Package 'CDMConnector'

January 24, 2023

Title Connect to an OMOP Common Data Model

Version 0.4.1

Description Provides tools for working with observational health data in the Observational Medical Outcomes Partnership (OMOP) Common Data Model format with a pipe friendly syntax.
Common data model database table references are stored in a single compound object along with metadata.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 4.0)

Imports dplyr, DBI (>= 0.3.0), checkmate, magrittr, dbplyr, pillar, cli, purrr, rlang, tidyselect, readr, glue, waldo, arrow, methods, withr, lifecycle, jsonlite, SqlRender

Suggests covr, knitr, rmarkdown, duckdb, RSQLite, RPostgres, odbc, ggplot2, testthat (>= 3.0.0), DatabaseConnector, lubridate

Enhances CirceR

Config/testthat/edition 3

VignetteBuilder knitr

Additional_repositories https://OHDSI.github.io/drat

NeedsCompilation no

Author Adam Black [aut, cre] (<https://orcid.org/0000-0001-5576-8701>)

Maintainer Adam Black <black@ohdsi.org>

Repository CRAN

Date/Publication 2023-01-24 08:30:19 UTC

R topics documented:

`addCohortTable` *Add a cohort table to a cdm object*

Description

This function creates an empty cohort table in a cdm and returns the cdm with the cohort table added.

Usage

```
addCohortTable(cdm, name = "cohort", overwrite = FALSE)
```

Arguments

cdm	A cdm reference created by CDMConnector. write_schema must be specified.
name	Name of the cohort table to be created.
overwrite	Should the cohort table be overwritten if it already exists? (TRUE or FALSE)

---

appendPermanent *Run a dplyr query and add the result set to an existing*

Description

Run a dplyr query and add the result set to an existing

Usage

```
appendPermanent(x, name, schema = NULL)
```

Arguments

x	A dplyr query
name	Name of the table to be appended. If it does not already exist it will be created.
schema	Schema where the table exists. Can be a length 1 or 2 vector. (e.g. schema = "my_schema", schema = c("my_schema", "dbo"))

Value

A dplyr reference to the newly created table

Examples

```
## Not run:
library(CDMConnector)
library(SqlUtilities)

con <- DBI::dbConnect(duckdb::duckdb(), dbdir = CDMConnector::eunomia_dir())
concept <- dplyr::tbl(con, "concept")

# create a table
rxnorm_count <- concept %>%
  dplyr::filter(domain_id == "Drug") %>%
  dplyr::mutate(isRxnorm = (vocabulary_id == "RxNorm")) %>%
  dplyr::count(domain_id, isRxnorm) %>%
  computePermanent("rxnorm_count")

# append to an existing table
rxnorm_count <- concept %>%
  dplyr::filter(domain_id == "Procedure") %>%
  dplyr::mutate(isRxnorm = (vocabulary_id == "RxNorm")) %>%
```

```
  dplyr::count(domain_id, isRxnorm) %>%
  appendPermanent("rxnorm_count")

DBI::dbDisconnect(con, shutdown = TRUE)


## End(Not run)
```

---

asDate *as.Date dbplyr translation wrapper*

Description

This is a workaround for using as.Date inside dplyr verbs against a database backend. This function should only be used inside dplyr verbs where the first argument is a database table reference. `asDate` must be unquoted with !! inside dplyr verbs (see example).

Usage

```
asDate(x)
```

Arguments

x	an R expression

Examples

```
## Not run:
con <- DBI::dbConnect(odbc::odbc(), "Oracle")
date_tbl <- dplyr::copy_to(con,
                           data.frame(y = 2000L, m = 10L, d = 10L),
                           name = "tmp",
                           temporary = TRUE)

df <- date_tbl %>%
  dplyr::mutate(date_from_parts = !!asDate(paste0(
    .data$y, "/",
    .data$m, "/",
    .data$d
  ))) %>%
  collect()

## End(Not run)
```

`assert_tables` *Assert that tables exist in a cdm object*

Description

A cdm object is a list of references to a subset of tables in the OMOP Common Data Model. If you write a function that accepts a cdm object as a parameter `assert_tables`/`assertTables` will help you check that the tables you need are in the cdm object, have the correct columns/fields, and (optionally) are not empty.

