Package ‘TKCat’

July 3, 2024

Type Package
Title Tailored Knowledge Catalog
Version 1.1.11
Description Facilitate the management of data from knowledge resources that are frequently used alone or together in research environments.

In ‘TKCat’, knowledge resources are manipulated as modeled database (MDB) objects. These objects provide access to the data tables along with a general description of the resource and a detail data model documenting the tables, their fields and their relationships.

These MDBs are then gathered in catalogs that can be easily explored and shared.

Finally, ‘TKCat’ provides tools to easily subset, filter and combine MDBs and create new catalogs suited for specific needs.


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

Depends R (>= 3.6), ReDaMoR (>= 0.7.0), magrittr, DBI, visNetwork, dplyr

Imports ClickHouseHTTP, rlang, tidyselect, getPass, shiny, shinydashboard, DT, htmltools, readr, jsonlite, jsonvalidate (> = 1.3.2), base64enc, markdown, promises, future, xml2, Matrix, uuid, crayon, roxygen2

Suggests knitr, rmarkdown, stringr, RClickhouse, data.tree, BED

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Patrice Godard [aut, cre, cph]

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

Repository CRAN

Date/Publication 2024-07-03 14:50:02 UTC
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>.format_bytes</td>
<td>4</td>
</tr>
<tr>
<td>add_chMDB_user</td>
<td>5</td>
</tr>
<tr>
<td>add_chTKCat_collection</td>
<td>6</td>
</tr>
<tr>
<td>add_collection_member</td>
<td>6</td>
</tr>
<tr>
<td>add_feature_def</td>
<td>7</td>
</tr>
<tr>
<td>add_helpers.MDB</td>
<td>7</td>
</tr>
<tr>
<td>add_km_feature</td>
<td>8</td>
</tr>
<tr>
<td>add_km_spec</td>
<td>9</td>
</tr>
<tr>
<td>add_km_table</td>
<td>9</td>
</tr>
<tr>
<td>add_property_values</td>
<td>10</td>
</tr>
<tr>
<td>add_table_def</td>
<td>10</td>
</tr>
<tr>
<td>add_table_features</td>
<td>11</td>
</tr>
<tr>
<td>add_unit_def</td>
<td>12</td>
</tr>
<tr>
<td>archive_chMDB</td>
<td>12</td>
</tr>
<tr>
<td>as_chMDB</td>
<td>13</td>
</tr>
<tr>
<td>as_fileMDB.chMDB</td>
<td>13</td>
</tr>
<tr>
<td>as_KMR</td>
<td>15</td>
</tr>
<tr>
<td>as_memoMDB</td>
<td>16</td>
</tr>
<tr>
<td>change_chTKCat_password</td>
<td>16</td>
</tr>
<tr>
<td>check_chTKCat</td>
<td>17</td>
</tr>
<tr>
<td>chMDB</td>
<td>17</td>
</tr>
<tr>
<td>chTKCat</td>
<td>19</td>
</tr>
<tr>
<td>ch_insert</td>
<td>20</td>
</tr>
<tr>
<td>collection_members.TKCat</td>
<td>20</td>
</tr>
<tr>
<td>compare_MDB</td>
<td>20</td>
</tr>
<tr>
<td>count_records.MDB</td>
<td>22</td>
</tr>
<tr>
<td>create_chMDB</td>
<td>23</td>
</tr>
<tr>
<td>create_chTKCat_user</td>
<td>23</td>
</tr>
<tr>
<td>create_KMR</td>
<td>24</td>
</tr>
<tr>
<td>create_POK</td>
<td>25</td>
</tr>
<tr>
<td>data_files</td>
<td>25</td>
</tr>
<tr>
<td>data_file_size</td>
<td>26</td>
</tr>
<tr>
<td>data_model.chMDB</td>
<td>26</td>
</tr>
<tr>
<td>data_tables.chMDB</td>
<td>27</td>
</tr>
<tr>
<td>db_disconnect.chMDB</td>
<td>28</td>
</tr>
<tr>
<td>db_info.chMDB</td>
<td>28</td>
</tr>
<tr>
<td>db_reconnect.POK</td>
<td>30</td>
</tr>
<tr>
<td>db_tables</td>
<td>30</td>
</tr>
<tr>
<td>decode_bin</td>
<td>31</td>
</tr>
<tr>
<td>dims.chMDB</td>
<td>31</td>
</tr>
<tr>
<td>drop_chMDB</td>
<td>33</td>
</tr>
<tr>
<td>drop_chTKCat_user</td>
<td>33</td>
</tr>
<tr>
<td>empty_chMDB</td>
<td>34</td>
</tr>
<tr>
<td>encode_bin</td>
<td>34</td>
</tr>
<tr>
<td>explore_MDBs.TKCat</td>
<td>35</td>
</tr>
<tr>
<td>filter.chMDB</td>
<td>36</td>
</tr>
<tr>
<td>Function/Property</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>filter.fileMDB</td>
<td>37</td>
</tr>
<tr>
<td>filter.memoMDB</td>
<td>37</td>
</tr>
<tr>
<td>filter.metaMDB</td>
<td>38</td>
</tr>
<tr>
<td>filter_mdb_matrix.chMDB</td>
<td>38</td>
</tr>
<tr>
<td>filter_with_tables.chMDB</td>
<td>39</td>
</tr>
<tr>
<td>format.chTKCat</td>
<td>40</td>
</tr>
<tr>
<td>get_chMDB_metadata</td>
<td>41</td>
</tr>
<tr>
<td>get_chMDB_timestamps</td>
<td>41</td>
</tr>
<tr>
<td>get_chTKCat_collection</td>
<td>42</td>
</tr>
<tr>
<td>get_collection_mapper</td>
<td>42</td>
</tr>
<tr>
<td>get_confrontation_report</td>
<td>43</td>
</tr>
<tr>
<td>get_hosts.DBIConnection</td>
<td>43</td>
</tr>
<tr>
<td>get_KMR</td>
<td>44</td>
</tr>
<tr>
<td>get_km_spec</td>
<td>44</td>
</tr>
<tr>
<td>get_local_collection</td>
<td>45</td>
</tr>
<tr>
<td>get_MDB.TKCat</td>
<td>45</td>
</tr>
<tr>
<td>get_POK</td>
<td>46</td>
</tr>
<tr>
<td>get_query.chMDB</td>
<td>46</td>
</tr>
<tr>
<td>get_R_helpers.MDB</td>
<td>47</td>
</tr>
<tr>
<td>get_shared_collections</td>
<td>48</td>
</tr>
<tr>
<td>has_km_spec</td>
<td>48</td>
</tr>
<tr>
<td>heads.chMDB</td>
<td>49</td>
</tr>
<tr>
<td>import_collection_mapper</td>
<td>50</td>
</tr>
<tr>
<td>import_local_collection</td>
<td>50</td>
</tr>
<tr>
<td>init_chTKCat</td>
<td>51</td>
</tr>
<tr>
<td>is.chMDB</td>
<td>51</td>
</tr>
<tr>
<td>is.chTKCat</td>
<td>52</td>
</tr>
<tr>
<td>is.fileMDB</td>
<td>52</td>
</tr>
<tr>
<td>is.KMR</td>
<td>53</td>
</tr>
<tr>
<td>is.MDB</td>
<td>53</td>
</tr>
<tr>
<td>is.memoMDB</td>
<td>54</td>
</tr>
<tr>
<td>is.metaMDB</td>
<td>54</td>
</tr>
<tr>
<td>is.POK</td>
<td>55</td>
</tr>
<tr>
<td>is.TKCat</td>
<td>55</td>
</tr>
<tr>
<td>is_chMDB_public</td>
<td>56</td>
</tr>
<tr>
<td>is_current_chMDB</td>
<td>56</td>
</tr>
<tr>
<td>join_mdb_tables</td>
<td>57</td>
</tr>
<tr>
<td>list_chMDB_timestamps</td>
<td>57</td>
</tr>
<tr>
<td>list_chMDB_users</td>
<td>58</td>
</tr>
<tr>
<td>list_chTKCat_collections</td>
<td>59</td>
</tr>
<tr>
<td>list_chTKCat_users</td>
<td>59</td>
</tr>
<tr>
<td>list_feature_properties</td>
<td>60</td>
</tr>
<tr>
<td>list_local_collections</td>
<td>60</td>
</tr>
<tr>
<td>list_MDBs.TKCat</td>
<td>61</td>
</tr>
<tr>
<td>list_measurements</td>
<td>61</td>
</tr>
<tr>
<td>list_measurement_units</td>
<td>62</td>
</tr>
<tr>
<td>list_POKs</td>
<td>62</td>
</tr>
<tr>
<td>list_property_values</td>
<td>63</td>
</tr>
<tr>
<td>Description</td>
<td>.format_bytes</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Format bytes numbers in human readable values</td>
<td>.format_bytes</td>
</tr>
</tbody>
</table>
add_chMDB_user

Usage

.\format_bytes(bytes)

Arguments

bytes a vector of integers

Value

A vector of character with human readable size

add_chMDB_user

Add a user to an MDB of a chTKCat object

Description

Add a user to an MDB of a chTKCat object

Usage

add_chMDB_user(x, mdb, login, admin = FALSE)

Arguments

x a chTKCat object
mdb name of the modeled database
login login of the user to drop
admin if the user is an admin of the MDB

Value

No return value, called for side effects
add_chTKCat_collection

Import a collection in a chTKCat database

Description

Import a collection in a chTKCat database

Usage

add_chTKCat_collection(x, json, overwrite = FALSE)

Arguments

x a chTKCat object
json a single character indicating the collection to import. Can be:
  • a path to a file
  • the name of a local collection (see list_local_collections())
  • the json text defining the collection
overwrite a logical indicating if the existing collection should be replaced.

