empty `@unit` must be ignored. If the attribute value is set by a resolved `@ref`, `@unit` must be the VOUnit counterpart of the `FIELD` or `PARAM` unit; an error must be risen otherwise.
`@arrayindex`	Index of the native value to be taken to set the `ATTRIBUTE`. The value must be >= 0. 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 17: XML attributes of `ATTRIBUTE` .

@dmrole	@dmtype	@ref	@value	@arrayindex	Pattern
MAND	MAND	OPT	OPT	OPT	(a)
MAND	MAND	NO	MAND	OPT	(b)
NO	MAND	OPT	OPT	OPT	(c)

Table 18: Valid attribute patterns for `ATTRIBUTE`. (a) Valid in a `TEMPLATES` context. The `ATTRIBUTE` value must be set with the value of the element referenced by `@ref`. If the `@ref` can not be resolved and `@value` is present, `@value` must be taken. Either `@ref` or `@value` must be present or both. (b) This pattern is valid in any context. (c) is valid in the context of a `COLLECTION` item. The `ATTRIBUTE` value must be set with `@value` as `ATTRIBUTE` value.

4.11 REFERENCE

The `REFERENCE` element provides a reference to an `INSTANCE` or `COLLECTION` serialized elsewhere in the mapping block. Reference can be used in the same contexts as `INSTANCE`.

The different uses for the `REFERENCE` element are:

- Static reference: the element has a `@dmref` attribute that matches the `@dmid` attribute of the referenced `INSTANCE` or `COLLECTION`
- Dynamic reference: The element has a `@sourceref` attribute identifying the foreign collection (`TEMPLATES` or `COLLECTION`) containing the referenced `INSTANCE` which is identified by its `PRIMARY_KEY`(s) children. In this case, `REFERENCE` must be located in a `TEMPLATES` and it must have one or more `FOREIGN_KEY` children. If the foreign collection has multiple `INSTANCE`-s with matching `PRIMARY_KEY`-s, the first match must be taken by default. If the `REFERENCE` has multiple `FOREIGN_KEY`-s, they must be evaluated against the target `PRIMARY_KEY`-s in order of occurrence. An error must be risen if the number of `FOREIGN_KEY`-s does not match this of target `PRIMARY_KEY`-s.

```
<INSTANCE dmid="IDds1" dmrole="" dmtype="ds:experiment.ObsDataset">
    <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductType"
               dmtype="ds:dataset.DataProductType" value="TIMESERIES"/>
    <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductSubtype"
               dmtype="ivoa:string" value="GAIA Time Series"/>
    <ATTRIBUTE dmrole="ds:experiment.ObsDataset.calibLevel"
               dmtype="ivoa:integer" value="1"/>
    <REFERENCE dmrole="ds:experiment.ObsDataset.target" dmref="_tg1"/>
</INSTANCE>
```

Listing 15: Simple `REFERENCE`, to be replaced with the `INSTANCE` having `@dmid=_tg1` (see line 157 in Appendix A).

```
<INSTANCE dmrole="meas:Measure.coord" dmtype="coords:PhysicalCoordinate">
    ...
```

```
    <!--
        The photometric system is given by the item of the COLLECTION[dmid=IDCoordinateSystems]
        having a primary key matching the value of the column IDband for that particular row
     -->
    <REFERENCE dmrole="coords:Coordinate.coordSys" sourceref="_CoordinateSystems">
        <FOREIGN_KEY ref="_band"/>
    </REFERENCE>
</INSTANCE>
```

Listing 16: Dynamic `REFERENCE`, to be replaced with the `INSTANCE` of the collection `_CoordinateSystems` having a `PRIMARY_KEY` matching the value of the column `_band`. This pattern is valid in the context of a TEMPLATES (see line 234 in Appendix A).

The operation of dynamic `REFERENCE`-s is detailed in Appendix B.

Attribute	Role
`@dmrole`	Role of the referenced `INSTANCE` or `COLLECTION` in the DM
`@sourceref`	`@dmid` of the `COLLECTION` or `TEMPLATES` to be searched in dynamic reference case
`@dmref`	`@dmid` of the referenced `INSTANCE` or `COLLECTION`

Table 19: `REFERENCE` attributes.

Element	Position	Occurs	
`FOREIGN_KEY`	First	0-*	Foreign key to be used to resolve a dynamic reference.

Table 20: Allowed children for `REFERENCE`.

@dmrole	@sourceref	@dmref	Pattern
MAND	MAND	NO	This is the dynamic reference pattern, `@sourceref` gives the `@dmid` of the foreign collection to be searched. In this case `REFERENCE` must have at least one `FOREIGN_KEY` child and the foreign collection must have a corresponding `PRIMARY_KEY`
MAND	NO	MAND	Simple reference to either an `INSTANCE` or `COLLECTION`, usually located in the `GLOBALS`

Table 21: Valid attribute patterns for `REFERENCE`.

4.12 JOIN

The `JOIN` element populates a `COLLECTION` with `INSTANCE` elements from another collection, called hereafter the foreign collection. The foreign collection

can either be a static element (COLLECTION child of GLOBALS) or a collection of INSTANCE elements resulting from the iteration over a TEMPLATES.

The JOIN element must contain XML attributes identifying the foreign collection like @sourceref and/or @dmref . It can have WHERE children elements stating the join condition and it must be the unique child of the host COLLECTION .

