DETQTIV - Data and Quality Service Dashboard

SHRINE Network Data Extraction Instructions

UCLA

Contents

[Introduction 3](#_Toc472927796)

[Ontology Enhancements 4](#_Toc472927797)

[Ontology Identifier 4](#_Toc472927798)

[Parent/child 🡪 Node 5](#_Toc472927799)

[Observations 6](#_Toc472927800)

[Observation Fact extraction 6](#_Toc472927801)

[Demographics 6](#_Toc472927802)

[Rest of demographic elements 7](#_Toc472927803)

[Age 7](#_Toc472927804)

[Results 8](#_Toc472927805)

[Export 8](#_Toc472927806)

|  |  |  |
| --- | --- | --- |
| Revision date | Author | Comment |
| 9/30/16 | Javier Sanz | Original document |
| 1/19/17 | Javier Sanz | Adaptation of original document to new method of extraction |
|  |  |  |

Document revision history

Data and Quality Service Dashboard

SHRINE NEtWork data extraction instructions

# Introduction

Thank you for taking the time to pull data for us! This document will detail the steps required to extract and send the data we need from your system. Our ultimate goal is to produce an extract of aggregated counts by year for each ontology element in your data. This will help us produce a dashboard like the one below which can be used for multiple purposes:

* Data quality
* Data density analysis
* Detect data growth
* Identify data trends

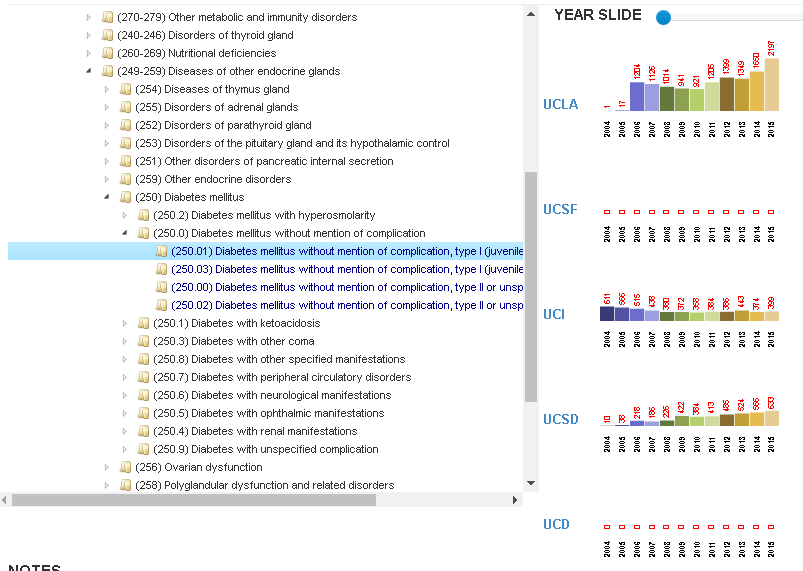


Figure 1 - Diagnosis dashboard.

# Ontology Enhancements

## Ontology Identifier

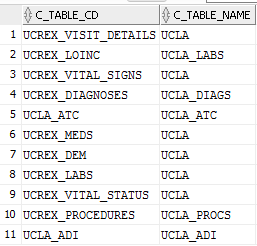
Considering the different ontology distributions that are available in each of the sites, we are providing a script that will use the [TABLE\_ACCESS] table to identify these elements and build an ad hoc merged version of them. See below an example of the process in UCLA system. [this stage corresponds to Step 1 in the script]. Additionally, this process will generate a numeric ont\_ID field to enhanced the counting/grouping process.

Figure 2 - Table\_access view

Figure 3 - Ontology conglomerate

## Parent/child 🡪 Node

The next action (Step 2) is to identify the parent/child relationship among elements and levels within the ontology, so it can be used later to generate roll up counts more efficiently. We use the **[c\_fullname]** field and the ‘\’ character to segment each element by name and level.

**LEVEL 1 LEVEL 2 LEVEL 3 LEVEL 4**

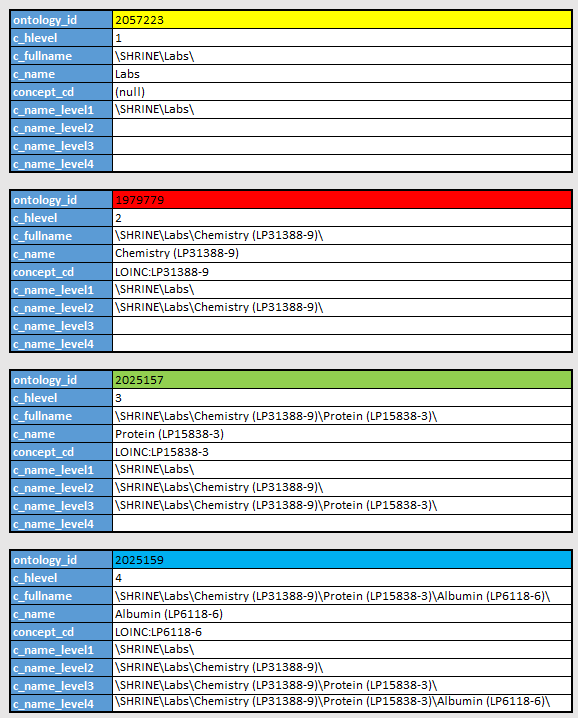
**\SHRINE\Labs\Chemistry (LP31388-9)\**Protein (LP15838-3)\Albumin (LP6118-6)\

Figure 4 - Ontology sample

This is finalized (Step 3) by a query with a series of self joins that, in combination with the **[c\_hlevel]** attribute, will ultimately generate a table that includes one row for each ontology record and one column for each parent ontology element corresponding to the ontology record row value (including itself). This is what you see moving up the tree structure in the ontology.

