Querying the UDS for Fun and Glory

The Universal Data Set (UDS) brings QPP datasets into one central data warehouse. Here we describe some pointers for extracting information and insights from this warehouse. This document assumes you are already generally familiar with the domains, data sources, and tables in the UDS (QPP Universal Data Set (UDS) User Guide).

Best Practices

1. Use the `active` schema instead of the `uds` schema.

Details: In the base-layer `uds` tables, we have multiple snapshots and versions of the data in order to preserve a history of how QPP data changes over time. For most querying needs, you likely only want the most recent, up to date information. The `active` schema handles selecting that most recent / latest / active data for you, in each domain.
Exceptions: The following views in `active` are not filtered compared to their `uds` table counterparts: submission_request_log, submission_request_json_flattened, state_region, tin_npi_ccn_mapping, change_reason, final_score_change, data_source, payment_adjustment_load, and all the wi_(x) tables.

Example: Instead of this...

```
SELECT COUNT(DISTINCT (o.tin, p.npi))
FROM uds.organization o
JOIN uds.data_load dl ON dl.db_id = o.db_data_load_id AND dl.active = true
JOIN uds.provider_org po ON po.db_organization_id = o.db_id
JOIN uds.provider p ON po.db_provider_id = p.db_id
WHERE o.year = 2018 and o.run = 4
LIMIT 1;
```

Do this...

```
SELECT COUNT(DISTINCT (o.tin, p.npi))
FROM active.organization o
JOIN uds.provider_org po ON po.db_organization_id = o.db_id
JOIN uds.provider p ON po.db_provider_id = p.db_id
WHERE o.year = 2018 and o.run = 4
LIMIT 1;
```

Notice that in the second example, we start from `active.organization`, but then for the next two tables, we use their `uds` counterparts. This is because all `active` objects are views, which means they are "rebuilt" as they are needed for every query. If we referenced the `active` view for all three tables, all three must be rebuilt, then joined. This ends up taking a lot of extra time. We can get around this by realizing that all three tables are in the same domain (Eligibility), and thus all foreign key - primary key associations must be unique within that domain. In other ones, every row's db_id in `active.organization` will only ever join to its active counterparts in `uds.provider_org` with db_organization_id.

To summarize: use one active view per domain in your query, but use uds objects for additional objects in a domain.

2. When joining *within* a domain, use the `db_..._id` foreign keys to connect tables.

Details: The `db_..._id` keys are created to efficiently join the normalized data together. In general, the tables within a given UDS domain are loaded from the same source and at the same time. Using the normalized foreign keys to join ensures you get the full 'snapshot' of the data.

Exceptions:
a. For tables without `db_..._id` keys, you must of course use natural keys instead. For example if you wanted to join the tables `neutral_payment_adjustment` and `final_score` within the Final Scoring domain, you would have to use the natural keys (tin, npi, year), as there is no normalized key between them.
b. Because the Submissions domain is loaded via two different processes, you may need to join using natural keys within the domain for certain use cases. Submissions is first populated via a streaming process, where information comes from audits and the `submission.db_submission_audit_log_id` foreign key is populated, and also by a batch "data load" process, where a snapshot of the state of every submission is recorded and the `submission.db_data_load_id` foreign key is populated. So for example if you wanted to know the time for which every 'apm' submission was last modified, and the submission window is under way, you can use the normalized keys:

```
SELECT s.submission_id, sal.create_time as last_modified
FROM active.submission s
JOIN active.submission_audit_log sal ON sal.db_id =
s.db_submission_audit_log_id
WHERE s.performance_year = 2018 AND s.entity_type = 'apm';
```

(Remember, the `active` schema filters Submission tables to the `is_latest=true` records only, so we don't have to worry about that here.)

But once the submission window is over and the Submissions domain moves on to the "batch loading" process, it gets more tricky.

```
SELECT s.submission_id, max(sal.create_time) over (partition by
s.submission_id) as last_modified
FROM active.submission s
```

```
JOIN active.submission_audit_log sal ON sal.submission_id = s.submission_id
WHERE s.performance_year = 2018 AND s.entity_type = 'apm';
```

Note the join changes to using the `submission_id` natural key; additionally, we have to partition all the audits to select the maximum timestamp.

