Straw-man Mapping from Gen3 Butler Schema to CAOM2

UPDATE FROM 2018-11-12 MEETING WITH CADC REPS

Present: Tim Jenness , Gregory Dubois-Felsmann , Jim Bosch , Pat Dowler, Séverin Gaudet, Stephen Gwyn

- We should just map CAOM2 Plane directly to LSST Dataset, and hence have a 1-1 mapping between Planes and Artifacts. The concerns about denormalization of Plane-linked entities with this approach expressed in in the Alternatives section below simply aren't valid; the CAOM2 UML is in some sense maximally denormalized for flexibility, but our internal representation can be as normalized as we'd like. This solves all of the problems we identified below with fine-grained provenance and CAOM2 productID vs. LSST dataset_id.

- Representing SkyMap Tracts and Patches as CAOM2 static CompositeObservations with membership defined based only on overlaps (not actual inclusion in any particular coadd) was regarded as completely natural. This is more or less how CADC use those concepts themselves (i.e. Observation membership represents intent, Plane provenance represents actual contributions). Not all CAOM2 producers do this (it sounded like perhaps MAST was the exception?), but if anything there would be a push to get others to adopt the philosophy we (and CADC) prefer.

- Generalizing CompositeObservation membership to permit CompositeObservation (not just SimpleObservation) members (to permit Visits rather than Snaps to be the direct children of Tracts or Patches) is indeed a change to CAOM2, but one that CADC is open to making (they simply didn't have a use case for it until now). This may involve just unifying SimpleObservation and CompositeObservation. No one has a use case for this that involves trees with dynamic depth, so we're not overly concerned about the potential for cycles here; we just need to take the usual care for backwards compatibility and due diligence. CADC will pursue that change over the natural course of negotiating the next version of CAOM.

ORIGINAL REVIEW, 2018-10-04

Collections

Both the Gen3 Butler and CAOM2 define a "collection" concept, and happily they're broadly consistent.

In Butler, collections are just groups of Datasets defined by a string tag, and they can be used for many purposes - sets of master calibrations to use, different selections of raw data, the outputs of processing runs, and any combination of these. That will probably include a collection for each data release, and some way of grouping public prompt processing data products into one or more collections.

Most of these should probably not be mapped to CAOM2 at all, but all public data collections almost certainly should be, and for these we should be able to have a one-to-one relationship between the Butler collection and the CAOM2 collection, which would avoid a lot of confusion (but see the Alternatives section at the bottom of this page).

While Datasets are permitted to belong to multiple Collections in the Butler data model, the mapping to CAOM2 is probably easiest if the collections exposed in a particular CAOM2 view do not overlap in the Datasets they contain (possibly with an exception for raw data).

Instruments, Telescopes, and Environment

Butler already has a Camera DataUnit, which we should at least consider renaming to Instrument for more generality, less conflict with cameraGeom. Camera, and consistency with CAOM2. We can put any static CAOM2 Telescope or Instrument information there easily. It doesn't seem to me that we gain anything by normalizing out those concepts in the Butler schema; it's probably more important to try to keep the number of DataUnit dimensions small.

Non-static Telescope "keywords" and CAOM2 Environment information should not go in the Butler Camera/Instrument table. Butler would normally put time-dependent observatory state information in Datasets such as a persisted afw.cameraGeom.Camera instance (which will use valid timestamp ranges for their data IDs, like any master calibration). We expect to copy some of that information into the Exposure and/or Visit tables to enable Dataset lookups and Preflight queries on it (fields TBD).

Observations

The CAOM2 Observation concept is extremely general, and maps to multiple very distinct concepts on the Butler side. The CAOM2 view into Butler content is thus essentially a union of several entirely different views, some of which work much better than others.

I'm interpreting the CAOM2 docs as saying that the unique key for an Observation is the (observationID, collection) tuple, and hence the same observationID can appear in multiple collections. That also maps slightly better to the Butler data model than the interpretation that says that observationIDs must be unique across all collections, but we can support that interpretation too by just appending the collection name to the observationID definitions below.

CAOM2 Observations that represent Butler Exposures:

- There is one CAOM2 SimpleObservation for every non-empty Butler Exposure×Collection combination.
- observationID should be a unique key field from the Butler Exposure table (a string key clearly needs to be added, TBD whether this replaces or augments the current integer primary key).
- This is a full-focal-plane quantity.
- Butler Exposure table has a foreign key to the Butler Camera table, so associating with (static) Telescope and Instrument content is easy.
- Butler Exposure table has (some) time-varying Telescope and Environment content, which we can normalize out in the CAOM2 view.
- Butler Exposure table should contain all Target and TargetPosition information (though perhaps not in identical form; not clear out to represent a Point type in generic SQL, for instance).
- Algorithm.name is a constant: "exposure".