JOIN may have 2 uses:

- JOIN with TEMPLATES data:
  - must have either a @sourceref attribute identifying the foreign TEMPLATES or a @dmref attribute identifying the foreign mapped INSTANCE or both.
    - if only @sourceref is given, the host collection is populated with the INSTANCE mapping the foreign table rows. This pattern is only valid when the foreign TEMPLATEs has one unique instance child.
    - if only @dmref is given, the host collection is populated with the INSTANCE having the matching @dmid. In this case, the client is in charge of locating the TEMPLATES containing that INSTANCE.
    - if both @sourceref and @dmref are given, it is assumed that the INSTANCE with @dmid matching @dmref is hosted by the TEMPLATES matching @sourceref; an error must be risen otherwise.
- JOIN with GLOBALS's COLLECTION data:
  - the foreign COLLECTION must be direct child of GLOBALS
  - JOIN must have one @sourceref attribute identifying the foreign COLLECTION
  - the host COLLECTION is populated with the foreign COLLECTION items eventually filtered by the WHERE condition.
  - the foreign COLLECTION must only have INSTANCE (eventually REFERENCE) as children.
  - the foreign COLLECTION items are supposed to have all the same @dmtype this rule is however out of the scope of the present standard.
  - If there are WHERE conditions, the foreign COLLECTION items must have PRIMARY_KEY

```
<JOIN dmref="IDtsIDdata">
    <WHERE foreignkey="_srcid" primarykey="_pksrcid" />
    <WHERE foreignkey="_band" primarykey="_pkband" />
</JOIN>
```

Listing 17: `JOIN example with 2 join conditions (see line 192 in Appendix A).` The joined condition is satisfied when the value of the column `_srcid` of the collection identified by `IDtsIDdata` matches the one of the `_band` column and the value of the column `_pkband` matches this of the `_pksrcid` column.

See more examples in the Appendix C.

Attribute	Role
`@sourceref`	Reference of the `TEMPLATES` or `COLLECTION` to be joined with.
`@dmref`	Reference of the foreign `INSTANCE`s that will populate the host `COLLECTION`

Table 22: XML attributes for a `JOIN`.

@sourceref	@dmref	Pattern
MAND	OPT	Join against the `TEMPLATES` or `COLLECTION` identified by `@sourceref`
OPT	MAND	Host `COLLECTION` populated with `INSTANCE` identified by `@dmref`

Table 23: Valid attribute patterns for `JOIN`.

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

Table 24: Allowed children for `JOIN`.

4.13 WHERE

The `WHERE` element is used to filter iteration outputs. Results are accepted when the key is equal to the specified `@value`.

- For now, we only define type-preserving comparisons; comparisons requiring type corrections will be defined in a future version of this standard and for now must raise an error.
- The `WHERE` statement is false if at least one of the values used for the comparison is NULL.

There are 2 different uses for this element:

1. 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`.

2. 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 either `@value` or `@primarykey` attribute. `@foreignkey` references the column of the foreign collection to be checked. The `WHERE` condition is satisfied for the rows having `@foreignkey` equals to either `@value` or `@primarykey` value.

```
<TEMPLATES tableref="table">
  <WHERE primarykey="col1" value="val1" />
  <WHERE primarykey="col2" value="val2" />
  <INSTANCE dmtype="type">
  ....
  </INSTANCE>
</TEMPLATES>
```

Listing 18: `WHERE` Example: only rows having `val1` as `col1` value and `val2` as `col2` value must be mapped.

```
<JOIN dmref="_ts_data">
    <WHERE foreignkey="_srcid" primarykey="_pksrcid" />
</JOIN>
```

Listing 19: `WHERE` Example: the join is satisfied when the value of the `_pksrcid` column is equal to the `_srcid` column of the foreign table (see line 194).

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 25: `WHERE` attributes.

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

Table 26: Valid attribute patterns for `WHERE`.

4.14 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 `REFERENCE` 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 the key must always be specified by the `@dmtype` attribute.

```
<INSTANCE dmtype="ds:experiment.ObsDataset">
    <PRIMARY_KEY dmtype="ivoa:string" value="5813181197970338560"/>
    ...
</INSTANCE>
```

Listing 20: The `INSTANCE` is identified within a `COLLECTION` by the `PRIMARY_KEY` value (see line 149 in Appendix A).

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 27: `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 28: Valid attribute patterns for `PRIMARY_KEY`.

4.15 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.

```
<REFERENCE dmrole="coords:Coordinate.coordSys" sourceref="_CoordinateSystems">
    <FOREIGN_KEY ref="IDband"/>
</REFERENCE>
```

Listing 21: The `REFERENCE` is resolved by the `INSTANCE` of the table `_CoordinateSystems` that has a primary key equals to the value of the column `_band` (see line 236 in Appendix A).

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

Table 29: `FOREIGN_KEY` attributes.

5 Schema and Validation

The MIVOT syntax is controlled by an XSD schema. This is part of this standard and available at http://ivoa.net/XML/MIVOT/mivot-v1.0.xsd. Implementations should always retrieve the schema from the namespace URI, http://www.ivoa.net/xml/mivot The XML schema conforms to XSD 1.1 (Shudi Gao, 2012) which enforces the syntax rules. The following features are validated:

- Element names
- Element attributes
- Element sequences
- Element ordering in specific sequences

In addition to this basic check, XSD 1.1 can assert which patterns are allowed or forbidden:

- Which elements are mandatory, optional or forbidden in the local context (parent and children elements).
- Which attributes are mandatory, optional or forbidden in the context of the host element.
- Attribute value requirements according to the context of the host element.

The scope of this schema-based validation is limited to the syntax however. The clients are still responsible for checking whether or not the attribute values are correct and for managing any semantic inconsistencies, for instance:

- Are references resolvable ?
- Do units conform to the requirements of the mapped models ?
- Is the mapping structure faithful to the models involved ? This however checked by the MIVOT validator (https://github.com/ivoa/mivot-validator).

6 Client APIs

The mapping syntax is pure XML. It can easily be processed by any XML package in any language. Although no API definition is part of the standard, the experience we got when exercising the syntax allows us to identify four processing levels.

1. Raw XML: The client resolves the reference and delivers an XML block corresponding to the searched models or model components. The client must then extract the search elements by using XPath queries applied to the the MIVOT namespace.

2. XML Wrapper class: The XPath strings mentioned above can be wrapped in generic objects providing accessors retrieving model nodes by selecting types or roles.

3. Model classes: For the most popular models and especially for the models that provide reusable components (Meas/Coord, PhotDM), the API can include objects instantiating model classes (e.g. Photometric filter).

4. Automatic extraction of class instances of common frameworks: One of the goal of the data annotation is to facilitate the connection between data and API code. This could be achieved with an API able to automatically build objects from the annotated data without asking the user to infer on meta data. Taking the Astropy example, a model-enable Python API should be able to build e.g. `Skycoord` instances in a transparent way and feed them with `FIELD` values from `TABLE` rows or from `PARAM`.

