

Object Oriented Data Model strategy in the context of IVOA Table Access Protocol services

francois.bonnarel@astro.unistra.fr

François Bonnarel¹, Mireille Louys^{1,2}, Laurent Michel³

¹ CDS-France, ² ICUBE-France, ³ ObAS-France

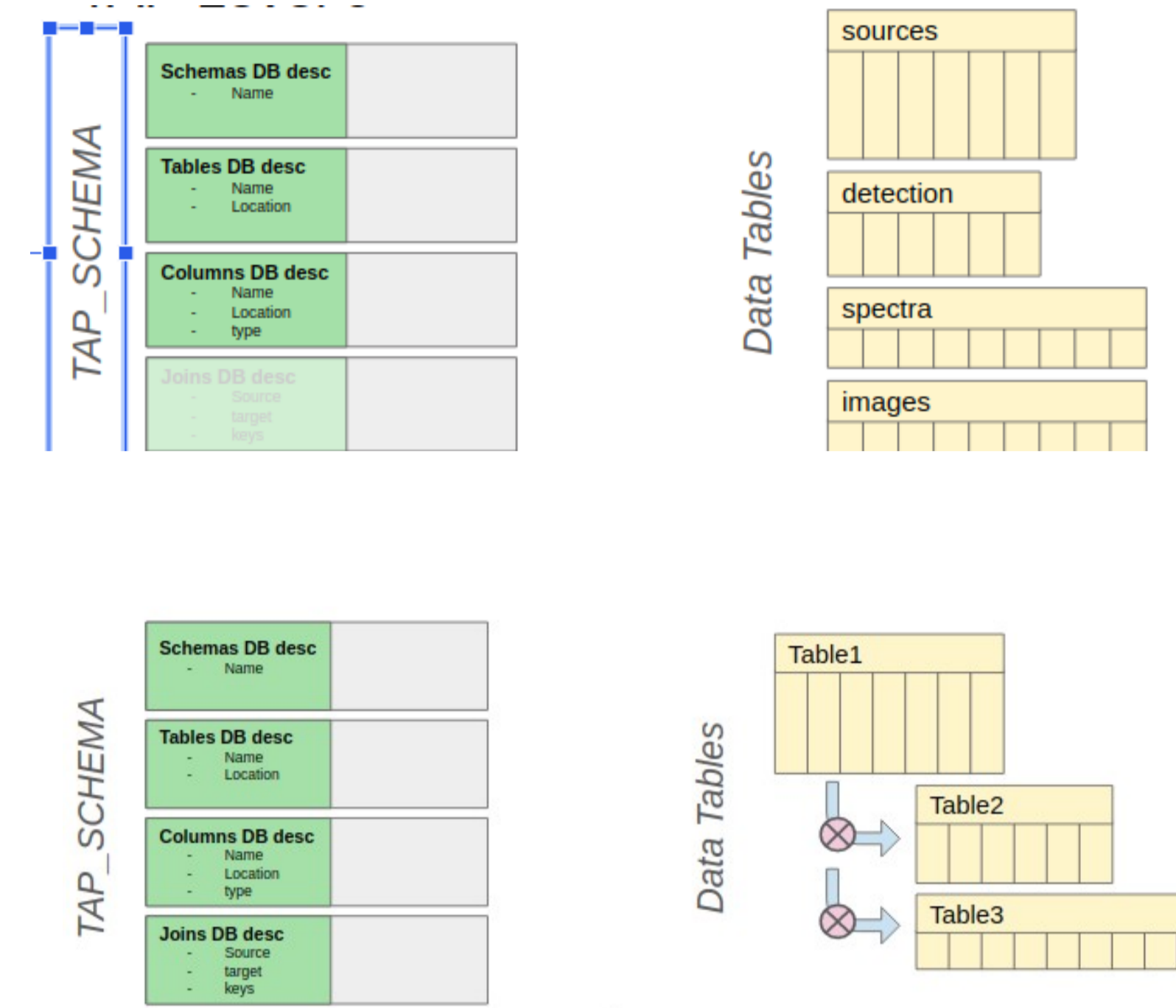
IVOA Table Access Protocol and IVOA datamodels

TAP

- TAP (Table access protocol) is an IVOA access standard for astronomical relational databases .
- TAP services registered in the IVOA registry, is widely used and currently expose more than 22 thousands tables.
- TAP makes use of ADQL a universal SQL-like language
- The database content is described in a standardized schema : the TAP_SCHEMA.
- TAP services built up a query response in one single table
- Default format for this table is the standard VOTable

