Maine APCD to OMOP ETL specification

This ETL specification contains the logic and business rules used to map the Maine All Payer Claims Data into the OMOP Common Data Model. Each table in the Extract-Transform-Load (ETL) process is described in detail. There are three schemas used in the database that hold tables at different stages of the ETL. This document outlines the tables in the order that they are created during the ETL process and describes the dependencies between them.

Schemas

Schema name	Description
raw	The source data with no transformation logic applied
stg (Stage)	The staged source data with some basic cleaning and restructuring applied
trm (Transformation)	The transformation schema that holds intermediate tables and mapping tables that are part of more complex transformation operations
cdm	The final CDM tables (see https://github.com/OHDSI/CommonDataModel/wiki/)

CDM vocabulary tables

The CDM vocabulary tables contain all of the code sets in the data and the relationships between them. Source codes are mapped to their standard code counterparts using the vocabulary tables. The tables can be downloaded from http://athena.ohdsi.org/. We need to make sure they include CPT, HCPCS, ICD9CM, ICD10CM, NDC, and NUCC code sets. These tables need to be loaded into the cdm schema before the ETL process begins.

Source tables

Table name	Description
raw.mc	Medical Claims - one row per claim line

raw.me	Medical Eligibility - one row per person-plan-month combination
raw.pc	Pharmacy Claims (pharmacy eligibility is not used)
raw.npi	NPPES NPI table - one row per NPI/provider - downloaded from https://download.cms.gov/nppes/NPI_Files.html
raw.uszips	US zipcode data - one row per zipcode - downloaded from https://simplemaps.com/data/us-zips

Source to target mappings

Each source table will be mapped into one or more target tables. One or more intermediate tables will be used during the mapping of each source table.

Source table	**Target tables**
raw.mc	cdm.visit_occurrence cdm.visit_detail cdm.condition_occurrence cdm.procedure_occurrence cdm.observation cdm.drug_exposure cdm.device_exposure cdm.person
raw.me	cdm.person cdm.observation_period
raw.pc	cdm.drug_exposure
raw.npi	cdm.provider cdm.care_site cdm.location
raw.uszips	cdm.location

Creation of staging tables

The creation of the staging from the source data as the first step in ETL process. The staging step involves applying some initial data cleaning rules and restructuring the source data into a format that is easier to work with. Variable number prefixes on the source data have been removed in this documentation for readability. The source variable names are uniquely identified without the number prefixes (e.g. raw.me.rx instead of raw.me.me019_rx).

stg.mc

stg.mc will contain all of the columns and rows in raw.mc except for the following rules

- Only final claims will be mapped to stg.mc by inner joining raw.mc with raw.claim_consolidation
- Claim lines that are missing **mhdo_personid** or **mhdo_claim** will be dropped
- Diagnosis code columns will not be included in stg.mc and instead be mapped to stg.dx
- Claims with more that one than one personid will be dropped

<full table specification is omitted>

stg.dx

stg.dx is a "long" format diagnosis code table with one row per diagnosis code. It also contains flags for primary diagnosis and admitting diagnosis as well as the vocabulary for the diagnosis code (ICD9CM or ICD10CM). Rows containing duplicate information are deduplicated.

destination field	**source field**	**Mapping rule**
mhdo_claim INT [not null]	raw.mc.mhdo_claim	This is simply the claim number on the source record.
dx VARCHAR(15)	**ICD9CM** raw.mc.dx1-24 raw.mc.admdx raw.mc.ecode **ICD10CM** raw.mc.ecom1-24 raw.mc.prindgns raw.mc.admdgns raw.mc.rfvdgns1-3 raw.mc.othdx1-othdx24	Each dx code + claim number combination will generate one record in the stg.dx table. Dx code fields with NULL values are not mapped.
vocabulary VARCHAR(10)	assigned	'ICD9CM' OR 'ICD10CM' based on the source field (see row above). Codes are not checked against known ICD code lists.

