Maine APCD to OMOP ETL specification

Technical documentation for mapping the Maine All Payer Claims Data to the OMOP Common Data Model version 6

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.

stg.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.

primary_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

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

- Claim lines that are missing **mhdo_memberid** or **mhdo_claim** will be dropped
- PC025_STATUS must be one of ('1','01','19', '2','02','20', '21', '04', '4') indicating that we are including paid claims and denied claims but not reversals or claim lines where the status code is NULL

- Claims with negative cost amounts should be excluded as well as they are part of a reversal.

<full table specification is omitted>

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_value 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	stg.npi.npi	Copied verbatim from source

VARCHAR(20)		
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	auto-generated integer	

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

The payer_plan_period table contains one row per unique combination of payer type (Medicare, Medicaid, commercial), person, and continuous period of insurance coverage. We did not make any distinction between different commercial plans since we did not have access to this information. Our primary purpose for this table was to record eligibility type in the CDM. The logic used to create this table is the same as the logic used for the observation_period table. This table was implemented by partitioning the medical eligibility records into three subsets (Medicare, commercial, Medicaid), running the same logic on each subset, and inserting the union of the results into the payer_plan_period table. Pharmacy eligibility was not considered for this table.

destination field	**source field**	**mapping rule**
payer_plan_period_id INTEGER	auto-generated integer	
person_id INTEGER	stg.me.patkey	Copied verbatim from source
contract_person_id INTEGER	NULL	
payer_plan_period_start_date DATE	derived from stg.me.month stg.me.year	See algorithm description for cdm.observation_period.observation_period_start_date

payer_plan_period_end_date DATE	derived from stg.me.month stg.me.year	See algorithm description for cdm.observation_period.observation_period_end_date
payer_concept_id INTEGER	280 for Medicare eligibility 289 for Medicaid eligibility 418 for commercial eligibility	Record assigned to Medicare when stg.me.medicare_flag = 1, Medicaid when stg.me.medicaid_flag = 1 and commercial when stg.me.commercial_flag = 1
payer_source_value VARCHAR(50)	‘MEDICAID’, ‘MEDICARE’, or ‘COMMERCIAL’	Same as above
payer_source_concept_id INTEGER	280 for Medicare eligibility 289 for Medicaid eligibility 418 for commercial eligibiliTY	Same as above
plan_concept_id INTEGER	0	
plan_source_value VARCHAR(50)	NULL	
plan_source_concept_id INTEGER	0	
contract_concept_id INTEGER	0	
contract_source_value VARCHAR(50)	NULL	
contract_source_concept_id INTEGER	0	
sponsor_concept_id INTEGER	0	
sponsor_source_value VARCHAR(50)	NULL	
sponsor_source_concept_id INTEGER	0	
family_source_value VARCHAR(50)	NULL	

stop_reason_concept_id INTEGER	0	
stop_reason_source_value VARCHAR(50)	NULL	
stop_reason_source_concept_id INTEGER	0	

cdm.person

This table has one row per mhdo_memberid (patkey).

Algorithm for assignment of provider_id

- Create a list of all possible primary care physicians in Maine by subsetting the stg.npi table to records where stg.npi.primary_taxonomy_code is one of '207Q00000X', '207R00000X', '208D00000X', '207QG0300X', '207RG0300X', '207V00000X', '208000000X', and stg.npi.prvstate = 'ME' and stg.npi.enttype = 'INDIVIDUAL'
- Select medical claims where the servicing provider is one of the potential PCPs
- Flag those medical claims as
    - Office visit if the place of service code (called factype) is 11
    - A physical exam if there is a cpt code on the claim that is one of '99381', '99382', '99383', '99384', '99385', '99386', '99387', '99391', '99392', '99393', '99394', '99395', '99396', '99397'
- For each patient-provider combination calculate the number of visits, the most recent physical exam date, the most recent office visit date, the most recent visit date (any visit)
- For each patient choose the primary care provider who had the most visits breaking ties using the date of the last physical exam, the date of the last office visit, and the date of the last visit of any type in that order.

