Butler Design Discussion 2018-02-14: Registry / Datastore boundary

Clarify boundaries between the Registry and Datastore components of the Butler v3 design.

Attending

- Jim Bosch
- Tim Jenness
- Michelle Gower
- Unknown User (pschella)

Discussion

Constraints

1. Datastores *must* record per-Dataset information indicating how each Dataset was stored (i.e. which Formatter was used); Datastore-level configuration *cannot* be used for this, as this is subject to change over the life of the Datastore. This information could be recorded...
    a. ...in the URI. But this may be fragile, as it means the URI depends on things like the file format, and hence Datastores cannot change the file format without also updating the URI stored in the Registry. *Michelle Gower does not like URI's (in particular when actively parsed for information), but Tim Jenness and Unknown User (pschella) do like them as human readable (but machine opaque) identifiers. Jim Bosch is neutral on the topic.*
    b. ...in a database table "owned" by the Datastore and not known to the Registry. This precludes simple database-free Datastore implementations. *We agreed that (almost) all Datastores need to store some metadata (such as formatter and checksum). So database free Datastores are unlikely, but the 'database' can probably be some files on disk rather than full SQL in some (many) cases.*
    c. ...in a database table known to both the Datastore and the Registry (and probably "owned" by some new third class of object). This is fragile in the same way as (A); it just puts the (e.g.) file format information in the schema of the shared table rather than packing it into the URI. In fact, URIs would probably not be useful in this design. *We all agree that some version of this is probably desired (see outcome).*
2. We have use cases for querying Datastore-specific Dataset metadata (MD5, size, whether the file exists in a particular Datastore) that may require some of these queries to be doable within a single database. These use cases include (at least):
    a. Is a Dataset present in an MRU cache shared by all Science Platform users, or must it be retrieved from the Data Backbone?
    b. Datasets of a particular DatasetType were (intentionally) not retrieved from compute node scratch storage after they were produced. How much storage would be required to keep them in a future production covering the same DataUnits?
    c. Datasets of another intermediate, not-retained DatasetType were produced both at IN2P3 and NCSA in what should have been identical runs. Can we verify that these (now deleted) files had the same MD5s?
    d. (see also Operations section of Generation 3 Butler Feature Requests)

Open Questions

- Is format information private to a Datastore? If not, do we require public access to it in general, or only in specific Datastore/Registry realizations? *It should be private (see outcome)*
- Do we have any use cases for changing formats or other storage details in a Datastore? *They have not been definitely identified, but the gain from making this information public appears small and the potential downside can be big (losing ability to compose / swap out Datastores or change storage details after the fact).*

Design Options

In all designs below, we have at least one database table for each physical Datastore (not Datastore proxies) associated with a Registry, with records for each Dataset. The table(s) store information such as md5sum and size that must have a public interface (which may or may not be direct access to the table itself) and may store format information that may be private to the Datastore. We will call these (for now) Storage Tables (not to be confused with Dataset Metadata tables associated with a particular StorageClass).

Datastore-Managed Storage Tables

Registries don't know about these tables, even if they exist in the same database server. Primary keys for Storage Tables are URIs. URIs are also in Registry's Dataset table. Butler passes URIs between Registry and Datastore Python objects.

Queries on public information must go through an (inefficient) two-stage lookup that involves getting URIs from Registry and then passing URIs to Datastore *or* (if the information is all in one database) cannot use our interfaces.

Registry-Managed Storage Tables

Datastores don't know about these tables, and they **only** contain public storage information fields. Primary keys for Storage Tables are (dataset_id, registry_id) tuples, and each Storage Table record has a URI that encodes the Datastore-private information. Registry's Dataset table does not have URIs. Butler passes URIs between Registry and Datastore Python objects.

Separate Storage Tables

Registry has a Storage Table that only contains public storage information, and Datastore has a Storage Table that only contains private format information. Both tables have a URI that does not encode the format information. The two tables may or may not be in the same database. Butler passes URIs between Registry and Datastore Python objects.

Jointly-Managed Storage Tables

Registry and Datastore jointly "own" tables in the same database (probably managed by a new third class). These tables include both standardized public information fields and Datastore-specific format fields, and are keyed by (dataset_id, registry_id). URIs do not exist; Butler passes the (dataset_id, registry_id) tuple between Registry and Datastore objects, which can both use them to access whatever tables they need.

Conditionally-Separate Storage Tables

Hybrid of Registry-Managed and Separate in which different Datastores can choose whether to encode format information in the URI or store it in a private table.

Outcome

- Formatter information (how was the file written) **should** be stored in Datastore **private** tables (which may live in SQL or in files on disk as preferred in some context by Tim Jenness).
  This allows for Datastores to change how files are stored without requiring Registry updates and will prevent breaking of compose-able Datastores.
- General information including, but not limited to: file existence, md5 and size **should** be stored in a Registry **public** and joinable table.
  One possibility could be a table keyed on (dataset_id, registry_id, datastore_id) with columns for "exists", "checksum", "size" (and perhaps "uri" or "location"?).
  This table could possibly be managed directly Registry, Datastore or by a new third class of object (to be decided by the implementer of the prototype).
- URI's are not a required part of the Datastore/Registry API, a (dataset_id, registry_id) primary-key would suffice, but are not (yet) ruled out.
  Formatting details are however **not** to be encoded in the URI (e.g. URI's, if kept, would only serve as convenient human readable but machine opaque identifiers).