Package ‘omopgenerics’

August 17, 2024

Title  Methods and Classes for the OMOP Common Data Model
Version  0.3.0
Description  Provides definitions of core classes and methods used by analytic pipelines that query the OMOP (Observational Medical Outcomes Partnership) common data model.
License  Apache License (>= 2)
Encoding  UTF-8
RoxygenNote  7.3.2
Imports  cli, dbplyr, dplyr, glue, lifecycle, methods, purrr, rlang, snakecase, stringr, tidyr, vctrs
Depends  R (>= 4.1)
Suggests  covr, here, knitr, jsonlite, readr, rmarkdown, testthat (>= 3.0.0), withr
URL  https://darwin-eu-dev.github.io/omopgenerics/
Config/testthat/edition  3
Config/testthat/parallel  true
VignetteBuilder  knitr
NeedsCompilation  no
Author  Martí Català [aut, cre] (<https://orcid.org/0000-0003-3308-9905>), Edward Burn [aut] (<https://orcid.org/0000-0002-9286-1128>), Mike Du [ctb] (<https://orcid.org/0000-0002-9517-8834>), Yuchen Guo [ctb] (<https://orcid.org/0000-0002-0847-4855>), Adam Black [ctb] (<https://orcid.org/0000-0001-5576-8701>), Marta Alcalde-Herraiz [ctb] (<https://orcid.org/0009-0002-4405-1814>)
Maintainer  Martí Català <marti.catalasabate@ndorms.ox.ac.uk>
Repository  CRAN
Date/Publication  2024-08-17 06:30:07 UTC
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>achillesColumns</td>
<td>4</td>
</tr>
<tr>
<td>achillesTables</td>
<td>5</td>
</tr>
<tr>
<td>assertCharacter</td>
<td>5</td>
</tr>
<tr>
<td>assertChoice</td>
<td>6</td>
</tr>
<tr>
<td>assertClass</td>
<td>7</td>
</tr>
<tr>
<td>assertDate</td>
<td>7</td>
</tr>
<tr>
<td>assertList</td>
<td>8</td>
</tr>
<tr>
<td>assertLogical</td>
<td>9</td>
</tr>
<tr>
<td>assertNumeric</td>
<td>10</td>
</tr>
<tr>
<td>assertTable</td>
<td>11</td>
</tr>
<tr>
<td>assertTrue</td>
<td>12</td>
</tr>
<tr>
<td>attrition</td>
<td>12</td>
</tr>
<tr>
<td>attrition.cohort_table</td>
<td>13</td>
</tr>
<tr>
<td>bind</td>
<td>14</td>
</tr>
<tr>
<td>bind.cohort_table</td>
<td>14</td>
</tr>
<tr>
<td>bind.summarised_result</td>
<td>15</td>
</tr>
<tr>
<td>cdmFromTables</td>
<td>16</td>
</tr>
<tr>
<td>cdmName</td>
<td>17</td>
</tr>
<tr>
<td>cdmReference</td>
<td>18</td>
</tr>
<tr>
<td>cdmSelect</td>
<td>19</td>
</tr>
<tr>
<td>cdmSource</td>
<td>20</td>
</tr>
<tr>
<td>cdmSourceType</td>
<td>21</td>
</tr>
<tr>
<td>cdmTableFromSource</td>
<td>22</td>
</tr>
<tr>
<td>cdmVersion</td>
<td>22</td>
</tr>
<tr>
<td>checkCohortRequirements</td>
<td>23</td>
</tr>
<tr>
<td>cohortCodelist</td>
<td>24</td>
</tr>
<tr>
<td>cohortColumns</td>
<td>25</td>
</tr>
<tr>
<td>cohortCount</td>
<td>26</td>
</tr>
<tr>
<td>cohortTables</td>
<td>27</td>
</tr>
<tr>
<td>collect.cdm_reference</td>
<td>28</td>
</tr>
<tr>
<td>collect.cohort_table</td>
<td>29</td>
</tr>
<tr>
<td>compute.cdm_table</td>
<td>29</td>
</tr>
<tr>
<td>dropSourceTable</td>
<td>30</td>
</tr>
<tr>
<td>dropTable</td>
<td>30</td>
</tr>
<tr>
<td>emptyAchillesTable</td>
<td>31</td>
</tr>
<tr>
<td>emptyCdmReference</td>
<td>32</td>
</tr>
<tr>
<td>emptyCodelist</td>
<td>33</td>
</tr>
<tr>
<td>emptyCodelistWithDetails</td>
<td>33</td>
</tr>
<tr>
<td>emptyCohortTable</td>
<td>34</td>
</tr>
<tr>
<td>emptyOmopTable</td>
<td>35</td>
</tr>
<tr>
<td>emptySummarisedResult</td>
<td>36</td>
</tr>
<tr>
<td>estimateTypeChoices</td>
<td>36</td>
</tr>
<tr>
<td>exportCodelist</td>
<td>37</td>
</tr>
<tr>
<td>exportConceptSetExpression</td>
<td>37</td>
</tr>
<tr>
<td>exportSummarisedResult</td>
<td>38</td>
</tr>
<tr>
<td>getCohortId</td>
<td>38</td>
</tr>
</tbody>
</table>
achillesColumns

Required columns for each of the achilles result tables

Description

Required columns for each of the achilles result tables

Usage

achillesColumns(table, onlyRequired = TRUE, version = "5.3")

Arguments

table Table for which to see the required columns. One of "achilles_analysis", "achilles_results", or "achilles_results_dist".
onlyRequired Whether to include only required fields.
version Version of the OMOP Common Data Model.

Value

Character vector with the column names

Examples

library(omopgenerics)
achillesColumns("achilles_analysis")
achillesColumns("achilles_results")
achillesColumns("achilles_results_dist")
achillesTables

| achillesTables | Names of the tables that contain the results of achilles analyses |

**Description**
Names of the tables that contain the results of achilles analyses

**Usage**

```r
achillesTables(version = "5.3")
```

**Arguments**

- `version` Version of the OMOP Common Data Model.

**Value**
Names of the tables that are contain the results from the achilles analyses

**Examples**

```r
library(omopgenerics)
achillesTables()
```

---

assertCharacter

| assertCharacter | Assert that an object is a character and fulfill certain conditions. |

**Description**
Assert that an object is a character and fulfill certain conditions.

**Usage**

```r
assertCharacter(
  x,
  length = NULL,
  na = FALSE,
  null = FALSE,
  unique = FALSE,
  named = FALSE,
  minNumCharacter = 0,
  call = parent.frame(),
  msg = NULL
)
```
assertChoice

Assert that an object is within a certain options.

Arguments

x      Variable to check.
length  Required length. If NULL length is not checked.
na      Whether it can contain NA values.
null    Whether it can be NULL.
unique  Whether it has to contain unique elements.
named   Whether it has to be named.
minNumCharacter  Minimum number of characters that all elements must have.
call    Call argument that will be passed to cli error message.
msg     Custom error message.

assertChoice(  
  x,  
  choices,  
  length = NULL,  
  na = FALSE,  
  null = FALSE,  
  unique = FALSE,  
  named = FALSE,  
  call = parent.frame(),  
  msg = NULL  
)

Description

Assert that an object is within a certain options.

Usage

assertChoice(  
  x,  
  choices,  
  length = NULL,  
  na = FALSE,  
  null = FALSE,  
  unique = FALSE,  
  named = FALSE,  
  call = parent.frame(),  
  msg = NULL  
)
assertClass

Assert that an object has a certain class.