Value

No return value, called for side effects

add_collection_member

Add a collection member to an MDB

Description

Add a collection member to an MDB

Usage

add_collection_member(x, collection, table, ...)

Arguments

x an MDB object
collection a collection title in list_local_collections()
table the table providing the collection member
... definition of the collection fields as lists (e.g. be=list(static=TRUE, value="Gene")
or organism=list(static=TRUE, value="Homo sapiens", type="Scientific name") )
add_feature_def

Add a feature definition to Knowledge Management Requirements (KMR)

Description

Add a feature definition to Knowledge Management Requirements (KMR)

Usage

add_feature_def(kmr, name, description, properties)

Arguments

- kmr: a KMR object
- name: the name of the feature type
- description: description of the feature type
- properties: properties of the feature. A list named with property names. For each property, a list with:
  - type: among "integer", "numeric", "logical", "character", "Date", "POSIXct", "base64", "table" and "field" ("table" and "field" type are used for referencing tables and fields; the other types come from ReDaMoR).
  - description (optional): a description of the feature property. Useful when the feature has more than one property
  - mandatory: a logical indicating the property is mandatory for the feature
  - measurement (optional and only for "integer" and "numeric" types): the name of the measurement for checking units

Value

The modified KMR MDB object

add_helpers.MDB

Add a set of helper functions to a compatible object

Description

Add a set of helper functions to a compatible object
add_km_feature

Add KM feature specifications to an MDB object

Usage

```r
add_km_feature(x, kmr, table, feature, fields, unit = as.character(NA))
```

Arguments

- `x`: An MDB object to update with specification tables.
- `kmr`: An MDB object with KM requirements.
- `table`: The name of an existing table in `x`.
- `feature`: The name of an existing feature in `kmr`.
- `fields`: Either a single character providing the name of an existing field in table or a list named with feature property names from kmr. Each element of the list should provide a "field" slot with the name of the corresponding field and a "unit" slot with the name of the unit if relevant.
- `unit`: A single character providing the unit if relevant. Unit information provided in fields override this parameter value.
Value

An **MDB** object with additional KM table feature specification

---

**add_km_spec**

Add empty KM specification tables to an **MDB** object

Description

Add empty KM specification tables to an **MDB** object

Usage

```
add_km_spec(x, kmr)
```

Arguments

- **x**
  - an **MDB** object to update with specification tables
- **kmr**
  - an **MDB** object with KM requirements

Value

An **MDB** object with empty KM specification tables

---

**add_km_table**

Add KM table specifications to an **MDB** object

Description

Add KM table specifications to an **MDB** object

Usage

```
add_km_table(x, kmr, name, type, features = list())
```

Arguments

- **x**
  - an **MDB** object to update with specification tables
- **kmr**
  - an **MDB** object with KM requirements
- **name**
  - the name of an existing table in x
- **type**
  - the name of an existing table type in kmr
- **features**
  - a list of feature definitions. Each element of a list is a list of parameters for the `add_km_feature()` function.

Value

An **MDB** object with additional KM table specification
add_property_values

Add possible values to an integer or a character feature property in KMR

Description

Add possible values to an integer or a character feature property in KMR

Usage

add_property_values(kmr, feature, property, values)

Arguments

- kmr: a KMR object
- feature: the name of the feature type
- property: the name of the property
- values: a vector of character or integer or a named vector with the description of the values

Value

The modified KMR MDB object

add_table_def

Add a table definition to Knowledge Management Requirements (KMR)

Description

Add a table definition to Knowledge Management Requirements (KMR)

Usage

add_table_def(
  kmr,
  name,
  description,
  collection = as.character(NA),
  mandatory_features
)
**add_table_features**

**Arguments**

- **kmr**  
  a KMR object
- **table**  
  the name of the table type
- **description**  
  description of the table type
- **collection**  
  the name of the collection of which this table type must be a member (default: NA)
- **mandatory_features**  
  a character vector with mandatory features for this table type

**Value**

The modified KMR MDB object

---

**add_table_features**  
*Add possible features to table type in KMR*

**Description**

Add possible features to table type in KMR

**Usage**

```
add_table_features(kmr, table, features)
```

**Arguments**

- **kmr**  
  a KMR object
- **table**  
  the name of the table type
- **features**  
  a character vector with optional features for this table type

**Value**

The modified KMR MDB object
add_unit_def  
*Add a unit definition to Knowledge Management Requirements (KMR)*

**Description**

Add a unit definition to Knowledge Management Requirements (KMR)

**Usage**

```
add_unit_def(kmr, measurement, unit, description)
```

**Arguments**

- `kmr`: an KMR object
- `measurement`: the name of the measurement (e.g. "volume")
- `unit`: the name of the unit (e.g. "ml")
- `description`: a description of the unit (e.g. "milliliter")

**Value**

The modified KMR object

---

archive_chMDB  
*Archive a chMDB in a chTKCat*

**Description**

Archive a chMDB in a chTKCat

**Usage**

```
archive_chMDB(
  x, name,
  defaultTS = as.POSIXct("1970-01-01 00:00.0", tz = "UTC")
)
```

**Arguments**

- `x`: a chTKCat object
- `name`: the name of the database to archive
- `defaultTS`: a default timestamp value to use when not existing in the DB (default: `as.POSIXct("1970-01-01 00:00.0", tz="UTC")`)

**Value**

No return value, called for side effects
as_chMDB  

Push an MDB object in a ClickHouse database

Description

Push an MDB object in a ClickHouse database

Usage

as_chMDB(x, tkcon, timestamp = Sys.time(), overwrite = FALSE, by = 10^5)

Arguments

x an MDB object
tkcon a chTKCat object
timestamp a single POSIXct value as a timestamp for the chMDB instance. The default value is the current system time. If this value is smaller or equal to the chMDB current value, an error is thrown. If NA, the current instance is overwritten (if the overwrite parameter is set to TRUE) without changing the existing timestamp.
overwrite a logical indicating if existing data should be overwritten (default: FALSE)
by the size of the batch: number of records to write together (default: 10^5)

Value

A chMDB object.

as_fileMDB.chMDB  

Write an MDB object

Description

Write an MDB object

Usage

## S3 method for class 'chMDB'
as_fileMDB(
  x,
  path,
  readParameters = list(delim = "\t", na = "<NA>")
)

Value

A chMDB object.
## S3 method for class 'fileMDB'
as_fileMDB(
x, path,
readParameters = list(delim = "\t", na = "<NA>"),
htmlModel = TRUE,
compress = TRUE,
by = 10^5,
...
)

## S3 method for class 'memoMDB'
as_fileMDB(
x, path,
readParameters = list(delim = "\t", na = "<NA>"),
htmlModel = TRUE,
compress = TRUE,
by = 10^5,
...
)

## S3 method for class 'metaMDB'
as_fileMDB(
x, path,
readParameters = list(delim = "\t", na = "<NA>"),
htmlModel = TRUE,
compress = TRUE,
by = 10^5,
...
)

### Arguments

- **x**: an MDB object
as_KMR

path the path where the MDB should be written

readParameters The following parameters are currently supported:

• **delim**: a single character used to separate fields within a record (default: '\t')

• **quoted_na**: a single logical indicating if missing values inside quotes should be treated as missing values or strings. **WARNING**: THIS PARAMETER IS NOT TAKEN INTO ACCOUNT WITH `readr>=2.0.0`.

