Package 'ReDaMoR'

May 19, 2026

Type Package

Title Relational Data Modeler

Version 1.0.0

Maintainer Patrice Godard <patrice.godard@gmail.com>

Description The aim of this package is to manipulate relational
data models in R.
It provides functions to create, modify and export data models
in json format.
It also allows importing models created
with 'MySQL Workbench' (<https://www.mysql.com/products/workbench/>).
These functions are accessible through a graphical user
interface made with 'shiny'.
Constraints such as types, keys, uniqueness and mandatory fields are
automatically checked and corrected when editing a model.
Finally, real data can be confronted to a model to check their compatibility.

URL https://patzaw.github.io/ReDaMoR/,
https://github.com/patzaw/ReDaMoR/

BugReports https://github.com/patzaw/ReDaMoR/issues

Depends R (>= 4.1), dplyr, visNetwork

Imports readr, shiny, shinyjs, jsonlite, DT, colourpicker, rintrojs,
markdown, rstudioapi, crayon, utils, graphics, stats, Matrix

Suggests knitr, rmarkdown, igraph

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Patrice Godard [aut, cre, cph],
Kai Lin [ctb]

Repository CRAN

Date/Publication 2026-05-19 05:10:30 UTC

Contents

`add_field` *Add a field to a table in a RelDataModel*

Description

Add a field to a table in a RelDataModel

Usage

```
add_field(x, tableName, name, type, nullable, unique, comment)
```

Arguments

`x`	a RelDataModel
`tableName`	the name of the table to modify (a single character)
`name`	the name of the field to add (a single character)
`type`	the type of the field (a single character)
`nullable`	if the field is nullable (a single logical)
`unique`	if the values are unique (a single logical)
`comment`	a description (a single character)

Value

A RelDataModel

add_foreign_key *Add a foreign key between two tables*

Description

Add a foreign key between two tables

Usage

```
add_foreign_key(
  x,
  fromTable,
  fromFields,
  toTable,
  toFields,
  fmin = 0L,
  fmax = -1L,
  tmin = 1L,
  tmax = 1L
)
```

Arguments

x	a RelDataModel
fromTable	the name of the referencing table
fromFields	the name of the referencing fields
toTable	the name of the referenced table
toFields	the names of the referenced fields
fmin	from minimum cardinality (default: 0L)
fmax	from maximum cardinality (default: -1L ==> Infinite)
tmin	to minimum cardinality (default: 1L)
tmax	to maximum cardinality (default: 1L)

Value

A RelDataModel

add_index — *Add an index to a table in a RelDataModel*

Description

Add an index to a table in a RelDataModel

Usage

```
add_index(x, tableName, fieldNames, unique)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
fieldNames	the names of the fields to include in the index
unique	a logical indicating if the indexed values are unique

Value

A RelDataModel

add_table — *Add a table to a RelDataModel*

Description

Add a table to a RelDataModel

Usage

```
add_table(x, newTable)
```

Arguments

x	a RelDataModel
newTable	the name of the new table or a RelTableModel

Value

A RelDataModel

as_type — *Convert an object into a specific type*

Description

Convert an object into a specific type

Usage

```
as_type(x, type)
```

Arguments

x	an object to convert
type	the targeted type

auto_layout — *Pre-compute RelDataModel layout when missing any x or y table position*

Description

Pre-compute RelDataModel layout when missing any x or y table position

Usage

```
auto_layout(
  x,
  layout = "layout_nicely",
  lengthMultiplier = 40 * length(x),
  force = FALSE
)
```

Arguments

x	a RelDataModel
layout	character name of igraph layout function to use (Default: "layout_nicely").
lengthMultiplier	a numeric value to scale x and y coordinate (default: 40*length(x))
force	if TRUE autolayout even if all tables have coordinates (default: FALSE)

Value

A RelDataModel

c.RelDataModel *Merge RelDataModel objects*

Description

Merge RelDataModel objects

Usage

```
## S3 method for class 'RelDataModel'
c(..., checkFK = TRUE)
```

Arguments

...	RelDataModel objects
checkFK	a logical indicating if foreign keys should be checked (default: TRUE)

Value

A RelDataModel objects

check_foreign_keys *Check the availability of foreign keys*

Description

Check the availability of foreign keys

Usage

```
check_foreign_keys(x)
```

Arguments

x	a RelDataModel object

Value

Nothing. The function throws an error if there is an issue with foreign keys.

check_types *Check if a set of types is supported*

Description

Check if a set of types is supported

Usage