Description

Assert that an object has a certain class.

Usage

```
assertClass(
  x,  # To check.
  class,  # Expected class or classes.
  length = NULL,  # Required length. If NULL length is not checked.
  null = FALSE,  # Whether it can be NULL.
  all = FALSE,  # Whether it should have all the classes or only at least one of them.
  extra = TRUE,  # Whether the object can have extra classes.
  call = parent.frame(),  # Call argument that will be passed to cli.
  msg = NULL  # Custom error message.
)
```

Arguments

- **x**: To check.
- **class**: Expected class or classes.
- **length**: Required length. If NULL length is not checked.
- **null**: Whether it can be NULL.
- **all**: Whether it should have all the classes or only at least one of them.
- **extra**: Whether the object can have extra classes.
- **call**: Call argument that will be passed to cli.
- **msg**: Custom error message.

assertDate

Assert Date

Description

Assert Date
assertList

Usage

assertList(
  x,
  length = NULL,
  na = FALSE,
  null = FALSE,
  unique = FALSE,
  named = FALSE,
  call = parent.frame(),
  msg = NULL
)

Arguments

x Expression to check.
length Required length.
a Whether it can contain NA values.
null Whether it can be NULL.
unique Whether it has to contain unique elements.
named Whether it has to be named.
call Call argument that will be passed to cli error message.
msg Custom error message

Value

x

---

assertList Assert that an object is a list.

Description

Assert that an object is a list.

Usage

assertList(
  x,
  length = NULL,
  na = FALSE,
  null = FALSE,
  unique = FALSE,
  named = FALSE,
  class = NULL,
  call = parent.frame(),
  msg = NULL
)
**assertLogical**

**Arguments**

- `x`: Variable to check.
- `length`: Required length. If NULL length is not checked.
- `na`: Whether it can contain NA values.
- `null`: Whether it can be NULL.
- `unique`: Whether it has to contain unique elements.
- `named`: Whether it has to be named.
- `class`: Class that the elements must have.
- `call`: Call argument that will be passed to cli error message.
- `msg`: Custom error message.

**assertLogical**  
*Assert that an object is a logical.*

**Description**

Assert that an object is a logical.

**Usage**

```r
assertLogical(
  x, 
  length = NULL,
  na = FALSE,
  null = FALSE,
  named = FALSE,
  call = parent.frame(),
  msg = NULL
)
```

**Arguments**

- `x`: Variable to check.
- `length`: Required length. If NULL length is not checked.
- `na`: Whether it can contain NA values.
- `null`: Whether it can be NULL.
- `named`: Whether it has to be named.
- `call`: Call argument that will be passed to cli error message.
- `msg`: Custom error message.
assertNumeric  Assert that an object is a numeric.

Description

Assert that an object is a numeric.

Usage

assertNumeric(
  x,
  integerish = FALSE,
  min = -Inf,
  max = Inf,
  length = NULL,
  na = FALSE,
  null = FALSE,
  unique = FALSE,
  named = FALSE,
  call = parent.frame(),
  msg = NULL
)

Arguments

x Variable to check.
integerish Whether it has to be an integer
min Minimum value that the object can be.
max Maximum value that the object can be.
length Required length. If NULL length is not checked.
na Whether it can contain NA values.
null Whether it can be NULL.
unique Whether it has to contain unique elements.
named Whether it has to be named.
call Call argument that will be passed to cli error message.
msg Custom error message.
assertTable

Assert that an object is a table.

Description

Assert that an object is a table.

Usage

```r
assertTable(
  x,
  class = NULL,
  numberColumns = NULL,
  numberRows = NULL,
  columns = character(),
  allowExtraColumns = TRUE,
  null = FALSE,
  unique = FALSE,
  call = parent.frame(),
  msg = NULL
)
```

Arguments

- **x**: Variable to check.
- **class**: A class that the table must have: "tbl", "data.fram", "tbl_sql", ...
- **numberColumns**: Number of columns that it has to contain.
- **numberRows**: Number of rows that it has to contain.
- **columns**: Name of the columns required.
- **allowExtraColumns**: Whether extra columns are allowed.
- **null**: Whether it can be NULL.
- **unique**: Whether it has to contain unique rows.
- **call**: Call argument that will be passed to cli error message.
- **msg**: Custom error message.
assertTrue  *Assert that an expression is TRUE.*

**Description**

Assert that an expression is TRUE.

**Usage**

```r
call(x, null = FALSE, call = parent.frame(), msg = NULL)
```

**Arguments**

- `x`: Expression to check.
- `null`: Whether it can be NULL.
- `call`: Call argument that will be passed to `cli` error message.
- `msg`: Custom error message.

**attrition**  *Get attrition from an object.*

**Description**

Get attrition from an object.

**Usage**

```r
attrition(x)
```

**Arguments**

- `x`: An object for which to get an attrition summary.

**Value**

A table with the attrition.
attrition.cohort_table

Get cohort attrition from a cohort_table object.

Description

Get cohort attrition from a cohort_table object.

Usage

## S3 method for class 'cohort_table'
attrition(x)

Arguments

x

A cohort_table

Value

A table with the attrition.

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)
observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)
cohort <- tibble(
  cohort_definition_id = c(1, 1, 1, 2),
  subject_id = 1,
  cohort_start_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
  cohort_end_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
)
cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "my_example_cdm",
  cohortTables = list("cohort1" = cohort)
)
attrition(cdm$cohort1)
bind

Bind two or more objects of the same class.

Description

Bind two or more objects of the same class.

Usage

bind(...)

Arguments

... Objects to bind.

Value

New object.

bind.cohort_table

Bind two or more cohort tables

Description

Bind two or more cohort tables

Usage

## S3 method for class 'cohort_table'
bind(..., name)

Arguments

... Generated cohort set objects to bind. At least two must be provided.

name Name of the new generated cohort set.

Value

The cdm object with a new generated cohort set containing all of the cohorts passed.
Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cohort1 <- tibble(
  cohort_definition_id = 1,
  subject_id = 1:3,
  cohort_start_date = as.Date("2010-01-01"),
  cohort_end_date = as.Date("2010-01-05")
)

cohort2 <- tibble(
  cohort_definition_id = c(2, 2, 3, 3, 3),
  subject_id = c(1, 2, 3, 1, 2),
  cohort_start_date = as.Date("2010-01-01"),
  cohort_end_date = as.Date("2010-01-05")
)

cohortTables <- list("cohort1" = cohort1, "cohort2" = cohort2)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3),
      gender_concept_id = 0,
      year_of_birth = 1990,
      race_concept_id = 0,
      ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3,
      person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock",
  cohortTables = cohortTables)

settings(cdm$cohort3)

bind(cdm$cohort1, cdm$cohort2, name = "cohort3")
```

bind.summarised_result

Bind two or summarised_result objects

Description

Bind two or summarised_result objects

Usage

```r
## S3 method for class 'summarised_result'
bind(...)```
Arguments

... summarised_result objects

Value

A summarised_result object the merged objects.

Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock",
  cohortTables = list("cohort1" = tibble(
    cohort_definition_id = 1,
    subject_id = 1:3,
    cohort_start_date = as.Date("2010-01-01"),
    cohort_end_date = as.Date("2010-01-05")
  ))
)

result1 <- summary(cdm)
result2 <- summary(cdm$cohort1)

mergedResult <- bind(result1, result2)
mergedResult
```

---

**cdmFromTables**  
*Create a cdm object from local tables*

**Description**

Create a cdm object from local tables
**cdmName**

Get the name of a `cdm_reference` associated object

**Description**

Get the name of a `cdm_reference` associated object

**Usage**

```r
cdmName(x)
```

**Usage**

```r
cdmFromTables(tables, cdmName, cohortTables = list(), cdmVersion = NULL)
```

**Arguments**

- `tables` List of tables to be part of the cdm object.
- `cdmName` Name of the cdm object.
- `cohortTables` List of tables that contains cohort, cohort_set and cohort_attrition can be provided as attributes.
- `cdmVersion` Version of the cdm_reference

**Value**

A `cdm_reference` object.

**Examples**

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)
observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test"
)
```
cdmReference

Arguments

x A cdm_reference or cdm_table object.

Value

Name of the cdm_reference.

Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

cdmName(cdm)

cdmName(cdm$person)
```

---

**cdmReference**

*Get the cdm_reference of a cdm_table.*

**Description**

Get the cdm_reference of a cdm_table.

**Usage**

```r
cdmReference(table)
```

**Arguments**

- **table** A cdm_table.
Value

A cdm_reference.

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

cdmReference(cdm$person)

---

cdmSelect

Restrict the cdm object to a subset of tables.

Description

Restrict the cdm object to a subset of tables.

Usage

cdmSelect(cdm, ...)

Arguments

cdm A cdm_reference object.

... Selection of tables to use, it supports tidyselect expressions.

Value

A cdm_reference with only the specified tables.
Examples

cdm <- emptyCdmReference("my cdm")
cdm

cdm |> 
    cdmSelect("person")

cdmSource

Get the cdmSource of an object.

Description

Get the cdmSource of an object.

Usage

cdmSource(x, cdm = lifecycle::deprecated())

Arguments

x
  Object to obtain the cdmSource.

cdm
  Deprecated, use x please.

Value

A cdm_source object.

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)
cdmSourceType

Get the source type of a cdm_reference object.

Description

[Deprecated]

Usage

cdmSourceType(cdm)

Arguments

cdm A cdm_reference object.

Value

A character vector with the type of source of the cdm_reference object.

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

cdmSourceType(cdm)
cdmTableFromSource

This is an internal developer focused function that creates a cdm_table from a table that shares the source but it is not a cdm_table. Please use insertTable if you want to insert a table to a cdm_reference object.

Description

This is an internal developer focused function that creates a cdm_table from a table that shares the source but it is not a cdm_table. Please use insertTable if you want to insert a table to a cdm_reference object.

Usage

cdmTableFromSource(src, value)

Arguments

src  A cdm_source object.
value  A table that shares source with the cdm_reference object.

Value

A cdm_table.

cdmVersion

Get the version of an object.

Description

Get the version of an object.

Usage

cdmVersion(x)

Arguments

x  Object to know the cdm_version of an object.

Value

A character vector indicating the cdm version.
Examples

```r
cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

cdmVersion(cdm)
cdmVersion(cdm$person)
```

---

**checkCohortRequirements**

*Check whether a cohort table satisfies requirements*

---

**Description**

[Deprecated]

**Usage**

```r
checkCohortRequirements(
  cohort,
  checkEndAfterStart = TRUE,
  checkOverlappingEntries = TRUE,
  checkMissingValues = TRUE,
  checkInObservation = TRUE,
  type = "error",
  call = parent.frame()
)
```

**Arguments**

- `cohort` cohort_table object.
If TRUE a check that all cohort end dates come on or after cohort start date will be performed.

If TRUE a check that no individuals have overlapping cohort entries will be performed.

If TRUE a check that there are no missing values in required fields will be performed.

If TRUE a check that cohort entries are within the individuals observation periods will be performed.

can be either "error" or "warning". If "error" any check failure will result in an error, whereas if "warning" any check failure will result in a warning.

The call for which to return the error message.

An error will be returned if any of the selected checks fail.

---

**cohortCodelist**  
Get codelist from a cohort_table object.

---

**Description**

Get codelist from a cohort_table object.

**Usage**

```r
cohortCodelist(
  cohortTable,  
  cohortId,  
  type = c("index event", "inclusion criteria", "exclusion criteria", "exit criteria")
)
```

**Arguments**

- **cohortTable**: A cohort_table object.
- **cohortId**: A particular cohort definition id that is present in the cohort table.
- **type**: The reason for the codelist. Can be "index event", "inclusion criteria", or "exit criteria".

**Value**

A table with the codelists used.
### Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cohort <- tibble(
  cohort_definition_id = c(1, 1, 1, 2),
  subject_id = 1,
  cohort_start_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
  cohort_end_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01"))
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "my_example_cdm",
  cohortTables = list("cohort1" = cohort)
)

cdm$cohort1 <- newCohortTable(table = cdm$cohort1,
  cohortCodelistRef = dplyr::tibble(
    cohort_definition_id = c(1, 1, 1, 2),
    codelist_name = c("disease X", "disease X", "disease X",
    "disease Y", "disease Y"),
    concept_id = c(1, 2, 3, 4, 5),
    type = "index event"
  ))

cohortCodelist(cdm$cohort1, cohortId = 1, type = "index event")
```

---

**cohortColumns**

*Required columns for a generated cohort set.*

**Description**

Required columns for a generated cohort set.

**Usage**

```r
cohortColumns(table, version = "5.3")
```
Arguments

- **table**: Either cohort, cohort_set or cohort_attrition
- **version**: Version of the OMOP Common Data Model.

Value

Character vector with the column names

Required columns

Examples

```r
library(omopgenerics)
cohortColumns("cohort")
```

cohortCount  Get cohort counts from a cohort_table object.

Description

Get cohort counts from a cohort_table object.

Usage

```r
cohortCount(cohort)
```

Arguments

- **cohort**: A cohort_table object.

Value

A table with the counts.

Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)
cohort <- tibble(
  cohort_definition_id = c(1, 1, 1, 2),
  subject_id = 1,
  cohort_start_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
  cohort_end_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01"))
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "my_example_cdm",
  cohortTables = list("cohort1" = cohort)
)

cohortCount(cdm$cohort1)

cohortTables

Cohort tables that a cdm reference can contain in the OMOP Common Data Model.

Description

Cohort tables that a cdm reference can contain in the OMOP Common Data Model.

Usage

cohortTables(version = "5.3")

Arguments

version Version of the OMOP Common Data Model.

Value

cohort tables

Examples

library(omopgenerics)
cohortTables()
**Description**

Retrieves the cdm reference into a local cdm.

**Usage**

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

**Arguments**

- `x` A `cdm_reference` object.
- `...` For compatibility only, not used.

**Value**

A local `cdm_reference`.