• **na**: String used for missing values. The default value for reading a fileMDB is "NA". But the default value for writing a fileMDB is ""<NA>"". This value is written in the DESCRIPTION.json file to avoid ambiguity when reading the fileMDB.

htmlModel a logical. If TRUE (default) the model is also plotted in an html file.

compress a logical specifying whether saving data is to use "gzip" compression (default: TRUE)

by the size of the batch: number of records to write together (default: 10^5)

... method specific parameters

Value

A fileMDB object.

---

as_KMR

Convert in a KMR object when possible

Description

Convert in a KMR object when possible

Usage

as_KMR(x)

Arguments

x an object

Value

A KMR object
as_memoMDB

Convert any MDB object in a memoMDB object

Description

Convert any MDB object in a memoMDB object

Usage

as_memoMDB(x, ...)

Arguments

x

a MDB object

... additional parameters for the memoMDB() function.

Value

A memoMDB object

See Also

get_confrontation_report, ReDaMoR::format_confrontation_report and ReDaMoR::format_confrontation_report_md for getting and formatting the report confronting the data to the model.

change_chTKCat_password

Change chTKCat password

Description

Change chTKCat password

Usage

change_chTKCat_password(x, login, password)

Arguments

x

a chTKCat object

login

user login

password

new user password

Value

No return value, called for side effects
check_chTKCat

Description

Check a chTKCat object

Usage

check_chTKCat(x, verbose = FALSE)

Arguments

x
  a chTKCat object
verbose
  a logical indicating if information messages should be displayed.

Value

Invisible result: chTKCat object

chMDB

An MDB (Modeled DataBase) relying on ClickHouse: chMDB

Description

An MDB (Modeled DataBase) relying on ClickHouse: chMDB

Rename tables of a chMDB object

Usage

chMDB(
  tkcon,  
  dbTables, 
  dbInfo, 
  dataModel, 
  collectionMembers = NULL, 
  check = TRUE, 
  n_max = 10, 
  verbose = FALSE
)

## S3 replacement method for class 'chMDB'
names(x) <- value

## S3 method for class 'chMDB'
rename(.data, ...)

## S3 method for class 'chMDB'

x[i]

## S3 method for class 'chMDB'

x[[i]]

### Arguments

- `tkcon`: a chTKCat object
- `dbTables`: a named vector of tables in tkcon$chcon with all(names(dbTables) %in% names(dataModel))
- `dbInfo`: a list with DB information: "name" (only mandatory field), "title", "description", "url", "version", "maintainer".
- `dataModel`: a ReDaMoR::RelDataModel object
- `collectionMembers`: the members of collections as provided to the collection_members<- function (default: NULL => no member).
- `check`: logical: if TRUE (default) the data are confronted to the data model
- `n_max`: maximum number of records to read for checks purpose (default: 10). If 0, the data are not checked. See also ReDaMoR::confront_data().
- `verbose`: if TRUE display the data confrontation report
- `x`: a chMDB object
- `value`: new table names
- `.data`: a chMDB object
- `...`: Use new_name = old_name to rename selected tables
- `i`: the index or the name of the tables to take

### Value

A chMDB object

### See Also

- MDB methods: db_info, data_model, data_tables, collection_members, count_records, dims, filter_with_tables, as_fileMDB
- Additional general documentation is related to MDB.
- filter.chMDB, slice.chMDB
- chTKCat, db_disconnect(), db_reconnect()
chTKCat  Connect to a ClickHouse TKCat instance

Description

Connect to a ClickHouse TKCat instance

Usage

chTKCat(
    host = "localhost",
    port = 9111L,
    user = "default",
    password,
    settings = list(max_query_size = 1073741824, use_uncompressed_cache = 0, load_balancing
        = "random", max_memory_usage = 0, allow_introspection_functions = 1,
        joined_subquery_requires_alias = 0),
    ports = NULL,
    drv = ClickHouseHTTP::ClickHouseHTTP(),
    ...
)

Arguments

  host         a character string specifying the host heberging the database (default: localhost)
  port         an integer specifying the port on which the database is listening (default: 9111)
  user         user name
  password     user password
  settings     list of Clickhouse settings
  ports        a named list of available ports for accessing ClickHouse (default: NULL; example: c(Native=9101, HTTP=9111))
  drv          a DBI driver for connecting to ClickHouse (default: ClickHouseHTTP::ClickHouseHTTP(); other supported driver: RClickhouse::clickhouse())
  ...          additional parameters for connection (see ClickHouseHTTP::dbConnect,ClickHouseHTTPDriver-
                method for the default driver)

Value

  a chTKCat object

See Also

  check_chTKCat(), db_disconnect(), db_reconnect()
### ch_insert

*Insert records by batches in a Clickhouse table*

**Description**

Insert records by batches in a Clickhouse table

**Usage**

```r
ch_insert(con, dbName, tableName, value, by = 10^6)
```

**Arguments**

- `con`: the clickhouse connection
- `dbName`: the name of the database
- `tableName`: the name of the table
- `value`: the table to import
- `by`: the size of the batch: number of records to import together (default: 10^6)

**Value**

No return value, called for side effects

### collection_members.TKCat

*Collection members*

**Description**

Collection members

**Usage**

```r
# S3 method for class 'TKCat'
collection_members(x, ...)

# S3 method for class 'chMDB'
collection_members(x, ...)

# S3 replacement method for class 'chMDB'
collection_members(x) <- value

# S3 method for class 'chTKCat'
```
collection_members(x, ...)  
### S3 method for class 'fileMDB'  
collection_members(x, ...)  
### S3 replacement method for class 'fileMDB'  
collection_members(x) <- value  
collection_members(x, ...)  
collection_members(x) <- value  
### S3 method for class 'memoMDB'  
collection_members(x, ...)  
### S3 replacement method for class 'memoMDB'  
collection_members(x) <- value  
### S3 method for class 'metaMDB'  
collection_members(x, ...)  

**Arguments**  
  
x  
an object with embedded collection members  
  
...  
names of the collections to focus on. By default, all of them are taken.  
  
value  
the new collection members. A data.frame with the following columns:  
  
  • collection (character): The name of the collection  
  • cid (character): Collection identifier  
  • resource (character): The name of the resource  
  • mid (integer): The identifier of the member  
  • table (character): The table recording collection information  
  • field (character): The collection field.  
  • static (logical): TRUE if the field value is common to all elements.  
  • value (character): The name of the table column if static is FALSE or the field value if static is TRUE.  
  • type (character): the type of the field. (not necessarily used ==> NA if not)  

**Value**  

A `dplyr::tibble` with the following columns:  
  
  • collection (character): The name of the collection  
  • cid (character): Collection identifier  
  • resource (character): The name of the resource  
  • mid (integer): The identifier of the member  
  • table (character): The table recording collection information
• **field** (character): The collection field.
• **static** (logical): TRUE if the field value is common to all elements.
• **value** (character): The name of the table column if static is FALSE or the field value if static is TRUE.
• **type** (character): the type of the field. (not necessarily used ==> NA if not)

---

**count_records.MDB**

**Description**

Count the number of records

**Usage**

```r
## S3 method for class 'MDB'
count_records(x, ...)
count_records(x, ...)
```

---

**compare_MDB**

**Compare two MDB objects**

**Description**

Compare two MDB objects

**Usage**

```r
compare_MDB(former, new)
```

**Arguments**

- **former** an MDB object
- **new** an MDB object

**Value**

A tibble with 4 columns:

- **Information**: Compared information
- **Former**: value for the former object
- **New**: value for the new object
- **Identical**: a logical indicating if the 2 values are identical
create_chMDB

Arguments

- **x** an object with embedded data tables
- ... the name of the tables to consider (default: all of them)

Value

A named vector with the number of records per table.

create_chMDB  
*Create a database in a chTKCat*

Description

Create a database in a chTKCat

Usage

create_chMDB(x, name, public = FALSE)

Arguments

- **x** a chTKCat object
- **name** the name of the new database
- **public** if the database data are accessible to any user (default: FALSE)

Value

No return value, called for side effects

create_chTKCat_user  
*Create a chTKCat user*

Description

Create a chTKCat user

Usage

create_chTKCat_user(*
  x, login, password, contact, admin = FALSE, provider = admin
*)
create_KMR