primary_flag BIT [not null]	assigned	1 if stg.dx source field is mc.dx1 or prindgns, 0 otherwise
rfv_flag BIT [not null]	assigned	1 if stg.dx source field is one of mc.rfvdgns1-3 (i.e. a “reason for visit” diagnosis), 0 otherwise
admit_flag BIT [not null]	assigned	1 if stg.dx source field is mc.admdgns, 0 otherwise
poa_flag BIT [not null]	raw.mc.poa (prindgns) raw.mc.poa1-24 (ecom1-24) raw.mc.othpoa1-24 (othdx1-24)	Some ICD10CM source dx codes have an associated “present on admission” (poa) field. The associations are shown in parentheses to the left. Assign NULL if source dx code does not have an associated poa field, 1 if associated poa field is ‘Y’ or ‘1’, and 0 otherwise.
ecode_flag BIT [not null]	assigned	1 if source diagnosis code field is an “external cause of injury” code (ecode, ecom1-24), 0 otherwise.

icd_proc

The stg.icd_proc table will store icd 10 procedure codes in a long format. The transformation is essentially the same as the stg.dx table. ICD procedure codes are header level variables.

destination field	source field	mapping rule
patkey INT [not null]	raw.mc.mhdo_memberid	This is simply the patient id number on the source record.
mhdo_claim INT [not null]	raw.mc.mhdo_claim	This is simply the claim number on the source record.
vocabulary VARCHAR(20) [not null]	assigned based on icd_proc source field	If the icd_proc source field is an icd9 procedure code (op) then assign ‘ICD9Proc’ otherwise assign ‘ICD10PCS’ for ICD 10 procedure codes.
icd_proc VARCHAR(10) [not null]	**ICD 9 Procedure codes** raw.mc.op **ICD 10 Procedure codes** raw.mc.prnprcdrcd raw.mc.othprcdrcd1-24	Each ICD procedure code + claim number combination will generate one record in the stg.dx table. Code fields with NULL values are not mapped.

prim_flag BIT [not null]	assigned based on icd_proc source field	1 if source code is op or prnprcdrcd (primary procedure), 0 otherwise

stg.me

The stage medical eligibility table will be a collapsed version of the source data, raw.me (one row per person-plan-month combination), to one row per person-month combination.

The medical eligibility table will be our source for the observation_period table.

destination field	source field	mapping rule
patkey INT [not null]	raw.me.mhdo_member_id	Simply copy the mhdo_memberid from the source records (all records with the same patient, year, and month)
year SMALLINT [not null]	raw.me.year	Copy directly from source records. Use only distinct combinations of patient, year, month.
month TINYINT	raw.me.month	Copy directly from source records. Use only distinct combinations of patient, year, month.
birthyear SMALLINT	raw.me.age	For each record in the source data a potential birthyear is calculated by subtracting age from year of eligibility. For each patient the most frequently occurring two birthyears are identified. These two most frequent birthyears should, in most cases, be one year apart. Finally take the smaller of the two most frequent possible birthyears which should be the correct birthyear in most

		cases.
gender VARCHAR(5)	raw.me.gender	Use records of raw.me where gender is either ‘M’ or ‘F’. Summarize them by patient counting up how many time ‘M’ or ‘F’ occurs for each patient. Assign the most frequently occurring gender to each patient. When there is a tie, choose one at random. Assume that gender is constant over time.
patcity VARCHAR(100)	raw.me.patcity	Use the most frequently occurring non-missing patcity for each unique combination of patient, year, and month. When there is a tie, choose one at random.
patst VARCHAR(10)	raw.me.patst	Use the most frequently occurring non-missing patst for each unique combination of patient, year, and month. When there is a tie, choose one at random.
county_fips VARCHAR(10)	raw.me.county_fips	Use the most frequently occurring non-missing county_fips for each unique combination of patient, year, and month. When there is a tie, choose one at random.
medicare_flag BIT [not null]	raw.me.payer_type	Was there a Medicare eligibility record for a given patient, year, month combination? (1 = yes, 0 = no)
medicaid_flag BIT [not null]	raw.me.payer_type	Was there a Medicaid eligibility record for a given patient, year, month combination? (1 = yes, 0 = no)
comercial_flag	raw.me.payer_type	Was there a Commercial

BIT [not null]		eligibility record for a given patient, year, month combination? (1 = yes, 0 = no)
rx_flag BIT [not null]	raw.me.rx	Did the medical eligibility record include pharmacy eligibility? i.e. Did the patient-year-month combination contain at least one eligibility record where raw.me.rx = ‘Y’? If so assign 1, otherwise assign 0.
pharmacy_flag BIT [not null]	raw.pe.mhdo_memberid raw.pe.year raw.pe.month	Did the patient, year, month combination have a matching record in the pharmacy eligibility table? If yes then assign 1, otherwise assign 0.