```
check_types(x)
```

Arguments

x	a character vector of types to be checked

clean_autosaved_RelDataModels *Remove all autosaved RelDataModel*

Description

Remove all autosaved RelDataModel

Usage

```
clean_autosaved_RelDataModels()
```

col_types *Get the types of the columns of a RelTableModel object*

Description

Get the types of the columns of a RelTableModel object

Usage

```
col_types(x)
```

Arguments

x	a RelTableModel object

Value

A col_spec object with the type of each column

confront_data *Confront a RelDataModel to actual data*

Description

Confront a RelDataModel to actual data

Usage

```
confront_data(
  x,
  data = list(),
  paths = NULL,
  returnData = FALSE,
  verbose = TRUE,
  n_max = Inf,
  checks = if (n_max == Inf) {
     c("unique", "not nullable", "foreign keys")
 } else
    {
     as.character()
 },
  delim = "\t",
  ...
)
```

Arguments

x	a RelDataModel
data	a list of data frames to be confronted with the model.
paths	a character vector with file paths taken into account if the data is empty. The file basename without extension will be considered as the table name.
returnData	a logical indicating if the data should be returned with the report (default: FALSE).
verbose	a single logical value indicating if some process information should be displayed (default: TRUE)
n_max	maximum number of records to read (default: Inf).
checks	a character vector with the name of optional checks to be done (Default: if n_max==Inf ==> all of them c("unique", "not nullable", "foreign keys"), else ==> none)
delim	single character used to separate fields within a record (default: "\t")
...	supplementary parameters for the readr::read_delim function.

Value

A report as a list

Examples

```
## Read the model ----
hpo_model <- read_json_data_model(
  system.file("examples/HPO-model.json", package = "ReDaMoR")
)
## Confront to data ----
confrontation_report <- confront_data(
  hpo_model,
  path = list.files(
    system.file("examples/HPO-subset", package = "ReDaMoR"),
    full.names = TRUE
  ),
  returnData = TRUE
)
```

`confront_table_data` *Confront a RelTableModel to actual data*

Description

Confront a RelTableModel to actual data

Usage

```
confront_table_data(x, d, checks = c("unique", "not nullable"))
```

Arguments

x	a RelTableModel
d	a data frame or a matrix for matrix model
checks	a character vector with the name of optional checks to be done (Default: all of them c("unique", "not nullable"))

Value

A report as a list

conv_type_ref *Convert a set of types from or to R supported types*

Description

Convert a set of types from or to R supported types

Usage

```
conv_type_ref(x, from = NULL, to = NULL, ignore.case = TRUE)
```

Arguments

x	a character vector of types to be converted. If from is not null, x should be a set of valid types in the from reference. If to is not null, x should be a set of supported R types (SUPPTYPES).
from	a character vector of length one: the type reference (list_type_ref) of x
to	a character vector of length one: the targeted type reference (list_type_ref)
ignore.case	should case be ignored when converting 'from" type reference (default: TRUE)

Details

Only from XOR to should be set

copy_fields *Copy fields from one table to another in a RelDataModel*

Description

Copy fields from one table to another in a RelDataModel

Usage

```
copy_fields(x, from, to, fields)
```

Arguments

x	a RelDataModel
from	the name of the table from which the fields are taken
to	the name of the table to which the fields are copied
fields	the names of the fields to copy

Value

A RelDataModel

correct_constraints *Correct the constraints of a table to make them consistent*

Description

Correct the constraints of a table to make them consistent

Usage

```
correct_constraints(x, createPKIndex = FALSE)
```

Arguments

x	a RelTableModel object
createPKIndex	should an index be create for the primary key (default: FALSE)

df_to_model *Create a RelDataModel object from column names of data frames*

Description

Create a RelDataModel object from column names of data frames

Usage

```
df_to_model(..., list = character(), pos = -1, envir = as.environment(pos))
```

Arguments

...	the data frame objects, as names (unquoted) or character strings (quoted)
list	a character vector naming data frame objects
pos	where to get the objects. By default, uses the current environment. See 'details' for other possibilities.
envir	the environment to use. See 'details'.

Details

The pos argument can specify the environment from which to get the objects in any of several ways: as an integer (the position in the search list); as the character string name of an element in the search list; or as an environment. The envir argument is an alternative way to specify an environment, but is primarily there for back compatibility.

Value

A RelDataModel object.

Examples