**Examples**

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = dplyr::tibble(
      person_id = c(1, 2, 3),
      gender_concept_id = 0,
      year_of_birth = 1990,
      race_concept_id = 0,
      ethnicity_concept_id = 0
    ),
    "observation_period" = dplyr::tibble(
      observation_period_id = 1:3,
      person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

collect(cdm)
```
collect.cohort_table  To collect a cohort_table object.

Description
To collect a cohort_table object.

Usage
## S3 method for class 'cohort_table'
collect(x, ...)

Arguments
x cohort_table object.
... Not used (for compatibility).

Value
A data frame with the cohort_table

compute.cdm_table  Store results in a table.

Description
Store results in a table.

Usage
## S3 method for class 'cdm_table'
compute(x, name = NULL, temporary = NULL, overwrite = TRUE, ...)

Arguments
x Table in the cdm.
name Name to store the table with.
temporary Whether to store table temporarily (TRUE) or permanently (FALSE).
overwrite Whether to overwrite previously existing table with name same.
... For compatibility (not used).

Value
Reference to a table in the cdm
**dropSourceTable**

*Drop a table from a cdm object.*

**Description**

Drop a table from a cdm object.

**Usage**

\[
\text{dropSourceTable}(\text{cdm}, \text{name})
\]

**Arguments**

- **cdm**
  - A cdm reference.
- **name**
  - Name(s) of the table(s) to insert. Tidyselect statements are supported.

**Value**

The table in the cdm reference.

---

**dropTable**

*Drop a table from a cdm object.*

**Description**

Drop a table from a cdm object.

**Usage**

\[
\text{dropTable}(\text{cdm}, \text{name})
\]

**Arguments**

- **cdm**
  - A cdm reference.
- **name**
  - Name(s) of the table(s) to drop. Tidyselect statements are supported.

**Value**

The cdm reference.
Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cohort <- tibble(
  cohort_definition_id = c(1, 1, 1, 2),
  subject_id = 1,
  cohort_start_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
  cohort_end_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01"))
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "my_example_cdm",
  cohortTables = list("cohort1" = cohort)
)

cdm

cdm <- dropTable(cdm = cdm, name = "cohort1")

cdm

emptyAchillesTable  Create an empty achilles table

Description

Create an empty achilles table

Usage

emptyAchillesTable(cdm, name)
emptyCdmReference

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdm</td>
<td>A cdm_reference to create the table.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the table to create.</td>
</tr>
</tbody>
</table>

Value

The cdm_reference with an achilles empty table

Examples

```r
library(omopgenerics)
cdm <- emptyCdmReference("my_example_cdm")
emptyAchillesTable(cdm = cdm, name = "achilles_results")
```

Description

Create an empty cdm_reference

Usage

`emptyCdmReference(cdmName, cdmVersion = NULL)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdmName</td>
<td>Name of the cdm_reference</td>
</tr>
<tr>
<td>cdmVersion</td>
<td>Version of the cdm_reference</td>
</tr>
</tbody>
</table>

Value

An empty cdm_reference

Examples

```r
library(omopgenerics)
emptyCdmReference(cdmName = "my_example_cdm")
```
### Description

An empty codelist object.

#### Usage

```python
emptyCodelist()
```

#### Value

An empty codelist object.

#### Examples

```python
emptyCodelist()
```

---

### Description

An empty codelist object.

#### Usage

```python
emptyCodelistWithDetails()
```

#### Value

An empty codelist object.

#### Examples

```python
emptyCodelistWithDetails()
```
emptyCohortTable  

Create an empty cohort_table object

Description

Create an empty cohort_table object

Usage

emptyCohortTable(cdm, name, overwrite = TRUE)

Arguments

cdm  
A cdm_reference to create the table.

name  
Name of the table to create.

overwrite  
Whether to overwrite an existent table.

Value

The cdm_reference with an empty cohort table

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdName = "test"
)

cdm <- emptyCohortTable(cdm, "my_empty_cohort")

cdm
cdm$my_empty_cohort

settings(cdm$my_empty_cohort)
attrition(cdm$my_empty_cohort)
cohortCount(cdm$my_empty_cohort)
emptyOmopTable

Create an empty omop table

Description

Create an empty omop table

Usage

emptyOmopTable(cdm, name)

Arguments

cdm
  A cdm_reference to create the table.

name
  Name of the table to create.

Value

The cdm_reference with an empty cohort table

Examples

library(omopgenerics)

person <- dplyr::tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- dplyr::tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test"
)

cdm <- emptyOmopTable(cdm, "drug_exposure")

cdm$drug_exposure
emptySummarisedResult  Empty summarised_result object.

Description

Empty summarised_result object.

Usage

emptySummarisedResult(settings = NULL)

Arguments

settings Tibble/data.frame with the settings of the empty summarised_result. It has to contain at least result_id column.

Value

An empty summarised_result object.

Examples

library(omopgenerics)

emptySummarisedResult()

---

estimateTypeChoices  Choices that can be present in estimate_type column.

Description

Choices that can be present in estimate_type column.

Usage

estimateTypeChoices()

Value

A character vector with the options that can be present in estimate_type column in the summarised_result objects.

Examples

library(omopgenerics)

estimateTypeChoices()
exportCodelist

Export a codelist object.

Description

Export a codelist object.

Usage

exportCodelist(x, path, type = "json")

Arguments

x A codelist
path Path to where files will be created.
type Type of files to export. Currently only "json" is supported.

Value

Files with codelists

exportConceptSetExpression

Export a concept set expression.

Description

Export a concept set expression.

Usage

exportConceptSetExpression(x, path, type = "json")

Arguments

x A concept set expression
path Path to where files will be created.
type Type of files to export. Currently only "json" is supported.

Value

Files with codelists
### exportSummarisedResult

*Export a summarised_result object to a csv file.*

**Description**

Export a summarised_result object to a csv file.

**Usage**

```r
definition = summarised_result(...,
    minCellCount = 5,
    fileName = \"results_{cdm_name}_{date}.csv\",
    path = getwd()
)
```

**Arguments**

**...**
A set of summarised_result objects.

**minCellCount**
Minimum count for suppression purposes.

**fileName**
Name of the file that will be created. Use \{cdm_name\} to refer to the cdmName of the objects and \{date\} to add the export date.

**path**
Path where to create the csv file.

### getCohtId

*Get the cohort definition id of a certain name*

**Description**

Get the cohort definition id of a certain name.

**Usage**

```r
definition = getCohtId(cohort, cohortName = NULL)
```

**Arguments**

**cohort**
A cohort_table object.

**cohortName**
Names of the cohort of interest. If NULL all cohort names are shown.

**Value**

Cohort definition ids
getCohortName

Get the cohort name of a certain cohort definition id

Description
Get the cohort name of a certain cohort definition id

Usage
getcohortName(cohort, cohortId = NULL)

Arguments
- cohort: A cohort_table object.
- cohortId: Cohort definition id of interest. If NULL all cohort ids are shown.

Value
Cohort names

getPersonIdentifier

Get the column name with the person identifier from a table (either subject_id or person_id), it will throw an error if it contains both or neither.

Description
Get the column name with the person identifier from a table (either subject_id or person_id), it will throw an error if it contains both or neither.

Usage
getPersonIdentifier(x, call = parent.frame())

Arguments
- x: A table.
- call: A call argument passed to cli functions.

Value
Person identifier column.
**importCodelist**  
*Import a codelist.*

**Description**  
Import a codelist.

**Usage**  
```python  
importCodelist(path, type = "json")  
```

**Arguments**  
- **path**  
  Path to where files will be created.
- **type**  
  Type of files to export. Currently only "json" is supported.

**Value**  
A codelist

---

**importConceptSetExpression**  
*Import a concept set expression.*

**Description**  
Import a concept set expression.

**Usage**  
```python  
importConceptSetExpression(path, type = "json")  
```

**Arguments**  
- **path**  
  Path to where files will be created.
- **type**  
  Type of files to export. Currently only "json" is supported.

**Value**  
A concept set expression
importSummarisedResult

**Description**

Import a set of summarised results.

**Usage**