stg.npi

The npi table will be a subset of the NPPES provider table. We do not use any of the MHDO supplied provider tables and providers will be uniquely identified using their NPI. The stg.npi table will only include NPIs that are actually in the medical claims table will only include a small number of columns in the raw.npi table. We only need the name, npi, type, and location of each provider.

destination field	source field	mapping rule
npi VARCHAR(10)	raw.npi.npi	Copied directly from source
enttype	raw.npi.entity_type_code	If entity type code = 1 then ‘INDIVIDUAL’, else if entity type code = 2 then ‘ORGANIZATION’, otherwise set to NULL
orgname	raw.npi.provider_organization_name	Copied directly from source
orgname2	raw.npi.provider_other_organization_name	Copied directly from source

prvname	raw.npi.provider_first_name raw.npi.provider_middle_name raw.npi.provider_last_name	Concatenation of the providers first, middle, and last name. If all three fields are NULL then assign NULL.
prvaddress	raw.npi.provider_first_line_business_practice_location_address	Copied directly from source
prvaddress2	raw.npi.provider_second_line_business_practice_location_address	Copied directly from source
prvcity	raw.npi.provider_business_practice_location_address_city_name	Copied directly from source
prvstate	raw.npi.provider_business_practice_location_address_state_name	Copied directly from source
prvzip	raw.npi.provider_business_practice_location_address_postal_code	Copied directly from source
primary_taxonomy_code	healthcare_provider_taxonomy_code_1-15 healthcare_provider_primary_taxonomy_switch_1-15	Each provider can have up to 15 taxonomy codes. Each taxonomy code has an associated switch that indicates whether or not that code is the providers primary taxonomy. Starting with 1, find the first taxonomy code = ‘Y’ and use its associated taxonomy code as the provider’s primary taxonomy. If none of the switches are = ‘Y’ then assign NULL.
last_update_date	raw.npi.last_update_date	Copied directly from source

stg.pc

Stage pharmacy claims

stg.patkey_list

This is a control table used to control batch processing and testing of the ETL execution. The ETL needs to batch process some of the operations to accommodate the large data size. Since all records for one patient are independent of all records for another patient we use patient key to break up the processing into batches.

This table contains the ids for all patients who will be mapped in the entire ETL run. There is one record per patient id. During batch operations subsets of patient keys will be made from this table. Each subset of patients will be processed in a batch until all all patients in the stg.patkey_list table have been processed.

During testing this table can be used to run the entire ETL on only a subset of patients which is helpful for debugging.

destination field	**source field**	**mapping rule**
patkey_chr VARCHAR(30)	raw.me.mhdo_memberid	Deduplicate and cast to VARCHAR
patkey_int INTEGER	raw.me.mhdo_memberid	Deduplicate and cast to INTEGER

Creation of health system tables

The three health system tables are used in later parts of the ETL process

cdm.location

destination field	**source field**	**mapping rule**
location_id INTEGER	auto-generated integer	
address_1 VARCHAR(50)	NULL	Assign NULL
address_2 VARCHAR(50)	NULL	Assign NULL
city VARCHAR(50)	raw.uszips.city	Copy verbatim from source (and convert to correct data type)

state VARCHAR(2)	raw.uszips.state_id	Copy verbatim from source
zip VARCHAR(9)	raw.uszips.zip	Copy verbatim from source
county VARCHAR(20)	raw.uszips.county_name	Copy verbatim from source
country VARCHAR(100)	‘USA’	Use ‘USA’ for all location records
location_source_value VARCHAR(50)	raw.uszips.zip	Copy verbatim from source
latitude FLOAT	raw.uszips.lat	Copy verbatim from source
longitude FLOAT	raw.uszips.lng	Copy verbatim from source

cdm.location_history

This table was not populated but might be useful to add at some point in the future.

cdm.care_site

dependencies: cdm.location

Care sites are meant to be physical locations where medical care is provided. These would include hospitals, clinics, outpatient surgery facilities, etc. In claims data we identify care sites using the NPI table. We can separate individuals from organizations easily but it is difficult to separate “brick and mortar” physical locations from organizational physician billing groups that are more like a business entity than a care site. Sometimes the line between the two types of organizations can even be blurry (e.g. a physician billing group with a number of offices that they submit professional bills out of).

We define care sites as organizational NPIs that submit facility claims and assume that this will capture most care sites while excluding billing entities. As a consequence we will not capture some outpatient care sites that do not submit facility claims in the care site table which is a limitation. Because of this we have created an extension table to the cdm called cdm.other_organization which will capture all other organization NPIs that are not in the cdm.care_site table.

