Data marts

Overview

Unizin has developed new data mart construction and delivery infrastructure that aligns more closely with the rest of the batch ingest process. Previously, we built the marts in a Unizin internal GCP project based on SQL queries scheduled directly in BigQuery for refresh. The marts were then “passed through” into each tenant’s prod GCP project in mart-prefixed datasets containing materialized views. Image 1 below shows the previous delivery of the marts:

Image 1: Previous Mart Presentation in BigQuery

Motivation for Change

Unizin’s data mart development began in 2021 as “proofs of concept” assets aiding use-case-driven analytics essential and shared across the consortium. Organically, the data marts have elevated into a first-party Unizin service core to the UDP and membership value.

The data marts have reached a point where convergence with the rest of Unizin’s cloud-based infrastructure and processes will enable more performant, scalable, and reliable data mart development moving forward.

Building the marts requires a series of dependencies and intermediary queries that we previously kept “behind the scenes” in favor of cleanliness in the prod GCP projects. However, Unizin has received feedback that seeing the “helper tables” has value on their own as well! The new infrastructure for building the marts will expose all dependencies and intermediary queries alongside the final mart presentations.

Refresh Frequency

The refresh frequency of the marts and helper tables depends mainly on the source UDP data that fuels the tables in scope.

- **Dependency on context_store data** - refresh frequency is daily; the current context_store data update daily, so there is no need to refresh these tables more often than daily.
- **Dependency on event_store data** - refresh frequency is hourly (usually); the event_store data refresh almost realtime. We approximate ‘realtime’ with hourly refreshes to strike a balance with refresh scanning costs in BigQuery. Nearing realtime requires more frequent scans of data which becomes more costly. Hourly refreshes is our current optimization of data freshness against refresh costs.
- **Mixed dependencies** - these tables often refresh on the most restrictive cadence. If a dependency refreshes hourly, these tables will refresh hourly too.
Each table in BigQuery has a *Details* tab that can be consulted for questions about refresh frequency; the field in scope is *Last Modified*:

![Image 2: Helpful Table Timestamp Information in BigQuery](image)

**BigQuery Datasets**

The delivery of marts for user consumption live in datasets prefixed with *mart*. The newly surfaced dependency tables all live together in *mart_helper*.

The final, use-case driven marts live in the other *mart* datasets:

- *mart_general*
- *martCourse_section*
- *martCourse_offering*
- *mart_taskforce*

These *mart_* datasets are unchanged from the previous mart infrastructure to the new one. The table schemas, field names, datatypes, and everything else are unchanged, so existing scripts and applications do not break. The only technical difference in these datasets is that the assets inside the datasets are now native tables instead of materialized views. The previous infrastructure favored surfacing materialized views since the computation and construction all occurred in an internal Unizin GCP project. However, this is no longer a requirement, so native BQ tables are preferred for mart presentation.

**mart_helper Deep Dive**

The newly visible dataset in prod BQ environments is called *mart_helper*. The tables in here are all the accumulated dependencies that build the final tables in the other *mart_* datasets. Since all the dependencies are compressed into one hierarchical level, a consistent naming convention is enforced on each asset in the *mart_helper* dataset. Double underscore characters are used as delimiters.

**First Substring: context, event, or utility**

The first part of a *mart_helper* table name is either “context,” “event,” or “utility.” These strings dictate the base data source on which the helper table depends:

- **Context** - the data source is the UDP context_store, governed by the UCDM
- **Event** - the data source is the UDP event_store, governed by the Caliper standard
- **Utility** - the data source is static mappings for tool names and such. This is the only part of the *udp_marts* that is manually compiled and maintained. However, these data will rarely (if ever) update.

**Second Substring: Scope**

The middle substring can be vague by looking at the tables in BQ, but its original purpose is as an organizational pointer to where SQL code lives in the *udp_marts* code repository. Values for pivot scope could include “base,” “course_offering,” “course_section,” “taskforce,” or “general.” Any table with “context__base” will have its SQL definition in the *context/base/* directory in the Gitlab repository.

Analytically, there is a loose interpretation of this substring, but we don't recommend focusing on these reasons too strictly:

- **Base** - these tables are the first layer on top of raw UCDM data
- **Course_offering** - a useful pivot in reporting for these tables may be per-course offering.
- **Course_section** - same interpretation as course_offering but at the more granular section level.
- **Taskforce** - these assets have this specific string to denote work directly related to the learning analytics taskforces in 2022.
- **General** - No direct pivot is specified or favored; appropriate usage can include looking across courses, terms, cohorts, etc.

### Third Substring: Asset Name

This names the scope and purpose of the table. Some values are more straightforward (such as person, learner_activity, course_grade, etc.), which point to a direct UCDM entity in scope for aggregations or denormalization. Other values are more use-case descriptive (such as interaction_sessions, last_activity, weekly_discussions, etc.). These tables likely have multiple dependencies on UCDM tables or Caliper events and are less tied to a single entity in scope. Instead, these tables likely answer a targeted analytics question.

### Code Repository and Visualized Dependencies

All of the SQL code and deployment scripts used to build the marts are stored in this Unizin Community Gitlab repository called Marts. For now, this repository is read-only and for logic reference purposes only; however, users are encouraged to create issues against the repository for additions, bug fixes, and requests.

The directories in scope for viewing SQL logic defining the marts are the following:

- **Context** - all helper tables that have UDP context_store dependencies
- **Event** - all helper tables that have UDP event_store dependencies
- **Mart** - highest level tables that combine both context and event dependencies together

The other Python scripts, directories, and configuration files are totally fine to see! However, their only purpose is internal to Unizin DevOps to maintain and deploy the marts at scale.

### Interactive Dependency Charts

Each corresponding mart documentation page nested beneath this page on the resources site has an interactive dependency visualization (built using Mermaid) showing the sequence of helper tables built from foundational UDP data to the use-case driven marts.

Below is an example of the mart_general.lms_tool dependency visualization:

![Image 3: mart_general.lms_tool dependency visualization](org.apache.batik.transcoder.TranscoderException: null Enclosed Exception: null:-1 The attribute "width" of the element is required)

Clicking on a node in the dependency diagram will take users to the corresponding SQL file in the Gitlab repository.

### Questions and Requests For Changes/Additions

Unizin’s [JIRA ticket queue](#) is the best place to ask questions and request additions or changes to the marts. The Services and DSS team will follow up directly on each ticket there.