CAOM2 Observations that represent Butler Visits:

- There is one CAOM2 CompositeObservation for every non-empty Butler Visit×Collection combination.
- observationID should be a unique key field from the Butler Visit table (a string key clearly needs to be added, TBD whether this replaces or augments the current integer primary key).
- This is a full-focal-plane quantity.
- Butler Visit table has a foreign key to the Butler Camera table, so associating with (static) Telescope and Instrument content is easy.
- Butler Visit table has (some) time-varying Telescope and Environment content, which we can normalize out in the CAOM2 view.
- Butler Visit table should contain all Target and TargetPosition information (though perhaps not in identical form; not clear out to represent a Point type in generic SQL, for instance).
- Butler Visit table could have CAOM2 Algorithm name added to it, or we could compute this in the view from other properties of the Visit (e.g. how many snaps it contains).
- Child SimpleObservations are a view on a Butler join table between Visit and Exposure (planned but not yet implemented in Butler schema: ⚠ DM-15536 - Jira project doesn't exist or you don't have permission to view it.).

CAOM2 Observations that represent coadds:

- There is one CAOM2 CompositeObservation for every non-empty Butler Tract×AbstractFilter×Collection combination.
  - We may want one more CompositeObservation for data products that utilize data from all AbstractFilters.
  - We may want different CompositeObservations for different coadd DatasetTypes (which would be shared with the data products derived from them); not doing this would imply that the set of child Observations is just the set of Exposures or Visits (see below) that *could* have contributed to the coadd, not those that *actually* did.
- Children should conceptually be other CompositeObservations (i.e. Visits), but this does not seem to be permitted by the CAOM2 UML model.
  - Do we need to flatten those CompositeObservations into SimpleObservations?
  - Does this mean it would actually be better to represent Visits as SimpleObservations, and maybe ignore snaps entirely? (probably not)
- observationID should be constructed by combining the Tract ID with the AbstractFilter name.
- Algorithm.name is probably just "coadd", unless we adopt the approach of having different Observations for different coadd DatasetTypes (in which case we can say a bit more based on the DatasetType, of course).
- Tract *could* be a Target.name; not sure if that's useful.
- TargetPosition information is essentially already present in the Tract table.
- Telescope, Instrument, and Environment should probably be null (Telescope and Instrument will frequently be the same for all child SimpleObservations, but not always, and I don't think restating it here is ever that useful).
- Child SimpleObservations are computed from more general QuantumGraph provenance representation.
- These are conceptually Tract-level quantities (though making them Patch-level would be no easier or harder).

CAOM2 Observations that represent difference images:

- Multiple options, none of them good:
  1. There is no new Observation for difference images; they are just a Plane associated with the CompositeObservation of their Visit (i.e. ignore the template as an input).
  2. A CompositeObservation comprised of all of the Exposure SimpleObservations that went into the Visit and template CompositeObservations (i.e. flatten out all inputs).
  3. A CompositeObservation comprised of the Visit CompositeObservation and the template CompositeObservation (this best represents the data product, but may not be valid).
- These are all full-focal-plane quantities.
- Telescope, instrument, Environment, Target, and TargetPosition are all the same as those of the Visit CompositeObservation.
- Child SimpleObservations are computed from more general QuantumGraph provenance representation.

CAOM2 Observations that represent master calibrations:

- Not generally (but frequently) meaningful at a full-focal-plane level.
- Probably needs to wait until we can (or have time to) enumerate master calibration DatasetTypes.
- Butler organizes master calibrations primarily around the Exposures they are valid to be used with, not the observations used to construct them.
- Child SimpleObservations are computed from more general QuantumGraph provenance representation. As with coadds and difference images, we will frequently want to nest CompositeObservations.

Analysis

CAOM2 maps well to our Exposure and Visit concepts, and it does so in a consistent and intuitive way. We can easily implement CAOM2 views into a Butler schema for these.

I toyed a bit with adding an Observation table to the Butler schema, with a string primary key field that would be used as both a foreign key and a primary key for both Exposure and Visit; it seemed a bit messier than what we have now on the whole, but I wouldn't rule it out entirely. It wouldn't necessarily make the mapping to CAOM2 much easier, anyway, as the primary difficulties there are in non-Exposure, non-Visit Observations and the relationship between Observation and Collection, which I definitely don't want to change on the Butler side.

Unless it can be extended to (or already does) support nesting of CompositeObservations, CAOM2 maps sufficiently poorly to coadds and difference images that I'm not sure we should even try.

I'm moderately confident we can make CAOM2 CompoundObservations for at least some master calibrations, but this may be confusing if they sometimes represent full-focal-plane concepts and sometimes do not.