First we find all NPIs with entity type = ‘ORGANIZATION’ and occur on at least 5 facility claims in either the billing or servicing NPI fields in the stg.mc table.

destination field	source field	mapping rule
care_site_id INTEGER	stg.npi.npi	Convert to integer
care_site_name VARCHAR(255)	stg.npi.orgname stg.npi.orgname2	WHEN ORGNAME2 IS NULL THEN ORGNAME ELSE CONCAT(ORGNAME, ': ', ORGNAME2)
place_of_service_concept_id INTEGER	stg.npi.primary_taxonomy_code	Map the primary taxonomy code to its standard concept. This is not strictly a place of service but seems like the most useful thing to do at the care site level.
location_id INTEGER	stg.npi.prvzip	zip code is mapped to cdm.location.zip and the corresponding location id is entered as cdm.care_site.location_id
care_site_source_value VARCHAR(50)	stg.npi.npi	NPI is copied verbatim
place_of_service_source_value VARCHAR(50)	stg.npi.primary_taxonomy_code	The primary taxonomy code is copied verbatim

cdm.other_organization

Dependencies: cdm.location

This table is an add-on that is not part of the CDM specification. It is needed to address the problem of handling billing organizations and physician groups. All organizational NPI that occur in the source data and are not mapped into the cdm.care_site table are mapped into this table.

destination field	source field	mapping rule
organization_id INTEGER	stg.npi.npi	Convert to integer
organization_name VARCHAR(255)	stg.npi.orgname stg.npi.orgname2	WHEN ORGNAME2 IS NULL THEN ORGNAME ELSE

		CONCAT(ORGNAME, ': ', ORGNAME2)
place_of_service_concept_id VARCHAR(255)	stg.npi.primary_taxonomy_code	Map the primary taxonomy code to its standard concept.
location_id INTEGER	stg.npi.prvzip	zip code is mapped to cdm.location.zip and the corresponding location id is entered as cdm.care_site.location_id. This could be a billing location.
organization_source_value VARCHAR(50)	stg.npi.npi	NPI is copied verbatim
place_of_service_source_val ue VARCHAR(50)	stg.npi.primary_taxonomy_code	The primary taxonomy code is copied verbatim

cdm.provider

Dependencies: cdm.care_site, cdm.other_organization

The cdm.provider table contains individual providers whose NPI occurs on at least one claim in stg.mc. Providers are identified by NPI and mapped essentially directly from the stg.npi table.

destination field	**source field**	**mapping rule**
provider_id INTEGER	stg.npi.npi	Convert to integer
provider_name VARCHAR(255)	stg.npi.prvname	Copied verbatim from source
npi VARCHAR(20)	stg.npi.npi	Copied verbatim from source
dea VARCHAR(20)	NULL	All records populated with NULL
specialty_concept_id INTEGER	stg.npi.primary_taxonomy_code	Map the primary taxonomy code to its standard concept.

care_site_id INTEGER	stg.mc.npi stg.mc.servicing_npi	For every record in stg.mc identify a care_site NPI and an inidividual provider NPI from the two NPI source filelds. Only consider records of stg.mc that have both a care site NPI (defined above) and an individual NPI. For every care_site-inividual provider combination, add up the number of claim lines and the total paid amount using stg.mc.tpay. For each individual provider, find the provider-care_site combination with the highest paid amount. Break ties using the number of claim lines. Convert the care_site's NPI to an integer and enter it as that provider's care_site_id.
year_of_birth INTEGER	NULL	All records populated with NULL
gender_concept_id INTEGER	0	All records populated with 0
provider_source_value VARCHAR(50)	stg.npi.npi	Copied verbatim from source
specialty_source_concept_id INTEGER	stg.npi.primary_taxonomy_code	Copied verbatim from source
gender_source_value VARCHAR(50)	NULL	All records populated with NULL
gender_source_concept_id INTEGER	0	All records populated with 0
organization_id (add-on field) INTEGER	stg.mc.npi stg.mc.servicing_npi	Use the exact same logic as care_site_id but substitute other_organization_id for care_site_id. This field might capture billing group affiliations.

Creation of Person/observation_period tables

cdm.observation_period

Dependencies: stg.me

The stg.me table contains one row per patient-month combination. It only contains rows for months when patients had at least one form of insurance. The logic used to collapse this format to one row per period of continuous enrollment (aka observation) is as follows.