```r
importSummarisedResult(path, recursive = FALSE)
```

**Arguments**

- `path`: Path to directory with CSV files containing summarised results or to a specific CSV file with a summarised result.
- `recursive`: If TRUE and path is a directory, search for files will recurse into directories.

**Value**

A summarised result

---

insertFromSource

**Description**

Convert a table that is not a cdm_table but have the same original source to a cdm_table. This Table is not meant to be used to insert tables in the cdm, please use insertTable instead.

**Usage**

```r
insertFromSource(cdm, value)
```

**Arguments**

- `cdm`: A cdm_reference object.
- `value`: A table that shares source with the cdm_reference object.

**Value**

A table in the cdm_reference environment
insertTable  
Insert a table to a cdm object.

Description
Insert a table to a cdm object.

Usage
```r
insertTable(cdm, name, table, overwrite = TRUE, temporary = FALSE)
```

Arguments
- `cdm`: A cdm reference or the source of a cdm reference.
- `name`: Name of the table to insert.
- `table`: Table to insert to the cdm.
- `overwrite`: Whether to overwrite an existent table.
- `temporary`: Whether to create a temporary table.

Value
The cdm reference.

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1,
  gender_concept_id = 0,
  year_of_birth = 1990,
  race_concept_id = 0,
  ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1,
  person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(tables = list("person" = person,
                                     "observation_period" = observation_period),
                      cdmName = "my_example_cdm")

x <- tibble(a = 1)

cdm <- insertTable(cdm = cdm, name = "new_table", table = x)

cdm$new_table
```

listSourceTables  
List tables that can be accessed though a cdm object.

Description
List tables that can be accessed though a cdm object.

Usage
```r
listSourceTables(cdm)
```
newAchillesTable

Arguments

cdm A cdm reference or the source of a cdm reference.

Value

A character vector with the names of tables.

Description

Create an achilles table from a cdm_table.

Usage

newAchillesTable(table, version = "5.3", cast = FALSE)

Arguments

table A cdm_table.
version version of the cdm.
cast Whether to cast columns to the correct type.

Value

An achilles_table object

newCdmReference cdm_reference objects constructor

Description

cdm_reference objects constructor

Usage

newCdmReference(tables, cdmName, cdmVersion = NULL, .softValidation = FALSE)

Arguments

tables List of tables that are part of the OMOP Common Data Model reference.
cdmName Name of the cdm object.
cdmVersion Version of the cdm. Supported versions 5.3 and 5.4.
.softValidation Whether to perform a soft validation of consistency. If set to FALSE, non overlapping observation periods are ensured.
Value
A cdm_reference object.

Examples
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdmTables <- list(
  "person" = tibble(
    person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
    race_concept_id = 0, ethnicity_concept_id = 0
  ) |> 
  newCdmTable(newLocalSource(), "person"),
  "observation_period" = tibble(
    observation_period_id = 1, person_id = 1,
    observation_period_start_date = as.Date("2000-01-01"),
    observation_period_end_date = as.Date("2023-12-31"),
    period_type_concept_id = 0
  ) |> 
  newCdmTable(newLocalSource(), "observation_period")
)
cdm <- newCdmReference(tables = cdmTables, cdmName = "mock")

cdm

newCdmSource
Create a cdm source object.

Description
Create a cdm source object.

Usage
newCdmSource(src, sourceType)

Arguments
src Source to a cdm object.
sourceType Type of the source object.

Value
A validated cdm source object.
newCdmTable

Create an cdm table.

Description

Create an cdm table.

Usage

newCdmTable(table, src, name)

Arguments

table A table that is part of a cdm.
src The source of the table.
name The name of the table.

Value

A cdm_table object

newCodelist 'codelist' object constructor

Description

'codelist' object constructor

Usage

newCodelist(x)

Arguments

x A named list where each element contains a vector of concept IDs.

Value

A codelist object.
newCodelistWithDetails

'codelist' object constructor

Description
'
codelist' object constructor

Usage
newCodelistWithDetails(x)

Arguments
x A named list where each element contains a tibble with the column concept_id

Value
A codelist object.

ewCohortTable cohort_table objects constructor.

Description
cohort_table objects constructor.

Usage
newCohortTable(
table,
cohortSetRef = attr(table, "cohort_set"),
cohortAttritionRef = attr(table, "cohort_attrition"),
cohortCodelistRef = attr(table, "cohort_codelist"),
.softValidation = FALSE
)

Arguments
table cdm_table object with at least: cohort_definition_id, subject_id, cohort_start_date, cohort_end_date.
cohortSetRef Table with at least: cohort_definition_id, cohort_name
cohortAttritionRef Table with at least: cohort_definition_id, number_subjects, number_records, reason_id, reason, excluded_subjects, excluded_records.
newConceptSetExpression

cohortCodelistRef
   Table with at least: cohort_definition_id, codelist_name, and concept_id.

.softValidation
   Whether to perform a soft validation of consistency. If set to FALSE four additional checks will be performed: 1) a check that cohort end date is not before cohort start date, 2) a check that there are no missing values in required columns, 3) a check that cohort duration is all within observation period, and 4) that there are no overlapping cohort entries

Value
   A cohort_table object

Examples

person <- dplyr::tibble(
   person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
   race_concept_id = 0, ethnicity_concept_id = 0
)
observation_period <- dplyr::tibble(
   observation_period_id = 1, person_id = 1,
   observation_period_start_date = as.Date("2000-01-01"),
   observation_period_end_date = as.Date("2023-12-31"),
   period_type_concept_id = 0
)
cohort1 <- dplyr::tibble(
   cohort_definition_id = 1, subject_id = 1,
   cohort_start_date = as.Date("2020-01-01"),
   cohort_end_date = as.Date("2020-01-10")
)
cdm <- cdmFromTables(
   tables = list(
      "person" = person,
      "observation_period" = observation_period,
      "cohort1" = cohort1
   ),
   cdmName = "test"
)
cdm
cdm$cohort1 <- newCohortTable(table = cdm$cohort1)
cdm
settings(cdm$cohort1)
attrition(cdm$cohort1)
cohortCount(cdm$cohort1)
newLocalSource

Description

'conceptSetExpression' object constructor

Usage

newConceptSetExpression(x)

Arguments

x a named list of tibbles, each of which containing concept set definitions

Value

A conceptSetExpression

---

newLocalSource A new local source for the cdm

Description

A new local source for the cdm

Usage

newLocalSource()

Value

A list in the format of a cdm source

Examples

library(omopgenerics)
newLocalSource()
newOmopTable

Create an omop table from a cdm table.

Description

Create an omop table from a cdm table.

Usage

newOmopTable(table, version = "5.3", cast = FALSE)

Arguments

table : A cdm_table.
version : version of the cdm.
cast : Whether to cast columns to the correct type.

Value

An omop_table object

newSummarisedResult

'summarised_results' object constructor

Description

'summarised_results’ object constructor

Usage

newSummarisedResult(x, settings = attr(x, "settings"))

Arguments

x : Table.
settings : Settings for the summarised_result object.