The above algorithm is meant to assign a primary care provider to each patient that is most relevant to retrospective analysis and not necessarily the patient's current primary provider.

destination field	**source field**	**mapping rule**
person_id INTEGER	stg.me.patkey	Copied verbatim from source
gender_concept_id INTEGER	stg.me.gender	WHEN GENDER = 'M' THEN 8507 WHEN GENDER = 'F' THEN 8532
year_of_birth	stg.me.birthyear	Copied verbatim from source

INTEGER		
month_of_birth INTEGER	NULL	
day_of_birth INTEGER	NULL	
birth_datetime DATETIME	NULL	
death_datetime DATETIME	NULL	
race_concept_id INTEGER	0	Populate all records with 0 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 choose 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	NULL	Not implemented in this ETL
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	stg.me.gender	Map the gender_source_value to its concept id. 'M' maps to id 8507 'F' maps to id 8532
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)

Intuitively a visit should represent a patient's encounter with the healthcare system from the patient's perspective. A visit should be uniquely identified by the combination of a patient, care site (hospital, clinic, etc) and date range. Constructing visits from claims is not all that simple to implement because of the variety of types of visits that are recorded and the ways in which those visits are represented in claims data.

trm.visit_detail

- A visit detail records are mapped 1:1 from claim lines
- Visit detail records roll up into visits
- Every visit detail must roll up into at least one visit occurrence
- A visit detail is a unique combination of first date, last date, provider, care site, and visit type
- We need to be able to tie procedures to a visit detail record
- Creation of trm.visit_detail is done as a batch process for performance reasons

destination field	source field	mapping rule
VISIT_DETAIL_ID BIGINT	stg.mc.idn (claim line identifier)	Copied verbatim
VISIT_OCCURRENCE_ID BIGINT	generated using stg.mc.person_id stg.mc.fdate stg.mc.ldate stg.mc.visit_detail_concept_id stg.mc.care_site_id/organization_id/provieder_id	This is a key field since it is defines which claim lines belong to which visits. The logic to create this field is difficult to describe succinctly. First a key is formed COALESCE(VISIT_DETAIL_CONCEPT_ID, CARE_SITE_ID, PROVIDER_ID, ORGANIZATION_ID, 0) AS KEY_, Within person_id and key_ we find the max end date. From there we identify new visits by making sure they do not overlap with prior visits. Apologies for the poor explanation. Please see the SQL code in trm_visit_detail.sql for an explicit implementation of the algorithm.
PERSON_ID BIGINT	stg.mc.patkey	Copied verbatim
MHDO_CLAIM BIGINT	stg.mc.mhdo_claim	Copied verbatim
VISIT_DETAIL_SOURCE_VALUE VARCHAR(20)	stg.mc.billtype (UB 04) stg.mc.factype (CMS 1500)	Copied verbatim based on claim form type.
VISIT_DETAIL_SOURCE_VOCABULARY VARCHAR(20)	NA	Set manually based on claim form (UB04 or CMS 1500) Set to 'UB04 Typ bill' for facility claims and set to

		'CMS Place of Service' for professional claims
VISIT_DETAIL_TYPE_CONCEPT_ID INTEGER	NA	Set manually to 32023 for UB04-facility claims and set to 32024 for CMS1500 professional claims.
FDATE DATETIME	stg.mc.fdate stg.mc.admdat	CAST(COALESCE(ADMDAT, FDATE) AS DATETIME) AS FDATE
LDATE DATETIME	stg.ldate stg.mc.disdat	CAST(COALESCE(DISDAT, LDATE) AS DATETIME) AS LDATE
NPI BIGINT	stg.mc.npi	Cast to big integer type
PRVLNAME VARCHAR(120)	stg.mc.prvlname	copied verbatim
SERVICING_NPI BIGINT	stc.mc.servicing_npi	copied verbatim
PRVLNAME_NPI BIGINT	stg.mc.prvlname stg.npi.npi	Sometimes prvlname is a useful field for identifying the care_site or organization associated with a visit. Each prvlname character string in the source data that contains at least two words and is at least 5 characters long is associated with the billing npi on the same claim. When more than one combination of prvlaname and billing npi exist in the source data the combination that occurs the most is used.
ADMITTED_FROM_SOURCE_VALUE VARCHAR(10)	stg.mc.admsr	copied verbatim
DISCHARGE_TO_SOURCE_VALUE	stg.mc.ptdis	copied verbatim