IVOA datamodels

- IVOA datamodels :
- are Object oriented
 - allow logical description of relationships Internal to data
 - can model measurements, coordinates, photometry, complex grouping and provenance
 - follow rules defined in the VO-DML standard and are represented in vo-dml-xml format



TAP schema and data tables :
by default no datamodel information.
Top : without joints
Bottom : with joints included

How do we combine them in practice? ORM solves this only theoretically.
Four possible solutions experimented or considered

Simple flat views (current IVOA standard)

- Simple flat views may be built on top of the data model and the data. In that case the TAP schema covers the needs through standard additional metadata such as ucd, utypes and xtype.
- Utypes are pointers to the model leaves. Ucd and xtypes express the semantics and format of columns.
- An example is the ObsCore data model.



TAP schema and data tables :
Standard data model attributes

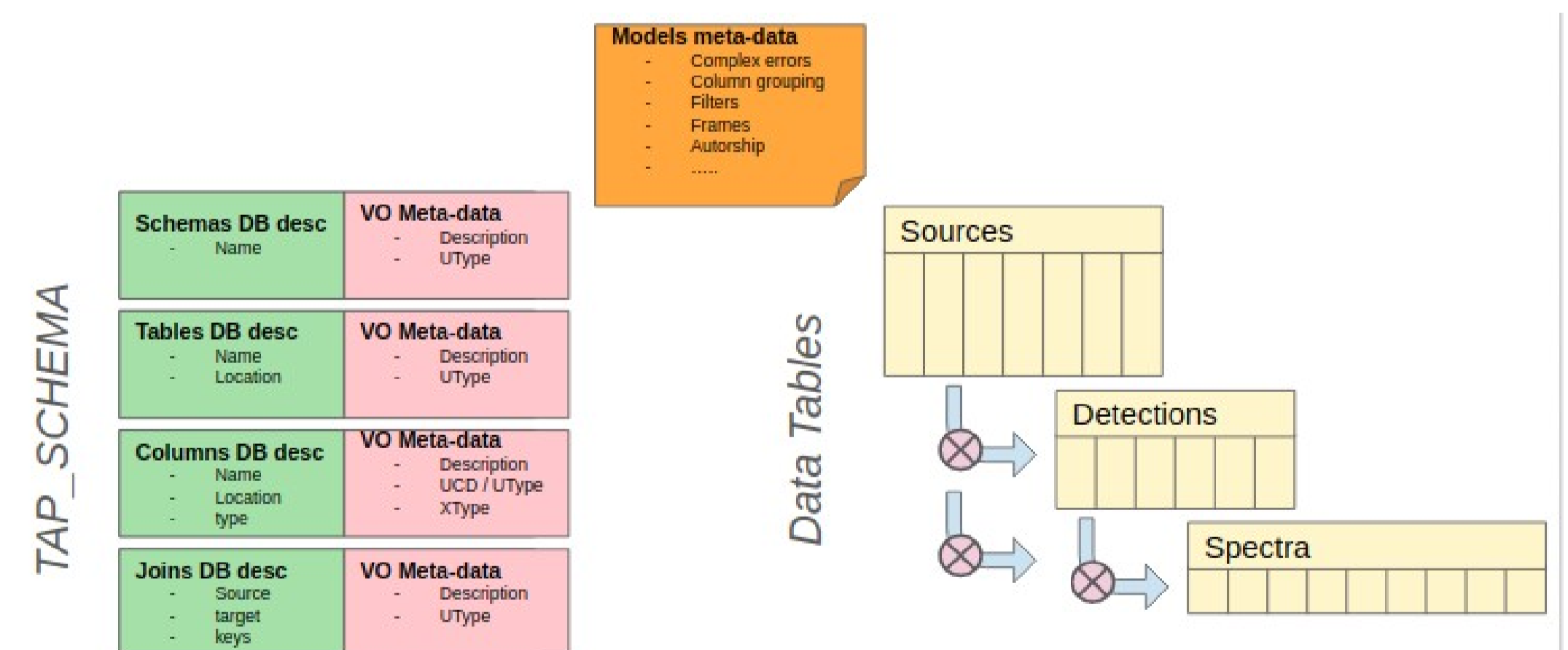
Table 6 TAP_SCHEMA.columns values for the mandatory fields of an ObsTAP table. All Utypes have the data model namespace prefix "obscore:" omitted in the table. The Datatype, Size, Principal, Index, and Std values shown here are informative for TAP 1.0 only; later versions of TAP may specify different values.

