Retrieving Complex Data in TAP services

Anaïs Oberto, Laurent Michel, Grégory Mantelet, Haoyun Liao

Retrieving Complex Data in TAP services.

Anaïs Oberto,[1] Laurent Michel,[1] Gregory Mantelet,[1] and Haoyun Liao,[2]

[1]*Observatoire astronomique de Strasbourg, Université de Strasbourg, CNRS, UMR 7550, Strasbourg, F-67000, France*

[2]*UTBM, Université de Technologie Belfort-Montbelliard, Belfort, 90010, France*

Abstract. Thanks to the Virtual Observatory, users can query services through a common language (ADQL) to get data directly from databases and they can choose an output format derived from a simple table serialization: VOTable, TSV and so on.

To write advanced queries, users must first understand the database schema to identify the tables containing data of interest and to figure out how to join them. Some tools (TOPCAT, TAPHandle, Aladin, ...) propose generic interfaces helping for this but none of these tools is capable of providing a convenient view of a table set containing complex data. It is furthermore difficult to reconstruct complex data structures from flattened tables usually returned by SQL queries. TAP Simbad is a significant example of this difficulty with data spread out over 25 tables linked by foreign keys.

We are working on a client module addressing this issue for TAP Web interfaces . The database schema, namely the TAP_SCHEMA, is analysed to build a JSON representation of complex data that can be annotated with query constraints set by the user. So that complex queries can be easily setup and run. Query results are parsed out to reconstruct the data structure in JSON strings, where top level data are explicitly listed and subcomponents can be fetched with appropriate synchronous TAP queries. These queries are generated by the parser and included within the JSON output. Thus, users can unfold on demand any part of the searched data.

This mechanism is being implemented in Javascript to be proposed as a new overlay for both SIMBAD (Wenger et al. 2000) and TAPHandle (Michel et al. 2014) interfaces.

1. What we had before

In a standard TAP service (Dowler et al. 2011)), users have to:

- Work with a flat representation of the database schema (TAP_SCHEMA tables)
- Deal with cryptic information about the relationships between tables
- Build ADQL queries “by hand”, including joins
- Get denormalized tables

SIMBAD example

Lets consider a user wants to retrieve objects (and their publications' bibcode) around a star called ‘Antares’ and having a publication from an author name `"OBERTO"`. The

user must search which tables will be necessary to fulfil this query, how to link them together, and write the corresponding ADQL query. This simple question would finally look like the following ADQL query:

```
SELECT main_id, bibcode
FROM
   (SELECT ra,dec FROM basic
    JOIN Ident ON oid=oidref WHERE id='ANTARES'
   ) AS antares,
   basic AS star
JOIN has_ref ON oid=oidref
JOIN ref ON oidbibref=oidbib
JOIN author USING(oidbibref)
WHERE
   CONTAINS(
     POINT('ICRS', star.ra, star.dec),
     CIRCLE('ICRS', antares.ra, antares.dec, 0.1)
   ) = 1
   AND author.name LIKE 'OBERTO%'
```

The output will be returned as a flat table (illustrated bellow in ASCII format):

```
      main_id          bibcode
   ----------------------------------
   * alf Sco B   1992BICDS..40...71P
   * alf Sco B   1976A&A....46...11A
   * alf Sco     1995ApJ...440L..93B
   * alf Sco     2005A&A...431..773R
   * alf Sco     2002AJ....124.1636K
   * alf Sco     1999ApJ...516..817K
```

The user obtains all information in a single file. Here we can see that it is not very convenient to have the same star name in multiple lines.

2. What we want

By retrieving hierarchical data in a TAP service, users could:

- Work with a graph representation of the database schema.
- Get a clear view on the relationships between tables.
- Put local constraints directly on graph nodes.
- Get data in a structured representation.

We are working on a Javascript library providing such features and that could be connected to any TAP service.

3. How would that work?

1. **Building a graph from TAP_SCHEMA**
    - **Query a TAP service** to fetch available schema, tables, columns, keys.

- **Create the graph representation** using retrieved tables and keys information on how to linked tables together.
- **Format the graph representation into JSON** in function of a root table (defined by the user).

2. **Annotate the JSON with the user constraints**.

3. **Query TAP service as usual:** conversion of user constraints on graph nodes into regular ADQL query(ies).