```
## Read data files ----
to_read <- list.files(
  system.file("examples/HPO-subset", package = "ReDaMoR"),
  full.names = TRUE
)
hpo_tables <- list()
for (f in to_read) {
  hpo_tables[[sub("[.]txt$", "", basename(f))]] <- readr::read_tsv(f)
}
## Build the model from a list of data frames ----
new_model <- df_to_model(
  list = names(hpo_tables),
  envir = as.environment(hpo_tables)
)
## Guess constraints and auto layout ----
new_model <- guess_constraints(new_model, data = hpo_tables) |>
  auto_layout(lengthMultiplier = 250)

## Plot the model ----
new_model |>
  plot()
```

`format.RelTableModel` *Format a RelTableModel object for printing*

Description

Format a RelTableModel object for printing

Usage

```
## S3 method for class 'RelTableModel'
format(x, ...)
```

Arguments

x	a RelTableModel object
...	for generics compatibility (not used)

Value

A single character

format_confrontation_report

Format confrontation report for printing in console

Description

Format confrontation report for printing in console

Usage

```
format_confrontation_report(cr, title = "Model")
```

Arguments

cr	the confrontation report from confront_data
title	a character with a single value corresponding to the report title (e.g. database/model name)

Examples

```
## Read the model ----
hpo_from_sql <- read_SQL_data_model(
  system.file("examples/HPO-model.sql", package = "ReDaMoR")
)
## Confront to data ----
confrontation_report <- confront_data(
  hpo_from_sql,
  path = list.files(
    system.file("examples/HPO-subset", package = "ReDaMoR"),
    full.names = TRUE
  ),
  verbose = FALSE,
  returnData = TRUE
)
## Show the report in console ----
format_confrontation_report(confrontation_report) |> cat()
## Format the report using markdown ----
format_confrontation_report_md(confrontation_report) |> cat()
```

format_confrontation_report_md

Format confrontation report in markdown format

Description

Format confrontation report in markdown format

Usage

```
format_confrontation_report_md(
  cr,
  title = "Model",
  level = 0,
  numbered = TRUE,
  bgSuccess = "green",
  txSuccess = "black",
  bgFailure = "red",
  txFailure = "white",
  bgMessage = "#FFBB33",
  txMessage = "white"
)
```

Arguments

cr	the confrontation report from confront_data
title	a character with a single value corresponding to the report
level	rmarkdown level in document hierarchy (default:0 ==> highest). It should be an integer between 0 and 4.
numbered	a logical. If TRUE (default) the sections are part of document numbering.
bgSuccess	background color for SUCCESS
txSuccess	text color for SUCCESS
bgFailure	background color for FAILURE
txFailure	text color for FAILURE
bgMessage	background color for a warning message
txMessage	text color for a warning message

Examples

```
## Read the model ----
hpo_from_sql <- read_SQL_data_model(
  system.file("examples/HPO-model.sql", package = "ReDaMoR")
)
## Confront to data ----
confrontation_report <- confront_data(
  hpo_from_sql,
  path = list.files(
    system.file("examples/HPO-subset", package = "ReDaMoR"),
    full.names = TRUE
  ),
  verbose = FALSE,
  returnData = TRUE
)
## Show the report in console ----
format_confrontation_report(confrontation_report) |> cat()
## Format the report using markdown ----
format_confrontation_report_md(confrontation_report) |> cat()
```

`fromDBM` *Convert a list of 5 normalized tibbles in a RelDataModel object*

Description

Convert a list of 5 normalized tibbles in a RelDataModel object

Usage

```
fromDBM(dbm)
```

Arguments

`dbm` a list with the following tibbles:

- **tables**: The tables in the model with the following information
  - **name**: the name of the table
  - **x**: the x coordinate of the table in the model drawing (NA ==> position undefined)
  - **y**: the y coordinate of the table in the model drawing (NA ==> position undefined)
  - **color**: the color of the table in the model drawing (NA ==> undefined)
  - **comment**: comment about the table
- **fields**: The fields in the model with the following information
  - **name**: the name of the field
  - **type**: the type of the field
  - **nullable**: a logical indicating if the field can be null
  - **comment**: comment about the field
  - **table**: the name of the table to which the field belongs
- **primaryKeys**: The primary keys in the model with the following information
  - **table**: the name of the relevant table
  - **field**: the name of the field participating to the primary key
- **foreignKeys**: The foreign keys in the model with the following information
  - **table**: the name of the referring table
  - **fki**: the identifier of the foreign key (by referring table)
  - **field**: the name of the referring field
  - **refTable**: the name of the referred table
  - **refField**: the name of the referred field
- **indexes**: The indexes in the model with the following information
  - **table**: the name of the relevant table
  - **idx**: the identifier of the index (by table)
  - **field**: the name of the field participating to the index
  - **unique**: a logical indicating if the field is unique

Value

A RelDataModel object

`get_foreign_keys` *Get a foreign key table from an object*

Description

Get a foreign key table from an object

Usage