Column Name	Datatype	Size	Units	ObsCoreDM Utype	UCD	Principal	Index	Std
dataprodtype	adql:VARCHAR	TBD	NULL	ObsDataset.dataProductType	meta.id	1	TBD	1
calib_level	adql:INTEGER	NULL	NULL	ObsDataset.calibLevel	meta.code:obs.calib	1	TBD	1
obs_collection	adql:VARCHAR	TBD	NULL	DataID.collection	meta.id	1	TBD	1
obs_id	adql:VARCHAR	TBD	NULL	DataID.observationID	meta.id	1	TBD	1
obs_publisher_id	adql:VARCHAR	TBD	NULL	Curation.publisherDID	meta.ref.uri;meta.curation	1	TBD	1
access_url	adql:CLOB	NULL	NULL	Access.reference	meta.ref.uri	1	0	1
access_format	adql:VARCHAR	NULL	NULL	Access.format	meta.code:mime	1	0	1
access_estsize	adql:BIGINT	NULL	kbyte	Access.size	phys.size;meta.file	1	0	1
target_name	adql:VARCHAR	TBD	NULL	Target.name	meta.id:src	1	0	1
s_ra	adql:DOUBLE	NULL	deg	Char.SpatialAxis.Coverage.Location.Coord.Position 2D.Value2.C1	pos.eq.ra	1	0	1
s_dec	adql:DOUBLE	NULL	deg	Char.SpatialAxis.Coverage.Location.Coord.Position 2D.Value2.C2	pos.eq.dec	1	0	1
s_fov	adql:DOUBLE	NULL	deg	Char.SpatialAxis.Coverage.Bounds.Extent.diameter	phys.angle;instr.fov	1	0	1
s_region	adql:REGION	NULL	NULL	Char.SpatialAxis.Coverage.Support.Area	pos.outline;obs.field	1	0	1
s_resolution	adql:DOUBLE	NULL	arcsec	Char.SpatialAxis.Resolution.Reval.value	pos.angle;resolution	1	TBD	1
s_xel1	adql:BIGINT	NULL	NULL	Char.SpatialAxis.numBins1	meta.number	1	TBD	1
s_xel2	adql:BIGINT	NULL	NULL	Char.SpatialAxis.numBins2	meta.number	1	TBD	1
t_min	adql:DOUBLE	NULL	d	Char.TimeAxis.Coverage.Bounds.Limits.StartTime	time.start;obs.exposure	1	0	1
t_max	adql:DOUBLE	NULL	d	Char.TimeAxis.Coverage.Bounds.Limits.StopTime	time.end;obs.exposure	1	0	1
t_exptime	adql:DOUBLE	NULL	s	Char.TimeAxis.Coverage.Support.Extent	time.duration;obs.exposure	1	TBD	1
t_resolution	adql:DOUBLE	NULL	s	Char.TimeAxis.Resolution.Reval.value	time.resolution	1	0	1
t_xel	adql:BIGINT	NULL	NULL	Char.TimeAxis.numBins	meta.number	1	TBD	1

Excerpt of ObsCore TAP Schema :
flat data model view through schema attributes

On the fly TAP response annotation (prototype)

- According to the nature of the datamodel Graph (hierarchical or not, with loops or not, etc.) it may be impossible to flatten the datamodel view.
- In that case (Mango, Provenance, etc..) we complete the response with a specific data model mapping syntax currently under development (ModelInstanceInVotable)
- In our prototype annotation is generated on the fly by the server using some mapping metadata (currently in json) stored in the TAP_SCHEMA
- The actual mapping depends from the query
- Mapping client interpreters will retrieve instances of the model and manipulate them as objects
- See poster X3-010 for details



Mapping elements (orange box) added on the fly allow structured vision of the response table.
!!!! This approach also works with legacy catalog data where the datamodel is mapped on some tables/columns a posteriori !!!

Ideas for the future : more complex solutions

- A : Renormalized TAP response TAP query response are denormalized.

- It may become strongly redundant.
- We are considering to use an extension of ADQL which will renormalize the output by adding joins in the multi table response document and qualifying them with help of ModelInstanceInVOTable syntax.
- This approach will work both for databases fully consistent with a data model (for example Provenance TAP services) as well as for legacy complex catalogs (as in VizieR database)

- B : Full data model instance retrieval Model structure may become too complex for on the fly mapping (loop graphs in some provenance use cases)

- We could create UDF (user defined function) able to retrieve instances of the model in json, yaml, or ad hoc xml, for direct processing by client software