OREGON HEALTH AND SCIENCE UNIVERSITY

DEPARTMENT OF MEDICAL INFORMATICS AND CLINICAL EPIDEMIOLOGY

Documentation of Huque's Python Program

Observational Medical Outcomes Partnership Common Data Model (OMOP CDM) Python Program

Documentation

Version 1.0

August 30, 2019

Revision History

Date	Version	Description	Author
August 30, 2019	1.0	Parses and prints XML	Alyssa Huque

Table of Contents

1. Introduction

The passage of the Health and Information Technology for Economic and Clinical Health (HITECH) Act of the American Recovery and Reinvestment Act (ARRA) of 2009 incentivized medical institutions to digitize health records[1]. However, with this transition, there were difficulties standardizing data in electronic health records (EHRs). As such, today there are many inconsistencies in the information captured by various institutions[2]. Observational Health Data Sciences and Informatics (OHDSI), an interdisciplinary group, has extended the Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) to improve the quality of EHRs across institutions[2].

The OMOP CDM standardizes the information captured by EHRs across various types of medical data sources[3]. It creates a consistent format, the data model, and a consistent representation, the standardization of medical codes and vocabularies[2]. This allows institutions with the OMOP CDM to perform systematic and standard analytics[2]. Standardizing the data in EHRs can accelerate research, improve collaborative efforts, and help to identify medical associations[2]. The OMOP CDM allows from data analysis from multiple, disparate data sources, supporting clinical practice and administrative claims[2].

The OMOP CDM is a patient-centric data model, organized by inter-connected tables that centered on a PERSON table[3]. Each table contains specific information about a patient's medical interactions that are all related to each other[3]. For example, if a patient has a medical visit this creates a VISIT_OCCURRENCE table. If at that visit the doctor prescribes a medication, this creates a DRUG_EXPOSURE table which is mapped back to the VISIT_OCCURRENCE table which is mapped back to the PERSON table.

Each table is organized in a similar fashion. There are fields called *[table]_id*, *[field]_concept_id*, *[field]_source_value*, and *[field]_source_concept_id*. *[table]_id* is a unique system-generated ID for each table that allows for calls to specific tables. This is normally the first field in a table. *[field]_source_value* maintains the information as a medical professional notes it. For example if the nurse makes a note of a patient's gender, this field, *gender_source_value*, would simply be "female." *[field]_source_concept_id* maintains the standardized vocabulary the medical institution uses. For example, if the medical institution uses SNOMED, *gender_source_concept_id* would be "24815202." The OMOP CDM does not lose the source data and the way it is represented at a specific medical institution. *[field]_concept_id* is the standardization that the OMOP CDM implements. This standardizes medical data one step further, using standardized codes that can be mapped to medical vocabularies[4]. This functions as a metathesaurus. These standardized codes can be found at http://athena.ohdsi.org [4].

1.1 Document Overview

The Observational Medical Outcomes Partnership Common Data Model (OMOP CDM) Python Script Business Requirements Document (BRD) provides insights into assumptions, constraints, and conclusions made during the process of the development. This is intended to help the research team who continues this process in understanding the foundation that has been laid and any modifications that needs to be done.

This document is organized into three sections – Introduction, Requirements Summary, and Process Mappings.

Within *1. Introduction,* there is background information under the title, "Introduction." *Section 1.1 Document Overview* explains the layout and purpose of this document. *Section 1.2 References* contains all the sources cited in this document. *Section 1.3 Glossary of Terms* has any important key words or phrases used in this document and their definitions.

Within *2. Requirements Summary*, there is a general overview explaining the structure of the code that this document summarizes underneath the title. *Section 2.1 Goals and Objectives* states the goal of moving OHSU's clinical data to the OMOP CDM. *Section 2.2 Problem Statement* explains the need to move to this standardized system. *Section 2.3 Problem Description* explains how the OMOP CDM resolves the issues explained in *Section 2.2*. *Section 2.4 Scope* states the full extent of the data set that will be transitioned to the OMOP CDM. *Section 2.5 Stakeholders* outlines the people and organizations at OHSU that are involved in this project. *Section 2.6 Constraints* discusses the difficulties and shortcomings of this python program. *Section 2.7 Assumptions and Dependencies* explains 12 assumptions made in mapping the data from OHSU to the OMOP CDM. Finally, *Section 2.8 Planning for Future Phases* lists the next steps required to fully implement the OMOP CDM at OHSU.