VARCHAR(10)		
ATT_NPI BIGINT	stg.mc.att_npi	Cast to int
OP_NPI BIGINT	stg.mc.op_npi	cast to int
ALLOWED_AMT MONEY	stg.mc.tpay stg.mc.prepaid stg.mc.copay stg.mc.prepaid stg.mc.ded	Sum of amounts paid for services by all parties: TPAY + PREPAID + COINS + COPAY + DED AS ALLOWED_AMT
VISIT_DETAIL_SOURCE_CONCEPT_ID INT	stg.mc.billtype (UB 04) stg.mc.factype (CMS 1500)	The concept id for whichever source value is not NULL.
VISIT_DETAIL_SOURCE_CONCEPT_NAME VARCHAR(255)	stg.mc.billtype (UB 04) stg.mc.factype (CMS 1500)	The concept name for whichever source value is not NULL.
VISIT_DETAIL_CONCEPT_ID INT	stg.mc.billtype (UB 04) stg.mc.factype (CMS 1500)	The standard concept id for whichever source value is not NULL.
VISIT_DETAIL_CONCEPT_NAME VARCHAR(255)	stg.mc.billtype (UB 04) stg.mc.factype (CMS 1500)	The standard concept name for whichever source value is not NULL.
ADMITTED_FROM_SOURCE_CONCEPT_ID INT	stg.mc.admsr	The concept id associated with the source value.
ADMITTED_FROM_SOURCE_CONCEPT_NAME VARCHAR(255)	stg.mc.admsr	The concept id associated with the source field.
ADMITTED_FROM_CONCEPT_ID INT	stg.mc.admsr	The standard concept id associated with the source field.
ADMITTED_FROM_CONCEPT_NAME VARCHAR(255)	stg.mc.admsr	The standard concept name associated with the source field.
DISCHARGE_TO_SOURCE_CONCEPT_ID INT	stg.mc.ptdis	The concept id associated with the source value.

DISCHARGE_TO_SOURCE_CONCEPT_NAME VARCHAR(255)	stg.mc.ptdis	The concept id associated with the source field.
DISCHARGE_TO_CONCEPT_ID INT	stg.mc.ptdis	The standard concept id associated with the source field.
DISCHARGE_TO_CONCEPT_NAME VARCHAR(255)	stg.mc.ptdis	The standard concept name associated with the source field.
CARE_SITE_ID BIGINT	stg.mc.npi stg.mc.servicing_npi stg.mc.prvlname	The first source value that appears in cdm.care_site.care_site_id is recorded as the care_site_id for each visit detail record.
PROVIDER_ID BIGINT	stg.mc.op_npi stg.mc.att_npi stg.mc.servicing_npi stg.mc.npi	The first source value that appears in cdm.provider.provider_id is recorded as the provider_id for each visit detail record.
ORGANIZATION_ID BIGINT	stg.mc.npi stg.mc.servicing_npi stg.mc.prvlname	The first source value that appears in cdm.other_organization.organization_id is recorded as the organization_id for each visit detail record.

cdm.visit_detail

dependencies: trm.visit_detail

This table is simply copied from trm.visit_detail dropping columns that are not in the final cdm specification.

cdm.visit_occurrence

dependencies: trm.visit_detail

This is essentially a "roll up" where multiple visit detail records are collapsed into a single visit_occurrence record When there are multiple values for a particular field (provider_id for example) a single value will be selected based on the total dollar amount on the claims. The value associated with the largest dollar amount in the visit detail rows is the value that is assigned to the visit. NULL values are only propagated from visit detail records to visit occurrence records if no other value is present on any visit detail records. A non-missing value will always supersede a missing value.