Usage

```
assert_tables(cdm, tables, empty.ok = FALSE, add = NULL)

assertTables(cdm, tables, empty.ok = FALSE, add = NULL)
```

Arguments

`cdm`	A cdm object
`tables`	A character vector of table names to check.
`empty.ok`	Should an empty table (0 rows) be considered an error? TRUE or FALSE (default)
`add`	An optional AssertCollection created by `checkmate::makeAssertCollection()` that errors should be added to.

Value

Invisibly returns the cdm object

Functions

- `assertTables()`: camelCase alias

Examples

```
## Not run:
# Use assertTables inside a function to check that tables exist
countDrugsByGender <- function(cdm) {
  assertTables(cdm, tables = c("person", "drug_era"), empty.ok = FALSE)

  cdm$person %>%
    dplyr::inner_join(cdm$drug_era, by = "person_id") %>%
    dplyr::count(.data$gender_concept_id, .data$drug_concept_id) %>%
    dplyr::collect()
}

library(CDMConnector)
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomia_dir())
```

```
cdm <- cdm_from_con(con)

countDrugsByGender(cdm)

DBI::dbDisconnect(con, shutdown = TRUE)


## End(Not run)
```

---

assert_write_schema *Assert that cdm has a writable schema*

Description

A cdm object can optionally contain a single schema in a database with write access. assert_write_schema checks that the cdm contains the "write_schema" attribute and tests that local dataframes can be written to tables in this schema.

Usage

```
assert_write_schema(cdm, add = NULL)

assertWriteSchema(cdm, add = NULL)
```

Arguments

cdm	A cdm object
add	An optional AssertCollection created by `checkmate::makeAssertCollection()` that errors should be added to.

Value

Invisibly returns the cdm object

Functions

- `assertWriteSchema()`: camelCase alias

`cdmFromCon` *Create a CDM reference object from a database connection*

Description

Create a CDM reference object from a database connection

Usage

```
cdmFromCon(
  con,
  cdmSchema = NULL,
  cdmTables = tbl_group("default"),
  writeSchema = NULL,
  cohortTables = NULL
)
```

Arguments

`con`	A DBI database connection to a database where an OMOP CDM v5.4 instance is located.
`cdmSchema`	The schema where the OMOP CDM tables are located. Defaults to NULL.
`cdmTables`	Which tables should be included? Supports a character vector, tidyselect selection helpers, or table groups. • tbl_group("all")all CDM tables • tbl_group("vocab")the CDM vocabulary tables • tbl_group("clinical")the clinical CDM tables
`writeSchema`	An optional schema in the CDM database that the user has write access to.
`cohortTables`	A character vector listing the cohort table names to be included in the CDM object. Cohort tables must be in the write_schema.

Value

A list of dplyr database table references pointing to CDM tables

`cdmFromFiles` *Create a CDM reference from a folder containing parquet, csv, or feather files*

Description

Create a CDM reference from a folder containing parquet, csv, or feather files

Usage

```
cdmFromFiles(
  path,
  cdmTables = deprecated(),
  format = "auto",
  asDataFrame = TRUE
)
```

Arguments

path	A folder where an OMOP CDM v5.4 instance is located.
cdmTables	deprecated
format	What is the file format to be read in? Must be "auto" (default), "parquet", "csv", "feather".
asDataFrame	TRUE (default) will read files into R as dataframes. FALSE will read files into R as Arrow Datasets.

