DAVICU-1 Database Mapping Document Source Data Mapping Approach to CDMV5.4.1

edenceHealth NV in collaboration with Dept. of Intensive Care (4131), Rigshospitalet

Benjamin Skov Kaas-Hansen, Davide Placido

2024-07-16

Table of contents

1 Introduction

This document describes how the RH4131 database is converted to the OMOP Common Data Model (CDM) version 5.4.1. It describes the definition of the ETL that will be used in the implementation.

RH4131 has requested support from edenceHealth to extract, transform and load historical data in the context of a first "pilot" project. This project will serve as a stepping stone to larger future projects that include building and maintaining a Danish national data warehouse to house real-world evidence. The pilot project of historical data contains ICU data from 10 hospitals. edenceHealth will co-develop an ETL for ICU data from 3 hospitals as a template for RH4131.

1.1 Document History

Version	Date	Changes
0.1	2023-12-14	Internal document used at the mapping workshop
0.2	2023-12-28	Updated with notes from mapping workshop
0.3	2024-01-12	Updated with notes from infrastructure workshop. And misc. additional details. Rename source tables to reflect final pre-processing.
0.4	2024-01-24	Finalise infrastructure section, clarification of remaining questions, technical instructions for the STEM table, removal of resolved threads
0.5	2024-01-31	Update of the person and death logic
0.6	2024-02-02	Final edits

Version	Date	Changes
1.0	2024-02-05	Signed-off structural mapping doc; instructions for v1.0 ETL
1.1	2024-07-14	Adaptations to reflect final ETL (work in progress)

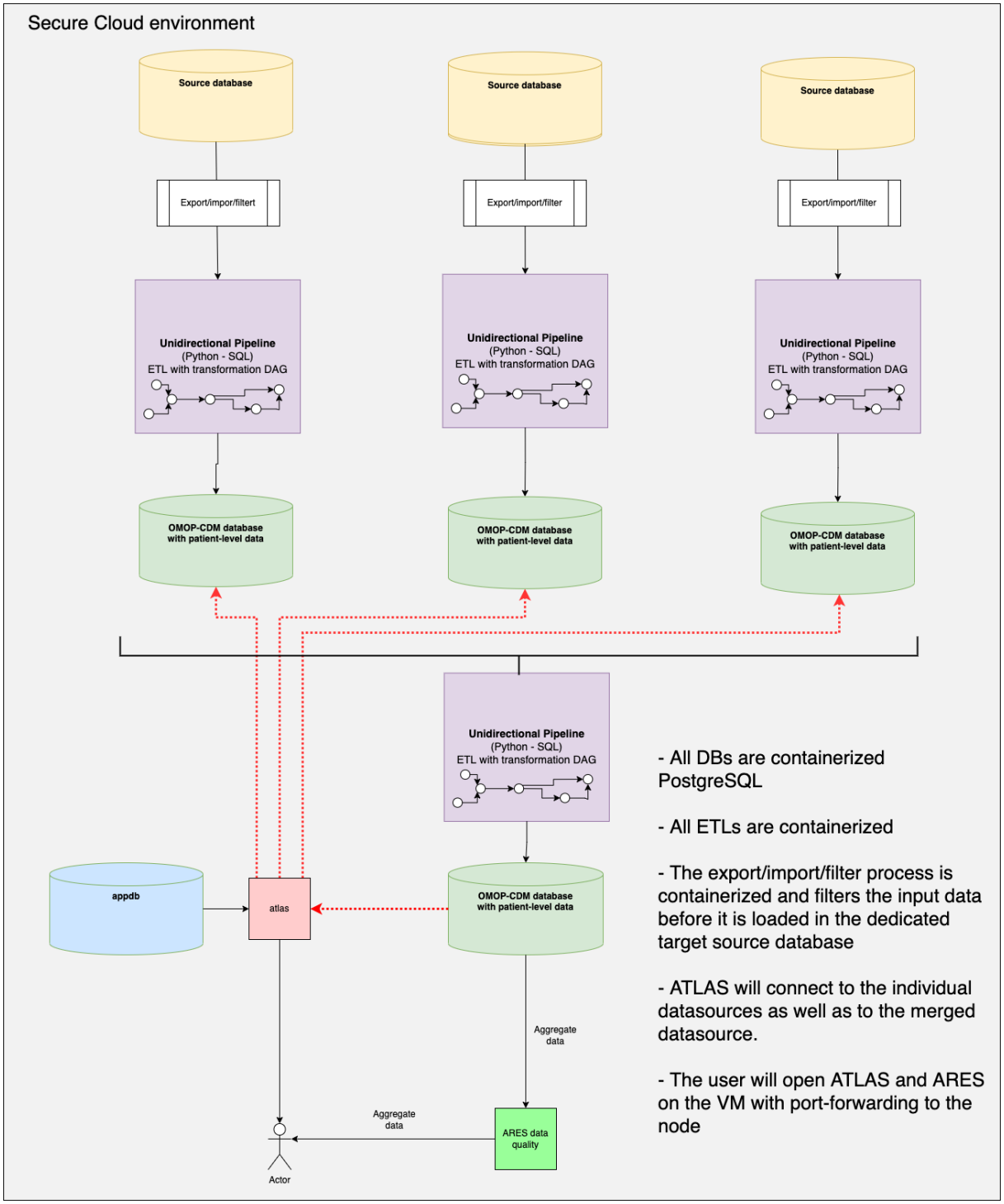
Part I

Background

2 Technical Infrastructure

The ETL will run on a secure HPC cloud hosted at the National Genome Center in Denmark and without internet access (although resources can be uploaded if packaged in a Singularity image file). The compute node available for this project has a 40-core 2.1GHz CPU, 192GB RAM, 1.9TB warm storage (XFS drive, NVMe SSD; approx. 1.5TB will be available).

All code will be containerised using Singularity to (i) conform with the HPC nature of the cloud that precludes root access, (ii) prevent version conflicts of software, and (iii) facilitate potential change of cloud system in the future. Singularity containers are brought onto the cloud through a semi-automated process that involves building a Dockerfile (with accompanying metadata such as python_requirements.txt). The SQL parts of the ETL will use PostgreSQL 16.x Non-SQL parts of the ETL will be written in Python 3.10.x or R >= v4.0; should the need arise, other languages can be used as well (e.g., Rust and Julia). All databases will be deployed as containers and the filtered source data will be loaded into the dedicated target source databases, see schematic. The database will use the SSD storage for filesystem.