Columns to roll up from the visit detail level to the visit level are

- visit start date
- visit end date
- provider_id
- care_site_id
- addmitted_from and discharged_to (always taken together from same visit detail row)
- organization_id

destination field	**source field**	**mapping rule**
visit_occurrence_id BIGINT	trm.visit_detail.visit_occurrence_id	copied verbatim
person_id BIGINT	trm.visit_detail.person_id	copied verbatim
visit_concept_id INTEGER	trm.visit_detail.visit_detail_concept_id	visit_detail_concept_id should be the same on all detail rows within a visit_occurrence_id. Just use the visit_detail_concept_id as the visit_occurrence_concept_id (i.e. type of visit)
visit_start_date DATE	trm.visit_detail.visit_start_date	Use earliest start date on the associated visit detail lines for each visit_occurrence_id
visit_start_datetime DATETIME2	trm.visit_detail.visit_start_datetime	Use earliest start date on the associated visit detail lines for a visit and cast to

		datetime.
visit_end_date DATE	trm.visit_detail.visit_end_date	Use latest enddate on the associated visit detail lines for each visit_occurrence_id
visit_end_datetime DATETIME2	trm.visit_detail.visit_end_datetime	Use latest end date on the associated visit detail lines for each visit_occurrence_id and cast to datetime.
visit_type_concept_id INTEGER	trm.visit_detail.visit_detail_source_vocabulary	For each visit occurrence id use the source field to determine if data for the visit come from professional claims, facility claims or both. If both: assign concept id 32021 If facility only: assign concept id 32023 If professional only: assign concept id 32024
provider_id BIGINT	trm.visit_detail.provider_id trm.visit_detail.allowed_amt	Weight each provider_id in a visit using the allowed amount. Assign the provider with the largest total allowed amount to the visit.
care_site_id BIGINT	trm.visit_detail.care_site_id trm.visit_detail.allowed_amt	Weight each care_site_id in a visit using the allowed amount. Assign the care site with the largest total allowed amount to the visit.
visit_source_value VARCHAR(50)	trm.visit_detail.visit_source_value	Use the source value associated with cdm.visit_occurrence.visit_concept_id assigned above.
visit_source_concept_id INTEGER	trm.visit_detail.visit_source_concept_id	Use the source concept id associated with cdm.visit_occurrence.visit_concept_id assigned above.
admitted_from_concept_id INTEGER	trm.visit_detail.admitted_from_concept_id	Weight each admitted_from_concept_id

		in a visit using the allowed amount. Assign the provider with the largest total allowed amount to the visit.
admitted_from_source_value VARCHAR(50)	trm.visit_detail.admitted_from_source_value	Use the source value associated with the admitted_from_concept_id assigned above
discharge_to_concept_id INTEGER	trm.visit_detail.discharge_to_concept_id	Use the discharge_to_concept_id that occurs on the same visit detail row use to populate admitted_from_concept_id
discharge_to_source_value VARCHAR(50)	trm.visit_detail.discharge_to_source_value	Use the discharge_to_source_value that occurs on the same visit detail row use to populate admitted_from_concept_id
preceding_visit_occurrence_id BIGINT	trm.visit_detail.visit_occurrence_id	Use the LAG function to assign this value ordering by visit start date and partitioning by person_id
organization_id BIGINT	trm.visit_detail.organization_id trm.visit_detail.allowed_amt	Weight each organization_id in a visit using the allowed amount. Assign the organization with the largest total allowed amount to the visit.

Creation of event tables (batch process)

The event tables are

- condition_occurrence
- procedure_occurrence
- drug_exposure,
- observation
- measurement
- device_exposure

The OMOP vocabulary assigns source codes to the appropriate table using the domain_id.

The source data for these tables comes from

- stg.mc (cpt codes)
- stg.dx (icd diagnosis codes)
- stg.icdproc (icd procedure codes)
- stg.pc (ndc codes)

There is a many to many relationship between source and target tables since each souce table can feed into one or more target tables and each target table can receive data from more than one source table. To accommodate this we used the following data processing steps.