Value

A list of dplyr database table references pointing to CDM tables

---

cdm_from_con *Create a CDM reference object from a database connection*

Description

Create a CDM reference object from a database connection

Usage

```
cdm_from_con(
  con,
  cdm_schema = NULL,
  cdm_tables = tbl_group("default"),
  write_schema = NULL,
  cohort_tables = NULL,
  cdm_version = "5.3"
)
```

Arguments

con	A DBI database connection to a database where an OMOP CDM v5.4 or v5.3 instance is located.
cdm_schema	The schema where the OMOP CDM tables are located. Defaults to NULL.
cdm_tables	Which tables should be included? Supports a character vector, tidyselect selection helpers, or table groups.

- tbl_group("all")all CDM tables
- tbl_group("vocab")the CDM vocabulary tables
- tbl_group("clinical")the clinical CDM tables

write_schema	An optional schema in the CDM database that the user has write access to.
cohort_tables	A character vector listing the cohort table names to be included in the CDM object.
cdm_version	The version of the OMOP CDM: "5.3" (default), "5.4", "auto". "auto" attempts to automatically determine the cdm version using heuristics. Cohort tables must be in the write_schema.

Value

A list of dplyr database table references pointing to CDM tables

---

cdm_from_files *Create a CDM reference from a folder containing parquet, csv, or feather files*

Description

Create a CDM reference from a folder containing parquet, csv, or feather files

Usage

```
cdm_from_files(
  path,
  cdm_tables = deprecated(),
  format = "auto",
  as_data_frame = TRUE
)
```

Arguments

path	A folder where an OMOP CDM v5.4 instance is located.
cdm_tables	deprecated
format	What is the file format to be read in? Must be "auto" (default), "parquet", "csv", "feather".
as_data_frame	TRUE (default) will read files into R as dataframes. FALSE will read files into R as Arrow Datasets.

Value

A list of dplyr database table references pointing to CDM tables

collect.cdm_reference *Bring a remote CDM reference into R*

Description

This function calls collect on a list of lazy queries and returns the result as a list of dataframes.

Usage

```
## S3 method for class 'cdm_reference'
collect(x, ...)
```

Arguments

x	A cdm_reference object.
...	Not used. Included for compatibility.

Value

A cdm_reference object that is a list of R dataframes.

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomia_dir())
vocab <- cdm_from_con(con, cdm_tables = c("concept", "concept_ancestor"))
local_vocab <- collect(vocab)
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

computePermanent *Run a dplyr query and store the result in a permanent table*

Description

Run a dplyr query and store the result in a permanent table

Usage

```
computePermanent(x, name, schema = NULL, overwrite = FALSE)
```

Arguments

x	A dplyr query
name	Name of the table to be created
schema	Schema to create the new table in Can be a length 1 or 2 vector. (e.g. schema = "my_schema", schema = c("my_schema", "dbo))
overwrite	If the table already exists in the remote database should it be overwritten? (TRUE or FALSE)

Value

A dplyr reference to the newly created table

Examples

```
## Not run:
library(CDMConnector)

con <- DBI::dbConnect(duckdb::duckdb(), dbdir = CDMConnector::eunomia_dir())
concept <- dplyr::tbl(con, "concept")

rxnorm_count <- concept %>%
  dplyr::filter(domain_id == "Drug") %>%
  dplyr::mutate(isRxnorm = (vocabulary_id == "RxNorm")) %>%
  dplyr::count(isRxnorm) %>%
  computePermanent("rxnorm_count")

DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

---

computeQuery *Execute dplyr query and save result in remote database*

Description

This function is a wrapper around `dplyr::compute` that is tested on several database systems. It is needed to handle edge cases where `dplyr::compute` does not produce correct SQL.

Usage

```
computeQuery(
  x,
  name = uniqueTableName(),
  temporary = TRUE,
  schema = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

x	A dplyr query
name	The name of the table to create.
temporary	Should the table be temporary: TRUE (default) or FALSE
schema	The schema where the table should be created. Ignored if temporary = TRUE.
overwrite	Should the table be overwritten if it already exists: TRUE or FALSE (default) Ignored if temporary = TRUE.
...	Further arguments passed on the `dplyr::compute`

Value

A `dplyr::tbl()` reference to the newly created table.

