Dataset and Collection Table Reorganization

Goals

- Normalize the dimension fields in the current (monolithic) dataset table.
- Add a place to put per-DatasetType metadata, including regions and timespans.
- Add support for special DatasetTypes:
  - "Global" DatasetTypes
    - Examples: "raw", maybe external reference catalogs.
    - Always ingested, never produced by pipelines.
    - Have data IDs that are unique across *all* collections.
  - "Standard" DatasetTypes
    - All we have now.
    - Data IDs are unique within any single collection.
  - "Nonsingular" DatasetTypes
    - Data IDs are not unique within "tagged" collections (see below); frequently have empty data IDs (no dimensions).
    - Cannot be used as PipelineTask inputs from "tagged" collections.
    - Must be used for PipelineTask initInputs and initOutputs.
    - May be used for PipelineTask prerequisite inputs.
    - Major use case is master calibrations (more on that later).
    - Using these for schemas and configs should solve some existing problems with those.
- Add support for special Collections:
  - "Run" Collections
    - A Collection associated with a Run.
    - Datasets are associated with this collection implicitly via their Run, and cannot be associated or disassociated later.
    - Only collection that can be used in filename templates (others don't guarantee uniqueness).
    - Only collection that can be used for PipelineTask outputs.
  - "Tagged" Collections
    - All we have now.
    - Datasets can be associated and disassociated freely.
  - "Calibration" Collections
    - Associates each member dataset with a validity range.
    - Validity ranges for a particular DatasetType are non-overlapping (probably not enforceable via DB constraint).
    - Member datasets must be nonsingular.

Designs

All designs below use the following example DatasetTypes:

- raw
  - global
  - dimensions=(instrument, exposure, detector)
  - metadata=(region, datetime_begin, datetime_end, npix)
- object
  - standard
  - dimensions=(skymap, tract, patch)
  - metadata=(nrows)
- bias
  - nonsingular
  - dimensions=(instrument, detector)
  - metadata=(npix)

Dimensions not used in these are not shown below for simplicity. Dimension tables, datastore tables, the `quantum` table, and the `execution` table are also not shown.
Database types shown as "enum" below may be either strings or ints - the point is that the set of possible values is fixed.

Current Schema + `(uid, site)` Split

This is the current schema, with autoincrement fields (eg. `dataset_id`) split into `(uid, site)`, and a table listing all collections We'll want to make those changes regardless.

This schema does not meet any of the goals stated above; it's shown here for reference as a starting point.

One field here that merits explanation is `dataset_ref_hash`. This is a secure (i.e. we require and assume no collisions) hash of the dataset's dataset_type_name and data ID (i.e. dimension) fields. It's included in `dataset` as the source of truth, and duplicated in `dataset_collection` so it can be used in the unique constraint there.

```
CREATE TABLE "dataset" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "dataset_type_name" varchar NOT NULL,
  "run_uid" integer NOT NULL,
  "quantum_uid" integer,
  "dataset_ref_hash" varchar NOT NULL,
  "instrument" varchar,
  "exposure" integer,
  "detector" integer,
  "skymap" varchar,
  "tract" integer,
  "patch" integer,
  FOREIGN KEY ("dataset_type_name") REFERENCES "dataset_type" ("dataset_type_name"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE dataset_type (
  "dataset_type_name" varchar PRIMARY KEY,
  "storage_class" varchar NOT NULL
);

CREATE TABLE "collection" (
  "name" varchar PRIMARY KEY,
  "type" enum NOT NULL,
  UNIQUE (name)
);

CREATE TABLE "run" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar NOT NULL,
  UNIQUE ("collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name")
);

CREATE TABLE "dataset_collection" (
  "dataset_uid" integer PRIMARY KEY,
  "dataset_site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "dataset_ref_hash" varchar NOT NULL,
  UNIQUE ("dataset_ref_hash", "collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("dataset_uid", "dataset_site") REFERENCES "dataset" ("uid", "site")
);
```