- Create event tables in the transformation schema as a set of intermediate tables to load the source data into.
- Iterating through the source tables:
    - create “mapping” helper tables used map source codes to their standard counterparts and
    - Load data from the source table into all of the intermediate transformation event tables.
    - Note that the vocabulary mapping determines which table (domain) the code is mapped to. (eg ICD codes with a condition domain in the vocabulary get mapped to the condition_occurence table while ICD codes with a domain_id = ‘Observation’ are mapped to the observation table.
- Finally load the completed transformation event tables into the final event tables in the cdm schema.

The helper tables that are used to map source concept codes to standard codes are not included in this ETL specification as they are simple to create and are derived directly from the OMOP vocabulary.

trm.condition_occurrence

condition_occurrence records come from

- stg.dx
- stg.icd_proc

destination field	source fields	mapping rule
condition_occurrence_id bigint	NA	Auto generated

person_id bigint	stg.dx.person_id stg.icd_proc.person_id	copied verbatim
condition_concept_id integer	stg.dx.dx stg.icd_proc.icd_proc	Map source field to standard concept using helper mapping table and joining on both the code and vocabulary.
condition_start_date date	stg.mc.fdate	Join source records to stg.mc on mhdo_claim. Use the earliest fdate value on the claim associated with the condition record being processed. (ie roll dates up to the claim level for the purpose of condition dates)
condition_start_datetime datetime2	stg.mc.fdate	Same as condition_start_date but converted to datetime format. Dates are rolled up to claim level
condition_end_date date	stg.mc.ldate	Join source table to stg.mc on mhdo_claim. Use the latest ldate value on the claim associated with this condition record.. (ie roll dates up to the claim level for the purpose of condition dates)
condition_end_datetime datetime2	stg.mc.ldate	Same as condition_end_date but converted to datetime format.
condition_type_concept_id integer	stg.dx.primary_flag stg.icd_proc.primary_flag	IF DX.PRIMARY_FLAG = 1 THEN 44786627 ELSE 0 Indicates whether or not condition was a primary diagnosis.
condition_status_concept_id integer	0	Set to 0 since we do not have this information
stop_reason varchar(20)	NULL	We do not have this information in the source data
provider_id	stg.mc.att_npi	Join stg.dx or stg.icd_proc

bigint	stg.mc.servicing_npi stg.mc.npi	record to the stg.mc table on mhdo_claim. Each provider variable is rolled up the the claim header level using the paid amount (tpay) to choose one provider on each claim. Next provider NPIs that are not in the cdm.provider table are ignored. Finally we coalesce(att_npi, servicing_npi, npi) and save the result as provider npi for the condition occurrence record.
visit_occurrence_id bigint	trm.visit_detail.visit_occurrence_id	Each source record is associated with a claim number and each claim number rolls up into one visit. This crosswalk from mhdo_claim to visit_occurrence_id is stored in the trm.visit_occurrence table. We assign the visit_occurrence_id associated with the claim number associated with the source record.
visit_detail_id bigint	NULL	Set to NULL for all records.
condition_source_value varchar(50)	stg.dx.dx stg.icd_proc.icd_proc	Copy the source value verbatim
condition_source_concept_id integer	stg.dx.dx stg.icd_proc.icd_proc	The concept is associated with source value.
condition_status_source_value varchar(50)	stg.dx.primary_flag stg.icd_proc.primary_flag	Copied verbatim and cast to varchar
source varchar(20)	NA	Manually assigned based on the source table for the condition_occurrence record. Possible values are “stg.dx” “stg.icd_proc”

trm.drug_exposure

drug_exposure records records come from

- stg.mc
- stg.pc

destination field	source field	mapping rule
drug_exposure_id bigint	NA	autogenerated
person_id bigint	stg.mc.patkey stg.pc.patkey	Copied verbatim
drug_concept_id integer	stg.mc.ndc stg.mc.cpt stg.pc.ndc	NDC and CPT codes are mapped to their standard concept representations. While most CPT codes map to the procedure domain some map to the drug domain. The target domain for each code is stored in the OMOP vocabulary.
drug_exposure_start_date date	stg.mc.fdate stg.pc.fdate	Dates are rolled up to the claim level. The earliest fdate on the claim with the NDC or drug CPT code is used as the drug_exposure_start_date.
drug_exposure_start_datetime datetime2	stg.mc.fdate stg.pc.fdate	Same as trm.drug_exposure.drug_exposure_start_date but cast to datetime
drug_exposure_end_date date	stg.mc.fdate stg.pc.fdate	When mapping CPT codes to drug_exposure from stg.mc.cpt we used the same day as drug_exposure_start_date. When mapping NDC codes from either stg.mc or stg.pc we used one day after the drug exposure_start_date.