The update frequency is expected to be quarterly/monthly, perhaps with intermittent need for more frequent runs in extreme cases such as pandemics. In those cases, more compute

resources would probably be available to compensate.

Part II

Mapping approach

3 Overview

The definition of the data mapping can be performed using the Rabbit-In-A-Hat tool that starts with a profile of the database made by the White Rabbit tool. RH4131 were unable to run WhiteRabbit on the cloud node. edenceHealth provided a format for the scan reports and RH4131 generated scan reports through tailored scripts based on the scan report format, so as to reverse-engineer conventional White Rabbit scan reports. In Appendix B, a data dictionary is presented for all the tables and fields that have been profiled.edenceHealth were then able to create White Rabbit-style scan reports based on the provided scan reports.

3.1 Source data

Rigshospitalet provided Parquet files for 2 of 3 sites (expectedly RH4131, Hvidovre, and Odense) containing ICU data. Each Parquet file contains data and/or information about one of the following five tables:

- prescriptions
- administrations
- diagnoses_procedures
- observations (actually contained in multiple files, named `observations-*.parquet`)
- course_metadata[1]
- t_person[2]

Initially, RH4131 provided three data-source scan report-like files for each site:

- database_scan: contains the number of rows for each table
- table_scan: includes information about the columns contained within each table, including data type, uniqueness, missing, etc.
- field_scan: contains the data for each column within each table

[1] Visits are called courses in the source data, from the Danish term *forløb*

[2] From the Danish Civil Registration System and holds data such as date of birth and sex ("CPR-Registeret - Sundhedsdatastyrelsen," n.d.))

Therefore, there is a slight nested quality to the data files. `database_scan` and `table_scan` contain information usually seen in the first two sheets of a scan report. `field_scan` contains the data usually seen in the following sheets (one per table) of a field scan; however, here it's all contained in one table, `field_scan`.

In addition, RH4131 has provided the following three files:

- `shak_lookup.tsv`: tab-separated file with SHAK codes and care-site metadata such as postal code and official name. This will be used during the ETL.
- `drug_mapping_helper.tsv`: tab-separated file with prescription data (including ATC, dose, dose unit, route, drug names) to be used before the ETL to populate the STEM table.
- `course_id_cpr_mapping.txt`: tab-separated file with three columns:
  - courseid: the visit identifier
  - timestamp: irrelevant for the purpose of the ETL
  - cpr_enc: the encrypted personal identifier

The exact columns included in each file are listed in Appendix B.

4 Conventional mapping

A "traditional" mapping including manual mappings for the following tables: PERSON, DEATH, VISIT_OCCURENCE, VISIT_DETAIL, LOCATION, CARE_SITE and CDM_SOURCE.

- These tables will be mapped directly from the source data using manual mappings.
- Some tables will require an additional environment variable and/or look-up table
  - VISIT_OCCURENCE, LOCATION, and CARE_SITE = SHAK code look-up table
  - CDM_SOURCE = environment variables

5 Stem-table mapping

The purpose of using an intermediate stem table between the source data and the clinical OMOP tables is to serve as an efficient, data-driven routing mechanism that allows tailored processing of the source data, and allows different source tables from data that will, eventually, land in the same clinical table. The table-level mechanism is illustrated in figure XXX.

The stem table relies on two key tables to do the routing: concept_lookup and concept_lookup_stem. These auxiliary tables are designed so as to enable using the same ETL pipeline for multiple sources (e.g., different ICUs from different hospitals).