Partition datasets (Only)

This design partitions the dataset table into a common table and one extra per-DatasetType table. That addresses the first two goals - data ID field normalization and metadata - but not the special kinds of dataset uniqueness and lookups we need.

```
CREATE TABLE "dataset" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "dataset_type_name" varchar NOT NULL,
  "dataset_ref_hash" varchar NOT NULL,
  "run_uid" integer NOT NULL,
  "quantum_uid" integer,
  FOREIGN KEY ("dataset_type_name") REFERENCES "dataset_type" ("dataset_type_name"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE dataset_type (
  "dataset_type_name" varchar PRIMARY KEY,
  "storage_class" varchar NOT NULL
);
```

```
CREATE TABLE "collection" (
  "name" varchar PRIMARY KEY,
  "type" enum NOT NULL,
  UNIQUE (name)
);

CREATE TABLE "run" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar NOT NULL,
  UNIQUE ("collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name")
);

CREATE TABLE "dataset_collection" (
  "dataset_uid" integer PRIMARY KEY,
  "dataset_site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "dataset_ref_hash" varchar NOT NULL,
  UNIQUE ("dataset_ref_hash", "collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("dataset_uid", "dataset_site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_raw" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "exposure" integer NOT NULL,
  "detector" integer NOT NULL,
  "region" varchar,
  "datetime_begin" datetime,
  "datetime_end" datetime,
  "npix" integer,
  UNIQUE ("instrument", "exposure", "detector"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_object" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "skymap" varchar NOT NULL,
  "tract" integer NOT NULL,
  "patch" integer NOT NULL,
  "nrows" integer,
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_bias" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "detector" integer NOT NULL,
  "npix" integer,
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);
```

Specialized uniqueness, with hashes

This is the first design that should meet all of the goals stated above. The primary difference is that `dataset_collection` is split into `dataset_collection_tagged` (for "tagged" collections) and `dataset_collection_calibration` (for "calibration" collections). Associations for "run" collections would be via the `run_uid` in the common dataset table (we could start doing this now with no schema changes, actually - just changes to query-generation logic).

The other changes involve the addition of UNIQUE constraints to several tables:

- "Global" DatasetTypes have a constraint in their per-DatasetType table (e.g. `dataset_raw`, below). Note that because that the table is restricted to a single DatasetType and all data ID fields are present, there's no need for `dataset_ref_hash` in this constraint.
- "Run" collections use the constraint involving `dataset_ref_hash` in the common `dataset` table (this is redundant but harmless for DatasetTypes with uniqueness="global").
- "Tagged" collections use the constraint involving `dataset_ref_hash` the `dataset_collections_tagged` table.

In addition, we define the `dataset_ref_hash` for "nonsingular" DatasetTypes to be a combination of their `uid` and `site`. That effectively makes the `dataset_ref_hash` constraints on them do nothing, which is what we want for tagged collections, and maybe acceptable for run collections.

```
CREATE TABLE "dataset" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "dataset_type_name" varchar NOT NULL,
  "dataset_ref_hash" varchar NOT NULL,
  "run_uid" integer NOT NULL,
  "quantum_uid" integer,
  UNIQUE ("dataset_ref_hash", "run_uid", "site"),
  FOREIGN KEY ("dataset_type_name") REFERENCES "dataset_type" ("dataset_type_name"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE dataset_type (
  "dataset_type_name" varchar PRIMARY KEY,
  "storage_class" varchar NOT NULL,
  "uniqueness" enum NOT NULL
);

CREATE TABLE "collection" (
  "name" varchar PRIMARY KEY,
  "type" enum NOT NULL,
  UNIQUE (name)
);

CREATE TABLE "run" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar NOT NULL,
  UNIQUE ("collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name")
);

CREATE TABLE "dataset_collection_tagged" (
  "dataset_uid" integer PRIMARY KEY,
  "dataset_site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "dataset_ref_hash" varchar NOT NULL,
  UNIQUE ("dataset_ref_hash", "collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("dataset_uid", "dataset_site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_collection_calibration" (
  "dataset_uid" integer PRIMARY KEY,
  "dataset_site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "datetime_begin" datetime NOT NULL,
  "datetime_end" datetime NOT NULL,
  FOREIGN KEY ("dataset_uid", "dataset_site") REFERENCES "dataset" ("uid", "site"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name")
);

CREATE TABLE "dataset_raw" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "exposure" integer NOT NULL,
  "detector" integer NOT NULL,
  "region" varchar,
  "datetime_begin" datetime,
  "datetime_end" datetime,
```

```
  "npix" integer,
  UNIQUE ("instrument", "exposure", "detector"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_object" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "skymap" varchar NOT NULL,
  "tract" integer NOT NULL,
  "patch" integer NOT NULL,
  "nrows" integer,
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_bias" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "detector" integer NOT NULL,
  "npix" integer,
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);
```