Arguments

- `x`: a chTKCat object
- `login`: user login
- `password`: user password (NA ==> no password)
- `contact`: contact information (can be NA)
- `admin`: a logical indicating if the user is an admin of the chTKCat instance (default: TRUE)
- `provider`: a logical indicating if the user is data provider (TRUE) or a data consumer (FALSE: default). If admin is set to TRUE provider will be set to TRUE

Value

No return value, called for side effects

Description

Create an MDB object with Knowledge Management Requirements (KMR)

Usage

`create_KMR(name, title, description, version, maintainer)`

Arguments

- `name`: the name of the requirements
- `title`: the title of the requirements
- `description`: the description of the requirements
- `version`: version of the requirements
- `maintainer`: maintainer of the requirements

Value

An MDB object with KM requirements data model
create_POK

Create a piece of knowledge (POK) from an MDB and a KMR object

Description
Create a piece of knowledge (POK) from an MDB and a KMR object

Usage
create_POK(mdb, kmr, tkcat = NULL)

Arguments
mdb  a MDB object with KM specifications
kmr  a KMR object with KM requirements
tkcat  A TKCat or chTKCat object to make available in helper environment

Value
A POK object: a list with 3 slots:
- $mdb: the provided MDB object
- $kmr: the provided KMR object
- $helpers: a list functions to leverage data from mdb and kmr

data_files

Get the data files from a fileMDB object

Description
Get the data files from a fileMDB object

Usage
data_files(x)

Arguments
x  a fileMDB object

Value
a list with "dataFiles" and "readParameters" for reading the files.
### data_file_size

*Get the size of data files from a fileMDB object*

**Description**

Get the size of data files from a fileMDB object

**Usage**

```r
data_file_size(x, hr = FALSE)
```

**Arguments**

- `x`: a fileMDB object
- `hr`: a logical indicating if the values should be "human readable". (default: FALSE)

**Value**

A numeric vector with size in bytes (hr=FALSE) or a character vector with size and units (hr=TRUE)

### data_model

*Get object data model*

**Description**

Get object data model

**Usage**

```r
## S3 method for class 'chMDB'
data_model(x, ...)

## S3 method for class 'fileMDB'
data_model(x, ...)

data_model(x, ...)

## S3 method for class 'memoMDB'
data_model(x, rtOnly = FALSE, recursive = FALSE, ...)

## S3 method for class 'metaMDB'
data_model(x, rtOnly = FALSE, recursive = FALSE, ...)
```
**Arguments**

- **x**
  - an object with an embedded data model
- **...**
  - method specific parameters
- **rtOnly**
  - if TRUE, the function only returns the relational tables and the corresponding foreign tables (default: FALSE)
- **recursive**
  - if TRUE and rtOnly, the function returns also the relational tables from embedded metaMDBs.

**Value**

A ReDaMoR::RelDataModel object

---

**data_tables.chMDB**

*Get object data tables*

---

**Description**

Get object data tables

**Usage**

```r
## S3 method for class 'chMDB'
data_tables(x, ..., skip = 0, n_max = Inf)
```

```r
## S3 method for class 'fileMDB'
data_tables(x, ..., skip = 0, n_max = Inf)
data_tables(x, ..., skip = 0, n_max = Inf)
```

```r
## S3 method for class 'memoMDB'
data_tables(x, ..., skip = 0, n_max = Inf)
```

```r
## S3 method for class 'metaMDB'
data_tables(x, ..., skip = 0, n_max = Inf)
```

**Arguments**

- **x**
  - an object with embedded data tables
- **...**
  - the name of the tables to get (default: all of them)
- **skip**
  - the number of rows to skip (default: 0)
- **n_max**
  - maximum number of rows to return (default: Inf)

**Value**

A list of dplyr::tibble and matrix
### db_disconnect.chMDB

**Disconnect an object from a database**

**Description**

Disconnect an object from a database

**Usage**

```r
## S3 method for class 'chMDB'
db_disconnect(x)

## S3 method for class 'chTKCat'
db_disconnect(x)

## S3 method for class 'metaMDB'
db_disconnect(x)
```

**Arguments**

- `x` an object with a database connection

**Value**

No return value, called for side effects

### db_info.chMDB

**DB information**

**Description**

DB information

**Usage**

```r
## S3 method for class 'chMDB'
db_info(x, ...)

## S3 replacement method for class 'chMDB'
db_info(x) <- value

## S3 method for class 'fileMDB'
```
db_info(x, ...)  

## S3 replacement method for class 'fileMDB'  
db_info(x) <- value  

db_info(x, ...)  

## S3 method for class 'memoMDB'  
db_info(x, ...)  

## S3 replacement method for class 'memoMDB'  
db_info(x) <- value  

## S3 method for class 'metaMDB'  
db_info(x, ...)  

## S3 replacement method for class 'metaMDB'  
db_info(x) <- value  

### Arguments  

- **x**: an object with embedded database information  
- **...**: method specific parameters  
- **value**: list with the following elements:  
  - **name**: a single character  
  - **title**: a single character  
  - **description**: a single character  
  - **url**: a single character  
  - **version**: a single character  
  - **maintainer**: a single character vector  
  - **size**: a numeric vector providing the size of the DB in bytes  

### Value  

A list with the following elements:  

- **name**: a single character  
- **title**: a single character  
- **description**: a single character  
- **url**: a single character  
- **version**: a single character  
- **maintainer**: a single character vector  
- **size**: a numeric vector providing the size of the DB in bytes
### db_reconnect.POK

Reconnect an object to a database

#### Description
Reconnect an object to a database

#### Usage
```r
# S3 method for class 'POK'
db_reconnect(x, user, password, ntries = 3, ...)

# S3 method for class 'chMDB'
db_reconnect(x, user, password, ntries = 3, ...)

# S3 method for class 'chTKCat'
db_reconnect(x, user, password, ntries = 3, ...)

db_reconnect(x, user, password, ntries = 3, ...)

# S3 method for class 'metaMDB'
db_reconnect(x, user, password, ntries = 3, ...)
```

#### Arguments
- **x**: an object with a database connection
- **user**: user name. If not provided, it’s taken from x
- **password**: user password. If not provided, first the function tries to connect without any password. If it fails, the function asks the user to provide a password.
- **ntries**: the number of times the user can enter a wrong password (default: 3)
- **...**: additional parameters for methods

#### Value
A new database connection object.

### db_tables

Get the DB tables from a chMDB or metaMDB object

#### Description
Get the DB tables from a chMDB or metaMDB object
**decode_bin**

**Usage**

db_tables(x, host)

**Arguments**

x 
a chMDB or a metaMDB object

host 
the name of host (as returned by `get_hosts`) to focus on. Only used with metaMDB objects.

**Value**

a list with a chTKCat object (tkcon) and a named vector of DB table names (dbTables).

---

**Description**

Decode base64 string

**Usage**

decode_bin(text)

**Arguments**

text 
the base64 character vector to decode

**Value**

One decoded value (e.g. a raw vector corresponding to a binary file)

---

**Description**

Detailed information about the format of the tables
Usage

```r
## S3 method for class 'chMDB'
dims(x, ...)

## S3 method for class 'fileMDB'
dims(
  x,
  ..., by = 1000,
  estimateThr = 5e+07,
  estimateSample = 10^6,
  showWarnings = TRUE
)

dims(x, ...)

## S3 method for class 'memoMDB'
dims(x, ...)

## S3 method for class 'metaMDB'
dims(x, ...)
```

Arguments

- `x` an object with embedded data tables
- `...` the name of the tables to consider (default: all of them)
- `by` the size of the batch: number of lines to count together (default: 1000)
- `estimateThr` file size threshold in bytes from which an estimation of row number should be computed instead of a precise count (default: 50000000 = 50MB)
- `estimateSample` number of values on which the estimation is based (default: 10^6)
- `showWarnings` a warning is raised by default if estimation is done.

Value

A tibble with one row for each considered table and the following columns:

- name: the name of the table
- format: "table", "matrix" or "MatrixMarket"
- ncol: number of columns
- nrow: number of rows
- records: number of records (nrow for tables and ncol*nrow for matrices)
- bytes: size in bytes
- transposed: FALSE by default. TRUE only for matrices stored in a transposed format.
**drop_chMDB**

*Drop a database from a chTKCat*

**Description**

Drop a database from a chTKCat

**Usage**

`drop_chMDB(x, name)`

**Arguments**

- `x`: a chTKCat object
- `name`: the name of the database to remove

**Value**

No return value, called for side effects

---

**drop_chTKCat_user**

*Drop a user from a chTKCat object*

**Description**

Drop a user from a chTKCat object

**Usage**

`drop_chTKCat_user(x, login)`

**Arguments**

- `x`: a chTKCat object
- `login`: login of the user to drop

**Value**

No return value, called for side effects
empty_chMDB

Empty a chMDB in a chTKCat

Description

Empty a chMDB in a chTKCat

Usage

empty_chMDB(x, name, timestamp = NA)

Arguments

x a chTKCat object
name the name of the database to empty
timestamp timestamp of the instance to empty. If NA (default) the current instance is emptied.