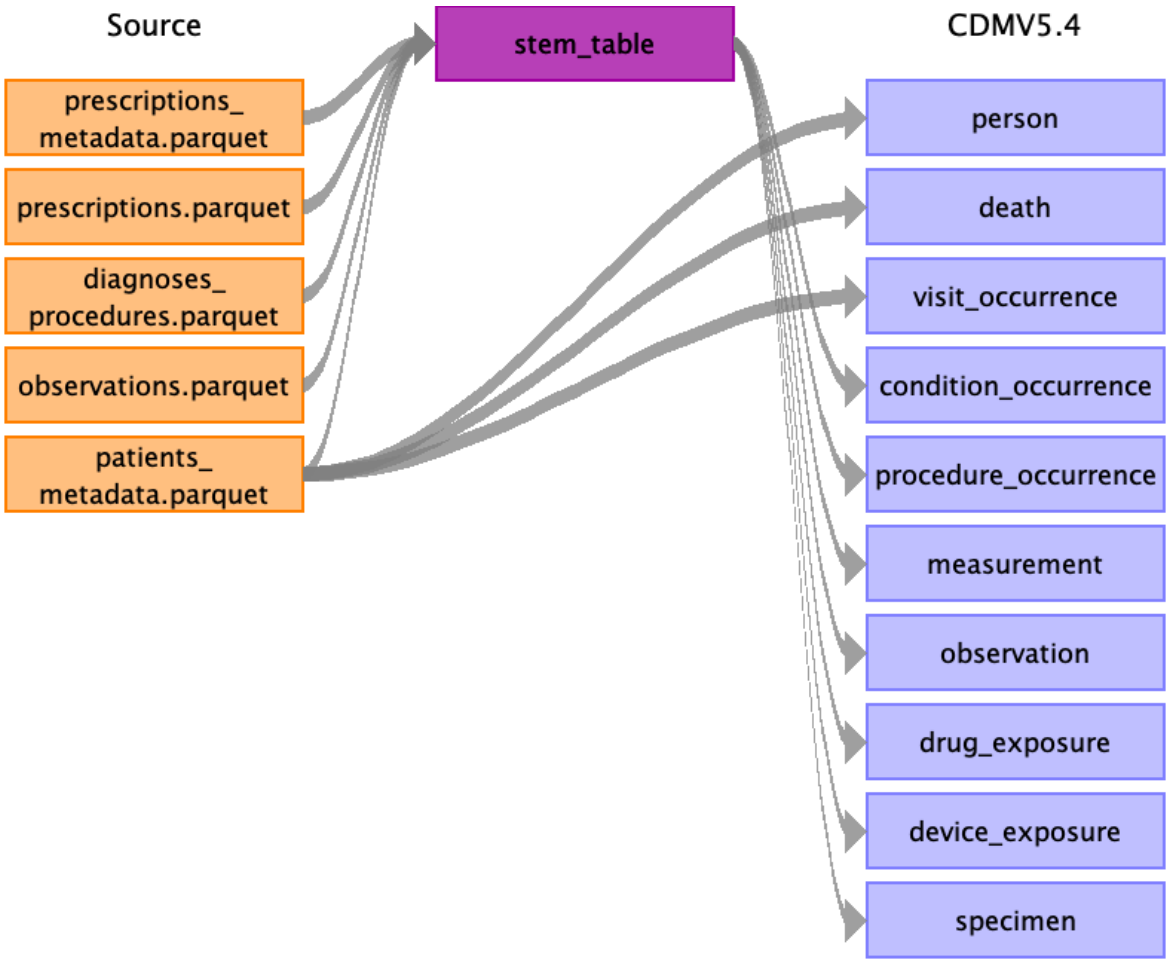
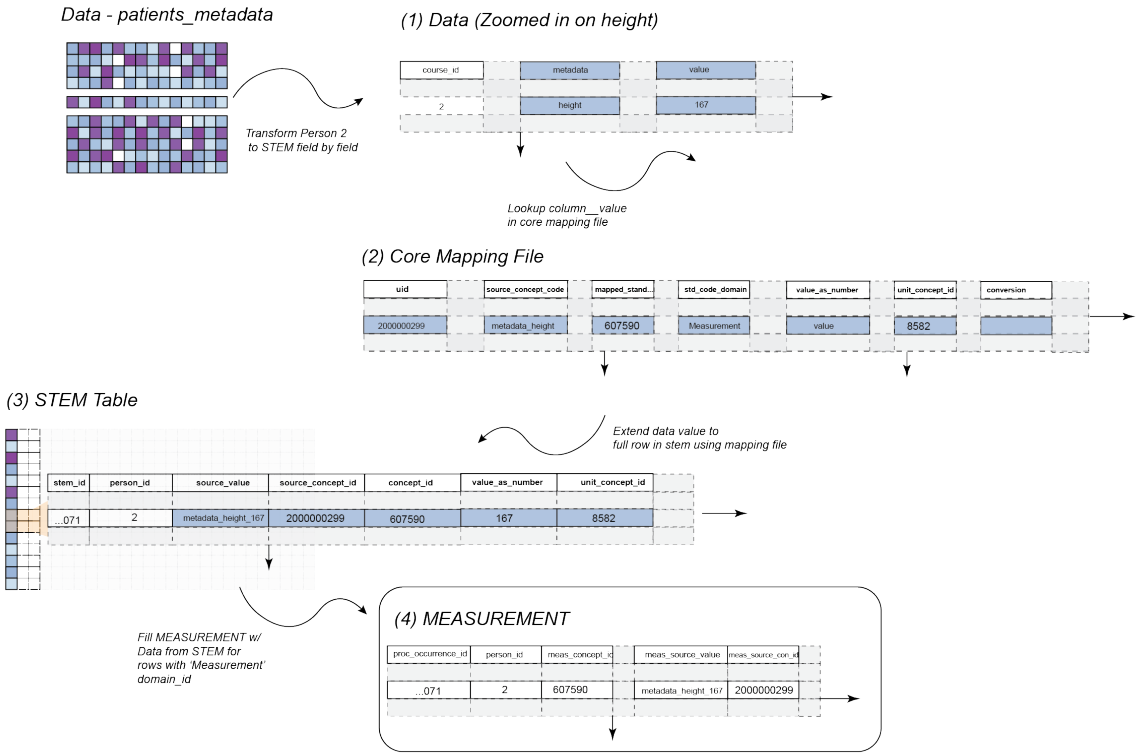


Table-level routing mechanism of the stem table

We, essentially, deploy two types of source->stem mechanisms: one tailored for drug data and a simple one (for the rest).



{align="center", background="white"}

5.0.1 Reading from `prescriptions` and `administrations`

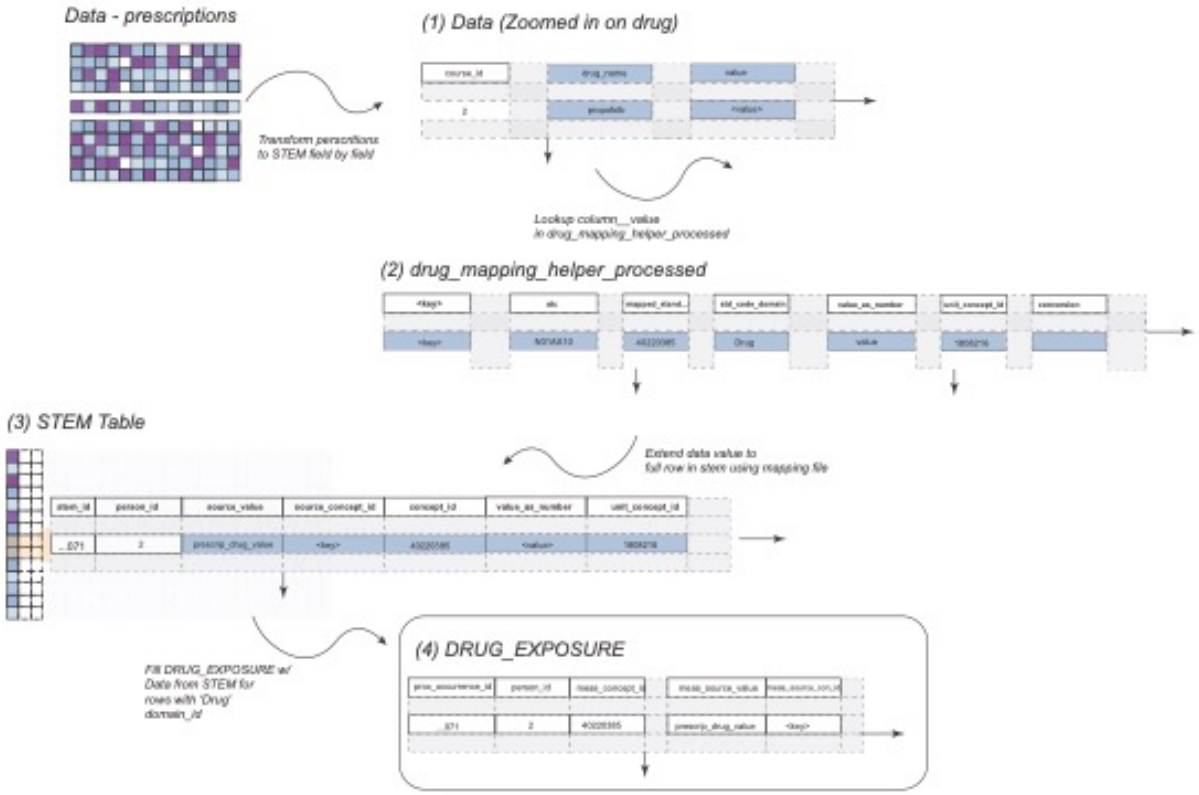
Because DRUG_EXPOSURE records contain data that are stored in two separate source tables, `prescriptions` and `administrations`, a more elaborate logic is required to build valid records.