```
get_foreign_keys(x)
```

Arguments

x	a RelTableModel or a RelDataModel

Value

A tibble with the following fields:

- from: the origin of the key
- ff: the key fields in from
- to: the target of the key
- tf: the key fields in to
- fmin: minimum cardinality of from
- fmax: maximum cardinality of from
- tmin: minimum cardinality of to
- tmax: maximum cardinality of to

`get_foreign_keys.RelDataModel`
Get foreign keys in RelDataModel

Description

Get foreign keys in RelDataModel

Usage

```
## S3 method for class 'RelDataModel'
get_foreign_keys(x)
```

Arguments

x	a RelDataModel

Value

A tibble with the following fields:

- from: the origin of the key
- ff: the key fields in from
- to: the target of the key
- tf: the key fields in to
- fmin: minimum cardinality of from
- fmax: maximum cardinality of from
- tmin: minimum cardinality of to
- tmax: maximum cardinality of to

---

get_foreign_keys.RelTableModel

Get foreign keys from RelTableModel

Description

Get foreign keys from RelTableModel

Usage

```
## S3 method for class 'RelTableModel'
get_foreign_keys(x)
```

Arguments

x	a RelTableModel

Value

A tibble with the following fields:

- from: the origin of the key
- ff: the key fields in from
- to: the target of the key
- tf: the key fields in to
- fmin: minimum cardinality of from
- fmax: maximum cardinality of from
- tmin: minimum cardinality of to
- tmax: maximum cardinality of to

guess_constraints *Guess RelDataModel constraints based on the provided or existing tables*

Description

Guess RelDataModel constraints based on the provided or existing tables

Usage

```
guess_constraints(
  x,
  data = NULL,
  env = parent.frame(n = 1),
  constraints = c("unique", "not nullable", "foreign keys")
)
```

Arguments

x	a RelDataModel
data	a named list of tables. All names of x should exist in data. If NULL, the data are taken from env.
env	the R environment in which to find the tables
constraints	the type of constraints to guess

Details

The guessed constraints should be carefully review, especially the foreign keys.

Complex foreign keys involving multiple fields are not guessed.

Value

A RelDataModel

Examples

```
## Read data files ----
to_read <- list.files(
  system.file("examples/HPO-subset", package = "ReDaMoR"),
  full.names = TRUE
)
hpo_tables <- list()
for (f in to_read) {
  hpo_tables[[sub("[.]txt$", "", basename(f))]] <- readr::read_tsv(f)
}
## Build the model from a list of data frames ----
new_model <- df_to_model(
```

```
  list = names(hpo_tables),
  envir = as.environment(hpo_tables)
)
## Guess constraints and auto layout ----
new_model <- guess_constraints(new_model, data = hpo_tables) |>
  auto_layout(lengthMultiplier = 250)

## Plot the model ----
new_model |>
  plot()
```

`identical_RelDataModel` *Check if two RelDataModel are identical*

Description

Check if two RelDataModel are identical

Usage

```
identical_RelDataModel(x, y, ...)
```

Arguments

x	a RelDataModel
y	a RelDataModel
...	additional parameters for `identical_RelTableModel()`

Value

A logical: TRUE if the 2 models are identical

`identical_RelTableModel` *Check if two RelTableModel are identical*

Description

Check if two RelTableModel are identical

Usage

```
identical_RelTableModel(x, y, includeDisplay = TRUE)
```

Arguments

x	a RelTableModel
y	a RelTableModel
includeDisplay	a single logical (default: TRUE) indicating if the display should be included in the comparison

Value

A logical: TRUE if the 2 models are identical

---

index_table *List indexes of a RelTableModel object*

Description

List indexes of a RelTableModel object

Usage