Value

No return value, called for side effects

encode_bin

Encode a binary file in a base64 string

Description

Encode a binary file in a base64 string

Usage

encode_bin(what)

Arguments

what a file path or a raw vector

Value

A character vector of length 1 with the base64 encoded file
explore_MDBs.TKCat

Explore available MDB in a shiny web interface

Description

Explore available MDB in a shiny web interface

Usage

## S3 method for class 'TKCat'
explore_MDBs(  
  x,  
  subSetSize = 100,  
  download = FALSE,  
  workers = 4,  
  title = NULL,  
  skinColors = "green",  
  logoDiv = TKCAT_LOGO_DIV,  
  rDirs = NULL,  
  tabTitle = "TKCat",  
  tabIcon = "www/TKCat-small.png",  
  ...
)

## S3 method for class 'chTKCat'
explore_MDBs(  
  x,  
  subSetSize = 100,  
  host = x$chcon@host,  
  download = FALSE,  
  workers = 4,  
  userManager = NULL,  
  title = NULL,  
  skinColors = c("blue", "yellow"),  
  logoDiv = TKCAT_LOGO_DIV,  
  tabTitle = "chTKCat",  
  tabIcon = "www/TKCat-small.png",  
  rDirs = NULL,  
  ...
)

explore_MDBs(x, ...)

Arguments

x a TKCat related object (e.g. chTKCat)
subSetSize the maximum number of records to show
download a logical indicating if data can be downloaded (default: FALSE). If TRUE a temporary directory is created and made available for shiny.

workers number of available workers when download is available (default: 4)

title A title for the application. If NULL (default): the chTKCat instance name

skinColors two colors for the application skin: one for default connection ("blue" by default) and one for user connection ("yellow" by default). Working values: "blue", "black", "purple", "green", "red", "yellow".

logoDiv a shiny::div object with a logo to display in side bar. The default is the TKCat hex sticker with a link to TKCat github repository.

rDirs a named character vector with resource path for shiny::addResourcePath

tabTitle a title to display in tab (default: "chTKCat")

tabIcon a path to an image (in available resource paths: "www", "doc" or in rDirs) to use as a tab icon.

... method specific parameters

host the name of the host to show in the application

userManager URL for user management interface (see manage_chTKCat_users()). If NULL (default), the functionality is not added.

Value

No return value, called for side effects

filter.chMDB Filter a chMDB object and return a memoMDB

Description

Filter a chMDB object and return a memoMDB

Usage

## S3 method for class 'chMDB'
filter(.data, ..., by = 10^5, .preserve = FALSE)

Arguments

.data a chMDB object

... each argument should have the name of one of the tables of the chMDB object and contain a simple logical expression involving the names of the corresponding table.

by the size of the batch: number of records to filter together (default: 10^5)

.preserve not used

Value

a memoMDB object
filter.fileMDB  
*Filter a fileMDB object and return a memoMDB*

**Description**

Filter a fileMDB object and return a memoMDB

**Usage**

```r
## S3 method for class 'fileMDB'
filter(.data, ..., .preserve = FALSE)
```

**Arguments**

- `.data`  a fileMDB object  
- `...` each argument should have the name of one of the tables of the fileMDB object and contain a simple logical expression involving the names of the corresponding table.  
- `.preserve` not used

**Value**

a memoMDB object

filter.memoMDB  
*Filter a memoMDB object*

**Description**

Filter a memoMDB object

**Usage**

```r
## S3 method for class 'memoMDB'
filter(.data, ..., .preserve = FALSE)
```

**Arguments**

- `.data`  a memoMDB object  
- `...` each argument should have the name of one of the tables of the memoMDB object and contain a simple logical expression involving the names of the corresponding table.  
- `.preserve` not used

**Value**

a filtered memoMDB object
filter.metaMDB  

Filter a metaMDB object

Description
Filter a metaMDB object

Usage
## S3 method for class 'metaMDB'
filter(.data, ..., .preserve = FALSE)

Arguments
.data       a metaMDB object
...                 each argument should have the name of one of the tables of the metaMDB object
    and contain a simple logical expression involving the names of the corresponding table.
.preserve   not used

Value
a filtered memoMDB object

filter_mdb_matrix.chMDB

Filter a matrix stored in an MDB

Description
Filter a matrix stored in an MDB

Usage
## S3 method for class 'chMDB'
filter_mdb_matrix(x, tableName, ...)

## S3 method for class 'fileMDB'
filter_mdb_matrix(x, tableName, .by = 10^5, ...)

filter_mdb_matrix(x, tableName, ...)

## S3 method for class 'memoMDB'
filter_mdb_matrix(x, tableName, ...)

## S3 method for class 'metaMDB'
filter_mdb_matrix(x, tableName, ...)
Arguments

- `x`: an **MDB** object
- `tableName`: a character vector of length 1 corresponding to the name of the table to filter (must be a matrix)
- `...`: character vectors with the row names and/or columns names to select. The names of the parameters must correspond to the name of the column and of the row fields (the matrix cannot be filtered from values).
- `.by`: the size of the batch: number of lines to process together (default: 10000)

Value

A sub-matrix of `tableName` in `x`. Only existing elements are returned. No error is raised if any element is missing. The result must be checked and adapted to user needs.

Examples

```
## Not run:
## Return the matrix of expression values focused on the selected genes
filter_mdb_matrix(x=db, "Expression_value", gene=c("SNCA", "MAPT"))
## End(Not run)
```

**filter_with_tables.chMDB**

*Filter an **MDB** object according to provided tables*

**Description**

Filter an **MDB** object according to provided tables

**Usage**

```
## S3 method for class 'chMDB'
filter_with_tables(x, tables, checkTables = TRUE, by = 10^5, ...)

## S3 method for class 'fileMDB'
filter_with_tables(x, tables, checkTables = TRUE, by = 10^5, ...)

## S3 method for class 'memoMDB'
filter_with_tables(x, tables, checkTables = TRUE, ...)

## S3 method for class 'metaMDB'
filter_with_tables(x, tables, checkTables = TRUE, ...)
```
Arguments

x an MDB object
tables a named list of tibbles to filter with. The names should correspond to the table names in x and the tibbles should fit the data model.
checkTables if TRUE, the tables are confronted to their model in the data model of x.
by the size of the batch: number of lines to process together (default: 10000)
... method specific parameters

Value

a memoMDB object

format.chTKCat Format a chTKCat object for printing

Description

Format a chTKCat object for printing

Usage

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

Arguments

x a chTKCat object
...
not used

Value

A single character
**get_chMDB_metadata**  
*Get the metadata of an MDB from a chTKCat connection*

**Description**  
Get the metadata of an MDB from a chTKCat connection

**Usage**  
```r
get_chMDB_metadata(x, dbName, timestamp = NA)
```

**Arguments**  
- `x`: a chTKCat object  
- `dbName`: the name of the MDB  
- `timestamp`: the timestamp of the instance to get. Default=NA: get the current version.

**Value**  
A list with the following elements:
- `dbInfo`: General information regarding the MDB  
- `dataModel`: The data model  
- `collectionMembers`: Members of different collections  
- `access`: type of access to the MDB

**See Also**  
- `get_MDB`

---

**get_chMDB_timestamps**  
*Get instance timestamps of an MDB in chTKCat*

**Description**  
Get instance timestamps of an MDB in chTKCat

**Usage**  
```r
get_chMDB_timestamps(x, name)
```

**Arguments**  
- `x`: a chTKCat object  
- `name`: the name of the database
get_collection_mapper

Value
A tibble with the instance "timestamp" and a logical indicating if it's the "current" one or not.

get_chTKCat_collection

Get a collection from a chTKCat

Description
Get a collection from a chTKCat

Usage
get_chTKCat_collection(x, title)

Arguments
x  a chTKCat object
title the title of the collection to get

Value
The definition of the collection as a JSON string.

get_collection_mapper

Get the default mapper function for a collection

Description
Get the default mapper function for a collection

Usage
get_collection_mapper(collection)

Arguments
collection the name of the targeted collection (it should belong to local collections: see list_local_collections()).