NOTE: This STEM table refers only to drugs and will use the STEM table logic with the addition of manual mappings to ensure relationships between the two tables can easily be referenced.

For clinical tables containing drug information, we will use the STEM table; however, the logic will differ from that described above for other clinical tables. Manual mappings will be included to ensure relationships between the source prescription table and drug look-up table are easily referenced.

- The STEM table will be used as an intermediate mapping table: the source data will first be mapped to the STEM table and the OMOP CDM tables will be populated with records from the STEM table.

- Manual mapping will be needed to link the source drug data (contained in prescriptions and administrations) to a drug look-up table (called drug_mapping_helper) and then to the STEM table.
- drug_mapping_helper only contains the ATC codes and their associated additional information. RxNorm builder will be used to create mappings to best level of granularity possible, aiming for Clinical Drug (or Clinical Drug Component) level to enable population of the drug_era and dose_era tables.
- The logic to link from the source data to drug_mapping_helper to the STEM table to the OMOP drug_exposure table will need to be done manually with a drug_key (will link prescriptions and drug_mapping_helper, epaspresbaseid will link administrations and prescriptions). The ETL will handle pump and non-pump drug data differently: drug_mapping_helper.numerator_value comes from administrations.value for non-pump drug data and from prescriptions.epaspresconc for pump drug data
- This slightly different STEM table logic will apply to the following table: DRUG_EXPOSURE.



A diagram of a drug exposure Description automatically generated

6 Merge ETL

Using the ETL generated based on this document, one target database per hospital will be created. To allow studies done on a combined data set, a 'merge ETL' would have to be written. The aim of this ETL is to combine the outputs of the different datasets into one big target database. For this, people would need to be deduplicated, as they could have person and death records from multiple hospitals which is in violation with OMOP conventions. A person and death entry should be unique. Once the person and death tables have been deduplicated and the person_ids were updated, this needs to be reflected in the other (clinical) tables. Lastly, when adding all events together there will be an overlap in the ids, these will have to be updated to ensure unique values are being used.

7 Vocabularies

In this section, an overview is provided of the vocabulary mapping step.

The following source code vocabularies are present in the database:

Vocabulary	Reference Link	Description	Data Domains
SKS	https://medinfo.dk/sks/		observation, procedure, condition_occurrence
ATC	https://www.whocc.no/atc/structure_and_principles/		drug_exposure
NPU	https://npu-terminology.org/	The source data use NPU codes for biochemical analyses of biological samples (blood, plasma, urin, spinal fluid, etc.). Patient-level NPU data will, in the end, not be available for this pilot project, but we will undertake semantic mapping of the 200 most common NPU codes so these are ready for structural mapping once possible.	measurement
drug_mapping_helper	Overview of drugs with ATC code, dosage, unit, form, drug name(s). Provided by RH4131.	Will be used to create mappings to best level of granularity using the RxNormBuilder scripts *(based on OHDSI's "Boiler")*	drug_exposure

Vocabulary	Reference Link	Description	Data Domains
<shak_lookup>	Overview of SHAK codes, departments, locations, address		care_site, location, visit_occurrence

Part III

Mapped OMOP tables

8 Health System Data Tables

8.1 Table name: LOCATION

8.1.1 Reading from environment variable (SHAK_code) and SHAK code lookup file

Note: to avoid adding in locations that will not be used by the ETL, we will ask the user via an environment variable what the SHAK code associated with the dataset is.

Destination Field	Source Field	Logic	Comment
location_id	Hospital_SHAK_code (SHAK code look up file)	Autogenerate integer	Use the SHAK_code environment variable to do the lookup in the SHAK code lookup file against the SHAK_code field. Only create one distinct record for the returned entry.
address_1			NULL
address_2			NULL
city			NULL
state			NULL
zip	postal_code (SHAK code look up file)	Use postal code (4 digits)	
county			NULL
location_source_value	Hospital_SHAK_code (environment variable)	<column_name>|<column_value>	
country_concept_id		4330435 [Denmark]	
conntry_source_value			NULL
latitude			NULL
longitude			NULL

8.2 Table name: CARE_SITE

8.2.1 Reading from environment variable (SHAK_code) and SHAK code lookup file

Note: to avoid adding in locations that will not be used by the ETL, we will ask the user via an environment variable what the SHAK code associated with the dataset is.