```
index_table(x)
```

Arguments

x	a RelTableModel object

Value

A tibble with the following columns:

- **index**: an integer corresponding to the index number
- **field**: a character corresponding to field belonging to the index
- **unique**: a logical indicating the uniqueness of the field

is.MatrixModel *Check if the object is a RelTableModel matrix object*

Description

A matrix model is a special RelTableModel object with 3 and only 3 fields: 2 of types 'row' and 'column' and the 3rd of your choice.

Usage

```
is.MatrixModel(x)
```

Arguments

x	any object

Value

A single logical: TRUE if x is a RelTableModel matrix object

is.RelDataModel *Check if the object is a RelDataModel object*

Description

Check if the object is a RelDataModel object

Usage

```
is.RelDataModel(x)
```

Arguments

x	any object

Value

A single logical: TRUE if x is a RelDataModel object

is.RelTableModel *Check if the object is a RelTableModel object*

Description

Check if the object is a RelTableModel object

Usage

```
is.RelTableModel(x)
```

Arguments

x	any object

Value

A single logical: TRUE if x is a RelTableModel object

is_MM *Identify if a file is in MatrixMarket text format*

Description

Identify if a file is in MatrixMarket text format

Usage

```
is_MM(file)
```

Arguments

file	the file to read

Value

A logical. If FALSE, the first line of the file is returned as an attribute named "r1": `attr(is_MM, "r1")`

lengths *Lengths of object elements*

Description

Lengths of object elements

Usage

```
lengths(x, use.names = TRUE)
```

Arguments

x	an object. If there is no method implemented for this object, the base::lengths() function is used.
use.names	logical indicating if the result should inherit the names from x.

Value

A non-negative integer of length length(x), except when any element has a length of more than 2^31 - 1 elements, when it returns a double vector. When use.names is true, the names are taken from the names on x, if any.

See Also

base::lengths()

list_autosaved_RelDataModel *List autosaved RelDataModel*

Description

List autosaved RelDataModel

Usage

```
list_autosaved_RelDataModel()
```

See Also

clean_autosaved_RelDataModels() to clean this list.

list_type_ref *List supported types references*

Description

List supported types references

Usage

```
list_type_ref()
```

modelToVn *VisNetwork representation of a RelDataModel object*

Description

VisNetwork representation of a RelDataModel object

Usage

```
modelToVn(
  model,
  color = "lightgrey",
  border = "black",
  highlightBorder = "orange"
)
```

Arguments

model	a RelDataModel
color	default table background color
border	border color (single character)
highlightBorder	color of highlighted borders
	Internal function

model_relational_data *Relational data modeler GUI*

Description

Relational data modeler GUI

Usage

```
model_relational_data(
  modelInput = RelDataModel(list()),
  fromR = interactive(),
  defaultColor = "#D9D9D9",
 availableColors = c("#9BC8FE", "#F67FC4", "#C6BDF1", "#DFFB86", "#F8DEC3", "#8FE6E0",
    "#FEFE8F", "#FAC6DC", "#A9ECC9"),
 example = system.file("examples/HPO-model.json", package = utils::packageName()),
  forceIntro = FALSE
)
```

Arguments

modelInput	the RelDataModel to start from
fromR	a logical indicating if the application is launched from R
defaultColor	a single color indicating the default table color
availableColors	a character of possible colors for tables
example	a file path to an sql or json model
forceIntro	if TRUE the help tour start when the application is launched (default: FALSE)

Value

The RelDataModel designed with the GUI.

norm_type_ref *Normalize type names*

Description

Normalize type names

Usage

```
norm_type_ref(x, typeRef, ignore.case = TRUE)
```

Arguments

x	a character vector to normalize
typeRef	a character vector of length one: the type reference (list_type_ref)
ignore.case	should case be ignored (default: TRUE)

---

order_fields *Order fields in a table in a RelDataModel*

Description

Order fields in a table in a RelDataModel

Usage

```
order_fields(x, tableName, order)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
order	a vector of integers all in (1:number_of_fields)

Value

A RelDataModel

---

order_indexes *Order index in a table in a RelDataModel*

Description

Order index in a table in a RelDataModel

Usage

```
order_indexes(x, tableName, order)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
order	a vector of integers all in (1:number_of_indexes)

Value

A RelDataModel

plot.RelDataModel *Plot a RelDataModel object*

Description

This function draw a visNetwork of the RelDataModel.

Usage