---

dateadd *Add days or years to a date in a dplyr query*

Description

This function must be "unquoted" using the "bang bang" operator (!!). See example.

Usage

```
dateadd(date, number, interval = "day")
```

Arguments

date	The name of a date column in the database table as a character string
number	The number of units to add. Can be a positive or negative whole number.
interval	The units to add. Must be either "day" (default) or "year"

Value

Platform specific SQL that can be used in a dplyr query.

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb())
date_tbl <- dplyr::copy_to(con, data.frame(date1 = as.Date("1999-01-01")),
                           name = "tmpdate", overwrite = TRUE, temporary = TRUE)

df <- date_tbl %>%
  dplyr::mutate(date2 = !!dateadd("date1", 1, interval = "year")) %>%
  dplyr::collect()

DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

datediff *Compute the difference between two days*

Description

This function must be "unquoted" using the "bang bang" operator (!!). See example.

Usage

```
datediff(start, end, interval = "day")
```

Arguments

start	The name of the start date column in the database as a string.
end	The name of the end date column in the database as a string.
interval	The units to use for difference calculation. Must be either "day" (default) or "year".

Value

Platform specific SQL that can be used in a dplyr query.

Examples

```
## Not run:
library(SqlUtilities)
con <- DBI::dbConnect(duckdb::duckdb())
date_tbl <- dplyr::copy_to(con, data.frame(date1 = as.Date("1999-01-01")),
                           name = "tmpdate", overwrite = TRUE, temporary = TRUE)

df <- date_tbl %>%
  dplyr::mutate(date2 = !!dateadd("date1", 1, interval = "year")) %>%
  dplyr::mutate(dif_years = !!datediff("date1", "date2", interval = "year")) %>%
  dplyr::collect()

DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

dbConnect,dbConnectDetails-method
Connect to a database using stored connection details

Description

Create connection details, save them in an object, and use them to connect to a database using dbConnect later during program execution.

Usage

```
## S4 method for signature 'dbConnectDetails'
dbConnect(drv)
```

Arguments

drv	An dbConnectDetails object created by dbConnectDetails()

Value

An S4 object that inherits from DBIConnection used to communicate with the database engine.

Examples

```
## Not run:
library(DBI)
connectionDetails <- dbConnectDetails(RPostgres::Postgres(),
                                      dbname = "cdm",
                                      host = "localhost",
                                      user = "postgres",
                                      password = Sys.getenv("password"))

 selfContainedQuery <- function(dbConnectDetails) {
   con <- dbConnect(connectionDetails)
   on.exit(dbDisonnect(con))
   dbGetQuery(con, "select count(*) as n from synthea1k.person")
 }

selfContainedQuery(connectionDetails)

## End(Not run)
```

dbConnectDetails *Save database connection information in an object*

Description

dbConnectDetails returns an object that can be used to connect to a database with dbConnect at some future point in execution. Functions that need to create a new connection to a database with dbConnect can accept a dbConnectDetails object as an argument and use it to create a new connection.

Usage

```
dbConnectDetails(drv, ...)
```

Arguments

drv	A DBI driver (e.g. RPostgres::Postgres())
...	DBI Driver parameters needed to create a connection using DBI::dbConnect()

Value

A dbConnectArgs object that can be passed to dbConnect

Examples

```
## Not run:
library(DBI)
connectionDetails <- dbConnectDetails(RPostgres::Postgres(),
                                      dbname = "cdm",
                                      host = "localhost",
                                      user = "postgres",
                                      password = Sys.getenv("password"))

 selfContainedQuery <- function(dbConnectDetails) {
   con <- dbConnect(connectionDetails)
   on.exit(dbDisonnect(con))
   dbGetQuery(con, "select count(*) as n from synthea1k.person")
 }

selfContainedQuery(connectionDetails)