Destination Field	Source Field	Logic	Comment
care_site_id	Autogenerate integer	Use the SHAK_code environment variable to do the lookup in the SHAK code lookup file against the SHAK_code field. Only create a record for the returned entry.	
care_site_name	department_name (SHAK code look up file)	use the department name from the SHAK lookup file	
place_of_service_concept_id	department_type (SHAK code look up file)	First join the SHAK_codes (env variable and the SHAK code look up file) then look up the department_type in the concept_look up file by joining on the source_code field and the department_type and filtering on 'care_site', use the corresponding standard_concept_id (lookup based on SHAK department code) E.g. 32037 [Intensive Care]	

Destination Field	Source Field	Logic	Comment
location_id	SHAK_code (environment variable)	Join in the location table using the location_source_value and the SHAK_code (<column_name>|<column_value> format) to find the corresponding location_id	
care_site_source_value	SHAK_code (environment variable)	<column_name>|<column_value>	SHAK code encodes hospital + department
place_of_service_source_value	department_type (SHAK code look up file)	<column_name>|<column_value>	

9 Clinical Data Tables

9.1 Table name: PERSON

9.1.1 Reading from T_PERSON

NOTE: course_id is the visit reference and unique within departments only. Raw course_id's are likely to recur across sites.

Destination Field	Source Field	Logic/remarks
person_id	cpr_enc	`hash(cpr_enc) // 2` The current implementation uses duckdb's hash function (implemented here; see also https://nullprogram.com/blog/2018/07/31). We halve because duckdb's `hash()` returns a uint64 (unsigned big integer) but we want a normal int64 to make the final CDM compatible with e.g. PostgreSQL which doesn't ship with uint64 support.
gender_concept_id	c_kon	'K' ~ 8532 'M' ~ 8507 else drop person
year_of_birth	d_foddato	Extract year
month_of_birth	d_foddato	Extract month
day_of_birth	d_foddato	Extract day
birth_datetime	d_foddato	Set time to 00:00:00
race_concept_id		Map to 0
ethnicity_concept_id		Map to 0
location_id		NULL

Destination Field	Source Field	Logic/remarks
provider_id		NULL
care_site_id		NULL
person_source_value	cpr_enc	`'cpr_enc|<cpr_enc'>`
gender_source_value	c_kon	`'c_kon|<c_kon>'`
gender_source_concept_id		NULL
race_source_value		NULL
race_source_concept_id		NULL
ethnicity_source_value		NULL
ethnicity_source_concept_id		NULL

9.2 Table name: DEATH

9.2.1 Reading from T_PERSON

Destination Field	Source Field	Logic/remarks
person_id	PERSON.person_id	Only for those patients in PERSON with c_status = 90 Look up the person_id in PERSON by matching the `'cpr_enc|<cpr_enc>'` with `PERSON.person_source_value`
death_date	d_status_hen_start	When `c_status == 90` [dead]. Format: YYYY-MM-DD
death_datetime		NULL
death_type_concept_id		32879 Registry
cause_concept_id		NULL
cause_source_value		NULL
cause_source_concept_id		NULL

9.3 Table name: VISIT_OCCURENCE

9.3.1 Reading from course_metadata, environment variable (SHAK_code) and SHAK code lookup file

COURSE_METADATA has a long format, effectively yielding a key-value data format so that a variable's name is looked up in the `variable` column and the corresponding `value` is used.

Destination Field	Source Field	Logic/remarks
visit_occurrence_id		`hash(<shak_code>|<courseid>)` See `PERSON.person_id` for details on hashing
person_id	PERSON.person_id	
visit_concept_id	SHAK_LOOKUP.department_type	Use environment variable `DEPARTMENT_SHAK_CODE` to find the department in `SHAK_LOOKUP`. Then, use `CONCEPT_LOOKUP.concept_id` where `CONCEPT_LOOKUP.concept_string == SHAK_LOOKUP.department_type` and `CONCEPT_LOOKUP.filter == 'care_site'`
visit_start_date	value	When `variable == 'admdate'` use corresponding value. Otherwise, use `admdatetime` cast to date.
visit_start_datetime	value	When `variable == 'admdatetime'` use corresponding value. Otherwise, use `admdate` with 00:00:00 suffix
visit_end_date	value	When `variable == 'dischdate'` use corresponding value. Otherwise, use `dischdtuse` cast to date

Destination Field	Source Field	Logic/remarks
visit_end_datetime	value	When `variable == 'dischdtuse'` use corresponding value. Otherwise, use `dischdate` with 00:00:00 suffix
visit_type_concept_id		32817 EHR
provider_id		NULL
care_site_id	CARE_SITE.care_site_id	Join with CARE_SITE on `CARE_SITE.care_site_source_value == 'department_shak_code|<DEPARTMENT_SHAK_CODE>'`
visit_source_value	course_id	`'course_id|<course_id>'`
visit_source_concept_id		NULL
admitted_from_concept_id		If `key == transfromid` and `value IS NOT NULL`, then look up `value` in CONCEPT_LOOKUP. The source data contain the following values: • "Non-ICU dept. - this hospital" • "Other ICU – other hospital" • "Non-ICU dept. - other hospital" • " " (blank) • "Other ICU – this hospital" • "Emergency room"
admitted_from_source_value		Use `'transfromid|<value>'`

Destination Field	Source Field	Logic/remarks
discharged_to_concept_id		If `key` == 'chkouttoid', look up `value` in CONCEPT_LOOKUP. The source data contain the following values: • "Non-ICU dept. - this hospital" • "Other ICU – other hospital" • "Non-ICU dept. - other hospital" • " " (blank) • "Other ICU – this hospital" • "Home"
discharged_to_source_value		Use `'chkouttoid|<value>'`
preceding_visit_occurrence_id		NULL

9.4 Table name: VISIT_DETAIL

Not in scope.

9.5 Table name: STEM

Most of the columns here come from CONCEPT_LOOKUP_STEM. When they do not, the origin table is denoted as prefix in the Source Field.