Planes

Once Observations mappings are established, expanding to Planes is relatively straightforward. For a CAOM2 Observation defined by a Butler Data ID and a Collection (which includes Observations associated with Exposures, Visits, and coadds), there is one CAOM2 Plane for every Butler DatasetType that has at least one Dataset in the Collection with a Data ID that is a superset of the Observation's Data ID (that sounds more complex than it is; need example or diagram).

Attributes:

- **productID:** unfortunately this can't be mapped to what would seem to be the most natural Butler analog, Dataset.dataset_id, because Butler Datasets don't appear in general until the Artifact level. From the Butler perspective, most Planes are just a DatasetType and a partial data ID, which is a useful grouping of Datasets to be sure, but only one of many useful groupings. I think our best option is probably to define the productID to be some combination of the observationID, the Butler DatasetType name, and the collection.
- **creatorID:** I don't know what to do with this; it seems to encode exactly the same information as productID but wants to be a URI, which doesn't make sense to me. Perhaps we leave it out.
- **metaRelease, dataRelease:** derived from the purpose of the particular Butler collection (not necessarily stored in the Butler schema, but easy to make a lookup table for)
- **calibrationLevel:** derived from the DatasetType associated with the Plane; we could just add this directly as a field to the Butler DatasetType table.
- **dataProductType:** similar to but coarser than Butler's StorageClass concept; we should probably just store the CAOM2 dataProductType in the standard StorageClass definitions in daf_butler (we haven't yet had a need to put any of that in the schema yet, but we easily could).

Associated Entities:

- **DataQuality:** don't have an clear use for this; ignore
- **Metrics:** might be meaningful for some Visit-associated Planes, probably as a view into some other more general metrics. Would be more natural at the Observation level since that already encodes a collection, and hence a unique processing of the data.
- **Provenance:** like productID, the natural Butler data model analog is at the Dataset/Artifact level, so at best this could be some sort of aggregation of the Dataset-level provenance. But unlike productID, where we at least have a straightforward definition we can use, aggregate provenance may be multi-valued (e.g. the patch-level Datasets in a full-tract Plane may not have been produced by the same processes).
- **Time, Position:** all straightforward to compute, but would be more natural at the Observation level; there's nothing Plane-specific for any of these for us.
- **Energy:** bandpassName is probably mapped to PhysicalFilter or AbstractFilter, depending on whether this is an Exposure/Visit entity or a coadd entity. I don't think anything here is Plane-specific; once again our versions of these are Observation-level.

Artifacts, Parts, and Chunks

It seems straightforward to map Butler Dataset directly to CAOM2 Artifact, and I see no problem with doing exactly that. Artifact.productType can work like Plane.calibrationLevel: something the DatasetType table can define. While Artifact's (mostly optional) fields seem to assume a bit more of a file-based system than the Butler requires, it will probably map reasonably well to the concrete mostly-file-based system (I think) we're actually building, and to the extent it doesn't, I imagine we'll still be able expose something that looks more-or-less file-based to users.

The CAOM2 Part concept can probably be used to expose the Butler's Dataset composition system, at least in some cases, and Chunks appear to be something we can straightforwardly define for at least Datasets with image-like StorageClasses.

Alternatives

Defining a CAOM2 Observation's "collection" to map directly to a Butler collection implies that (unlike Butler DataUnits), CAOM2 Observations are collection-dependent, and hence there are no CAOM2 concepts that span multiple Butler collections. But the notion that Butler's collection-independent DataUnits are static even across the collections that define different data releases is at some level aspirational (i.e. I imagine we'll prepare for schema evolution by adding levels of indirection above or outside the Butler schema). It might make sense to treat CAOM2 Observations the same way, and map all Butler collections to a much smaller number of CAOM2 collections (i.e. start with just one, and add more only when really necessary). While I'd need to work through the implications of that in detail to be certain, I think that would make the mapping from Visit/Exposure to Observation much more natural (at least within any scope in which each Exposures is not assigned to multiple Visits), at the expense of making the already more tenuous Observation mapping from coadds, difference images, and master calibrations probably unworkable: the CompositeObservation membership of those quantities is *defini tely* Butler-collection-specific (well, coadd and difference image CompositeObservations might be salvageable if we define their children to be possible rather than actual inputs).

The other clear mismatch here is in Plane, which might work better if we let it be a Dataset-level quantity (and hence always have a 1-1 relationship with Artifact in our realization of CAOM2). That would let its productID and Provenance map to much more meaningful Butler-side entities. On the other hand, Plane has other attributes we could define at the Observation level (Energy, Time, Position, Metrics), in some cases (all but Metrics) even if we follow the last paragraph and extend the domain of each CAOM2 Observation across multiple Butler collections. Those would be denormalized even more if we pushed the Plane definition down to match Dataset.