## End(Not run)
```

dbms *Get the database management system (dbms) from a cdm_reference or DBI connection*

Description

Get the database management system (dbms) from a cdm_reference or DBI connection

Usage

```
dbms(con)
```

Arguments

con	A DBI connection or cdm_reference

Value

A character string representing the dbms that can be used with SqlRender

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomia_dir())
cdm <- cdm_from_con(con)
dbms(cdm)
dbms(con)

## End(Not run)
```

downloadEunomiaData *Download Eunomia data files*

Description

Download the Eunomia data files from https://github.com/darwin-eu/EunomiaDatasets

Usage

```
downloadEunomiaData(
  datasetName = "GiBleed",
  cdmVersion = "5.3",
  pathToData = Sys.getenv("EUNOMIA_DATA_FOLDER"),
  overwrite = FALSE
)
```

Arguments

datasetName	The data set name as found on https://github.com/darwin-eu/EunomiaDatasets. The data set name corresponds to the folder with the data set ZIP files
cdmVersion	The OMOP CDM version. This version will appear in the suffix of the data file, for example: datasetName_cdmVersion.zip. Default: '5.3'
pathToData	The path where the Eunomia data is stored on the file system., By default the value of the environment variable "EUNOMIA_DATA_FOLDER" is used.
overwrite	Control whether the existing archive file will be overwritten should it already exist.

Value

Invisibly returns the destination if the download was successful.

Examples

```
## Not run:
downloadEunomiaData("GiBleed")

## End(Not run)
```

---

download_optional_data *Download the example Eunomia CDM dataset*

Description

This function is deprecated. Use downloadEunomiaData instead.

Usage

```
download_optional_data()
```

---

eunomiaDir *Create a copy of a Eunomia dataset*

Description

Creates a copy of a Eunomia database, and returns the path to the new database file. If the dataset does not yet exist on the user's computer it will attempt to download the source data to the the path defined by the EUNOMIA_DATA_FOLDER environment variable.

Usage

```
eunomiaDir(
  datasetName = "GiBleed",
  cdmVersion = "5.3",
  pathToData = Sys.getenv("EUNOMIA_DATA_FOLDER"),
  dbms = "duckdb",
  databaseFile = tempfile(fileext = paste0(".", dbms))
)
```

Arguments

datasetName	The data set name as found on https://github.com/darwin-eu/EunomiaDatasets. The data set name corresponds to the folder with the data set ZIP files
cdmVersion	The OMOP CDM version. This version will appear in the suffix of the data file, for example: datasetName_cdmVersion.zip. Default: '5.3'
pathToData	The path where the Eunomia data is stored on the file system., By default the value of the environment variable "EUNOMIA_DATA_FOLDER" is used.
dbms	The database system to use. "sqlite" or "duckdb" (default)
databaseFile	The path where the database file will be copied to. By default, the database will be copied to a temporary folder, and will be deleted at the end of the R session.

Value

The file path to the new Eunomia dataset copy

Examples

```
## Not run:
 conn <- DBI::dbConnect(RSQLite::SQLite(), eunomiaDir("GiBleed"))
 DBI::dbDisconnect(conn)

 conn <- DBI::dbConnect(duckdb::duckdb(), eunomiaDir("GiBleed", dbms = "duckdb"))
 DBI::dbDisconnect(conn, shutdown = TRUE)

 conn <- DatabaseConnector::connect(dbms = "sqlite", server = eunomiaDir("GiBleed"))
 DatabaseConnector::disconnect(conn)

## End(Not run)
```

---

eunomia_dir *Create a new Eunomia CDM*

Description

Create a copy of the duckdb Eunomia CDM and return the file path

Usage

```
eunomia_dir(exdir = NULL)
```

Arguments

exdir	Enclosing directory where the Eunomia CDM should be created. If NULL (default) then a temp folder is created.