Destination Field	Source Field	Logic/remarks
domain_id	std_code_domain	
datasource		Appropriate idenfier of the provenance of the data (e.g. file name)
stem_id		Auto-generated integer

Destination Field	Source Field	Logic/remarks
person_id	PERSON.course_id	`PERSON.person_source_value == 'cpr_enc|<SOURCE_TABLE.cpr_enc>'`
concept_id	mapped_standard_code	Joining source data with CONCEPT_LOOKUP_STEM depends on the type of source data. For details, please refer to the actual implementation here. Generally, the idea is that for categorcial values, we join on `'<SOURCE_TABLE>.<variable>__<value>' == CONCEPT_LOOKUP_STEM.source_concept_code`, while we join on `'<SOURCE_TABLE>.<variable>' == CONCEPT_LOOKUP_STEM.source_concept_code` for numerical and free-text values. The standard concept id's of free-text values, then, are fetched from CONCEPT_LOOKUP. This way, free-text values serve as an extension of categorical values, when there are so many possible values that explicitly mapping them each via CONCEPT_LOOKUP_STEM would be too cumbersome.
start_date	<start_date>	Use column with the name defined in the source field, cast to DATE

Destination Field	Source Field	Logic/remarks
start_datetime	<start_date>	Use column with the name defined in the source field, cast to TIMESTAMP after converting to Europe/Copenhagen timezone
end_date	<end_date>	Use column with the name defined in the source field, cast to DATE
end_datetime	<end_date>	Use column with the name defined in the source field, cast to TIMESTAMP after converting to Europe/Copenhagen timezone
type_concept_id	type_concept_id	
provider_id		NULL
visit_occurrence_id	VISIT_OCCURRENCE.visit_occurrence_id	Join with VISIT OCCURRENCE on `VISIT_OCCURRENCE.visit_source_value == 'courseid|<SOURCE_TABLE>.courseid'`
visit_detail_id		NULL
care_site_id		NULL
source_value		The exact source_value depends on the provenance of the data, but generally consists of the variable name and the value separated by two underscores.

Destination Field	Source Field	Logic/remarks
source_concept_id		Will be `CONCEPT_LOOKUP_STEM.uid` for everything except drug administrations with an ATC code. For these, the `source_concept_id` will either be for the ATC concept (for drugs with custom mappings, in CONCEPT_LOOKUP_STEM) or for the ingredients (the rest, called *automapped*)
quantity_or_value_as_number		Used in all target tables that have a quantity of value_as_number column. Combined into one in the stem table for efficient routing, see **?@sec-stem-table-mapping**.
value_as_string	value_as_string	
value_as_concept_id	value_as_concept_id	Used for `value_type == 'categorical'` to encode the different levels of the categorial variable
unit_concept_id	unit_concept_id	
value_source_value	value	
unit_source_concept_id		NULL
unit_source_value	unit_source_value	
verbatim_end_date		NULL
days_supply		NULL
dose_unit_source_value	dose_unit_source_value	
modifier_concept_id	modifier_concept_id	
modifier_source_value		NULL
measurement_datetime		NULL
operator_concept_id	operator_concept_id	
range_low		Coalesce of the lower bound as per the source data, when available, and `CONCEPT_LOOKUP_STEM.range_low`.

Destination Field	Source Field	Logic/remarks
range_high		Coalesce of the upper bound as per the source data, when available, and `CONCEPT_LOOKUP_STEM.range_high`.
stop_reason		NULL
refills		NULL
sig		NULL
route_concept_id		Join with CONCEPT_LOOKUP on `CONCEPT_LOOKUP.concept_string == PRESCRIPTIONS.epaspresadmroute` and use `CONCEPT_LOOKUP.concept_id`
route_source_value	PRESCRIPTIONS.epaspresadmroute	
era_lookback_interval	era_lookback_interval	
lot_number		NULL
unique_device_id		NULL
production_id		NULL
anatomic_site_concept_id		NULL
disease_status_concept_id		NULL
specimen_source_id		NULL
anatomic_site_source_value		NULL
disease_status_source_value		NULL
condition_status_concept_id		NULL
condition_status_source_value		NULL
qualifier_concept_id		NULL
qualifier_source_value		NULL
event_id		NULL
event_field_concept_id		NULL
episode_id_source		NULL

9.6 Table name: CONDITION_OCCURRENCE

9.6.1 Reading from STEM (filtered on domain_id = 'Condition')

Destination Field	Source Field	Logic/remarks
condition_occurrence_id	id	

Destination Field	Source Field	Logic/remarks
person_id	person_id	
condition_concept_id	concept_id	If environment variable `INCLUDE_UNMAPPED_CODES == 'FALSE'` (default), we discard records whose concept_id's are 0 or NULL
condition_start_date		`coalesce(start_date, end_date)`
condition_start_datetime		`coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)`
condition_end_date		`coalesce(end_date, start_date)`
condition_end_datetime		`coalesce(end_datetime, end_date::timestamp, start_datetime, start_date::timestamp)`
condition_type_concept_id	type_concept_id	
stop_reason		NULL
provider_id		NULL
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
condition_source_value	source_value	
condition_source_concept_id	concept_id	
condition_status_source_value		NULL

9.7 Table name: PROCEDURE_OCCURRENCE

9.7.1 Reading from STEM (filtered on domain_id = 'Procedure')

Destination Field	Source Field	Logic/remarks
procedure_occurrence_id	id	
person_id	person_id	

Destination Field	Source Field	Logic/remarks
procedure_concept_id	concept_id	If environment variable INCLUDE_UNMAPPED_CODES == 'FALSE' (default), we discard records whose concept_id's are 0 or NULL
procedure_date		`coalesce(start_date, end_date)`
procedure_datetime		`coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)`
procedure_end_date		`coalesce(end_date, start_date)`
procedure_end_datetime		`coalesce(end_datetime, end_date::timestamp, start_datetime, start_date::timestamp)`
procedure_type_concept_id	type_concept_id	
modifier_concept_id	modifier_concept_id	
quantity	quantity	
provider_id		NULL
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
procedure_source_value	source_value	
procedure_source_concept_id	concept_id	
modifier_source_value	modifier_source_value	

9.8 Table name: DEVICE_EXPOSURE

9.8.1 Reading from STEM (filtered on domain_id = 'Device')

Destination Field	Source Field	Logic/remarks
device_exposure_id	uid	
person_id	person_id	
device_concept_id	concept_id	
device_exposure_start_date		`coalesce(start_date, end_date)`

Destination Field	Source Field	Logic/remarks
device_exposure_start_datetime		coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)
device_exposure_end_date		coalesce(end_date, start_date)
device_exposure_end_datetime		coalesce(end_datetime, end_date::timestamp, start_datetime, start_date::timestamp)
device_type_concept_id	type_concept_id	
unique_device_id		NULL
production_id		NULL
quantity		NULL
provider_id		NULL
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
device_source_value	source_value	
device_source_concept_id	source_concept_id	
unit_concept_id	unit_concept_id	
unit_source_value	unit_source_value	
unit_source_concept_id		NULL