		Basically we treat drug exposures as one day in length and use drug eras to capture continuous periods of time when a person was exposed to a drug ingredient.
drug_exposure_end_date datetime2	stg.mc.fdate stg.pc.fdate	Same as trm.drug_exposure.drug_exposure_end_date but cast to datetime
verbatim_end_date date	NULL or stg.mc.fdate	NULL if mapping an NDC code from stg.mc or stg.pc. stg.mc.fdate if mapping a CPT code from stg.mc
drug_type_concept_id integer	0	Assigned to zero
stop_reason varchar(20)	NULL	assigned static value
refills integer	NULL	assigned static value
quantity float	NULL or stg.pc.qty	Set to null for records coming from stg.mc Otherwise copied verbatim from stg.pc.qty
days_supply integer	NULL	assigned static value
sig varchar(max)	NULL	assigned static value
route_concept_id integer	0	assigned static value
lot_number varchar(50)	NULL	assigned static value
provider_id bigint	stg.mc.att_npi stg.mc.servicing_npi stg.mc.op_npi stg.mc.npi	NPIs that are not in cdm.provider (individual providers) are ignored. The first non-missing source NPI

	stg.pc.prescribing_npi	from stg.mc is used for data coming from stg.mc. For data coming from stg.pc the prescribing_npi is always used.
visit_occurrence_id bigint	trm.visit_detail.visit_occurrence_id or NULL	Each drug_exposure record is associated with a claim number and each claim number is associated with a visit_occurrence_id. The crosswalk between claim_id and visit_id is in the trm_visit_detail table. We used the visit_id associated with the claim number on the source record. For data coming from stg.pc we assigned NULL since these records were not linked to visits.
visit_detail_id bigint	NULL	assigned static value
drug_source_value varchar(50)	stg.mc.ndc stg.mc.cpt stg.pc.ndc	copied verbatim from source
drug_source_concept_id integer	stg.mc.ndc stg.mc.cpt stg.pc.ndc	Use the concept id representing the source value
route_source_value varchar(50)	NULL	assigned static value
dose_unit_source_value varchar(50)	NULL	assigned static value
source varchar(20)		Assigned ‘stg.pc’ if source data comes from stg.pc. Assigned ‘stg.mc’ if source data comes from stg.mc.ndc or stg.mc.cpt

trm.measurement

Typically measurements are generally not recorded in claims data. These usually represent lab values. Still some source codes do map to the measurement domain.

measurement records come from

- stg.dx.dx
- stg.mc.cpt

destination field	source field	mapping rule
measurement_id bigint	NA	autogenerated
person_id bigint	stg.mc.patkey	copied verbatim (When mapping from stg.dx a join with stg.mc is required)
measurement_concept_id integer	stg.dx.dx stg.mc.cpt	Mapped to standard concept id. Only souce codes associated with the measurement domain are included in the trm.measurement table.
measurement_date date	stg.mc.fdate	The earliest fdate on the claim is used as the measurement date
measurement_datetime datetime2	stg.mc.fdate	Same as measurement_date but cast to datetime
measurement_time varchar(10)	NULL	assigned static value
measurement_type_concept_id integer	32466	Assigned static value. 32466 = INFERRED FROM CLAIM
operator_concept_id integer	NULL	
value_as_number float	NULL	
value_as_concept_id integer	NULL	