Level 1 and 2 may only be useful for client developpers, they should be hidden to the end user. Using MIVOT annotations makes sense only if the client code provides the users with a direct access to both modeled quantities and links that connect them. MIVOT is a machine level system that relies primarily on higher level APIs to provide representations of the models (level 3 and 4) to end users

The first feature that a user-friendly API processing MIVOT annotations should implement is the ability to transparently build from the annotation, the objects usually exposed to the user. These objects could be extended to support features made possible by the annotations such as e.g. the parameter grouping or the connection with calibration data.

7 Changes from Previous Versions

- The standards has been published as an IVOA WD under the MIVOT name in April 2022 (https://ivoa.net/documents/MIVOT/20220407/

index.html). Since that time, most of the changes were focused on the unite tests, the text and the client code.

- June 2022 (PR #124): The `dm-mapping:` prefix in MIVOT element has been dropped.
- December 2022 (PR #173): Allow `ATTRIBUTE` located in the `GLOBALS` block to refer to VOTable `PARAM`.
- December 2022 (MR #174): More flexible location of the MIVOT block. It must be enclosed in a `RESOURCE@type=meta` which must be enclosed in another `RESOURCE`.
- January 2023 :
  - (PR #179) Refine the allowed locations for the MIVOT block according to the issue #176.
  - (PR #184): Add appendix E (MIVOT and the registry).
  - (PR #185): Add some clarification on the client API, the TAP implementation and the NULL value for attributes.
- March 2023 :
  - (PR #187): Tell how dmrole are built.
  - (PR #189): Set more details in some snippet captions.
  - (PR #190): Add appendix F (Using VOMDL resources).
- April 2023 :
  - (PR #196): Extend appendix F with tips for going from VODML types to MIVOT objects.

A Complete VOTable

This listing shows an annotated light curve from Gaia DR2. It is based on a data file provided by the Gaia DPAC which mixes photometric points of multiple sources through different filters taken all at different time. This dataset has been reworked out and mapped by hand in a way to provide examples of mapping patterns applied to real data. The mapped models are the prototypes that were used in 2021 to develop the mapping syntax and the associated tooling. Models are however inspired by CubeDM, Measure, Coords and PhotDM. This sample must not be understood as a proposal for serializing light curves but as a demonstrator for annotating complex data structure.

Most of the XML snippets of the normative sections are taken out from this table. In this case, links connecting the text with the relevant locations in the listing have been setup.