Value

A summarised_result object
Examples

```r
library(dplyr)
library(omopgenerics)

x <- tibble(
  "result_id" = 1L,
  "cdm_name" = "cprd",
  "group_name" = "cohort_name",
  "group_level" = "acetaminophen",
  "strata_name" = "sex &&& age_group",
  "strata_level" = c("male &&& <40", "male &&& >=40"),
  "variable_name" = "number_subjects",
  "variable_level" = NA_character_,
  "estimate_name" = "count",
  "estimate_type" = "integer",
  "estimate_value" = c("5", "15"),
  "additional_name" = "overall",
  "additional_level" = "overall"
) |> newSummarisedResult()

x
settings(x)
summary(x)

x <- tibble(
  "result_id" = 1L,
  "cdm_name" = "cprd",
  "group_name" = "cohort_name",
  "group_level" = "acetaminophen",
  "strata_name" = "sex &&& age_group",
  "strata_level" = c("male &&& <40", "male &&& >=40"),
  "variable_name" = "number_subjects",
  "variable_level" = NA_character_,
  "estimate_name" = "count",
  "estimate_type" = "integer",
  "estimate_value" = c("5", "15"),
  "additional_name" = "overall",
  "additional_level" = "overall"
) |> newSummarisedResult(settings = tibble(
  result_id = 1L, result_type = "custom_summary", mock = TRUE, value = 5
))

x
settings(x)
summary(x)
```
omopTables

Required columns that the standard tables in the OMOP Common Data Model must have.

Description

Required columns that the standard tables in the OMOP Common Data Model must have.

Usage

omopColumns(table, onlyRequired = TRUE, version = "5.3")

Arguments

table Table to see required columns.
onlyRequired Whether to include only required fields.
version Version of the OMOP Common Data Model.

Value

Character vector with the column names

Examples

library(omopgenerics)

omopColumns("person")

omopTables

Standard tables that a cdm reference can contain in the OMOP Common Data Model.

Description

Standard tables that a cdm reference can contain in the OMOP Common Data Model.

Usage

omopTables(version = "5.3")

Arguments

version Version of the OMOP Common Data Model.

Value

Standard tables
Examples

```r
library(omopgenerics)

omopTables()
```

Participants

It returns the participants that contributed to a particular analysis

Description

It returns the participants that contributed to a particular analysis

Usage

```r
participants(result, ...)
```

Arguments

- `result`: A result object.
- `...`: ...

Value

Table with the participants

print.cdm_reference

Print a CDM reference object

Description

Print a CDM reference object

Usage

```r
## 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
Examples

library(omopgenerics)

cdm <- cdmFromTables(
  tables = list(
    "person" = dplyr::tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = dplyr::tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    ),
  ),
  cdmName = "mock"
)

print(cdm)

----------------------------------------
print.codelist  Print a codelist
----------------------------------------

Description

Print a codelist

Usage

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

Arguments

x  A codelist

...  Included for compatibility with generic. Not used.

Value

Invisibly returns the input

Examples

codes <- list("disease X" = c(1, 2, 3), "disease Y" = c(4, 5))
codes <- newCodelist(codes)
print(codes)
### print.codelist_with_details

*Print a codelist with details*

**Description**

Print a codelist with details

**Usage**

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

**Arguments**

- `x` A codelist with details
- `...` Included for compatibility with generic. Not used.

**Value**

Invisibly returns the input

**Examples**

```r
codes <- list("disease X" = dplyr::tibble(concept_id = c(1, 2, 3),
other= c("a", "b", "c")))
codes <- newCodelistWithDetails(codes)
print(codes)
```

### print.conceptSetExpression

*Print a concept set expression*

**Description**

Print a concept set expression

**Usage**

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

**Arguments**

- `x` A concept set expression
- `...` Included for compatibility with generic. Not used.
**Value**

Invisibly returns the input

**Examples**

```r
asthma_cs <- list("asthma_narrow" = dplyr::tibble(
  "concept_id" = 1,
  "excluded" = FALSE,
  "descendants" = TRUE,
  "mapped" = FALSE
),
  "asthma_broad" = dplyr::tibble(
  "concept_id" = c(1,2),
  "excluded" = FALSE,
  "descendants" = TRUE,
  "mapped" = FALSE
))
asthma_cs <- newConceptSetExpression(asthma_cs)
print(asthma_cs)
```

---

**readSourceTable**  
*Read a table from the cdm_source and add it to the cdm.*

**Description**

Read a table from the cdm_source and add it to the cdm.

**Usage**

```r
readSourceTable(cdm, name)
```

**Arguments**

- `cdm`  
  A cdm reference.

- `name`  
  Name of a table to read in the cdm_source space.

**Value**

A cdm_reference with new table.
recordCohortAttrition  

Update cohort attrition.

Description
Update cohort attrition.

Usage
recordCohortAttrition(cohort, reason, cohortId = NULL)

Arguments
- cohort: A cohort_table object.
- reason: A character string.
- cohortId: Cohort definition id of the cohort to update attrition. If NULL all cohort_definition_id are updated.

Value
cohort_table with updated attrition.

Examples
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cohort <- tibble(
  cohort_definition_id = c(1, 1, 1, 2),
  subject_id = 1,
  cohort_start_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01")),
  cohort_end_date = as.Date(c("2020-01-01", "2021-01-01", "2022-01-01", "2022-01-01"))
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmdName = "my_example_cdm",
  cohortTables = list("cohort1" = cohort)
)
resultColumns

resultColumns

---

resultColumns

Required columns that the result tables must have.

Description

Required columns that the result tables must have.

Usage

resultColumns(table = "summarised_result")

Arguments

table Table to see required columns.

Value

Required columns

Examples

library(omopgenerics)

resultColumns()
settings  
*Get settings from an object.*

**Description**

Get settings from an object.

**Usage**

```r
settings(x)
```

**Arguments**

- `x`  
  Object

**Value**

A table with the settings of the object.

---

settings.cohort_table  
*Get cohort settings from a cohort_table object.*

**Description**

Get cohort settings from a cohort_table object.

**Usage**

```r
# S3 method for class 'cohort_table'
settings(x)
```

**Arguments**

- `x`  
  A cohort_table object.

**Value**

A table with the details of the cohort settings.
**Examples**

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0)

cohort <- tibble(
  cohort_definition_id = 1,
  subject_id = 1,
  cohort_start_date = as.Date("2010-01-01"),
  cohort_end_date = as.Date("2012-01-01")
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test",
  cohortTables = list("my_cohort" = cohort))

settings(cdm$my_cohort)

cdm$my_cohort <- cdm$my_cohort |> 
  newCohortTable(cohortSetRef = tibble(
    cohort_definition_id = 1, cohort_name = "new_name")
)

settings(cdm$my_cohort)
```

---

**settings.summarised_result**

*Get settings from a summarised_result object.*

**Description**

Get settings from a summarised_result object.

**Usage**

```r
## S3 method for class 'summarised_result'
settings(x)
```
sourceType

Get the source type of an object.

Description

Get the source type of an object.

Usage

sourceType(x)
Arguments

- `x` Object to know the source type.

Value

A character vector that defines the type of `cdm_source`.

Description

Summary a cdm reference

Usage

```r
## S3 method for class 'cdm_reference'
summary(object, ...)
```

Arguments

- `object` A cdm reference object.
- `...` For compatibility (not used).

Value

A `summarised_result` object with a summary of the data contained in the cdm.

Examples

```r
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)
observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test"
)

summary(cdm)
```
summary.cohort_table  Summary a generated cohort set

Description
Summary a generated cohort set

Usage
```r
## S3 method for class 'cohort_table'
summary(object, ...)
```

Arguments
- `object` : A generated cohort set object.
- `...` : For compatibility (not used).