The most common queries for datasets with this schema will have one of the following forms. Frequently these will be used as subqueries in much larger queries (such as in QuantumGraph generation); in that context the WHERE constraints below on data ID fields (instrument, exposure, detector, skymap, tract, patch) will move to JOIN ON clauses. A key point for all of these queries is that we can expect the collections to be fully provided by the user; that means we'll be able to perform initial queries (and cache results as necessary) to identify the collection type and look up associated `run_uid` values as necessary (and hence WHERE constraints below on collections will stay in the WHERE clause even when these are subqueries).

```
-- Data ID search for raw (uniqueness=GLOBAL) in any collection:
SELECT uid, site FROM dataset_raw WHERE instrument=? AND exposure=? AND detector=?;

-- Data ID search for object (uniqueness=STANDARD) in a type=RUN collection:
SELECT dataset.uid, dataset.site
  FROM dataset
  JOIN dataset_object ON (dataset.uid = dataset_object.uid AND dataset.site = dataset_object.site)
  WHERE dataset.run_uid = ? AND dataset.site = ?
        AND dataset_object.skymap=? AND dataset_object.tract=? AND dataset_object.patch=?;

-- Data ID search for object (uniqueness=STANDARD) in a type=TAGGED collection:
SELECT dataset_object.uid, dataset_object.site
  FROM dataset_object
  JOIN dataset_collection_tagged ON (dataset_object.uid = dataset_collection_tagged.dataset_uid
                                     AND dataset_object.site = dataset_collection_tagged.dataset_site)
  WHERE dataset_collection_tagged.collection = ?
    AND dataset_object.skymap=? AND dataset_object.tract=? AND dataset_object.patch=?;

-- Temporal search for bias (uniqueness=NONSINGULAR) in a type=CALIBRATION collection:
SELECT dataset_bias.uid, dataset_bias.site
  FROM dataset_bias
  JOIN dataset_collection_calibration ON (dataset_bias.uid = dataset_collection_tagged.dataset_uid
                                          AND dataset_bias.site = dataset_collection_tagged.dataset_site)
  WHERE dataset_collection_calibration.collection = ?
    AND dataset_bias.instrument=? AND dataset_bias.detector=?
    AND dataset_collection_calibration.datetime_begin < ? AND dataset_collection_calibration.datetime_end > ?;
```

Dropping Hashes and Eliminating Joins

This design also meets all of the stated goals, but also makes the most common queries join-free at the expense of more per-dataset tables.

Differences from the previous design include:

- The `dataset_ref_hash` field has been removed from all tables. This a nice simplification for the Python code, as we no longer need code to compute the hash, and it also makes the behavior of the unique constraints more natural and obvious from the database side, as they no longer depend on a field whose meaning can change from row to row.

