VERA C. RUBIN OBSERVATORY

User-Generated Data Products

Fritz Mueller – SLAC National Accelerator Laboratory

NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

User-Generated Data Products

- Support for “User-Generated Data Products” (aka “Level-3 tabular products”) is one of the remaining construction deliverables for Qserv (at least), but has not yet been designed in detail.
- Fritz, Gregory, and Colin were tasked at the last DMLT F2F to meet and push this forward.
- The following slides summarize the landscape after some preliminary discussion, with aim to publicize intended direction and enable initial implementation planning across groups.
- Formal requirements, as known today extracted from LDM-554 (“Data Management LSST Science Platform Requirements”) are appended here for reference.

Temporary Tables: VO Perspective

- Use TAP **`UPLOAD`** parameter
  - Table(s) provided in-line, or indirectly via URI
  - Transient (exist only for lifetime of query, but possibly cached)
  - Table schemas are neither provided nor published in TAP_SCHEMA
  - Must support VOTable uploads at minimum
  - We can assume (enforce) relatively small tables (able to be fully replicated, not sharded in Qserv)
- Use cases:
  - **`JOIN`** w/ tables already retrieved or loaded into user's portal session
  - **`JOIN`** w/ tables existing in notebook filespace (VOSpace, or…?)
- Concerns:
  - How complete is the implementation of this feature in the CADC TAP server codebase we are using? How much work to plumb through Rubin's Qserv connector extensions?
  - Underlying Qserv ingest is a distributed, asynchronous, operation. Can we get acceptable user feedback on progress and errors through this channel?

Temporary Tables: Qserv Perspective

- Current situation:
  - Qserv has well-developed ingest REST API, but as is accepts only staged CSV ingest products and requires many touches to accomplish an ingest.
  - Qserv ingest as is practically requires access to “admin only” web dashboard for status monitoring and troubleshooting (though polling access to status is available via current REST API)
- Proposal:
  - Add simplified API for specific case of ingesting modestly-sized fully-replicated (unsharded) tables in fewer touches, to support TAP **`UPLOAD`**
  - We would do this as an ingest REST API extension, not Qserv-specific SQL extensions over mysql-client connection (will this additional channel be okay for TAP service?)
  - Support VOTable inputs on this API
  - Expose/improve status/progress monitoring for above as necessary
  - Exceed prevailing standard for user-intelligible Qserv error messages…
  - [Stretch] support CSV/TSV, FITS bintable inputs; explore caching strategies

Persistent Tables: VO Perspective

- Contender here seems to be CADC “YouCat” TAP
  - Adds PUT and DELETE support to VOSI tables endpoint
  - Addresses publication of TAP_SCHEMA metadata
  - Permissions model for schema and data
  - Claimed “built in to regular CADC tap library” (in 2019; current status?)
  - Permissions hard-wired to CADC GMS instance (2019; current status?)
- Use Cases:
  - Upload moderately large datasets, to be sharded within Qserv for efficient parallelized **`JOIN`**s with official survey data products.
  - Permissions model would facilitate sharing these uploads among groups of users.
  - Published TAP_SCHEMA metadata makes these “first class” products in the portal and notebooks.

Persistent Tables: VO Perspective (cont.)

- Concerns:
  - As with temp tables, how well/completely supported in TAP codebase we are currently using, and how much work to push through Rubin's Qserv extensions?
  - Implies dynamism in TAP_SCHEMA implementation (currently TAP_SCHEMA data is assumed relatively static, and is baked in to TAP_SCHEMA service containers at build time.)
  - Implies per-user TAP_SCHEMA "views".
  - Table naming issues (avoiding collisions across users, etc.) tricky…
  - Auth*n issues (e.g. tie-ins to RSP user/group management) tricky…
  - Quota management issues tricky…
  - For sharded tables, Qserv will require additional inputs/guidance from user (anticipated `JOIN` targets, indication of sharding coordinate columns, indication of FK relation columns, etc.)
  - Portal effort required to support this (as-yet CADC-specific) protocol extension?

Persistent Tables: Qserv perspective

- Current situation:
  - Qserv ingest API currently needs pre-partitioned products for ingestion of sharded datasets; partitioning to date has been handled by separate front-end frameworks/tooling.
  - Same observability issues as mentioned above for temporary tables (admin console access needed)
  - Significant technical understanding of Qserv architecture specifics is needed to write correct and effective Qserv ingest control/configuration files (lengthy, technical, documentation)
- Proposal:
  - Given the many concerns/unknowns at both the VO service and Qserv levels, we propose starting with an MVP consisting of a web-based ingest “wizard” developed by the Qserv team, along with known-needed enhancements to the Qserv input-conversion and partitioning tooling
  - Would “interview” a user regarding partitioning options, provide default/pre-populated options, as-you-go explanations/guidance, and progress/status monitoring
  - Would allow Qserv to make progress on uncovering risk/complication on back-end, and help sharpen exactly what information is needed from users and must be supported in VO-to-Qserv APIs
  - VO standards, service, and portal work can be conducted concurrently

Requirements (per LDM-554)

1.3.2 User Database Workspace

ID: DMS-LSP-REQ-0012

Specification: The LSP shall provide for the creation, use, and management of user databases (User Generated tabular data products), and shall enable interaction with user databases with the same facilities as for Project-created database to the extent feasible.

Discussion: Some database-related capabilities of the LSP rely on the availability of detailed metadata on the Project-created databases that goes beyond the normal content of a database schema (e.g., IVOA UCDs for table columns). Users will be enabled, but not required, to supply such metadata for their own databases (and they may do so incorrectly), so LSP functionality that depends on it may not be available for user databases.

Requirements (per LDM-554, cont.)

2.6.1 Access to User Databases

ID: DMS-PRTL-REQ-0109

Specification: The Portal aspect shall provide read/write access to user databases (Level 3 tabular data products) and shall implement any access restrictions placed on such data.

Requirements (per LDM-554, cont.)

3.2.5 User Database Workspace Access

ID: DMS-NB-REQ-0020

Specification: Users will be able to interact with their User Database through the Notebook Aspect to insert, delete, and control access to their tables.

Discussion: This will be possible via TAP, at least, and possibly through lower-level access.

Requirements (per LDM-554, cont.)

4.3

User Storage Requirements for the User Database Workspace are still being developed