Value
A function to map collection members.
get_confrontation_report

Get the last generated MDB confrontation report

Description

Get the last generated MDB confrontation report

Usage

text

get_confrontation_report()

Value

A confrontation report generated by ReDaMoR::confront_data()

get_hosts.DBIConnection

Description

Get database hosts

Usage

## S3 method for class 'DBIConnection'
get_hosts(x, ...)

## S3 method for class 'chMDB'
get_hosts(x, ...)

## S3 method for class 'chTKCat'
get_hosts(x, ...)

## S3 method for class 'metaMDB'
get_hosts(x, ...)

Arguments

x an object with database connection(s)

... additional parameters for methods.
get_km_spec

Value
A character vector with hosts information (generally 1) in the following shape: "host:port"

get_KMR

Get a KMR object from a TKCat or a chTKCat object

Description
Get a KMR object from a TKCat or a chTKCat object

Usage
get_KMR(...)

Arguments
...

parameters for the get_MDB() function

Value
A KMR object

get_km_spec

Get KM specifications from an MDB object

Description
Get KM specifications from an MDB object

Usage
get_km_spec(x, kmr)

Arguments

x an MDB object with specification tables

kmr an MDB object with KM requirements

Value
An MDB object with kmr specification tables from x


---

**get_local_collection** 
*Get the json definition of a local collection of concepts*

**Description**

Get the json definition of a local collection of concepts

**Usage**

```r
get_local_collection(title)
```

**Arguments**

- `title`: the title of the collection to get

**Value**

The definition of the collection as a JSON string.

---

**get_MDB.TKCat** 
*Get an MDB object from a TKCat related object*

**Description**

Get an MDB object from a TKCat related object

**Usage**

```r
## S3 method for class 'TKCat'
get_MDB(x, dbName, ...)

## S3 method for class 'chTKCat'
get_MDB(x, dbName, timestamp = NA, check = TRUE, n_max = 10, ...)

get_MDB(x, dbName, ...)
```

**Arguments**

- `x`: a TKCat related object (e.g. chTKCat)
- `dbName`: the name of the database
- `...`: method specific parameters
- `timestamp`: the timestamp of the instance to get. Default=NA: get the current version.
- `check`: logical: if TRUE (default) the data are confronted to the data model
- `n_max`: maximum number of records to read for checks purpose (default: 10). See also `ReDaMoR::confront_data()`.

---
get_POK

Get a POK from a \texttt{chTKCat} connection or a \texttt{TKCat} object

Description

Get a POK from a \texttt{chTKCat} connection or a \texttt{TKCat} object

Usage

\texttt{get\_POK(x, \texttt{mdb}, \texttt{kmr})}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} \texttt{a chTKCat or a TKCat object}
\item \texttt{mdb} \hspace{1cm} \texttt{MDB object with KM specifications or its name in tkcat}
\item \texttt{kmr} \hspace{1cm} \texttt{KMR object with KM requirements or its name in tkcat}
\end{itemize}

Value

A POK object

get_query.chMDB

Get SQL query

Description

Get SQL query

Usage

\begin{verbatim}
## S3 method for class 'chMDB'
get_query(x, query, autoalias = !is_current_chMDB(x), ...)

## S3 method for class 'chTKCat'
get_query(x, query, ...)

get_query(x, query, ...)
\end{verbatim}
Arguments

x an object with a database connection
query the SQL query
autoalias Change this parameter only if you know what you’re doing. if TRUE, make relevant alias to query the chMDB using the table names from the data model. If FALSE, the user must know the table instance name in the remote database. By default, autoalias is set to TRUE when using a non-current instance of the database.

... method specific parameters

Value

A tibble with query results

Description

Get a set of helper functions from an object

Usage

```r
## S3 method for class 'MDB'
get_R_helpers(x, hnames = NA, kmr, tkcat = NULL, ...)

## S3 method for class 'KMR'
get_R_helpers(x, hnames = NA, tkcat = NULL, mdb = NULL, ...)

get_R_helpers(x, hnames, ...)
```

Arguments

x an object with helpers
hnames the names of the helper sets. If NA (default), all available are sourced.
kmr a KMR object
tkcat A TKCat or chTKCat object to make available in helper environment
... method specific parameters
mdb An MDB object to make available in helper environment

Details

x, kmr and tkcat objects are made available in helpers environment as 'THISMDB', 'THISKMR' and 'THISTKCAT' objects respectively and can be used as such within helpers code.
x, tkcat and mdb objects are made available in helpers environment as 'THISKMR', 'THISTKCAT' and 'THISMDB' objects respectively and can be used as such within helpers code.
has_km_spec

Value

Check if KM specifications are available in an MDB object

Description

Check if KM specifications are available in an MDB object

Usage

has_km_spec(x, kmr)

Arguments

- x: an MDB object with specification tables
- kmr: an MDB object with KM requirements

get_shared_collections

Get collections shared by 2 objects and return member combinations

Description

Get collections shared by 2 objects and return member combinations

Usage

get_shared_collections(x, y)

Arguments

- x: an MDB object
- y: an MDB object

Value

A tibble with the following fields:

- collection: the name of the collection
- mid.x: the collection member identifier in x
- table.x: the table of the collection member in x
- mid.y: the collection member identifier in y
- table.y: the table of the collection member in y
Value

A logical: TRUE if the MDB has KM specifications

Description

Get the first records of each object data tables

Usage

```r
## S3 method for class 'chMDB'
heads(x, ..., n = 6L)

## S3 method for class 'fileMDB'
heads(x, ..., n = 6L)

heads(x, ..., n = 6L)

## S3 method for class 'memoMDB'
heads(x, ..., n = 6L)

## S3 method for class 'metaMDB'
heads(x, ..., n = 6L)
```

Arguments

- `x`: an object with embedded data tables
- `...`: the name of the tables to get (default: all of them)
- `n`: maximum number of records to return (default: 6)

Value

A list of `dplyr::tibble` and `matrix`
import_collection_mapper

*Import a function to map collection members*

**Description**

Import a function to map collection members

**Usage**

import_collection_mapper(collection, fun)

**Arguments**

- **collection**
  the name of the targeted collection (it should belong to local collections: see `list_local_collections()`).
- **fun**
  a function which takes 2 data.frames (x an y) with fields described in the collection definition and map the different elements.

**Value**

No return value, called for side effects. The function will be used to map collection members.

import_local_collection

*Import a the definition of a collection of concepts in the local environment*

**Description**

Import a the definition of a collection of concepts in the local environment

**Usage**

import_local_collection(txt, overwrite = FALSE)

**Arguments**

- **txt**
  a JSON string or file
- **overwrite**
  a single logical. If TRUE the collection is overwritten if it already exists (default: FALSE)

**Value**

No return value, called for side effects. The collection will be available and operations will be possible on its members.
**init_chTKCat**

*Initialize a chTKCat database*

**Description**

The initialization can only be done locally (host="localhost")

**Usage**

`init_chTKCat(x, instance, version, path, login, password, contact)`

**Arguments**

- `x`: a chTKCat object
- `instance`: instance name of the database
- `version`: version name of the database
- `path`: path to ClickHouse folder
- `login`: login of the primary administrator of the database
- `password`: password for the primary administrator of the database
- `contact`: contact information for the primary administrator of the database

**Value**

a chTKCat

---

**is.chMDB**

*Check if the object is a chMDB object*

**Description**

Check if the object is a chMDB object

**Usage**

`is.chMDB(x)`

**Arguments**

- `x`: any object

**Value**

A single logical: TRUE if `x` is a chMDB object
is.chTKCat  Check the object is a chTKCat object

Description
Check the object is a chTKCat object

Usage
is.chTKCat(x)

Arguments
x  any object

Value
A single logical: TRUE if x is a chTKCat object

is.fileMDB  Check if the object is a fileMDB object

Description
Check if the object is a fileMDB object

Usage
is.fileMDB(x)

Arguments
x  any object

Value
A single logical: TRUE if x is an fileMDB object
is.KMR

Check if an object represents Knowledge Management Requirements (KMR)

Description
Check if an object represents Knowledge Management Requirements (KMR)

Usage
is.KMR(x)

Arguments
x
an object

Value
TRUE if x is a KMR, FALSE if not

---

is.MDB

Check if the object is an MDB object

Description
Check if the object is an MDB object

Usage
is.MDB(x)