9.9 Table name: MEASUREMENT

9.9.1 Reading from STEM (filtered on domain_id = 'Measurement')

Destination Field	Source Field	Logic/remarks
measurement_id	uid	
person_id	person_id	
measurement_concept_id	concept_id	
measurement_date		coalesce(start_date, end_date)
measurement_datetime		coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)
measurement_time		NULL

Destination Field	Source Field	Logic/remarks
measurement_type_concept_id	type_concept_id	
operator_concept_id	operator_concept_id	
value_as_number	quantity_or_value_as_number	
value_as_concept_id	value_as_concept_id	
unit_concept_id	unit_concept_id	
range_low	range_low	
range_high	range_high	
provider_id		NULL
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
measurement_source_value	source_value	
measurement_source_concept_id	source_concept_id	
unit_source_value	unit_source_value	
unit_source_concept_id	unit_source_concept_id	
value_source_value	value_source_value	
measurement_event_id	event_id	
meas_event_field_concept_id	event_field_concept_id	

9.10 Table name: SPECIMEN

9.10.1 Reading from STEM (filtered on domain_id = 'Specimen')

Destination Field	Source Field	Logic/remarks
person_id	person_id	
specimen_id	specimen_id	
specimen_concept_id	concept_id	
specimen_type_concept_id	type_concept_id	
specimen_date		`coalesce(start_date, end_date)`
specimen_datetime		`coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)`
quantity	quantity_or_value_as_number	
unit_concept_id	unit_concept_id	
anatomic_site_concept_id	anatomic_site_concept_id	
disease_status_concept_id	disease_status_concept_id	
specimen_source_id	source_concept_id	

Destination Field	Source Field	Logic/remarks
specimen_source_value	source_value	
unit_source_value	unit_source_value	
anatomic_site_source_value	anatomic_site_source_value	
disease_status_source_value	disease_status_source_value	

9.11 Table name: OBSERVATION

9.11.1 Reading from STEM (filtered on domain_id = 'Observation')

Destination Field	Source Field	Logic/remarks
observation_id	uid	
person_id	person_id	
observation_concept_id	concept_id	
observation_date		`coalesce(start_date, end_date)`
observation_datetime		`coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)`
observation_type_concept_id	type_concept_id	
value_as_number	quantity_or_value_as_number	
value_as_string	value_as_string	
value_as_concept_id	value_as_concept_id	
qualifier_concept_id		NULL
unit_concept_id	unit_concept_id	
provider_id	provider_id	
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
observation_source_value	source_value	
observation_source_concept_id	concept_id	
unit_source_value	unit_source_value	
qualifier_source_value	qualifier_source_value	
value_source_value	value_source_value	
observation_event_id	event_id	
obs_event_field_concept_id	event_field_concept_id	

9.12 Table name: DRUG_EXPOSURE

9.12.1 Reading from STEM (filtered on domain_id = 'Drug')

Destination Field	Source Field	Logic/remarks
drug_exposure_id	uid	
person_id	person_id	
drug_concept_id	concept_id	
drug_exposure_start_date		`coalesce(start_date, end_date)`
drug_exposure_start_datetime		`coalesce(start_datetime, start_date::timestamp, end_datetime, end_date::timestamp)`
drug_exposure_end_date		`coalesce(end_date, start_date)`
drug_exposure_end_datetime		`coalesce(end_datetime, end_date::timestamp, start_datetime, start_date::timestamp)`
verbatim_end_date		NULL
drug_type_concept_id	type_concept_id	
stop_reason		NULL
refills		
quantity	quantity_of_value_as_number	
days_supply		NULL
sig		NULL
route_concept_id	route_concept_id	
lot_number		NULL
provider_id		NULL
visit_occurrence_id	visit_occurrence_id	
visit_detail_id		NULL
drug_source_value	source_value	
drug_source_concept_id	source_concept_id	
route_source_value	route_source_value	
dose_unit_source_value		NULL

9.13 Table name: OBSERVATION_PERIOD

9.13.1 Reading from clinical tables (including visit_occ)

NOTE: min/max dates all established from dates across all filled in clinical tables

Destination Field	Source Field	Logic/remarks
observation_period_id		Auto-generated integer
person_id	person_id	
observation_period_start_date		MIN(EVENT [START] DATES)
observation_period_end_date		MAX(EVENT [END] DATES)
period_type_concept_id		32817 EHR or 32879 Registry

10 Standardised Derived Elements

10.1 T able name: DRUG_ERA

A Drug Era is defined as a span of time when the Person is assumed to be exposed to a particular active ingredient. A Drug Era is not the same as a Drug Exposure: Exposures are individual records corresponding to the source when Drug was delivered to the Person, while successive periods of Drug Exposures are combined under certain rules to produce continuous Drug Eras.

Generated as part of ETL process using standard OHDSI SQL script.

10.2 T able name: DOSE_ERA

A Dose Era is defined as a span of time when the Person is assumed to be exposed to a constant dose of a specific active ingredient.

Generated as part of ETL process using standard OHDSI SQL script.

10.3 T able name: CONDITION_ERA

A Condition Era is defined as a span of time when the Person is assumed to have a given condition. Condition Eras are chronological periods of Condition Occurrences.

Generated as part of ETL process using standard OHDSI SQL script.

11 Metadata Tables

11.1 Table Name: CDM_SOURCE

NOTE: Single-record table containing information about the site, source, and cdm.

Destination Field	Source Field	Logic	Comment
cdm_source_name		Add in an environment variable	
cdm_source_abbreviation		Add in an environment variable	
cdm_holder		Add in an environment variable	
source_description		Add in an environment variable	
source_documentation_reference			
cdm_etl_reference		Includes GitHub/GitLab tag, if provided	https://github.com/edencehealth/rh4131/r tag>
source_release_date		Add in an environment variable	Request date of last export at the start of ETL run
cdm_release_date		Date of ETL run	
cdm_version			'5.4.1'
cdm_version_concept_id			798878 [OMOP CDM Version 5.4.1]
vocabulary_version	vocabulary.vocabulary_version	where vocabulary_id='None'	

A OMOP CDM tables not included in mapping

The following tables were not included in the mapping as they were not relevant for the source data available. These still need to be created as part of the ETL run as they are needed for some of the OHDSI tooling to successfully complete.