The VOTable below contains many comments explaining how mapping patterns must be interpreted.

```
<?xml version='1.0' encoding='UTF-8'?>
<!--
   File: GAIA Multi-band Time Series

   Original file submitted as a use-case example for validating/testing the
   models and annotation syntax at the DM Workshop 2021.
      * https://github.com/ivoa/dm-usecases/blob/main/usecases/time-series/raw\_data/gaia_multiband.xml

   This file annotates that file according to the merged syntax proposal
   which resulted from the workshop effort.
      - migrated by hand from the VODML-Mapping syntax example.

   Notable Features:
     + uses ORM mechanism to select the appropriate coordinate system for each Observable.
     + uses JOIN to pull filtered set of records from the table to form the time series data.

-->

<VOTABLE version="1.4"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.3
    https://www.ivoa.net/xml/VOTable/v1.3">
  <DESCRIPTION>
  Epoch photometry. This table contains light curve data points.
  Each entry is a photometric flux-time pair for a given object, band and time.
  It follows the (evolving) VO convention for time series as much as possible.
  At the time of writing, a VO recommendation has not yet been released.
  </DESCRIPTION>
  <RESOURCE type="results">
    <RESOURCE type="meta">
      <VODML
        xmlns="http://www.ivoa.net/xml/mivot" >
        <!--  back to 4.4 -->
        <REPORT status="OK">Mapping compiled by hand</REPORT>
        <!--  back to 4.5 -->
        <MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
        <MODEL name="mango" url="https://github.com/ivoa-std/MANGO/blob/master/vo-dml/mango.vo-dml.xml" />
        <MODEL name="cube" url="https://github.com/ivoa-std/Cube/vo-dml/Cube-1.0.vo-dml.xml" />
        <MODEL name="ds" url="https://github.com/ivoa-std/DatasetMetadata/vo-dml/DatasetMetadata-1.0.vo-dml.xml" />
        <MODEL name="coords" url="https://www.ivoa.net/xml/VODML/Coords-v1.vo-dml.xml" />
        <MODEL name="meas" url="https://www.ivoa.net/xml/VODML/Meas-v1.vo-dml.xml" />
        <!--
            place holder for singular data model instances not associated with a singular V
            OTable TABLE
          -->
        <!--  back to 4.6 -->
        <GLOBALS>
          <!--
              Container for the coordinate systems
            -->
```

```
<COLLECTION dmid="\_CoordinateSystems" dmrole="" >
  <!--
      Time coordinate system
    -->
  <!--  back to 4.9 -->
  <INSTANCE dmid="\_timesys" dmrole="" dmtype="coords:TimeSys">
    <PRIMARY\_KEY dmtype="ivoa:string" value="TCB"/>
    <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:TimeFrame">
      <ATTRIBUTE dmrole="coords:TimeFrame.timescale" dmtype="ivoa:string" value="TCB" />
      <INSTANCE dmrole="coords:TimeFrame.refPosition" dmtype="coords:StdRefLocation">
        <ATTRIBUTE dmrole="coords:StdRefLocation.position"
                   dmtype="ivoa:string" value="BARYCENTER"/>
      </INSTANCE>
    </INSTANCE>
  </INSTANCE>
  <!--
      Space coordinate system
    -->
  <INSTANCE dmid="\_spacesys1" dmrole="" dmtype="coords:SpaceSys">
    <PRIMARY\_KEY dmtype="ivoa:string" value="ICRS"/>
    <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:SpaceFrame">
      <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame"
                 dmtype="ivoa:string" value="ICRS"/>
      <ATTRIBUTE dmrole="coords:SpaceFrame.equinox"
                 dmtype="coords:Epoch" value="J2015.5"/>
    </INSTANCE>
  </INSTANCE>
  <!--
      Photometric coordinate system: At the time of writing, PhotDM was not available
      in VODML.
      We use a class of the first MANGO draft
    -->
  <INSTANCE dmid="\_photsys\_G" dmtype="mango:coordinates.PhotometryCoordSys">
    <PRIMARY\_KEY dmtype="ivoa:string" value="G"/>
    <INSTANCE dmrole="coords:PhysicalCoordSys.frame"
              dmtype="mango:coordinates.PhotFilter">
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.name"
                 dmtype="ivoa:string" value="GAIA/GAIA2r.G"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.zeroPointFlux"
                 dmtype="ivoa:RealQuantity" value="2.49524e-9"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.magnitudeSystem"
                 dmtype="ivoa:string" value="Vega"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.effectiveWavelength"
                 dmtype="ivoa:RealQuantity" value="6246.77"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.unit"
                 dmtype="ivoa:Unit" value="Angstrom" />
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.bandWidth"
                 dmtype="ivoa:real" value="4578.32"/>
    </INSTANCE>
  </INSTANCE>
  <INSTANCE dmid="\_photsys\_RP" dmrole=""
              dmtype="mango:coordinates.PhotometryCoordSys">
    <PRIMARY\_KEY dmtype="ivoa:string" value="RP"/>
    <INSTANCE dmrole="coords:PhysicalCoordSys.frame"
              dmtype="mango:coordinates.PhotFilter">
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.name"
                 dmtype="ivoa:string" value="GAIA/GAIA2r.Grp"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.zeroPointFlux"
                 dmtype="ivoa:RealQuantity" value="1.29363e-9"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.magnitudeSystem"
                 dmtype="ivoa:string" value="Vega"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.effectiveWavelength"
                 dmtype="ivoa:RealQuantity" value="7740.87"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.unit"
                 dmtype="ivoa:Unit" value="Angstrom"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.bandWidth"
                 dmtype="ivoa:real" value="2943.72"/>
    </INSTANCE>
  </INSTANCE>
  <INSTANCE dmid="\_photsys\_BP" dmrole="" dmtype="mango:coordinates.PhotometryCoordSys">
    <PRIMARY\_KEY dmtype="ivoa:string" value="BP"/>
    <INSTANCE dmrole="coords:PhysicalCoordSys.frame"
              dmtype="mango:coordinates.PhotFilter">
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.name"
                 dmtype="ivoa:string" value="GAIA/GAIA2r.Gbp"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.zeroPointFlux"
                 dmtype="ivoa:RealQuantity" value="4.03528e-9"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.magnitudeSystem"
                 dmtype="ivoa:string" value="Vega"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.effectiveWavelength"
                 dmtype="ivoa:RealQuantity" value="5278.58"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.unit"
                 dmtype="ivoa:Unit" value="Angstrom"/>
      <ATTRIBUTE dmrole="mango:coordinates.PhotFilter.bandWidth"
```

```
                dmtype="ivoa:real" value="2279.45"/>
        </INSTANCE>
      </INSTANCE>
    </COLLECTION>
    <!--
        Container for the datasets (one per timeseries)
      -->
    <!-- back to 4.8 -->
    <COLLECTION dmid="\_Datasets" dmrole="">
      <INSTANCE dmid="\_ds1" dmrole="" dmtype="ds:experiment.ObsDataset">
        <!--
            The primary keys of that collection items are set by the mapping
          -->
        <!-- back to 4.14 -->
        <PRIMARY\_KEY dmtype="ivoa:string" value="5813181197970338560"/>
        <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductType"
                dmtype="ds:dataset.DataProductType" value="TIMESERIES"/>
        <ATTRIBUTE dmrole="ds:dataset.Dataset.dataProductSubtype"
                dmtype="ivoa:string" value="GAIA Time Series"/>
        <ATTRIBUTE dmrole="ds:experiment.ObsDataset.calibLevel"
                dmtype="ivoa:integer" value="1"/>
        <!-- back to 4.11 -->
        <REFERENCE dmrole="ds:experiment.ObsDataset.target" dmref="\_tg1"/>
      </INSTANCE>
    </COLLECTION>
    <!--
        Time series target
        Value set as a literal by the mapping
      -->
    <INSTANCE dmid="\_tg1" dmrole="" dmtype="ds:experiment.Target">
      <ATTRIBUTE dmrole="ds:experiment.BaseTarget.name"
              dmtype="ivoa:string" value="5813181197970338560"/>
    </INSTANCE>
  </GLOBALS>
  <!--
      Mapping of the rows of the table \_PKTable
    -->
  <TEMPLATES tableref="\_PKTable">
    <INSTANCE dmid="\_TimeSeries" dmrole="" dmtype="cube:SparseCube">
      <!--
          The dataset object is the item of the COLLECTION[dmid=\_Datasets]
          having a primary key matching the value of the column \_pksrcid
          for that particular row
        -->
      <REFERENCE dmrole="cube:DataProduct.dataset" sourceref="\_Datasets">
        <FOREIGN\_KEY ref="\_pksrcid"/>
      </REFERENCE>
      <!--
          Sparse cube data are given by the INSTANCE[dmid=\_ts\_data]
          These instances are located in the Results table (must be found by the client)
          The matching row are identified by a double join condition
              \_PKTable[\_pksrcid] = Results[\_srcid]
              \_PKTable[\_pkband] = Results[\_band]
        -->
      <!-- back to 4.8 -->
      <COLLECTION dmrole="cube:SparseCube.data">
        <!-- back to 4.12 -->
        <JOIN dmref="\_ts\_data">
          <!-- back to 4.13 -->
          <WHERE foreignkey="\_srcid" primarykey="\_pksrcid" />
          <WHERE foreignkey="\_band" primarykey="\_pkband" />
        </JOIN>
      </COLLECTION>
    </INSTANCE>
  </TEMPLATES>
  <!--
      Mapping of the rows of the table Results
    -->
  <!-- back to 4.7 -->
  <TEMPLATES tableref="Results">
    <INSTANCE dmid="\_ts\_data" dmrole="" dmtype="cube:NDPoint">