3. When joining *across* domains, use natural keys.

Details: UDS domains are loaded at different times and cadences. When a particular domain is loaded, like Final Scoring, the normalized `db_..._id` keys that refer to tables in different domains are set up to point to the 'active' or 'latest' version of that domain's data. So for example when we load a new snapshot of `uds.final_score`, the `db_submission_id` key is filled in by finding the row in the `uds.submission` table that corresponds to the natural key `submission_id` and where `is_latest=true`. Similarly the `db_organization_id` key is filled in by finding the row in the `uds.organization` table that corresponds to the natural keys [`tin`, `year`], the maximum valid run of the year, and where `organization.db_data_load_id` corresponds to the row in `uds.data_load` where `uds_domain='Eligibility'`, `year=<natural key year>`, and `active=true`.

Now consider what happens if, after we load Final Scoring, Eligibility fixes a bug in their data, or adds new organizations. We have to create a new data load of the Eligibility domain and make it active, to show that it is the most recent version of the domain's data. Now, the `final_score.db_organization_id` that we originally populated no longer points to the most recent Eligibility data, but rather the Eligibility data at the time of the Final Scoring load. So in order to join the most recent/active `final_score` to the most recent/active `organization` table, we must use the natural keys instead.

Example:

When Final Scoring was loaded, the following query providers the most-up-to-date information on the final score and the eligibility information for a certain TIN. (**WARNING:** This query is not recommended, but rather presented to show the difference between this and the correct, second query.)

```
SELECT fs.tin, fs.npi, o.name, o.city
FROM active.final_score fs
JOIN active.organization o ON o.db_id = fs.db_organization_id
WHERE fs.tin = '000123456';
```

Then Eligibility is loaded again, with a new data load marked active (let's say a bug with the `city` gets fixed). The above query won't work at all, because `active.organization` now points to a completely new version of the Eligibility data, and that `db_organization_id` key doesn't appear in the view. We have to use the natural keys to get the most up to date information for both domains:

```
SELECT fs.tin, fs.npi, o.name, o.city
FROM active.final_score fs
JOIN active.organization o ON o.tin = fs.tin and o.year = fs.year and (o.run
= 4 OR o.run = 3 OR o.run = 0)
```

```
WHERE fs.tin = '000123456'
ORDER BY o.run desc LIMIT 1;
```

We switch from using the `db_organization_id` normalized key to using the natural keys: `tin`, `year`, and a filter on `run`.

Exceptions:
a. The Submission Scoring domain has no natural defining key of its own, because a score must always result from a submission. When joining Submission Scoring tables to the Submission domain, we must always use the `submission_score.db_submission_id` key.
b. If you want to know exactly how all of the data looked at a particular point in time you may want to use the `db_..._id` normalized keys. For example, perhaps we want to know what the organization's city looked like at the time the Final Score was made. So the above query would become:

```
SELECT fs.tin, fs.npi, o.name, o.city
FROM active.final_score fs
JOIN uds.organization o ON o.db_id = fs.db_organization_id
WHERE fs.tin = '000123456';
```

We are using the `active` schema to get us the latest Final Scoring information, but we have to use the `uds` tables to see what the older data looked like with the normalized key.

4. Use the `udsdm` data marts rather than querying the tables directly.

Details: The A&R data scientists have set up a number of data marts that handle the task of joining various domains and tables together into a denormalized, 'flat' version of the data. If your use case falls into one of the situations described in the Data Mart documentation, https://confluence.cms.gov/display/QPPAR/UDS+Data+Mart+Data+Dictionary?src=contextnavpagetreemode, using these data marts will result in faster information retrieval, decreased potential of missing some vital 'gotcha' when writing your own queries, simpler queries, and access to derived business logic fields that do not exist in the UDS.