4. **Store result in the graph:** conversion of denormalized tables into the JSON structure.

   Prepare more subqueries: setup of TAP URLs allowing to fetch more content inside individual graph sub-nodes on demand.

SIMBAD example

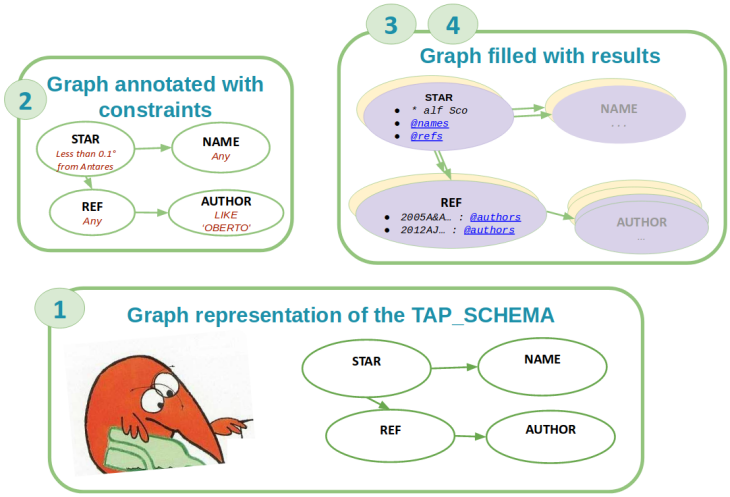

Figure 1. Example on how to query SIMBAD with a TAP graph representation.

From the same example explained before in section1, the user would follow the steps described in figure 1:

- select the "`basic`" node as the root node of what (s)he wants
- set position constraint around "`Antares`" position.
- select "`author`" node, and put the constraint on the author name "`OBERTO%`".
- run the query, and wait for the result to appear in the selected nodes.
- eventually click on the author node to see all other authors concerned by this selection.

The output would look like this sort of JSON structure:

```
[
  { "oid"  : "2401184",
    "name" : "* alf Sco B",
    "ra"   : 247.35083, "dec" : -26.43083,
    "publi": [
     { "oidbib" : 80228, "bibcode" : "1992BICDS..40...71P",
       "@authors" : "SELECT...FROM author...WHERE oidbibref=80228"
     }, ...
    ],
    "@ids" : "SELECT...FROM ident...WHERE oidref=2401184"
  },
  { "oid"  : "401703",
    "name" : "* alf Sco",
    "ra"   : 247.35191, "dec" : -26.43200,
    "publi": [
     { "oidbib" : 97299, "bibcode" : "1995ApJ...440L..93B",
       "@authors" : "SELECT...FROM author...WHERE oidbibref=97299"
     }, ...
       ],
    "@ids" : "SELECT...FROM ident...WHERE oidref=401703"
  }
]
```

Figure 2. Json output example of structured data from TAP Simbad.

User interface.

We can imagine in the future a web site that shows an interactive graph where the user could select the nodes, and directly put the constraints on them. Once the corresponding TAP query completed, the returned result is used to fill theses nodes so that being immediately and graphically visible to the user. By selecting more nodes, the prepared TAP URLs are run allowing the user to fetch more information.

4. Conclusion

With our Javascript module, the user has a clear view of the database thanks to the proposed hierarchical structure. Thus, this is more user friendly to understand links between tables and so to put constraints. The complexity of the tables relationships is hidden: the user does not have to manually write ADQL joins' constraints. Finally, the graph representation allows a more interactive output view, which avoid being flooded in too much data.

References

Dowler, P., Rixon, G., & Tody, D. 2011, ArXiv e-prints. 1110.0497, URL http://www.ivoa.net/documents/TAP/20100327/REC-TAP-1.0.html

Michel, L., Louys, M., & Bonnarel, F. 2014, in Astronomical Data Analysis Software and Systems XXIII, edited by N. Manset, & P. Forshay, vol. 485 of Astronomical Society of the Pacific Conference Series, 15

Wenger, M., Ochsenbein, F., Egret, D., Dubois, P., Bonnarel, F., Borde, S., Genova, F., Jasniewicz, G., Laloë, S., Lesteven, S., & Monier, R. 2000, A&AS, 143, 9. astro-ph/0002110