      <COLLECTION dmrole="cube:NDPoint.observable">
        <!-- back to 4.10 -->
        <INSTANCE dmtype="cube:Observable">
          <ATTRIBUTE dmrole="cube:DataAxis.dependent" dmtype="ivoa:boolean" value="False"/>
          <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:Time">
            <INSTANCE dmrole="meas:Measure.coord" dmtype="coords:MJD">
              <ATTRIBUTE dmrole="coords:MJD.date" dmtype="ivoa:real" ref="\_obstime"/>
              <!--
                  The time coodinate is givel by the INSTANCE[dmid=\_timesys]
                  This instance must be located by the client
                -->
              <REFERENCE dmrole="coords:Coordinate.coordSys" dmref="\_timesys"/>
            </INSTANCE>
```

```
            </INSTANCE>
          </INSTANCE>
          <INSTANCE dmtype="cube:Observable">
            <ATTRIBUTE dmrole="cube:DataAxis.dependent" dmtype="ivoa:boolean" value="True"/>
            <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:GenericMeasure">
              <INSTANCE dmrole="meas:Measure.coord" dmtype="coords:PhysicalCoordinate">
               <!--  back to 4.9 -->
               <ATTRIBUTE dmrole="coords:PhysicalCoordinate.cval" dmtype="ivoa:RealQuantity"
                          ref="\_mag" unit="mag"/>
                 <!--
                    The photometric system is given by the item of the
                    COLLECTION[dmid=\_CoordinateSystems] having a primary key matching
                    the value of the column \_band for that particular row
                  -->
                <!--  back to 4.11 -->
                <REFERENCE dmrole="coords:Coordinate.coordSys" sourceref="\_CoordinateSystems">
                  <!--  back to 4.15 -->
                  <FOREIGN\_KEY ref="\_band"/>
                </REFERENCE>
              </INSTANCE>
            </INSTANCE>
          </INSTANCE>
          <INSTANCE dmtype="cube:Observable">
            <ATTRIBUTE dmrole="cube:DataAxis.dependent" dmtype="ivoa:boolean" value="True"/>
            <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:GenericMeasure">
              <INSTANCE dmrole="meas:Measure.coord" dmtype="coords:PhysicalCoordinate">
                <ATTRIBUTE dmrole="coords:PhysicalCoordinate.cval" dmtype="ivoa:RealQuantity"
                           ref="\_flux" unit="e-/s" />
                 <!--
                    The photometric system is given by the item of
                    the COLLECTION[dmid=\_CoordinateSystems] having a primary key matching
                    the value of the column \_band for that particular row
                  -->
                <REFERENCE dmrole="coords:Coordinate.coordSys" sourceref="\_CoordinateSystems">
                  <FOREIGN\_KEY ref="\_band"/>
                </REFERENCE>
              </INSTANCE>
              <INSTANCE dmrole="meas:Measure.error" dmtype="meas:Error">
                <INSTANCE dmrole="meas:Error.statError" dmtype="meas:Symmetrical">
                  <ATTRIBUTE dmrole="meas:Symmetrical.radius" dmtype="ivoa:RealQuantity"
                             ref="\_fluxerr" unit="e-/s" />
                </INSTANCE>
              </INSTANCE>
            </INSTANCE>
          </INSTANCE>
        </COLLECTION>
      </INSTANCE>
    </TEMPLATES>
  </VODML>
  <!--
  mapping block end
-->
</RESOURCE>
<TABLE name="\_PKTable">
  <FIELD ID="\_pksrcid" datatype="char" arraysize="*" name="pksrcid">
    <DESCRIPTION>Source \_</DESCRIPTION>
  </FIELD>
  <FIELD ID="\_pkband" name="pkband" datatype="char" arraysize="*">
    <DESCRIPTION>Photometric Band</DESCRIPTION>
  </FIELD>
  <DATA>
    <TABLEDATA>
      <TR>
        <TD>5813181197970338560</TD>
        <TD>G</TD>
      </TR>
      <TR>
        <TD>5813181197970338560</TD>
        <TD>BP</TD>
      </TR>
      <TR>
        <TD>5813181197970338560</TD>
        <TD>RP</TD>
      </TR>
    </TABLEDATA>
  </DATA>
</TABLE>
<TABLE name="Results">
  <FIELD ID="\_srcid" datatype="char" arraysize="*"
         name="source\_id" utype="meta.id;meta.main">
    <DESCRIPTION>Source Id.</DESCRIPTION>
  </FIELD>
  <FIELD datatype="long" name="transit\_id" ucd="meta.version">
    <DESCRIPTION>Transit unique identifier.</DESCRIPTION>
```

```
</FIELD>
<FIELD ID="\_band" arraysize="*" datatype="char"
       name="band" ucd="instr.bandpass" utype="ssa:Data\_.Bandpass">
  <DESCRIPTION>Photometric band. Values: G .</DESCRIPTION>
</FIELD>
<FIELD ID="\_obstime" datatype="double" name="time" ucd="time.epoch;VOX:Image\_MJDateObs"
       unit="d" utype="spec:Spectrum.Data.TimeAxis.Value">
  <DESCRIPTION>Different times are defined for each band. </DESCRIPTION>
</FIELD>
<FIELD ID="\_mag" datatype="float" name="mag" ucd="phot.mag;em.opt" unit="mag">
  <DESCRIPTION>Vega magnitude</DESCRIPTION>
</FIELD>
<FIELD ID="\_flux" datatype="float" name="flux" ucd="em.opt;phot.flux;stat.mean"
       unit="e-/s" utype="spec:Spectrum.Data.SpectralAxis.Value">
  <DESCRIPTION>Band flux value for the transit. </DESCRIPTION>
</FIELD>
<FIELD ID="\_fluxerr" datatype="float" name="flux\_error"
       ucd="em.opt;phot.flux;stat.error" unit="e-/s">
  <DESCRIPTION>Flux error.</DESCRIPTION>
</FIELD>
<FIELD datatype="float" name="flux\_over\_error" ucd="em.opt;phot.flux;stat.error">
  <DESCRIPTION>Band flux divided by its error.</DESCRIPTION>
</FIELD>
<FIELD datatype="boolean" name="rejected\_by\_photometry" ucd="meta.code.status">
  <DESCRIPTION>Rejected by DPAC photometry processing.</DESCRIPTION>
</FIELD>
<FIELD datatype="boolean" name="rejected\_by\_variability" ucd="meta.code.status">
  <DESCRIPTION>Rejected by DPAC variability processing.</DESCRIPTION>
</FIELD>
<FIELD datatype="long" name="other\_flags" ucd="meta.code.status"></FIELD>
<FIELD datatype="long" name="solution\_id" ucd="meta.version">
  <DESCRIPTION>Solution identifier. </DESCRIPTION>
</FIELD>
<DATA>
  <TABLEDATA>
    <TR>
      <TD>5813181197970338560</TD>
      <TD>17091923342681275</TD>
      <TD>G</TD>
      <TD>1705.9437360200984</TD>
      <TD>15.216574774452164</TD>
      <TD>15442.456273273616</TD>
      <TD>44.151258712309364</TD>
      <TD>349.76254</TD>
      <TD>F</TD>
      <TD>F</TD>
      <TD>4097</TD>
      <TD>369295551293819386</TD>
    </TR>
    <TR>
      <TD>5813181197970338560</TD>
      <TD>17096015648964756</TD>
      <TD>G</TD>
      <TD>1706.0177100217386</TD>
      <TD>14.767336693604877</TD>
      <TD>23356.70699319823</TD>
      <TD>33.53035403015752</TD>
      <TD>696.584</TD>
      <TD>F</TD>
      <TD>F</TD>
      <TD>4194817</TD>
      <TD>369295551293819386</TD>
    </TR>
    <TR>
      <TD>5813181197970338560</TD>
      <TD>19103616164443503</TD>
      <TD>G</TD>
      <TD>1742.3215763366886</TD>
      <TD>15.278342999137502</TD>
      <TD>14588.447956240941</TD>
      <TD>15.054069973748831</TD>
      <TD>969.07</TD>
      <TD>F</TD>
      <TD>F</TD>
      <TD>1</TD>
      <TD>369295551293819386</TD>
    </TR>
    <TR>
      <TD>5813181197970338560</TD>
      <TD>19107708466271149</TD>
      <TD>G</TD>
      <TD>1742.3955784801215</TD>
      <TD>14.799456395369246</TD>
      <TD>22675.8581669284</TD>
```

```
    <TD>36.37833986932219</TD>
    <TD>623.33405</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>1</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>17091923342681275</TD>
    <TD>BP</TD>
    <TD>1705.9440504175118</TD>
    <TD>15.64539174200359</TD>
    <TD>7627.787250948564</TD>
    <TD>111.47726591016318</TD>
    <TD>68.4246</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>0</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>17096015648964756</TD>
    <TD>BP</TD>
    <TD>1706.0180527092407</TD>
    <TD>14.945026143036898</TD>
    <TD>14539.343967569965</TD>
    <TD>145.0521624616696</TD>
    <TD>100.23528</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>0</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>19103616164443503</TD>
    <TD>BP</TD>
    <TD>1742.3218911236327</TD>
    <TD>15.530661923152225</TD>
    <TD>8477.94313079077</TD>
    <TD>91.74578776556334</TD>
    <TD>92.40689</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>0</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>17091923342681275</TD>
    <TD>RP</TD>
    <TD>1705.9441391177577</TD>
    <TD>14.76056670662418</TD>
    <TD>10012.471954702154</TD>
    <TD>101.25870322095649</TD>
    <TD>98.88011</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>0</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>17096015648964756</TD>
    <TD>RP</TD>
    <TD>1706.018140557839</TD>
    <TD>14.422572989291783</TD>
    <TD>13669.064712119256</TD>
    <TD>123.67648827643211</TD>
    <TD>110.52274</TD>
    <TD>F</TD>
    <TD>F</TD>
    <TD>0</TD>
    <TD>369295551293819386</TD>
  </TR>
  <TR>
    <TD>5813181197970338560</TD>
    <TD>19103616164443503</TD>
    <TD>RP</TD>
    <TD>1742.3219778490015</TD>
    <TD>14.820081408614794</TD>
    <TD>9478.408715049112</TD>
```

```
            <TD>96.68089453665782</TD>
            <TD>98.03807</TD>
            <TD>F</TD>
            <TD>F</TD>
            <TD>0</TD>
            <TD>3692955512938193 86</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

Listing 22: Gaia multiband example. Notice that due to a Latex tweak, all _ characters are prefixed with a \

B Dynamic References

The example below illustrates how `GLOBALS` objects can be referenced from a `TEMPLATES`.