1. Choose a reference month. This could be anything but we chose January 1960. Create a new variable for each month record called numeric_month that maps the month of the eligibility record to an integer that is the number of months since the reference month. (e.g. Jan 1960 -> 0, Feb 1960 -> 1, March 1960 -> 3, etc)
2. Order the data by patient and the new numeric_month variable. Add a new variable called new_period_flag that has the value 1 if the record represents a new eligibility period and 0 if it is part of the same period of continuous eligibility as the previous record. We know that a record is the start of a new period if the previous record is for a different patient or is for the same patient but not the previous month.
3. Add a new variable called period_id that is the cumulative sum of the new_period_flag field. This period_id variable will give the same number to every row in a given patient's continuous period of eligibility.
4. Grouping by patient and period_id, calculate the min(numeric_month) as the period_start_month and max(numeric_month) as period_end_month. This should leave us with one row per continuous period of eligibility (observation).
5. Convert the period_start_month and period_end_month variables to actual dates. Use the first day of the month for the period_start_date and the last day of the month for the period end date.

destination field	source field	mapping rule
observation_period_id INTEGER	auto-generated integer	
person_id INTEGER	stg.me.patkey	Copied verbatim from source
observation_period_start_date DATE	derived from stg.me.month stg.me.year	See algorithm description above.

observation_period_end_date DATE	derived from stg.me.month stg.me.year	See algorithm description above.
period_type_concept_id INTEGER	44814722	Assign 44814722 to all records which is the standard concept for “period while enrolled in insurance”.

cdm.payer_plan_period

Not implemented yet.

destination field	source field	mapping rule

cdm.person

destination field	source field	mapping rule
person_id INTEGER	stg.me.patkey	
gender_concept_id INTEGER	stg.me.gender	
year_of_birth INTEGER	stg.me.age	
month_of_birth INTEGER		
day_of_birth INTEGER		
birth_datetime DATETIME		
death_datetime DATETIME		
race_concept_id	0	Populate all records with 0

INTEGER		since we do not have race information in this data
ethnicity_concept_id INTEGER	0	Populate all records with 0 since we do not have ethnicity information in this data
location_id INTEGER	stg.me.zip	Assign each patient to a zip code using the majority vote rule. For each patient choose the zip where they spent the most time as measured by their member eligibility records. In the case of ties chose one zip at random.
provider_id INTEGER		Assign a provider id to each patient using the algorithm described above. “Of the possible PCPs in Maine which one did this person see most often or most recently, if any?”
care_site_id INTEGER		Assign a care site to each patient using the algorithm described above. Of the care_sites in the cdm.care_site table, which one did this patient receive the most care at as measured by the paid amount on the claims?
person_source_value VARCHAR(50)	stg.me.patkey	Copy the patient key verbatim
gender_source_value VARCHAR(50)	stg.me.gender	Find the most commonly occurring non-missing source gender value for each patient that is either ‘M’ or ‘F’. Copy that value here as the gender source value
gender_source_concept_id INTEGER		Map the gender_source_value to its concept id. ‘M’ maps to id 00000 ‘F’ maps to id 00000

race_source_value VARCHAR(50)	NULL	Populate all records with NULL
race_source_concept_id INTEGER	0	Populate all records with 0
ethnicity_source_value VARCHAR(50)	NULL	Populate all records with NULL
ethnicity_source_concept_id INTEGER	0	Populate all records with 0

Creation of Visit/visit detail tables (batch process)

A visit is defined as ___.

trm.visit_detail

destination field	source field	mapping rule

cdm.visit_detail

destination field	source field	mapping rule

cdm.visit

dependencies: cdm.visit_detail

destination field	source field	mapping rule

Creation of event tables (batch process)

The event tables are condition_occurrence, procedure_occurrence, drug_exposure, observation, measurement,

trm.dx_map

destination field	source field	mapping rule

trm.cpt_map

destination field	source field	mapping rule

trm.ndc_map

destination field	source field	mapping rule

Creation of the cost table

cdm.cost

destination field	source field	mapping rule

Creation of the derived tables

Derived tables are

- cdm.drug_era
- cdm.condition_era

cdm.dose_era?

Era tables

ETL testing

Description of what was run: 100K pats
2015-2018
Mapping rate
12 hours to run
no testing done

Appendix