Value
A summarised_result object with a summary of a cohort_table.

Examples
```r
library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)
observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)
cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test",
  cohortTables = list("cohort1" = tibble(
    cohort_definition_id = 1,
    subject_id = 1,
    cohort_start_date = as.Date("2010-01-01"),
    cohort_end_date = as.Date("2010-01-05")
  ))
)

summary(cdm$cohort1)
```
Summary a summarised_result

Description
Summary a summarised_result

Usage
## S3 method for class 'summarised_result'
summary(object, ...)

Arguments

object
A summarised_result object.

...    
For compatibility (not used).

Value
A summary of the result_types contained in a summarised_result object.

Examples

library(dplyr, warn.conflicts = FALSE)

person <- tibble(
  person_id = 1, gender_concept_id = 0, year_of_birth = 1990,
  race_concept_id = 0, ethnicity_concept_id = 0
)

observation_period <- tibble(
  observation_period_id = 1, person_id = 1,
  observation_period_start_date = as.Date("2000-01-01"),
  observation_period_end_date = as.Date("2023-12-31"),
  period_type_concept_id = 0
)

cdm <- cdmFromTables(
  tables = list("person" = person, "observation_period" = observation_period),
  cdmName = "test"
)

result <- summary(cdm)

summary(result)
 suppress (result, minCellCount = 5)

Arguments

result           Result object
minCellCount     Minimum count of records to report results.

Value

Table with suppressed counts

== suppress.summarised_result ==

 suppress.summarised_result

Function to suppress counts in result objects

Description

Function to suppress counts in result objects

Usage

## S3 method for class 'summarised_result'
suppress(result, minCellCount = 5)

Arguments

result           summarised_result object.
minCellCount     Minimum count of records to report results.

Value

summated_result with suppressed counts.
Examples

```r
library(dplyr, warn.conflicts = FALSE)
library(omopgenerics)

my_result <- tibble(
  "result_id" = "1",
  "cdm_name" = "mock",
  "result_type" = "summarised_characteristics",
  "package_name" = "omopgenerics",
  "package_version" = as.character(utils::packageVersion("omopgenerics")),
  "group_name" = "overall",
  "group_level" = "overall",
  "strata_name" = c(rep("overall", 6), rep("sex", 3)),
  "strata_level" = c(rep("overall", 6), "male", "female", "female"),
  "variable_name" = c("number records", "age_group", "age_group",
                     "age_group", "age_group", "my_variable", "number records", "age_group",
                     "age_group"),
  "variable_level" = c(NA, "<50", "<50", ">=50", ">=50", NA, NA,
                     "<50", "<50"),
  "estimate_name" = c("count", "count", "percentage", "count", "percentage",
                     "random", "count", "count", "percentage"),
  "estimate_type" = c("integer", "integer", "percentage", "integer",
                     "percentage", "numeric", "integer", "integer", "percentage"),
  "estimate_value" = c("10", "5", "50", "3", "30", "1", "3", "12", "6"),
  "additional_name" = "overall",
  "additional_level" = "overall"
)
my_result <- newSummarisedResult(my_result)
my_result |> glimpse()
my_result <- suppress(my_result, minCellCount = 5)
my_result |> glimpse()
```

tableName

Get the table name of a cdm_table.

Description

Get the table name of a cdm_table.

Usage

tableName(table)

Arguments

table A cdm_table.
Value

A character with the name.

Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

tableName(cdm$person)
```

---

`tableSource`  
*Get the table source of a cdm_table.*

Description

Get the table source of a cdm_table.

Usage

`tableSource(table)`

Arguments

`table`  
A cdm_table.

Value

A cdm_source object.
Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3),
      gender_concept_id = 0,
      year_of_birth = 1990,
      race_concept_id = 0,
      ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3,
      person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

tableSource(cdm$person)
```

---

tmpPrefix  
Create a temporary prefix for tables, that contains a unique prefix that starts with tmp.

Description

Create a temporary prefix for tables, that contains a unique prefix that starts with tmp.

Usage

```r
tmpPrefix()
```

Value

A temporary prefix.

Examples

```r
library(omopgenerics)
tmpPrefix()
```
### toSnakeCase

*Convert a character vector to snake case*

**Description**

Convert a character vector to snake case

**Usage**

```r
toSnakeCase(x)
```

**Arguments**

- `x` Character vector to convert

**Value**

A snake_case vector

**Examples**

```r
toSnakeCase("myVariable")
toSnakeCase(c("cohort1", "Cohort22b"))
```

---

### uniqueId

*Get a unique Identifier with a certain number of characters and a prefix.*

**Description**

Get a unique Identifier with a certain number of characters and a prefix.

**Usage**

```r
uniqueId(n = 1, exclude = character(), nChar = 3, prefix = "id_")
```

**Arguments**

- `n` Number of identifiers.
- `exclude` Columns to exclude.
- `nChar` Number of characters.
- `prefix` A prefix for the identifiers.

**Value**

A character vector with n unique identifiers.
uniqueTableName

Create a unique table name

Description

Create a unique table name

Usage

uniqueTableName(prefix = "")

Arguments

prefix Prefix for the table names.

Value

A string that can be used as a dbplyr temp table name

Examples

library(omopgenerics)
uniqueTableName()

validateAgeGroupArgument

validateAgeGroupArgument

Description

validateAgeGroupArgument

Usage

validateAgeGroupArgument(
  ageGroup,
  multipleAgeGroup = TRUE,
  overlap = FALSE,
  null = TRUE,
  call = parent.frame()
)
validateCdmArgument

Arguments

- **ageGroup**: age group in a list
- **multipleAgeGroup**: allow multiple age group
- **overlap**: allow overlapping ageGroup
- **null**: null age group allowed true or false
- **call**: parent frame

Value

validate ageGroup

---

validateCdmArgument validateCdmArgument

Description

validateCdmArgument

Usage

validateCdmArgument(
  cdm,
  checkOverlapObservation = FALSE,
  checkStartBeforeEndObservation = FALSE,
  checkPlausibleObservationDates = FALSE,
  validation = "error",
  call = parent.frame()
)

Arguments

- **cdm**: A cdm_reference object
- **checkOverlapObservation**: TRUE to perform check on no overlap observation period
- **checkStartBeforeEndObservation**: TRUE to perform check on correct observational start and end date
- **checkPlausibleObservationDates**: TRUE to perform check that there are no implausible observation period start dates (before 1800-01-01) or end dates (after the current date)
- **validation**: How to perform validation: "error", "warning".
- **call**: A call argument to pass to cli functions.

Value

A cdm_reference object
validateCohortArgument

Validate a cohort table input.

Description

Validate a cohort table input.

Usage

validateCohortArgument(
  cohort,
  checkEndAfterStart = FALSE,
  checkOverlappingEntries = FALSE,
  checkMissingValues = FALSE,
  checkInObservation = FALSE,
  validation = "error",
  call = parent.frame()
)

Arguments

cohort Object to be validated as a valid cohort input.

checkEndAfterStart If TRUE a check that all cohort end dates come on or after cohort start date will be performed.

checkOverlappingEntries If TRUE a check that no individuals have overlapping cohort entries will be performed.

checkMissingValues If TRUE a check that there are no missing values in required fields will be performed.

checkInObservation If TRUE a check that cohort entries are within the individuals observation periods will be performed.

validation How to perform validation: "error", "warning".

call A call argument to pass to cli functions.
validateCohortIdArgument

Validate cohortId argument.