```
<GLOBALS>
  <COLLECTION dmid="_Filters">
    <INSTANCE dmtype="photdm:Filter">
              <PRIMARY_KEY dmtype="ivoa:string" value="TCB" />
              ...
        </INSTANCE>
        <INSTANCE dmtype="photdm:Filter">
              <PRIMARY_KEY dmtype="ivoa:string" value="B" />
              ...
        </INSTANCE>
  </COLLECTION>
</GLOBALS>

<TEMPLATES tableref="Results">
  ....
  <REFERENCE dmrole="model:Class.filter" sourceref="_Filters">
    <FOREIGN_KEY ref="_band" />
  </REFERENCE>
  ...
</TEMPLATES>
```

Listing 23: Dynamic reference example.

In this example, each object mapped in the `TEMPLATES` references the filter for which the primary key equals to value of the `_band` column.

The global `COLLECTION @dmid=_Filters` can be seen as a static data table being part of the mapping (not of the native data).

C Join Examples

In listing 24, the `GLOBALS` collection having `cube:SparseCube.data` as role is populated with instances of another `TEMPLATES` (or global `COLLECTION`)

- The joined `TEMPLATES` is the one containing the `INSTANCE` whose `@dmid` equals '_ts_data'
- The joined items are those matching the condition `Results[_band]='G'`

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
  ....
  <GLOBALS>
    <COLLECTION dmid="_SparseCubes">
      <INSTANCE dmid="_TimeSeries" dmrole="" dmtype="cube:SparseCube">
        <REFERENCE dmrole="cube:DataProduct.dataset" dmref="_ds1" />

        <COLLECTION dmrole="cube:SparseCube.data">
          <JOIN dmref="_ts_data">
            <WHERE value="G" primarykey="_band" />
          </JOIN>
        </COLLECTION>

      </INSTANCE>
    </COLLECTION>
  </GLOBALS>

  <TEMPLATES tableref="Results">
    <INSTANCE dmid="_ts_data" dmtype="cube:NDPoint">
      ....
      ....
    </INSTANCE>
  </TEMPLATES>
</VODML>
```

Listing 24: Joining a global `COLLECTION` with a `TEMPLATES` identified by a `(@dmid,@dmref)` pair.