```
## S3 method for class 'RelDataModel'
plot(x, ...)
```

Arguments

x	a RelDataModel
...	additional parameters: • **color** default table background color • **border** border color (single character) • **highlightBorder** color of highlighted borders

Examples

```
## Read the model ----
hpo_model <- read_json_data_model(
  system.file("examples/HPO-model.json", package = "ReDaMoR")
)
## Plot the model ----
plot(hpo_model)
```

read_json_data_model *Read a data model from JSON*

Description

Read a data model from JSON

Usage

```
read_json_data_model(txt)
```

Arguments

txt	a JSON string, URL or file

Examples

```
## Read the model ----
hpo_model <- read_json_data_model(
  system.file("examples/HPO-model.json", package = "ReDaMoR")
)
## Confront to data ----
confrontation_report <- confront_data(
  hpo_model,
  path = list.files(
    system.file("examples/HPO-subset", package = "ReDaMoR"),
    full.names = TRUE
  ),
  returnData = TRUE
)
```

`read_named_MM` *Read a named sparse matrix in MatrixMarket text format*

Description

Read a named sparse matrix in MatrixMarket text format

Usage

```
read_named_MM(
  file,
  skip = 0,
  n_max = Inf,
  class = c("dgCMatrix", "tibble"),
  guess_max = 20
)
```

Arguments

`file`	the file to read
`skip`	the number of records to skip (default: 0)
`n_max`	the maximum number of records to read (default: Inf)
`class`	the class of object to return. By default a "dgCMatrix". If "tibble" is chosen, the sparse matrix is returned as a tibble with 3 columns: i (row index), j (column index) and x (values) and an "header" attribute containing the matrix rownames and colnames.
`guess_max`	the number of lines to read to find the header. (see `read_named_MM_header()`)

Value

By default a dgCMatrix. If the "tibble" class is chosen, the sparse matrix is returned as a tibble with 3 columns: i (row index), j (column index) and x (values) and an "header" attribute containing the matrix rownames and colnames.

read_named_MM_header *Read the header of a named sparse matrix in MatrixMarket text format*

Description

Read the header of a named sparse matrix in MatrixMarket text format

Usage

```
read_named_MM_header(file, guess_max = 20)
```

Arguments

file	the file to read
guess_max	the number of lines to read to find the header. (4 should be sufficient. Default: 20)

Value

A list with the following fields:

- rownames: a character vector with the matrix row names
- colnames: a character vector with the matrix column names
- rows: the number of matrix rows
- columns: the number of matrix columns
- values: the number of values in the matrix
- header_length: the number of lines in the header

read_SQL_data_model *Read a data model from an SQL file from the MySQL Workbench*

Description

Read a data model from an SQL file from the MySQL Workbench

Usage

```
read_SQL_data_model(f, typeRef = "MySQLWB", mysqlcomments = TRUE)

readSQLDataModel(...)
```

Arguments

f	the SQL file to read
typeRef	the reference for type conversion (Default: "MySQLWB"; see list_type_ref())
mysqlcomments	if MySQL comments (starting with #) should be removed (Default: TRUE)
...	params for read_SQL_data_model

Details

Database, table and field names should be surrounded by "`".

Value

A RelDataModel object

Functions

- readSQLDataModel(): Deprecated version of read_SQL_data_model

Examples

```
## Read the model ----
hpo_from_sql <- read_SQL_data_model(
  system.file("examples/HPO-model.sql", package = "ReDaMoR")
)
## Confront to data ----
confrontation_report <- confront_data(
  hpo_from_sql,
  path = list.files(
    system.file("examples/HPO-subset", package = "ReDaMoR"),
    full.names = TRUE
  ),
  returnData = TRUE
)
```

---

recover_RelDataModel *Recover an autosaved RelDataModel*

Description

Recover an autosaved RelDataModel

Usage

```
recover_RelDataModel(name = NA)
```

Arguments

name	The name of the autosaved RelDataModel to bring back. Available autosaved RelDataModel can be listed using the `list_autosaved_RelDataModel()`. If NA (default) the latest model is returned.

RelDataModel *Create a RelDataModel object*

Description

Create a RelDataModel object

Usage

```
RelDataModel(l, checkFK = TRUE, createFKIndex = FALSE)
```

Arguments

l	the list of table models (RelTableModel objects)
checkFK	a logical indicating if foreign keys should be checked (default: TRUE)
createFKIndex	should an index be create for the foreign keys (default: FALSE)

Value

A RelDataModel object.

RelTableModel *Create a RelTableModel object*

Description

Create a RelTableModel object

Usage