Arguments
x
any object

Value
A single logical: TRUE if x is an MDB object.
is.memoMDB

*Check if the object is a memoMDB object*

**Description**

Check if the object is a memoMDB object

**Usage**

```r
is.memoMDB(x)
```

**Arguments**

- `x`: any object

**Value**

A single logical: TRUE if `x` is an memoMDB object

---

is.metaMDB

*Check if the object is a metaMDB object*

**Description**

Check if the object is a metaMDB object

**Usage**

```r
is.metaMDB(x)
```

**Arguments**

- `x`: any object

**Value**

A single logical: TRUE if `x` is an metaMDB object
is.POK

Check if the object is a POK object

Description
Check if the object is a POK object

Usage
is.POK(x)

Arguments
x any object

Value
A single logical: TRUE if x is a POK object

is.TKCat

Check the object is a TKCat object

Description
Check the object is a TKCat object

Usage
is.TKCat(x)

Arguments
x any object

Value
A single logical: TRUE if x is a TKCat object
### is_chMDB_public

**Description**
Is a chMDB public

**Usage**
\[
\text{is\_chMDB\_public}(x, \text{mdb})
\]

**Arguments**
- \(x\) a chTKCat object
- \(\text{mdb}\) name of the modeled database

**Value**
A logical indicating if the chMDB is public or not.

---

### is_current_chMDB

**Description**
Check if the chMDB object refers to the current instance of the MDB

**Usage**
\[
\text{is\_current\_chMDB}(x)
\]

**Arguments**
- \(x\) a chMDB object

**Value**
A single logical: TRUE if \(x\) refers to the current instance of the MDB.
**join_mdb_tables**  
*Join connected tables*

**Description**
Join connected tables

**Usage**

```r
code
join_mdb_tables(
  x,
  ...,
  type = c("left", "right", "inner", "full"),
  jtName = NA
)
```

**Arguments**
- **x**: an MDB object
- **...**: at least 2 names of tables to join
- **type**: the type of join among:
  - "left": includes all rows of the first provided table
  - "right": includes all rows of the last provided table
  - "inner": includes all rows in all provided tables
  - "full": includes all rows in at least one provided table
- **jtName**: the name of the joint. IF NA (default), the name is then the name is the first provided table name.

**Value**
A metaMDB corresponding to x with the joined tables replaced by the joint. If less than 2 table names are provided, the function returns the original x MDB.

---

**list_chMDB_timestamps**  
*List instance timestamps of an MDB in chTKCat*

**Description**
List instance timestamps of an MDB in chTKCat

**Usage**

```r
code
list_chMDB_timestamps(x, name)
```
Arguments

\( x \)  

a `chTKCat` object

\( \text{name} \)  

the name of the database

Value

A tibble with the instance of each table at each timestamp. The "current" attribute indicate the current timestamp instance. If there is no recorded timestamp, the function returns NULL.

```
list_chMDB_users  

List users of an MDB of a `chTKCat` object
```

Description

List users of an MDB of a `chTKCat` object

Usage

```
list_chMDB_users(x, mdbs = NULL)
```

Arguments

\( x \)  

a `chTKCat` object

\( \text{mdbs} \)  

names of the modeled databases. If NULL (default), all the databases are considered.

Value

A tibble with 3 columns:

- user: the user login
- mdb: the name of the modeled database
- admin: if the user is an admin of the MDB
**list_chTKCat_collections**

*List collections available in a chTKCat*

**Description**

List collections available in a chTKCat

**Usage**

`list_chTKCat_collections(x, withJson = FALSE)`

**Arguments**

- `x` a chTKCat object
- `withJson` if TRUE, returns the json strings of the collection (default: FALSE)

**Value**

A tibble with the title, the description and optionally the json definition of the collections

---

**list_chTKCat_users**

*List chTKCat user*

**Description**

List chTKCat user

**Usage**

`list_chTKCat_users(x)`

**Arguments**

- `x` a chTKCat object

**Value**

A tibble with 3 columns:

- `login`: user login
- `contact`: user contact information
- `admin`: if the user is an admin of the chTKCat object
### list_feature_properties

*List properties of a feature*

**Description**

List properties of a feature

**Usage**

```r
list_feature_properties(kmr, feature)
```

**Arguments**

- `kmr`: a KMR object
- `feature`: the name of the feature

**Value**

A `dplyr::tibble` with the description of feature properties

### list_local_collections

*List local collections of concepts*

**Description**

List local collections of concepts

**Usage**

```r
list_local_collections(withJson = FALSE)
```

**Arguments**

- `withJson`: if TRUE, returns the json strings of the collection (default: FALSE)

**Value**

A tibble with the title, the description and optionally the json definition of the collections
list_MDBs.TKCat

List available MDB

Description
List available MDB

Usage
## S3 method for class 'TKCat'
list_MDBs(x, withInfo = TRUE)

## S3 method for class 'chTKCat'
list_MDBs(x, withInfo = TRUE)

list_MDBs(x, withInfo = TRUE)

Arguments
x a TKCat related object (e.g. chTKCat)
withInfo if TRUE (default), the function returns a table with db_info. If FALSE, it returns only MDB names.

Value
A tibble with information about the MDB available in a TKCat related object.

list_measurements
List supported types of measurement

Description
List supported types of measurement

Usage
list_measurements(kmr)

Arguments
kmr a KMR object

Value
A vector of character with measurement names
list_measurement_units

List possible units for a type of measurement

Description
List possible units for a type of measurement

Usage
list_measurement_units(kmr, measurement)

Arguments
- kmr: a KMR object
- measurement: the type of measurement

Value
A dplyr::tibble with the description of supported units

list_POKs

List available POK

Description
List available POK

Usage
list_POKs(x, kmr)

Arguments
- x: a chTKCat or TKCat object
- kmr: KMR object with KM requirements or its name in tkcat

Value
The names of available POKs in x with kmr requirements.
list_property_values  List of supported values for an integer or a character property

Description
List of supported values for an integer or a character property

Usage
list_property_values(kmr, feature, property)

Arguments
kmr a KMR object
feature the name of the feature
property the name of the property

Value
A dplyr::tibble with value and their description (if available)

list_tables.DBIConection
List tables in a clickhouse database

Description
List tables in a clickhouse database

Usage
## S3 method for class 'DBIConection'
list_tables(x, dbNames = NULL, ...)

## S3 method for class 'chTKCate'
list_tables(x, dbNames = NULL, ...)

list_tables(x, ...)

Arguments
x an object with a clickhouse connection
dbNames the name of databases to focus on (default NULL ==> all)
... method specific parameters
Value

A tibble with at least the following columns:

- **database**: the name of the database
- **name**: the name of the table
- **total_rows**: the number of rows in the table
- **total_bytes**: the size of the table
- **total_columns**: the number of columns in the table

---

**list_table_features**  
List the features provided by a set of tables

Description

List the features provided by a set of tables

Usage

```
list_table_features(kmr, tables = NULL)
```

Arguments

- **kmr**: a KMR object
- **tables**: the name of the tables. If NULL (default), all the features are listed.

Value

A `dplyr::tibble` with feature description and properties

---

**list_table_types**  
List types of tables defined in a KMR object

Description

List types of tables defined in a KMR object

Usage

```
list_table_types(kmr)
```

Arguments

- **kmr**: a KMR object

Value

A `dplyr::tibble` with the names of table types, their descriptions and the related collections
**manage_chTKCat_users**

*Manage user information in a shiny interface*

**Description**

Manage user information in a shiny interface

**Usage**

```r
manage_chTKCat_users(x, pwdFile = NULL)
```

**Arguments**

- `x` - a `chTKCat` object
- `pwdFile` - a local file in which the password for `x` can be found. If `NULL` (default), the connection is shared by all sessions and can be disabled at some point.

---

**map_collection_members**

*Map different collection members*

**Description**

Map different collection members

**Usage**

```r
map_collection_members(
  x, 
  y, 
  collection, 
  xm, 
  ym, 
  suffix = c("_x", "_y"), 
  fun = NA, 
  ... 
)
```

**Arguments**

- `x` - a `data.frame`
- `y` - a `data.frame`
- `collection` - the name of the collection.
collection member x: a data.frame with the fields "field", "static", "value", "type" as returned by the read_collection_members() function.

collection member y: a data.frame with the fields "field", "static", "value", "type" as returned by the read_collection_members() function.

the suffix to append to field names from x and y tables. Default: c("_x", "_y")

the function used to map x and y collection members. By default (NA) it is automatically identified if recorded in the system. The way to write this function is provided in the details section.

additional parameters for the fun function.

fun must have at least an x and a y parameters. Each of them should be a data.frame with all the field values given in xm and ym. Additional parameters can be defined and will be forwarded using ... fun should return a data frame with all the fields values given in xm and ym followed by "_x" and "_y" suffix.