Listing 25 below shows another way to map the same join but by using a `@sourceref`.

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
  ....
  <GLOBALS>
    <COLLECTION dmid="_SparseCubes">
      <INSTANCE dmid="_TimeSeries" dmrole="" dmtype="cube:SparseCube">
        <REFERENCE dmrole="cube:DataProduct.dataset" dmref="_ds1" />

        <COLLECTION dmrole="cube:SparseCube.data">
          <JOIN sourceref="Results">
            <WHERE value="G" primarykey="_band" />
          </JOIN>
        </COLLECTION>

      </INSTANCE>
    </COLLECTION>
  </GLOBALS>

  <TEMPLATES tableref="Results">
    <INSTANCE dmid="_ts_data" dmtype="cube:NDPoint">
      ....
      ....
    </INSTANCE>
  </TEMPLATES>
</VODML>
```

Listing 25: Joining a global `COLLECTION` with a `TEMPLATES` identified by a `@sourceref`.

This pattern works since the joined `TEMPLATES` has only one child. If it had more than one child, the mapped instances would have to be identified by `(@dmid,@dmref)` pairs as shown in listing 26.

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
  ....
  <GLOBALS>
    <COLLECTION dmid="_SparseCubes">
      <INSTANCE dmid="_TimeSeries" dmrole="" dmtype="cube:SparseCube">
        <REFERENCE dmrole="cube:DataProduct.dataset" dmref="_ds1" />

        <COLLECTION dmrole="cube:SparseCube.data">
          <JOIN sourceref="Results" dmref="_ts_data">
            <WHERE value="G" primarykey="_band" />
          </JOIN>
        </COLLECTION>
```

```
      </INSTANCE>
    </COLLECTION>
  </GLOBALS>

  <TEMPLATES tableref="Results">
    <INSTANCE dmid="_ts_data" dmtype="cube:NDPoint">
      ....
    </INSTANCE>
    <INSTANCE dmid="_other_data" dmtype="cube:OtherType">
      ....
    </INSTANCE>
  </TEMPLATES>
</VODML>
```

Listing 26: Joining a `TEMPLATES` with a global `COLLECTION` identified by both `@sourceref` and `(@dmid,@dmref)` pairs.

In the example 27, the collection having `cube:SparseCube.data` as role is populated with instances of another `TEMPLATES`

- The joined `TEMPLATES` is the one containing the `INSTANCE` whose `@dmid` equals '_ts_data'
- The joined items are those matching the condition `_PKTable[_pksrcid]=Results[_srcid] AND _PKTable[_PKband]=Results[_band]`

```
<VODML xmlns="http://www.ivoa.net/xml/mivot">
  ....
  <GLOBALS>
  ....
  ....
  </GLOBALS>

  <TEMPLATES tableref="_PKTable">
    <INSTANCE dmid="_TimeSeries" dmrole="" dmtype="cube:SparseCube">
      <COLLECTION dmrole="cube:SparseCube.data">
        <JOIN dmref="_ts_data">
          <WHERE foreignkey="_srcid" primarykey="_pksrcid" />
          <WHERE foreignkey="_band" primarykey="_pkband" />
        </JOIN>
      </COLLECTION>
    </INSTANCE>
  </TEMPLATES>

  <TEMPLATES tableref="Results">
    <INSTANCE dmid="_ts_data" dmtype="cube:NDPoint">
      ....
      ....
    </INSTANCE>
  </TEMPLATES>
</VODML>
```

Listing 27: Joining two `TEMPLATES` together with `(@dmid,@dmref)` pairs.

D TAP and the data models

The aim of the mapping syntax is to enable services to associate a model view to searched data including legacy data. The usage of such model views can range for a simple enhancement of the metadata up to the representation of full model instances. A step toward a better DM integration in the VO consists in enabling services to annotate legacy data by providing complete model views. To automatically generate model annotations, the server must

check that the selected columns match the model definitions and thus can be mapped on that model. To operate the mapping, the server needs further information such as coordinate frames and data profile resources giving the binding between table columns and model leaves. A prototype (Louys and Michel et al., 2022) implementing this feature has been demonstrated at the ADASS conference in 2021.

This prototype underlined the lack of a query pattern able to tell users which columns must be selected to get a complete model view on the table rows. The safest solution to get that view is to query all columns with `MAXREC=0`. This would provide the user with a quick look on the table columns that are required to build instances of the model(s) of interest. This feature could also be handled by a registry extension as mentioned in Appendix E.

E Registering the MIVOT capability

MIVOT adds a new feature to the VOTable format which allows a new way to consume data. The standard was designed so that this new functionality could be ignored by regular clients, as services are not required to provide annotated data anyway. This flexibility requires a standard way to declare the availability of annotated data. The registration of this capability must be applicable to any service running any protocol that possibly serves data as VOTables.

In this context it is important to be able inform clients in advance that a service can deliver annotated data. We identified 2 ways to do this:

- Defining a specific mime-type extension allowing clients to tell the server that annotated responses are accepted if available. This has been exercised with the XTapDB (`https://xcatdb.unistra.fr/xtapdb/`) service which understands the `application/mango` ad-hoc accepted mime type as a request for mapping the query results on the Mango data model (still a draft at the time of writing.). The limitation if this approach is that one mime-type has to be defined per model or if we do not want to specify any specific model, e.g. `mime-type:application/mivot` clients will have to parse the VOTable to discover which models are mapped.