Description

Validate cohortId argument.

Usage

validateCohortIdArgument(
    cohortId,
    cohort,
    validation = "error",
    call = parent.frame()
)

Arguments

    cohortId       A cohortId vector to be validated.
    cohort         A cohort_table object.
    validation     How to perform validation: "error", "warning".
    call           A call argument to pass to cli functions.

validateConceptSetArgument

Validate conceptSet argument.

Description

Validate conceptSet argument.

Usage

validateConceptSetArgument(
    conceptSet,
    cdm = NULL,
    validation = "error",
    call = parent.frame()
)
validateNameArgument

Arguments

- **conceptSet**: It can be either a named list of concepts or a codelist, codelist_with_details or conceptSetExpression object.
- **cdm**: A cdm_reference object, needed if a conceptSetExpression is provided.
- **validation**: How to perform validation: "error", "warning".
- **call**: A call argument to pass to cli functions.

validateNameArgument Validate name argument.

Description

Validate name argument.

Usage

```r
validateNameArgument(
  name,
  cdm = NULL,
  validation = "error",
  null = FALSE,
  call = parent.frame()
)
```

Arguments

- **name**: Name of a new table to be added to a cdm object.
- **cdm**: A cdm_reference object. It will check if a table named name already exists in the cdm.
- **validation**: How to perform validation: "error", "warning".
- **null**: If TRUE, name can be NULL.
- **call**: A call argument to pass to cli functions.
validateNameLevel | Validate if two columns are valid Name-Level pair.

**Description**

Validate if two columns are valid Name-Level pair.

**Usage**

```
validateNameLevel(x, nameColumn, levelColumn, sep = " &&& ", warn = FALSE)
```

**Arguments**

- `x` A tibble.
- `nameColumn` Column name of the name.
- `levelColumn` Column name of the level.
- `sep` Separation pattern.
- `warn` Whether to throw a warning (TRUE) or an error (FALSE).

---

validateResultArgument | validateResultArgument

**Description**

validateResultArgument

**Usage**

```
validateResultArgument(result, validation = "error", call = parent.frame())
```

**Arguments**

- `result` summarise result object to validate
- `validation` message to return
- `call` parent.frame

**Value**

summarise result object
validateWindowArgument

Description

validateWindowArgument

Usage

validateWindowArgument(window, snakeCase = TRUE, call = parent.frame())

Arguments

- **window**: time window
- **snakeCase**: return default window name in snake case if TRUE
- **call**: A call argument to pass to cli functions.

Value

- time window

[[.cdm_reference

Subset a cdm reference object.

Description

Subset a cdm reference object.

Usage

## S3 method for class 'cdm_reference'
x[[name]]

Arguments

- **x**: A cdm reference
- **name**: The name or index of the table to extract from the cdm object.

Value

- A single cdm table reference
Examples

```r
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    )
  ),
  cdmName = "mock"
)

cdm[["person"]]
```

Description

Assign a table to a cdm reference.

Usage

```r
## S3 replacement method for class 'cdm_reference'
cdm[[name]] <- value
```

Arguments

- `cdm` A cdm reference.
- `name` Name where to assign the new table.
- `value` Table with the same source than the cdm object.

Value

The cdm reference.
Description

Subset a cdm reference object.

Usage

## S3 method for class 'cdm_reference'

x$name

Arguments

x

A cdm reference.

name

The name of the table to extract from the cdm object.

Value

A single cdm table reference

Examples

library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- cdmFromTables(
  tables = list(
    "person" = tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    ),
  ),
  cdmName = "mock"
)

cdm$person
$<-.cdm_reference

Assign an table to a cdm reference.

Description

Assign an table to a cdm reference.

Usage

```r
## S3 replacement method for class 'cdm_reference'
cdm$name <- value
```

Arguments

- **cdm**: A cdm reference.
- **name**: Name where to assign the new table.
- **value**: Table with the same source than the cdm object.

Value

The cdm reference.

Examples

```r
library(omopgenerics)

cdm <- cdmFromTables(
  tables = list(
    "person" = dplyr::tibble(
      person_id = c(1, 2, 3), gender_concept_id = 0, year_of_birth = 1990,
      race_concept_id = 0, ethnicity_concept_id = 0
    ),
    "observation_period" = dplyr::tibble(
      observation_period_id = 1:3, person_id = 1:3,
      observation_period_start_date = as.Date("2000-01-01"),
      observation_period_end_date = as.Date("2023-12-31"),
      period_type_concept_id = 0
    ),
  ),
  cdmName = "mock"
)

cdm$person
```
Index

[[.cdm_reference, 75
[[<- .cdm_reference, 76
$.cdm_reference, 77
$<- .cdm_reference, 78

achillesColumns, 4
achillesTables, 5
assertCharacter, 5
assertChoice, 6
assertClass, 7
assertDate, 7
assertList, 8
assertLogical, 9
assertNumeric, 10
assertTable, 11
assertTrue, 12
attrition, 12
attrition.cohort_table, 13

bind, 14
bind.cohort_table, 14
bind.summarised_result, 15

cdmFromTables, 16
cdmName, 17
cdmReference, 18
cdmSelect, 19
cdmSource, 20
cdmSourceType, 21
cdmTableFromSource, 22
cdmVersion, 22
checkCohortRequirements, 23
cohortCodelist, 24
cohortColumns, 25
cohortCount, 26
cohortTables, 27
collect.cdm_reference, 28
collect.cohort_table, 29
compute.cdm_table, 29

dropSourceTable, 30
dropTable, 30
emptyAchillesTable, 31
emptyCdmReference, 32
emptyCodelist, 33
emptyCodelistWithDetails, 33
emptyCohortTable, 34
emptyOmopTable, 35
emptySummarisedResult, 36
estimateTypeChoices, 36
exportCodelist, 37
exportConceptSetExpression, 37
exportSummarisedResult, 38

g getCohortId, 38
g getCohortName, 39
g getPersonIdentifier, 39

importCodelist, 40
importConceptSetExpression, 40
insertSummarisedResult, 41
insertFromSource, 41
insertTable, 42

listSourceTables, 42

newAchillesTable, 43
newCdmReference, 43
newCdmSource, 44
newCdmTable, 45
newCodelist, 45
newCodelistWithDetails, 46
newCohortTable, 46
newConceptSetExpression, 47
newLocalSource, 48
newOmopTable, 49

newSummarisedResult, 49

omopColumns, 50
omopTables, 51
participants, 52
print.cdm_reference, 52
print.codelist, 53
print.codelist_with_details, 54
print.conceptSetExpression, 54
readSourceTable, 55
recordCohortAttrition, 56
resultColumns, 57

settings, 58
settings.cohort_table, 58
settings.summarised_result, 59
sourceType, 60
summary.cdm_reference, 61
summary.cohort_table, 62
summary.summarised_result, 63
suppress, 64
suppress.summarised_result, 64

tableName, 65
tableSource, 66
tmpPrefix, 67
toSnakeCase, 68

uniqueId, 68
uniqueTableName, 69

validateAgeGroupArgument, 69
validateCdmArgument, 70
validateCohortArgument, 71
validateCohortIdArgument, 72
validateConceptSetArgument, 72
validateNameArgument, 73
validateNameLevel, 74
validateResultArgument, 74
validateWindowArgument, 75