For example, the first ontology record in Figure 5 is ONT\_ID 2025159, which has a hierarchical level of 4. This ontology record will display 4 additional columns – the ONT\_IDs for parental records from levels 1,2 and 3 and itself. The next ontology record, ONT ID 2025157 demonstrates the same column attributes structure except it will only show 3 additional columns.

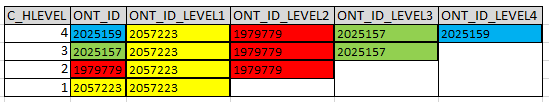


Figure 5 - Ontology nodes sample

# DATA PROCESSING

## Observation Fact extraction

The next step is to query [OBSERVATION\_FACT] and build a temporary table [DASH\_OBSERVATION] that includes the generated [ONT\_ID] values and their associated parent ONT\_IDs. This process corresponds to Step 4.1 in the script.The [DASH\_OBSERVATION] temporary table provides us with a modified version of the [OBSERVATION\_FACT] table that is more efficient in aggregating and rolling up the unique patient counts at each ontology level.

Figure 6 - Observation\_fact workflow

It is important to point out that by copying [OBSERVATION\_FACT] into [DASH\_OBSERVATION], it will be necessary to allow sufficient space in your schema for it. Please note this table will be deleted after we generate the final output.

## visit

This step uses a dynamic query approach where the different elements in the ontology are parsed together to match the relevant items from the ontology to the [VISIT\_DIMENSION] table.

The script generates a series of INSERT/SELECT statements that mimics what I2B2 does every time a request is made. It parses different elements into a final command that will ultimately query [VISIT\_DIMENSION] and aggregate patient counts accordingly. [this step corresponds to Step 6 in the script]

## Demographics

The next step is to gather the demographic information recorded in [PATIENT\_DIMENSION] and add it to [DASH\_OBSERVATION] table. In this stage, we need to make a differentiation between age and the rest of elements (race, gender, marital status, etc…)

### Rest of demographic elements

For the later set, we will use the same approach as in the previous case by querying [PATIENT\_DIMENSION] and replacing the [C\_DIMCODE] with the new ont\_id field on top of adding the parents’ [ont\_id]s fields. [this step corresponds to Step 4.1 in the script]

Figure 7 - Demographics workflow

In this case, we will dynamically generate a series of insert-select statements based on the different demographic elements included in the patient dimension (*'language\_cd', 'religion\_cd', 'marital\_status\_cd', 'race\_cd', 'sex\_cd'*). The final result is added to the [DASH\_OBSERVATION] table.

### Age

For age however, we will use again a more dynamic approach by generating a series of queries defined by some of the elements in the new version of the ontology.

The results are temporarily stored in [DASH\_AGE] before being merged with the rest of the results on Step 5.

# Results

The final step consists on iterating through the [DASH\_OBSERVATION] table and count records grouping them by year and each [ONT\_ID\_LEVEL]. This will produce the counts at each level of the ontology which ultimately provides individual and rolled up counts for each ontology element present in the original [OBSERVATION\_FACT] table.

Because each site might have a different depth of ontology level utilization, the dynamic query generates the exact number of queries required in order to account for each level in your system.(i. e. UCLA is currently using 17 levels). The final results are stored in the [DASH\_< SiteAcronym>] table. [this step corresponds to Step 6 in the script].

## Export

The last step is to export the contents of [DASH\_<SiteAcronym>] to a csv named **MMDDYYYY\_<SiteAcronym> \_COUNTS.csv** and load it into the corresponding Box folder.

In addition to the counts, it will be necessary to provide the [DASH\_ONTOLOGY] table using the same pattern. Name the file **MMDDYYYY\_<SiteAcronym> \_ONTOLOGY.csv.**

The [DASH\_ONTOLOGY] table is needed in order to reconcile results from all sites and be able to map all records while respecting the original ontologies used by the I2B2 framework. The ont\_id field is a feature to simplify the counting process but in no way represents a modification of the ontologies in place and maintain by our organizations.

Once the system briefly mentioned at the beginning of this document is available, you will receive the instructions and credentials to access it to directly benefit from the data extract produced here.

I’m including my contact info in case you need to get a hold of me:

Javi Sanz

[jsanz@ucla.mednet.edu](mailto:jsanz@ucla.mednet.edu?subject=Macey%20-%20RC:196%20-%2015-001792%20-%20characteristics%20of%20the%20OSA%20population%20)