```
RelTableModel(
  l = NULL,
  tableName,
  fields,
  primaryKey = NULL,
  foreignKeys = NULL,
  indexes = NULL,
 display = list(x = as.numeric(NA), y = as.numeric(NA), color = as.character(NA),
    comment = as.character(NA))
)
```

Arguments

`l`	DEPRECATED. A named list with the function parameters. If `NULL` (default) the function parameters are used. If not `NULL`, the function parameters are ignored and taken from l.
`tableName`	a character vector of length one
`fields`	a tibble with the following columns: • *name*: character • *type*: character • *nullable*: logical (optional, defaults to TRUE) • *unique*: logical (optional, defaults = FALSE) • *comment*: character (optional, defaults to NA_character_)
`primaryKey`	a character vector of any length. All values should be in fields$name
`foreignKeys`	a list of foreign keys. Each foreign key is defined as a list with the following elements: • *refTable*: a character vector of length one (the referenced table) • *key*: a tibble with a "from" and a "to" columns • (*cardinality*): an optional integer vector with 4 values: – fmin: from minimum cardinality – fmax: from maximum cardinality – tmin: to minimum cardinality – tmax: to maximum cardinality
`indexes`	a list of indexes. Each index is defined by 3 columns: • *field*: character (all in fields$name) • *order*: character • *unique*: logical
`display`	a list gathering: • *x*: single numeric value for the x position of the table • *y*: single numeric value for the y position of the table • *color*: single character value corresponding to the color of the table • *comment*: single character value with some description of the table

Details

When defining a matrix, 3 and only 3 fields must be defined: 2 of types 'row' and 'column' and the 3rd of your choice. In this case primaryKey is defined automatically as the combination of row and column.

Value

A RelTableModel object.

remove_field *Remove a field from a table in a RelDataModel*

Description

Remove a field from a table in a RelDataModel

Usage

```
remove_field(x, tableName, fieldName, rmForeignKeys = FALSE)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
fieldName	the name of the field to remove (a single character)
rmForeignKeys	a single logical indicating if the corresponding foreign keys should be removed. If FALSE (default), the function will throw an error if it encounter a foreign key using the field.

Value

A RelDataModel

remove_foreign_key *Remove a foreign key between two tables*

Description

Remove a foreign key between two tables

Usage

```
remove_foreign_key(x, fromTable, fromFields, toTable, toFields)
```

Arguments

x	a RelDataModel
fromTable	the name of the referencing table
fromFields	the name of the referencing fields
toTable	the name of the referenced table
toFields	the names of the referenced fields

Value

A RelDataModel

remove_index *Remove an index from a table in a RelDataModel*

Description

Remove an index from a table in a RelDataModel

Usage

```
remove_index(x, tableName, fieldNames)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
fieldNames	the names of the fields composing the index

Value

A RelDataModel

remove_table *Remove a table from a RelDataModel*

Description

Remove a table from a RelDataModel

Usage

```
remove_table(x, tableName, rmForeignKeys = FALSE)
```

Arguments

x	a RelDataModel
tableName	the name of the table to remove
rmForeignKeys	if TRUE, remove foreign keys which are not available after extraction. If FALSE (default) the function will throw an error if any foreign keys does not exist in the extracted RelDataModel.

Value

A RelDataModel

rename_field *Rename an existing field in a RelDataModel table*

Description

Rename an existing field in a RelDataModel table

Usage

```
rename_field(x, tableName, current, new)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
current	the current name of the field to modify (a single character)
new	the new name of the field (a single character)

Value

A RelDataModel

rename_table *Rename a table in a RelDataModel*

Description

Rename a table in a RelDataModel

Usage

```
rename_table(x, old, new)
```

Arguments

x	a RelDataModel object
old	a single character corresponding to the table name to change
new	the new table name

Value

A RelDataModel

set_primary_key *Set the primary key a table in a RelDataModel*

Description

Set the primary key a table in a RelDataModel

Usage

```
set_primary_key(x, tableName, fieldNames)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
fieldNames	the names of the fields to include in the primary key

Value

A RelDataModel

set_unique_index *Set table index uniqueness in a RelDataModel*

Description

Set table index uniqueness in a RelDataModel

Usage

```
set_unique_index(x, tableName, fieldNames, unique)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
fieldNames	the names of the fields composing the index
unique	a logical value

