YouCat TAP
User Managed Catalogs and Permissions

Brian Major, Patrick Dowler, Adrian Damian

October 2019

IVOA Groningen, NL

National Research Council Canada
Conseil national de recherches Canada

Canada

YouCat - User managed tables

- adds other HTTP methods to VOSI Tables (/tables/tableName) (PUT & DELETE)

- POST to /load to append rows. No support for updating--complexity boundary reached where it's now better to use a db tool.

- POST to /table-update for (async) index creation

- New: GET and POST to /permissions
  - Allows schema and table 'owners' to get and modify permissions to schemas and tables

API Overview

- VOSI /tables/[table]
  - GET (regular VOSI get tables support)
  - PUT (VOSI table, VOTable)
  - DELETE
- 3 additional endpoints:
  - POST /load - append rows (FITS bintable, csv, tsv) *
  - POST /table-update/{table} - add indexes
  - **GET POST /permissions/{schema|table}**
    - **view/set permissions**

* thanks to the support of the stil and nom-tam java libraries

cadc-tap - python client for TAP and YouCat

```
> cadc-tap --help  (abridged)

usage: cadc-tap <subcommand>

subcommands:

    schema                  Print the tables available
                            for querying
    query                   Run an adql query
    create                  Create a table
    delete                  Delete a table
    index                   Create a table index
    load                    Load data to a table
    permission              Control table access
```

cadc-tap - python client for TAP and YouCat

```
> cadc-tap permission --help  (abridged)

usage: cadc-tap permission mode TARGET
       [groups [groups ...]]

Update access permissions of a table or a schema.

positional arguments:
  TARGET                table or schema name
  groups                name(s) of group(s) to
                        assign read/write permission
                        to. One group per r or w
                        permission.
  mode                  permission setting
  accepted modes:       (og|go|o|g)[+-=](rw|wr|r|w)
```

cadc-tap - python client for TAP and YouCat

```
> cadc-tap query --help  (abridged)

  -s, --service SERVICE

Set the TAP service. For the CADC TAP services both
the ivo and the short formats
(ivo://cadc.nrc.ca/youcat or youcat) are accepted.

External TAP services can be referred to by their
URL

(https://almascience.nrao.edu/tap).

Default is ivo://cadc.nrc.ca/youcat
```

TAP Schema and Table Permissions

Permissions at the schema and table level:

- Schema permissions apply to the **metadata** of the schema (table list) and metadata of the tables (columns, indexes, etc) in that schema
- Table permissions apply to the **data** (rows) of the table
- roughly modelled after the permissions model in VOSpace

Permissions at the schema and table levels:

- owner:
  - full permissions for all operations
  - always set to the creator of a table
- anonRead - if true then anyone can read the metadata/data
- readGroup - members of this group can read the metadata/data
- readWriteGroup - members of this group can read and write to the metadata/data
-

Permissions enforcement - queries

GET to /tables - filters out tables on which the user may not read

TAP sync and async queries:

- queries could see a view of the TAP schema to which they have read access, but:

- If a query includes a table to which you do not have read permissions, should you say 'permission denied' (403) or 'not found' (404)? (answer: users want 403)

- If a query includes one of the supporting tap_schema tables (schemas, tables, columns) it must be re-written by the service to include access control constraints.

Permissions enforcement - sync and async queries

- In order to inject access control constraints on the group columns, the user's group memberships is queried upfront. eg:

  where group_read in ('ivo://cadc.nrc.ca/gms?projectX', 'ivo://cadc.nrc.ca/projectY')

  - The IN clause is formed by getting all the user's group memberships. Two potential problems:

    - The membership list could be quite long

    - Which GMS service do you ask? (cont ->)

Permissions enforcement - GMS Issue?

- Example of the 2nd issue: A user is a member of:

  ivo://**cadc.nrc.ca**/gms?projectX and of:
  ivo://**oats.inaf.it**/gms?projectY

- These group URIs have different authorities that resolve to different GMS instances with different membership information.

- To get the complete list of a user's memberships, one would have find (via RegTAP) all GMS instances and query them all.

- We had to reluctantly admit (for now) that YouCat is tied to a single well-known GMS instance (the CADC one).

CADC open source TAP implementation

github.com/opencadc/tap

- YouCat is built into regular CADC TAP library
- Out-of-box configuration is 'anonymous read'
- Owner column will be populated if you provide a user mapping plugin
- Permissions enforced by enabling 1 other plugin

Interest? Standardization?

- VOSI /tables/[table]
  - GET (regular VOSI get tables support)
  - **PUT (VOSI table, VOTable)**
  - **DELETE**
- 3 additional endpoints:
  - **POST /load - append rows (FITS bintable, csv, tsv)**
  - **POST /table-update/{table} - add indexes**
  - **GET POST /permissions/{schema|table}**
    - **view/set permissions**