Value

The full path to the new Eunomia CDM that can be passed to `dbConnect()`

Examples

```
## Not run:
library(DBI)
library(CDMConnector)
con <- dbConnect(duckdb::duckdb(), dbdir = getEunomiaPath())
dbListTables(con)
dbDisconnect(con)

## End(Not run)
```

---

eunomia_is_available *Has the eunomia dataset been cached?*

Description

Has the eunomia dataset been cached?

Usage

```
eunomia_is_available(datasetName = "GiBleed", cdmVersion = "5.3")
```

Arguments

datasetName	Name of the Eunomia dataset to check. Defaults to "GiBleed".
cdmVersion	Version of the Eunomia dataset to check. Must be "5.3" or "5.4".

Value

TRUE if the eunomia example dataset is available and FASLE otherwise

extractLoadData — *Extract the Eunomia data files and load into a SQLite or duckdb database*

Description

Extract files from a .ZIP file and creates a SQLite or duckdb OMOP CDM database that is then stored in the same directory as the .ZIP file.

Usage

```
extractLoadData(from, to, dbms = "sqlite", verbose = FALSE)
```

Arguments

from	The path to the .ZIP file that contains the csv CDM source files
to	The path to the .sqlite or .duckdb file that will be created
dbms	The file based database system to use: 'sqlite' (default) or 'duckdb'
verbose	Print progress notes? TRUE or FALSE

generateCohortSet — *Generate a set of cohorts*

Description

This function generates a set of cohorts in the cohort table.

Usage

```
generateCohortSet(
  cdm,
  cohortSet,
  cohortTableName = "cohort",
  overwrite = FALSE
)
```

Arguments

cdm	cdm reference object
cohortSet	A cohort definition set dataframe
cohortTableName	The name of the cohort table in the cdm. Defaults to 'cohort'.
overwrite	Should the cohort table be overwritten if it already exists? TRUE or FALSE (default).

Value

cdm reference object with the added cohort table containing generated cohorts

---

`listTables` *List tables in a schema*

Description

DBI::dbListTables can be used to get all tables in a database but not always in a specific schema. `listTables` will list tables in a schema.

Usage

```
listTables(con, schema = NULL)
```

Arguments

`con`	A DBI connection to a database
`schema`	The name of a schema in a database. If NULL, returns DBI::dbListTables(con).

Value

A character vector of table names

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomia_dir())
listTables(con, schema = "main")

## End(Not run)
```

`print.cdm_reference` *Print a CDM reference object*

Description

Print a CDM reference object

Usage

```
## S3 method for class 'cdm_reference'
print(x, ...)
```

Arguments

x	A cdm_reference object
...	Included for compatibility with generic. Not used.

Value

Invisibly returns the input

---

readCohortSet *Read a set of cohort definitions into R*

Description

A "cohort set" is a collection of cohort definitions. In R this is stored in a dataframe with cohortId, cohortName, cohort, and sql columns. On disk this is stored as a folder with a CohortsToCreate.csv file and one or more json files. If the CohortsToCreate.csv file is missing then all of the json files in the folder will be used, cohortIds will be automatically assigned in alphabetical order, and cohortNames will match the file names.

Usage

```
readCohortSet(path)
```

Arguments

path	The path to a folder containing Circe cohort definition json files and optionally a csv file named CohortsToCreate.csv with columns cohortId, cohortName, and jsonPath.

---

snapshot *Extract CDM metadata*

Description

Extract the name, version, and selected record counts from a cdm.

Usage

```
snapshot(cdm)
```

Arguments

cdm	A cdm object

Value

A list of attributes about the cdm including selected fields from the cdm_source table and record counts from the person and observation_period tables

Examples

```
## Not run:
library(CDMConnector)
con <- DBI::dbConnect(duckdb::duckdb(), eunomia_dir())
cdm <- cdm_from_con(con, cdm_tables = c(tbl_group("default"), "cdm_source"))
snapshot(cdm)

DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

---

stow *Collect a list of lazy queries and save the results as files*

Description

Collect a list of lazy queries and save the results as files

Usage

```
stow(cdm, path, format = "parquet")
```

Arguments

cdm	A cdm object
path	A folder to save the cdm object to
format	The file format to use: "parquet", "csv", "feather".