Within *3. Process Mappings,* an explanation of the mapping documents under *Section 3.1* is under the title. *Section 3.1 Mapping Files* contains the links to two different forms of the same files, both showing the OHSU to OMOP mappings.

1.2 References

[1]Rodriquez, Alison. "HITECH Act Resulted in Significant Gains in HER Adoption in Hospitals." *AJMC,* 15 Aug. 2017, www.ajmc.com/newsroom/hitech-act-resulted-in-significant-gains-in-ehr-adoption-in-hospitals.

[2]"OMOP Common Data Model." *OHDSI,* www.ohdsi.org/data-standardization/the-common-data-model. Accessed 26 Jun. 2019.

[3] Ohdsi. “OHDSI/CommonDataModel.” *GitHub*, github.com/OHDSI/CommonDataModel/wiki/Standardized-Clinical-Data-Tables.

[4] Belenkaya, R., Blacketer, C., Hripcsak, G., Natarajan, K., O’Hara, D., Reich, D., Roa, G., Torok, D., Zandt, M., Velez M., Voss E.. *OMOP Common Data Model and Standardized Vocabularies*. https://www.ohdsi.org/wp-content/uploads/2017/10/OHDSI-Vocabulary-CDM-Tutorial-2017-2017.10.11.pdf. PowerPoint Presentation.

1.3 Glossary of Terms

[table]_id	A field within the OMOP CDM table which is a unique system-generated ID for each table that allows for calls to specific tables
[field]_source_value	A field within the OMOP CDM table which has the information as a medical professional notes it; information prior to mapping to medical codes/vocabularies
[field]_source_concept_id	A field within the OMOP CDM table which has the standardized vocabulary/code used by the medical institution
[field]_concept_id	A field within the OMOP CDM table which has standardized vocabularies unique to the OMOP CDM. Can be found at http://athena.ohdsi.org.
EHR	Electronic Health Records
OHDSI	Observational Health Data Science and Informatics – the name of a collaborative, interdisciplinary group that has extended the OMOP CDM
OMOP CDM	Observational Medical Outcomes Partnership Common Data Model – the name of the standardized system
Queue	a data structure; the data that is first entered into the queue is the first that is removed from the queue
XML	Extensible Markup Language file – a type of file, similar to HTML, that has a tree structure

1.4 Helpful Resources

The Book of OHDSI

https://ohdsi.github.io/TheBookOfOhdsi/

OMOP CDM GitHub

https://github.com/OHDSI/CommonDataModel/wiki

OMOP-CM and Standardized Vocabularies Tutorial YouTube video

https://youtu.be/wLTpWVmuuxg

Common Data Model & Extract, Transform and Load Tutorial YouTube video

https://youtu.be/T03GC4IBrLo

OHDSI Forum

https://forums.ohdsi.org/

2. Requirements Summary

The program this document summarizes a python program that is able to extract OHSU EHR data and print a new XML document that is formatted and organized by the standards set by the OMOP CDM. There are five files that are used in this program. These files are parse_OHSU.py, queues.py, print_OMOP_xml.py, xml_content.py, and main.py. main.py is only used to run the program. This program works with four files in six steps:

1. Parse the OHSU XML source files (parse_OHSU.py)
2. Store the EHR patient data into a list (parse_OHSU.py)
3. Add all lists to a queue (queues.py)
4. Remove one list at a time from the queue (queues.py)
5. Index the list and assign the information to the OMOP field (xml_content.py)
6. Print a new XML file that follows the standards and organizational pattern of the OMOP CDM (print_OMOP_xml.py)

2.1 Goals and Objectives

The goal of this project is move all of OHSU's clinical data stored in the data warehouse into the OMOP CDM. This requires reformatting and reorganizing the data as OHSU stores and implementing OMOP's standardized codes and data model. This is in order to improve medical research and analysis as this standardization allows for network studies, ease in combining data sets, and use of available standardized tools.