- The other option is to define a new IVOA Registry capability (see VOResource, Plante and Demleitner et al. (2018)) that would list the mapped models and that could be appended to the service capabilities.

It is to be noted that these 2 options can be mixed together. The definitions of both capability and mime type are out of the scope of this standard which is focused on the syntax.

F Tips for the VODML to MIVOT translation

The VOMDL standard has been designed to provide machine readable descriptions of data models. This feature makes MIVOT working for any model element described in this language. For serializing data model instances, it is therefore strongly recommended to build the model annotations directly from VODML resources.

IVOA models are released with the following resources:

- The standard specification document (pdf)
- The VODML specification (model-vx.y.vodml.xml)
- An HTML representation of the model generated from the VODML file using a dedicated style sheet.

Both PDF and HTML documents are accessible through the standard landing page on `https://ivoa.net/documents/index.html`. VODML model description files are accessible from `https://ivoa.net/xml/index.html`.

This document does not specify algorithms converting VODML objects into MIVOT elements. However, the snippets in the following listing illustrate the translation process for a simple PhotDM class.

```
<objectType>
  <vodml-id>Access</vodml-id>
  <name>Access</name>
  <description>Gathers all properties to access a resource : uri, format and size .
  </description>
  <attribute>
    <vodml-id>Access.reference</vodml-id>
    <name>reference</name>
    <description>URI to access the resource.</description>
    <datatype>
      <vodml-ref>ivoa:anyURI</vodml-ref>
    </datatype>
    <multiplicity>
      <minOccurs>1</minOccurs>
      <maxOccurs>1</maxOccurs>
    </multiplicity>
  </attribute>
  <attribute>
    <vodml-id>Access.size</vodml-id>
    <name>size</name>
    <description>Approximate estimated size of the dataset, specified in kilobytes.</description>
    <datatype>
      <vodml-ref>ivoa:integer</vodml-ref>
    </datatype>
    <multiplicity>
      <minOccurs>1</minOccurs>
      <maxOccurs>1</maxOccurs>
    </multiplicity>
  </attribute>
  <attribute>
    <vodml-id>Access.format</vodml-id>
    <name>format</name>
    <description>Format of the accessed resource. </description>
    <datatype>
      <vodml-ref>ivoa:string</vodml-ref>
    </datatype>
    <multiplicity>
      <minOccurs>1</minOccurs>
      <maxOccurs>1</maxOccurs>
    </multiplicity>
  </attribute>
</objectType>
```

Listing 28: VODML representation of the PhotDM class `Access`. This is an object type with 3 attributes, each one with a cardinality equal to 1. At this stage, we do not know whether attributes are typed complex or primitive type. This will be completed later on by going through their own VODML definitions.

Basically the VODML to MIVOT translation obeys the following rules:

- a VODML <objectType> or <dataType> is converted into a MIVOT `INSTANCE`.
- The VODML <datatype> of the ivoa model element is taken to set the xml attribute `@dmtype` into a MIVOT `ATTRIBUTE` or `INSTANCE`.
- The VODML <vodml-id> of the ivoa model element is taken to set the xml attribute `@dmrole` into a MIVOT `COLLECTION`, `ATTRIBUTE` or `INSTANCE`.
- a VODML <attribute> with a cardinality equal to 1 and a primitive type is converted into a MIVOT `ATTRIBUTE`.
- a VODML <attribute> with a cardinality greater than 1 is enclosed into a MIVOT `COLLECTION`.
- a VODML <composition> with a cardinality greater than 1 is enclosed into a MIVOT `COLLECTION`.

```
<INSTANCE dmrole="Phot:TransmissionCurve.access" dmtype="Phot:Access">
  <ATTRIBUTE dmrole="Phot:Access.reference" dmtype="ivoa:anyURI" ref="@@@@@" value=""/>
  <ATTRIBUTE dmrole="Phot:Access.size" dmtype="ivoa:integer" unit="" ref="@@@@@" value=""/>
  <ATTRIBUTE dmrole="Phot:Access.format" dmtype="ivoa:string" ref="@@@@@" value=""/>
</INSTANCE>
```

Listing 29: MIVOT instanciation of the PhotDM class `Access`. VODML attributes are mapped as simple `ATTRIBUTE`s since their cardinality is equal to 1 and they have primitive types. In this example, automatically generated, `ATTRIBUTE`s come with both `@ref` and `@value`. Using one, the other or both depends on the actual data being annotated (see 4.10). The cryptic `"@@@@@"` label is a tag used by the annoter tool to indicate a place holder for column reference. It must be replaced with the actual name of the table column (reference to a `FIELD` id or name, actually) to be used to set the `ATTRIBUTE` values.

The conversion rules are a bit more complex when we have to tackle with inheritance or constraints. Abstract classes should not be mapped into MIVOT as such. They must be mapped as concrete classes whose type depends on the mapped data (e.g. the error type associated with one measurement).

This simple example has been generated by the `snippet_builder` module of the MIVOT validator (`https://github.com/ivoa/mivot-validator`).

References

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

Derriere, S., Gray, N., Demleitner, M., Louys, M. and Ochsenbein, F. (2014), 'Units in the VO Version 1.0', IVOA Recommendation 23 May 2014, arXiv:1509.07267.
http://doi.org/10.5479/ADS/bib/2014ivoa.spec.0523D

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

Louys, M., Michel, L., Bonnarel, F. and Vetter, J. (2022), 'Annotating tap responses on-the-fly against an ivoa data model'.
http://doi.org/10.48550/arXiv.2201.01732

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

Plante, R., Demleitner, M., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T. and Rixon, G. (2018), 'VOResource: an XML Encoding Schema for Resource Metadata Version 1.1', IVOA Recommendation 25 June 2018.
http://doi.org/10.5479/ADS/bib/2018ivoa.spec.0625P

Shudi Gao, e. a. (2012), 'W3c xml schema definition language (xsd) 1.1'.
https://www.w3.org/TR/xmlschema11-1/