- Clinical Data Tables: NOTE
- Clinical Data Tables: NOTE_NLP
- Clinical Data Tables: FACT RELATIONSHIP
- Health System Data Tables: PROVIDER
- Health Economics Data Tables: PAYER_PLAN_PERIOD
- Health Economics Data Tables: COST
- Standardised Derived Elements: EPISODE
- Standardised Derived Elements: EPISODE_EVENT
- Metadata Tables: METADATA

B Source tables

RH4131 has provided scan reports for 3 ICUs describing the ICU data. Initially, there were 3 data source files for each hospital: table_scan, database_scan, and field_scan. Each TSV file contains data and/or information about the following five tables: prescriptions, administrations, diagnoses_procedures, observations, and course_metadata.

table_scan and database_scan contain information usually seen in the first two sheets of a scan report. field_scan contains the data usually seen in the following sheets (one per table) of a field scan; however, here everything is contained in one table = field_scan.

The actual ETL development will be based on the RH4131 dataset (as well as Odense and one other site) with the assumptions that technically the ETL should be able to run on all sites if the data structure remains the same.

Table: prescriptions.parquet included in field_scan

Field	Type	Most freq. value	Comment
courseid	BIGINT		
timestamp	TIMESTAMP		
epaspresid	BIGINT		
epaspresbaseid	BIGINT		
epaspresstarttime	TIMESTAMP		
epaspresdose	DOUBLE		
epaspresdosemax	DOUBLE		
epaspresdosestart	DOUBLE		
epaspresdrugunit	VARCHAR		
epaspresdrugunitact	VARCHAR		
epaspresconc	DOUBLE		
epaspresfluids	VARCHAR		
epaspresmaxconc	DOUBLE		
epaspresmaxbag	BIGINT		
epasprescreatetime	TIMESTAMP		
epaspresdisolved	VARCHAR		
epaspresmixammount	DOUBLE		
epasprespn	VARCHAR		
epaspresinint	VARCHAR		
epaspresfreq	VARCHAR		

Field	Type	Most freq. value	Comment
epasprescreattype	VARCHAR		
epaspresgsubst	VARCHAR		
epasprespsubst	VARCHAR		
epaspresdosemaxdaily	DOUBLE		
epaspresdosemaxtotal	BIGINT		
epaspresscheduletype	VARCHAR		
epaspresdosemaxdailyunit	VARCHAR		
epaspresdosemaxtotalunit	VARCHAR		
epaspressecuritydose	DOUBLE		
epaspressecuritydoseunit	VARCHAR		
epaspressecuritydosemindays	BIGINT		
epaspresminadmtime	BIGINT		
epaspresprotname	VARCHAR		
epaspresprotname_vgh	VARCHAR		
epaspresprotkey	VARCHAR		
epaspresdrugname	VARCHAR		
epaspresadmmthd	VARCHAR		
epaspresdrugatc	VARCHAR		
epaspresindication	VARCHAR		
epaspresindictext	VARCHAR		
epaspresindicsks	VARCHAR		
epaspresdisctime	TIMESTAMP		
epaspresdiscreason	VARCHAR		
epaspresadmroute	VARCHAR		
epaspresgestage	BIGINT		
epaspresweight	BIGINT		
epaspresage	BIGINT		
epaspresbsa	BIGINT		
epasadmdoseunit	VARCHAR		
epasadmdose	DOUBLE		
epaspresinfusionmax	DOUBLE		

Table: administrations.parquet in field_scan

Field	Type	Most freq. value	Comment
courseid	BIGINT		
timestamp	TIMESTAMP		
epaspresbaseid	BIGINT		
drug_name	VARCHAR		
value	DOUBLE		

Field	Type	Most freq. value	Comment
from_file	VARCHAR		

Table: diagnoses_procedures.parquet in field_scan

Field	Type	Most freq. value	Comment
courseid	BIGINT		
timestamp	VARCHAR		
diag_proc	VARCHAR		
value	DOUBLE		
from_file	VARCHAR		

Table: observations.parquet in field_scan

Field	Type	Most freq. value	Comment
courseid	BIGINT		
timestamp	TIMESTAMP		
observation	VARCHAR		
value	DOUBLE		
from_file	VARCHAR		

Table: course_metadata.parquet in field_scan

Field	Type	Most freq. value	Comment
courseid	BIGINT		
timestamp	VARCHAR		
metadata	VARCHAR		
value	DOUBLE		
from_file	VARCHAR		

Table: drug_mapping_helper.tsv

Field	Type	Most freq. value	Comment
drug_key	VARCHAR		
atc	VARCHAR		
numerator_value	VARCHAR		
numerator_unit	VARCHAR		

Field	Type	Most freq. value	Comment
denominator_value	DOUBLE		
denominator_unit	VARCHAR		
actual_unit	VARCHAR		The actual unit prescribed (e.g. mg/min/kg.) In the enacted prescriptions, the per-kg. part goes out and the strength reflects this
route	VARCHAR		The original two-letter administration-route code
route_long	VARCHAR		Expanded administration route
n	INT		
drug_names	VARCHAR		

Table: shak_lookup.tsv

Field	Type	Most freq. value	Comment
department_shak_code	VARCHAR		
hospital_name	VARCHAR		
department_name	VARCHAR		
region	VARCHAR		Geographical region in Denmark (of which there are five)
include	INT		All 1's, not used for now
surgical	INT		Whether the department accepts surgical patients, 1 = yes, 0 = no
postal_code	INT		
hospital_shak_code	INT		

C DQD Results

This appendix lists the final DQD results and addresses the cases where the CDM failed the DQD checks and why we have allowed these to stay.

"CPR-Registeret - Sundhedsdatastyrelsen." n.d. https://sundhedsdatastyrelsen.dk/da/registre-og-services/om-de-nationale-sundhedsregistre/personoplysninger-og-sundhedsfaglig-beskaeftigelse/cpr-registeret.