2.2 Problem Statement

Currently there is no way to cohesively view all the data a person accumulates while receiving healthcare. Additionally, there are many inconsistencies in the way EHR data is stored across various medical institutions. The OMOP CDM creates consistency across data sources that aims to resolve these issues.

2.3 Project Description

The OMOP CDM standardizes the information captured by EHRs across various types of medical data sources[3]. It creates a consistent format, the data model, and a consistent representation, the standardization of medical codes and vocabularies[2]. This allows institutions with the OMOP CDM to perform systematic and standard analytics[2]. Standardizing the data in EHRs can accelerate research, improve collaborative efforts, and help to identify medical associations[2]. The OMOP CDM allows from data analysis from multiple, disparate data sources, supporting clinical practice and administrative claims[2].

2.4 Scope

To transform all the data within the OHSU data warehouse into the OMOP CDM.

2.5 Stakeholders

Stakeholder	Support Role
OHSU Data Warehouse	Data Warehouse
Dr. William Hersh	Mentor
Dr. Aaron Cohen	Researcher
Dr. Steven Bedrick	Researcher
Dr. Steven Chamberlin	Researcher
National Library of Medicine	Funding through Grant 2T15LM007088
Alyssa Huque	Summer 2019 Intern
Aaron Berger	Summer 2019 Intern

2.6 Constraints

Printing Multiple XML Files: This python program does not have the capabilities to print multiple tables into the same XML file. This was due to being unable to have the time within a 10 week internship to develop this software capability. This is important if a patient has multiple data tables. For example, if a patient has multiple prescriptions there would be multiple DRUG_EXPSOURE tables, one for each medication. Every time an XML file is printed it is named [OMOP TABLE]_[person_id].xml. However, if there are multiple data tables it would simply write over the file, losing any data that was previously in the file. This is an egregious error. However, if these files are written into a relational database it would be beneficial to have separate XML files for each table in order to create the relationally between XML files. This just requires distinguishing between XML files of the same data table (for example, could be named [OMOP TABLE]_[person_id]_[counter].xml). This inability to print multiple data files made mapping the OHSU vitals table difficult and it was left unmapped in the python code (see **vitals** below).

result_comments: Mapping the OHSU result_comments table is a large issue. Currently, it does not appear that the OMOP CDM has a place for free-text. As such, this OHSU table does not have a place it can be mapped to in the OMOP CDM despite containing valuable information. It may be mapped to the OMOP NOTE table but I avoided doing such as I did not want to pollute the OMOP notes table with excess information (the NOTE table is not a catch-all bucket table so I avoided making it such). One could create a new table for this information but it would deviate from the standardization of the CDM. Additionally, any OHSU customization would be unused in network studies. This table was left unmapped in the python code.

[table]_id: Currently, the OHSU data field SOURCE_SYSTEM_PAT_ID is used in the *person_id* field. However, person_id is intended to be a system generated ID, just like all the other *[table]_id* so eventually this will need to be changed to have a unique ID to the OMOP system. Additionally, this SOURCE_SYSTEM_PAT_ID starts with a “Z” and the *person_id* field is an integer. As a result, eventually any unique ID OHSU generates to identify a patient, SOURCE_SYSTEM_PAT_ID and OHSU_MRN, will be lost. It may be possible to use the SOURCE_SYSTEM_PAT_ID without the Z as the *person_id* but it would be inconsistent with the other *[table]_id*.

Administrative Claims: The OMOP CDM claims to handle both administrative claims and EHR information[2]. However, there was difficulty in finding the tables in which administrative claims could be mapped to. As such, currently any fields containing information about administrative claims are considered to a part of the data loss of transitioning to the OMOP CDM. This includes (ambulatory_encounters) INSURANCE_CLASS, INSURANCE_GROUP, INSURANCE_PAYOR, (procedures_ordered) PROC_BILLING_TYPE, and (encounter_diagnosis) BILLING_DX_FLAG.

vitals: (VITALS) BMI, BP_DIASTOLIC, BP_SYSTOLIC, BSA, HEIGHT_CM, PAINT_SCORE_NUMBER, PULSE, RESPIRATION_RATE, SPO2, TEMPERATURE_C, and WEIGHT_KG are all measurements within the OHSU vitals table that are mapped to the OMOP measurement table. Since the python program is currently unable to distinguish between multiple of the same tables for the same patient (see **Printing Multiple XML files** above) and this is essential for mapping the vitals table to the MEASUREMENT table of the OMOP CDM, this table was left unmapped in the python code.