Value

A RelDataModel

SUPPTYPES *Supported R types*

Description

Supported R types

Usage

```
SUPPTYPES
```

toDBM *Convert a RelDataModel object in a list of 5 normalized tibbles*

Description

Convert a RelDataModel object in a list of 5 normalized tibbles

Usage

```
toDBM(rdm)
```

Arguments

rdm	a RelDataModel object

Value

A list with the following tibbles:

- **tables**: The tables in the model with the following information
  - **name**: the name of the table
  - **x**: the x coordinate of the table in the model drawing (NA ==> position undefined)
  - **y**: the y coordinate of the table in the model drawing (NA ==> position undefined)
  - **color**: the color of the table in the model drawing (NA ==> undefined)
  - **comment**: comment about the table
- **fields**: The fields in the model with the following information
  - **name**: the name of the field
  - **type**: the type of the field
  - **nullable**: a logical indicating if the field can be null
  - **comment**: comment about the field
  - **table**: the name of the table to which the field belongs
- **primaryKeys**: The primary keys in the model with the following information

  - **table**: the name of the relevant table
  - **field**: the name of the field participating to the primary key
- **foreignKeys**: The foreign keys in the model with the following information
  - **table**: the name of the referring table
  - **fki**: the identifier of the foreign key (by referring table)
  - **field**: the name of the referring field
  - **refTable**: the name of the referred table
  - **refField**: the name of the referred field
- **indexes**: The indexes in the model with the following information
  - **table**: the name of the relevant table
  - **idx**: the identifier of the index (by table)
  - **field**: the name of the field participating to the index
  - **unique**: a logical indicating if the field is unique

`update_field` *Update field information in a table of a RelDataModel*

Description

Update field information in a table of a RelDataModel

Usage

```
update_field(
  x,
  tableName,
  fieldName,
  type = NULL,
  nullable = NULL,
  unique = NULL,
  comment = NULL
)
```

Arguments

`x`	a RelDataModel
`tableName`	the name of the table to modify (a single character)
`fieldName`	the name of the field to modify (a single character)
`type`	the type of the field (a single character)
`nullable`	if the field is nullable (a single logical)
`unique`	if the values are unique (a single logical)
`comment`	a description (a single character)

Value

A RelDataModel

`update_foreign_key` *Update a the cardinalities of a foreign key between two tables*

Description

Update a the cardinalities of a foreign key between two tables

Usage

```
update_foreign_key(
  x,
  fromTable,
  fromFields,
  toTable,
  toFields,
  fmin,
  fmax,
  tmin,
  tmax
)
```

Arguments

`x`	a RelDataModel
`fromTable`	the name of the referencing table
`fromFields`	the name of the referencing fields
`toTable`	the name of the referenced table
`toFields`	the names of the referenced fields
`fmin`	from minimum cardinality
`fmax`	from maximum cardinality
`tmin`	to minimum cardinality
`tmax`	to maximum cardinality

Value

A RelDataModel

update_table_display *Update the display of a table of a RelDataModel*

Description

Update the display of a table of a RelDataModel

Usage

```
update_table_display(
  x,
  tableName,
  px = NULL,
  py = NULL,
  color = NULL,
  comment = NULL
)
```

Arguments

x	a RelDataModel
tableName	the name of the table to modify (a single character)
px	the position of the table: x value
py	the position of the table: y value
color	the color of the table
comment	a table description/comment

Value

A RelDataModel

view_confrontation_report *View confrontation report in rstudio viewer*

Description

View confrontation report in rstudio viewer

Usage

```
view_confrontation_report(cr, ...)
```

Arguments

cr	the confrontation report from confront_data
...	additional params for the format_confrontation_report_md() function

---

write_json_data_model *Write a data model in a JSON file*

Description

Write a data model in a JSON file

Usage

```
write_json_data_model(x, path)
```

Arguments

x	the model to be written
path	file on disk

[.RelDataModel *Subset a RelDataModel*

Description

Subset a RelDataModel

Usage

```
## S3 method for class 'RelDataModel'
x[i, rmForeignKeys = FALSE, ...]
```

Arguments

x	the RelDataModel objcet
i	the index or the names of the elements to extract
rmForeignKeys	if TRUE, remove foreign keys which are not available after extraction. If FALSE (default) the function will throw an error if any foreign keys does not exist in the extracted RelDataModel.
...	additional arguments for the RelDataModel function.

Index