Value

Invisibly returns the cdm input

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = eunomia_dir())
vocab <- cdm_from_con(con, cdm_tables = c("concept", "concept_ancestor"))
stow(vocab, here::here("vocab_tables"))
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

`tbl_group` *CDM table selection helper*

Description

The OMOP CDM tables are grouped together and the `tbl_group` function allows users to easily create a CDM reference including one or more table groups.

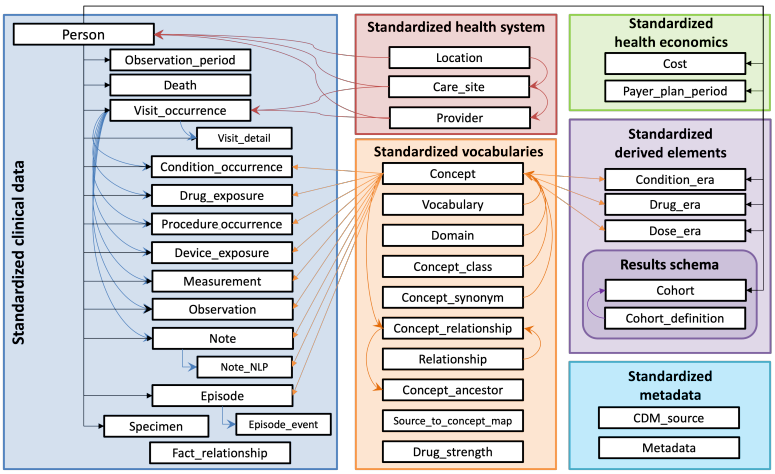
Usage

```
tbl_group(group)
```

Arguments

`group`	A character vector of CDM table groups: "vocab", "clinical", "all", "default", "derived".

Details



The "default" table group is meant to capture the most commonly used set of CDM tables. Currently the "default" group is: person, observation_period, visit_occurrence, visit_detail, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, death, note, note_nlp, specimen, fact_relationship, location, care_site, provider, payer_plan_period, cost, drug_era, dose_era, condition_era, concept, vocabulary, concept_relationship, concept_ancestor, concept_synonym, drug_strength

Value

A character vector of CDM tables names in the groups

Examples

```
## Not run:
con <- DBI::dbConnect(RPostgres::Postgres(),
                      dbname = "cdm",
                      host = "localhost",
                      user = "postgres",
                      password = Sys.getenv("PASSWORD"))

cdm <- cdm_from_con(con, cdm_tables = tbl_group("vocab"))

## End(Not run)
```

`validate_cdm` *Validation report for a CDM*

Description

Print a short validation report for a cdm object. The validation includes checking that column names are correct and that no tables are empty. A short report is printed to the console. This function is meant for interactive use.

Usage

```
validate_cdm(cdm)

validateCdm(cdm)
```

Arguments

`cdm`	A cdm reference object.

Value

Invisibly returns the cdm input

Functions

- `validateCdm()`: camelCase alias

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(), eunomia_dir())
cdm <- cdm_from_con(con, cdm_tables = c("person", "observation_period"))
validate_cdm(cdm)
DBI::dbDisconnect(con)

## End(Not run)
```

version *Get the CDM version*

Description

Extract the CDM version attribute from a cdm_reference object

Usage

```
version(cdm)
```

Arguments

cdm	A cdm object

Value

"5.3" or "5.4"

Examples

```
## Not run:
library(CDMConnector)
con <- DBI::dbConnect(duckdb::duckdb(), eunomia_dir())
cdm <- cdm_from_con(con, cdm_tables = c(tbl_group("default"), "cdm_source"))
version(cdm)

DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

Index