- We now have two (for most datasets) or three (for master calibrations) per-dataset tables that duplicate the data ID fields; the additional tables replace the monolithic `dataset_collection_*` tables. The total number of rows is conserved between these designs, while the average width should be about the same (depending on how many data ID fields are used on average, and how many bytes are used in `dataset_ref_hash`).
- The `run_uid` field is duplicated in the per-DatasetType tables, allowing the unique constraint for run collections to be defined there. That also gives us the option of whether or not to define a unique constraint on nonsingular datasets in run collections.

```
CREATE TABLE "dataset" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "dataset_type_name" varchar NOT NULL,
  "run_uid" integer NOT NULL,
  "quantum_uid" integer,
  FOREIGN KEY ("dataset_type_name") REFERENCES "dataset_type" ("dataset_type_name"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE dataset_type (
  "dataset_type_name" varchar PRIMARY KEY,
  "storage_class" varchar NOT NULL,
  "uniqueness" enum NOT NULL
);

CREATE TABLE "collection" (
  "name" varchar PRIMARY KEY,
  "type" enum NOT NULL,
  UNIQUE (name)
);

CREATE TABLE "run" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar NOT NULL,
  UNIQUE ("collection"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name")
);

CREATE TABLE "dataset_raw" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "run_uid" integer NOT NULL,
  "instrument" varchar NOT NULL,
  "exposure" integer NOT NULL,
  "detector" integer NOT NULL,
  "region" varchar,
  "datetime_begin" datetime,
  "datetime_end" datetime,
  "npix" integer,
  UNIQUE ("instrument", "exposure", "detector"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE "dataset_tag_raw" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "exposure" integer NOT NULL,
  "detector" integer NOT NULL,
  UNIQUE ("collection", "instrument", "exposure", "detector"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_object" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "run_uid" integer NOT NULL,
```

```
  "skymap" varchar NOT NULL,
  "tract" integer NOT NULL,
  "patch" integer NOT NULL,
  "nrows" integer,
  UNIQUE ("run_uid", "site", "skymap", "tract", "patch"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE "dataset_tag_object" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "skymap" varchar NOT NULL,
  "tract" integer NOT NULL,
  "patch" integer NOT NULL,
  UNIQUE ("collection", "skymap", "tract", "patch"),
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_bias" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "run_uid" integer NOT NULL,
  "instrument" varchar NOT NULL,
  "detector" integer NOT NULL,
  "npix" integer,
  UNIQUE (run_uid, site, instrument, detector),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site"),
  FOREIGN KEY ("run_uid", "site") REFERENCES "run" ("uid", "site")
);

CREATE TABLE "dataset_tag_bias" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar NOT NULL,
  "instrument" varchar NOT NULL,
  "detector" integer NOT NULL,
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);

CREATE TABLE "dataset_validity_bias" (
  "uid" integer PRIMARY KEY,
  "site" integer PRIMARY KEY,
  "collection" varchar PRIMARY KEY,
  "instrument" varchar NOT NULL,
  "detector" integer NOT NULL,
  "datetime_begin" datetime NOT NULL,
  "datetime_end" datetime NOT NULL,
  FOREIGN KEY ("collection") REFERENCES "collection" ("name"),
  FOREIGN KEY ("uid", "site") REFERENCES "dataset" ("uid", "site")
);
```

As promised, the common queries are now much simpler:

```
-- Data ID search for raw (uniqueness=GLOBAL) in any collection:
SELECT uid, site FROM dataset_raw WHERE instrument=? AND exposure=? AND detector=?;

-- Data ID search for object (uniqueness=STANDARD) in a type=RUN collection:
SELECT uid, site FROM dataset_object
  WHERE run_uid=? AND site=? AND skymap=? AND tract=? AND patch=?;

-- Data ID search for object (uniqueness=STANDARD) in a type=TAGGED collection:
SELECT uid, site FROM dataset_tag_object
  WHERE collection=? AND skymap=? AND tract=? AND patch=?;
```

```
-- Temporal search for bias (uniqueness=NONSINGULAR) in a type=CALIBRATION collection:
SELECT uid, site FROM dataset_validity_bias
  WHERE collection=? AND instrument=? AND detector=? AND datetime_begin < ? AND datetime_end > ?;
```