unit_concept_id integer	NULL	
range_low float	NULL	
range_high float	NULL	
provider_id bigint	stg.mc.att_npi stg.mc.servicing_npi stg.mc.npi	NPIs that are not in cdm.provider.provider_id are ignored. (Only individual providers are considered for this field.) The first non-missing value in the source fields is used. (ie coalesce)
visit_occurrence_id bigint	trm.visit_detail.visit_occurrence_id	The claim id associated with the source record is matched to a visit_occurrence_id using the trm.visit_detail table.
visit_detail_id bigint	NULL	
measurement_source_value varchar(50)	stg.dx.dx stg.mc.cpt	copied verbatim
measurement_source_concept_id integer	stg.dx.dx stg.mc.cpt	The possibly non-standard concept ID associated with the source code in in the OMOP vocabulary
unit_source_value varchar(50)	NULL	
value_source_value varchar(50)	NULL	
source varchar(20)	NA	Assigned a static value based on the table where the source data came from. Possible values are: ‘stg.dx’ ‘stg.mc’

trm.procedure_occurrence

Most procedures come from CPT or ICD procedure codes.

Source tables/fields for procedures are

- stg.mc.cpt
- stg.icd_proc.icd_proc
- stg.dx.dx

destination field	source field	mapping rule
procedure_occurrence_id bigint	NA	Autogenerated
person_id bigint	stg.mc.patkey	Copied verbatim. When data is mapped from stg.dx or stg.icd_proc a join with stg.mc is required.
procedure_concept_id integer		
procedure_date date		
procedure_datetime datetime2		
procedure_type_concept_id integer		
modifier_concept_id integer		
quantity integer		
provider_id bigint		
visit_occurrence_id bigint		
visit_detail_id bigint		

procedure_source_value varchar(50)		
procedure_source_concept_id integer		
modifier_source_value varchar(50)		
source varchar(20)		Assigned a static value based on the table where the source data came from. Possible values are: ‘stg.dx’ ‘stg.mc’

trm.observation

observation record come from

destination field	source field	mapping rule
observation_id bigint		
person_id bigint		
observation_concept_id integer		
observation_date date		
observation_datetime datetime2		
observation_type_concept_id integer		
value_as_number float		
value_as_string varchar(60)		

value_as_concept_id integer		
qualifier_concept_id integer		
unit_concept_id integer		
provider_id integer		
visit_occurrence_id bigint		
visit_detail_id bigint		
observation_source_value varchar(50)		
observation_source_concept_id integer		
unit_source_value varchar(50)		
qualifier_source_value varchar(50)		
observation_event_id bigint		
obs_event_field_concept_id integer		
value_as_datetime datetime2		
source varchar(20)		

trm.device_exposure

destination field	source field	mapping rule
device_exposure_id bigint identity(1, 1)		
person_id bigint		
device_concept_id		

integer		
device_exposure_start_date date		
device_exposure_start_datetime datetime2		
device_exposure_end_date date		
device_exposure_end_datetime datetime2		
device_type_concept_id integer		
unique_device_id varchar(50)		
quantity integer		
provider_id bigint		
visit_occurrence_id bigint		
visit_detail_id bigint		
device_source_value varchar(100)		
device_source_concept_id integer		
source varchar(20)		

cdm.condition_occurrence

Copied verbatim from trm.condition_occurrence dropping columns not in the cdm specification.

cdm.procedure_occurrence

Copied verbatim from trm.procedure_occurrence dropping columns not in the cdm specification.

cdm.drug_exposure

Copied verbatim from trm.exposure dropping columns not in the cdm specification.

cdm.observation
Copied verbatim from trm.observation dropping columns not in the cdm specification.

cdm.device_exposure
Copied verbatim from trm.device_exposure dropping columns not in the cdm specification.

cdm.measurement
Copied verbatim from trm.measurement dropping columns not in the cdm specification.

cdm.cost
This table was unfortunately not implemented in this ETL due to time and resource constraints.

Creation of the derived tables
Derived tables are

- cdm.drug_era
- cdm.condition_era
- cdm.dose_era

Standard SQL code is provided by the OHDSI community to implement the derived era tables.

ETL testing
Description of what was run: 100K pats
Maine APCD 2015-2018
12 hours to run
no testing done

Appendix

trm.tmp

destination field	source field	mapping rule