microbiology_results: I lack the medical knowledge to fully understand the purpose and difference between the (microbiology_results) ANTIBIOTIC, GRAM_STAIN, LAB_ORGANISM, ORGANISM_FOUND, and SEE_NOTE fields. I could not properly map these fields so they are currently marked as data lost when transitioning to the OMOP CDM. I could not find an OMOP CDM field for this information. There may be a field that would be applicable and I simply did not understand the medical terminology equivalent in the OMOP CDM.

ICD Code Hierarchies: For ICD codes, the type of code used (ICD10, ICD9, or just ICD) depends on the year the data was taken. The program needs a way to distinguish that if there is no data in the ICD10 field for OHSU, then use the data in the ICD9 field. If there is no data in the ICD9 field, then use the ICD code. Currently this functionality does not exist within the python program and could logically be created by numerous conditional (if/else) statements.

hospital_encounters: Within the OHSU hospital_encounters table there are different forms of diagnoses (DX, DX2, DX3, DX4). The OMOP CDM does not have a way to record multiple diagnoses within the same table. I lacked the medical knowledge to distinguish between these diagnosis fields as well. These fields are either repetitive and would be lost information when transitioning to the OMOP CDM or these fields are important and would need their own CONDITION_OCCURRENCE table within the OMOP CDM for each of those fields. If the latter is true, this is hindered by the program's inability to print multiple XML files with the same table for the same patient (see Printing Multiple XML Files above).

surgeries: Logically the OHSU surgeries table would be mapped to the PROCEDURE_OCCURRENCE table within the OMOP CDM. However, we noticed most of the information retained within this table are prognosis/diagnosis information. As such we mapped it to the OMOP CDM. It would be beneficial to some with a deeper understanding of medicine to verify specifically this table.

2.7 Assumptions and Dependencies

Assumption 1: In the OHSU administered_medications table, RX_ORDER_PLACED, RX_TAKEN_TIME, and RX_ORDERED_STARTED, though capturing different types of starts datetimes, are essentially repetitive information and it is appropriate to have this data loss. Only the RX_ORDER_TO_START_DATE is kept in the OMOP CDM, mapped to (DRUG_EXPOSURE) drug_exposure_start_date. This same assumption applies to medications_ordered.

Assumption 2: In the OHSU administered_medications table, RX_DESCRIPTION, though capturing different information than RX_DISPLAY_NAME, it is essentially repetitive information and it is appropriate to have this data loss. Only RX_DISPLAY_NAME is kept in the OMOP CDM, mapped to (DRUG_EXPOSURE) drug_source_value.

Assumption 3: The OMOP CDM does not need to contain information about how to contact the patient since the OMOP CDM is primarily used for research. Thus, data fields containing patient contact information is lost in the data transition to the OMOP CDM. The OHSU data fields ADDRESS_PHONE and EMAIL are both unmapped to the OMOP system.

Assumption 4: any type of flag OHSU records does not have a place in the OMOP CDM. (demographics) BIO_SAMPL_OPT_OUT_FLG, GENETC_OPT_OUT_FLG, (encounter_diagnosis) BILLING_DX_FLAG, ENC_DX_FLAG, FOLLOWUP_DX_FLAG, HOSP_ADMIT_FLAG, MED_HX_DX_FLAG, ORDER_MED_DX_FLAG, ORDER_PROC_DX_FLAG, and REFERRAL_DX_FLAG are all lost when transitioning to the OMOP CDM despite containing valuable information.

Assumption 5: There is no place within the OMOP CDM for keeping the information of the (notes) COSIGNER_NAME and COSIGNER_SPECIALTY within the OMOP CDM NOTE table.

Assumption 6: In the OHSU notes table, NOTE_FILING_DATE and NOTE_CREATED_DATE, though capturing different types of date, are essentially repetitive information and it is appropriate to have this data loss. Only NOTE_DATE is kept in the OMOP CDM, mapped to (NOTE) note_datetime.

Assumption 7: In the OHSU lab_results table, RX_DESCRIPTION and RX_DISPLAY_NAME used to be mapped to (DRUG_EXPOSURE) drug_exposure_source_value. Once we started programming we realized it may difficult to have both fields mapped to the same place so it was established that RX_DISPLAY_NAME was more important to keep and RX_DESCRIPTION is appropriate to lose in the data transition to the OMOP CDM.

Assumption 8: In the OHSU procedures_ordered table, PROC_CODE, though capturing different information from the PROC_CPT_CODE, is essentially repetitive information and it is appropriate to have this data loss. Only PROC_CPT_CODE is kept in the OMOP CDM, mapped to (PROCEDURE_OCCURRENCE) procedure_source_concept_id.

Assumption 9: In the OHSU current_medications table, GENERIC_NAME_1 and GENERIC_NAME_2, though capturing different types of medication information, are essentially repetitive information and it is appropriate to have this data loss. Only the MED_NAME is kept in the OMOP CDM, mapped to (DRUG_EXPOSURE) drug_source_value. This same assumption applied for the OHSU medications_ordered table.

Assumption 10: The OMOP CDM does not have a field for free text (other than the NOTE table but that is intended for the doctor’s note – notes in the OHSU table). As a result, (microbiology_results) ITEM_FREE_TEXT does not have a place in the OMOP CDM.

Assumption 11: The OMOP CDM does not have a field for (hospital_encounters) CHIEF_COMPLAINT.

Assumption 12: In the OHSU hospital_encounters table, INPATIENT_ADMIT_TIME, though capturing slightly different information than HOSPITAL_ADMIT_TIME, is essentially repetitive information and it is appropriate to have this data loss. Only HOSPITAL_ADMIT_TIME is kept in the OMOP CDM, mapped to (VISIT_OCCURRENCE) visit_start_datetime.

2.8 Planning for Future Phases

Mappings: refine the OHSU to OMOP mappings to ensure minimal data loss while transitioning to the OMOP CDM.

Efficiency: improve the efficiency of the python program to handle a few million data records.

Relationally: create relational linkages of multiple XML files for a given patient

[field]_concept_id: implement the OMOP standardized codes in the *[field]_concept_id* fields using http://athena.ohdsi.org

Multiple Data Tables: as discussed in *2.7 Constraints* under *Printing Multiple XML Files*, the program needs the ability to print multiple data tables for the same patient without overwriting the data.

Hierarchies: as discussed in *2.7 Constraints* under *ICD Code Hierarchies*, the program needs the ability to distinguish what code to map into the OMOP CDM depending on the data available in the file and/or the date the data was recorded.

PROVIDER: Within the OMOP CDM there is a provider table which is not created in this code. We have various fields mapped to the PROVIDER table. However, since this is a domain that other tables are mapped to I did not have the time to create it. It is important this table is created when modifying the code.

3. Process Mappings

Mapping the OHSU data fields and tables to the OMOP CDM data fields and tables is a large difficulty in the transition to the OMOP CDM. In *3.1 Mapping Files* you'll see two different links to the same data that shows the attempts we made at mapping OHSU to the OMOP CDM. Link 1A is the file in excel. Link 1B is the file as a google spreadsheets. Both spreadsheets have certain cells that are highlighted with different colors. The explanation of those colors are explained in the **COLOR CODING KEY** below. The first sheet shows the mappings from OHSU to the OMOP CDM. The second sheet shows how the OHSU data is mapped onto the OMOP CDM.

COLOR CODING KEY	
🟦	does not map to the OMOP CDM
🟪	we thought it mapped but once we started programming realizing it was repetitive
🟩	Not in the OMOP CDM but important information
🟨	Need to research some more
🟥	These fields inability to map no longer makes the information mapped from the table unique
🟩	unique system generated IDs

3.1 Mapping Files

Link 1A - Mapping Excel File

OMOP_CDM_Mapping.xlsx

Link 1B - Mapping Google Spreadsheets File

https://docs.google.com/spreadsheets/d/17Yy4m2f38PUrM5krUJZtaOrqtMqbjzbXeIvPs3cdSCE/edit?usp=sharing