A tibble giving necessary information to map elements in x and y. The columns corresponds to the field values in xm and ym followed by a suffix (default: c("_x", "_y"))

Only fields documented as non static in xm and ym are kept.

The class "MDB" provides general functions for handling modeled databases. The MDB classes implemented in the TKCat package are: fileMDB, memoMDB, chMDB and metaMDB. These classes provide additional functions.

## S3 method for class 'MDB'
names(x)

## S3 method for class 'MDB'
length(x)

## S3 method for class 'MDB'
lengths(x, use.names = TRUE)

## S3 method for class 'MDB'
as.list(x, ...)

---

** MDB**

** MDB**

---

**Description**

The class "MDB" provides general functions for handling modeled databases. The MDB classes implemented in the TKCat package are: fileMDB, memoMDB, chMDB and metaMDB. These classes provide additional functions.

Usage

```r
## S3 method for class 'MDB'
names(x)

## S3 method for class 'MDB'
length(x)

## S3 method for class 'MDB'
lengths(x, use.names = TRUE)

## S3 method for class 'MDB'
as.list(x, ...)
```
## S3 method for class 'MDB'
select(.data, ...)

## S3 method for class 'MDB'
pull(.data, var = -1, name = NULL, ...)

## S3 method for class 'MDB'
c(...)

## S3 method for class 'MDB'
merge(
  x,
  y,
  by = get_shared_collections(x, y),
  dbInfo = list(name = paste(db_info(x)$name, db_info(y)$name, sep = "_")),
  dmAutoLayout = TRUE,
  rtColor = "yellow",
  funs = list(),
  ...)

### Arguments

- **x**: an MDB object
- **use.names**: return the names of the tables
- **...**: additional parameters
- **.data**: an MDB object
- **var**: a variable specified as in dplyr::pull
- **name**: not used but kept for compatibility with the generic function
- **y**: an MDB object
- **by**: a tibble as returned by the get_shared_collections() function which indicates which collection members should be merged through a relational table. If the collection is NA, the relational table is built by merging identical columns in table.x and table.y. If the collection is provided, the relational table is build using the map_collection_members() function.
- **dbInfo**: a list with DB information: "name" (only mandatory field), "title", "description", "url", "version", "maintainer".
- **dmAutoLayout**: if TRUE (default) the layout of the merged data model is automatically adjusted.
- **rtColor**: the color of the relational tables in the merged data model (default: "yellow")
- **funs**: a named list of functions (default: list()). If there is no function for mapping a collection in this list, it is taken automatically using the get_collection_mapper() function.
Value

names() returns the table names.

length() returns the number of tables in x.

lengths() returns the number of fields for each table in x.

as.list.MDB() returns a simple list of tibbles with all the data from the tables in x.

A metaMDB object gathering x and y along with relational tables between them created using collection members and mapping functions automatically chosen or provided by the funs parameter. . . . can be used to send parameters to the mapper functions.

See Also

MDB methods: db_info, data_model, data_tables, collection_members, count_records, filter_with_tables, as_fileMDB Additional documentation is provided for each specific class: fileMDB, memoMDB, chMDB and metaMDB.

---

**MDBs**

Get a list of MDB from metaMDB object

**Description**

Get a list of MDB from metaMDB object

**Usage**

MDBs(x)

**Arguments**

x a metaMDB object

**Value**

A list of MDB objects
Description

An MDB (Modeled DataBase) in memory: memoMDB

Rename tables of a memoMDB object

Usage

memoMDB(
  dataTables,
  dataModel,
  dbInfo,
  collectionMembers = NULL,
  check = TRUE,
  checks = c("unique", "not nullable", "foreign keys"),
  verbose = FALSE
)

## S3 replacement method for class 'memoMDB'
names(x) <- value

## S3 method for class 'memoMDB'
rename(.data, ...)

## S3 method for class 'memoMDB'
x[i]

## S3 method for class 'memoMDB'
x[[i]]

## S3 method for class 'memoMDB'
x$i

Arguments

dataTables a list of tibbles
dataModel a ReDaMoR::RelDataModel object
dbInfo a list with DB information: "name" (only mandatory field), "title", "description", "url", "version", "maintainer".
collectionMembers the members of collections as provided to the collection_members<- function (default: NULL ==> no member).
check logical: if TRUE (default) the data are confronted to the data model
checks: a character vector with the name of optional checks to be done (all of them `c("unique", "not nullable", "foreign keys")`)

verbose: if TRUE display the data confrontation report (default: FALSE)

x: a memoMDB object

value: new table names

.data: a memoMDB object

... Use new_name = old_name to rename selected tables

i: the index or the name of the tables to take

Value

A memoMDB object

See Also

- MDB methods: `db_info`, `data_model`, `data_tables`, `collection_members`, `count_records`, `dims`, `filter_with_tables`, `as_fileMDB`
- Additional general documentation is related to `MDB`.
- `filter.memoMDB`, `slice.memoMDB`

Examples

```r
hpo <- read_fileMDB(
  path = system.file("examples/HPO-subset", package="ReDaMoR"),
  dataModel = system.file("examples/HPO-model.json", package="ReDaMoR"),
  dbInfo = list(
    "name" = "HPO",
    "title" = "Data extracted from the HPO database",
    "description" = paste(  
      "This is a very small subset of the HPO!",
      "Visit the reference URL for more information"
    ),
    "url" = "http://human-phenotype-ontology.github.io/"
  )
)  
```

as_memoMDB()

```r
count_records(hpo)
```

## Too long on win-builder.r-project.org

## Not run:

```r
hpoSlice <- slice(hpo, HPO_diseases=1:10)
count_records(hpoSlice)
```

```r
if("stringr" %in% installed.packages()[,"Package"]){
  epilHP <- filter(
    hpo,
    HPO_diseases=stringr::str_detect(
      label, stringr::regex("epilepsy", ignore_case=TRUE)
    )
  )
  ``
mergeTrees_from_RelDataModel

Create ClickHouse MergeTree tables from a ReDaMoR::RelDataModel

Description

Create ClickHouse MergeTree tables from a ReDaMoR::RelDataModel

Usage

mergeTrees_from_RelDataModel(con, dbName, dbm)

Arguments

- con: the clickhouse connection
- dbName: the name of the database in which the tables should be written
- dbm: a ReDaMoR::RelDataModel object

Value

No return value, called for side effects

mergeTree_from_RelTableModel

Create a ClickHouse MergeTree table from a ReDaMoR::RelTableModel

Description

Create a ClickHouse MergeTree table from a ReDaMoR::RelTableModel

Usage

mergeTree_from_RelTableModel(con, dbName, tm)
Arguments
  
  con the clickhouse connection
  dbName the name of the database in which the table should be written
  tm a ReDaMoR::RelTableModel object

Value
  
  No return value, called for side effects

metaMDB

A metaMDB object

Description

A metaMDB object is an MDB gathering several other MDBs glued by relational tables.

Usage

metaMDB(MDBs, relationalTables, dataModel, dbInfo, check = TRUE)

## S3 replacement method for class 'metaMDB'
names(x) <- value

## S3 method for class 'metaMDB'
rename(.data, ...)

## S3 method for class 'metaMDB'
x[i]

## S3 method for class 'metaMDB'
x[[i]]

## S3 method for class 'metaMDB'
x$i

Arguments

MDBs a list of MDB objects
relationalTables a list of tibbles corresponding to the relational tables between the different MDBs
dataModel a ReDaMoR::RelDataModel object gathering all the data model of all the MDBs plus the relational tables
dbInfo a list with DB information: "name" (only mandatory field), "title", "description", "url", "version", "maintainer".
check logical: if TRUE (default) the data are confronted to the data model
x a metaMDB object
value new table names
.data a metaMDB object
... Use new_name = old_name to rename selected tables
i the index or the name of the tables to take

Value
A metaMDB object

See Also
• MDB methods: db_info, data_model, data_tables, collection_members, count_records, dims,
  filter_with_tables, as_fileMDB
• Additional general documentation is related to MDB.
• filter.metaMDB, slice.metaMDB
• get_confrontation_report, ReDaMoR::format_confrontation_report and ReDaMoR::format_confrontation_report_md
  for getting and formatting the report confronting the data to the model.

Description
Parse source code to get R helpers

Usage
parse_R.helpers(code, ...)

Arguments
code the source code as a character vector
... other objects to add in the environment of the returned functions

Details
Functions in code must be documented with roxygen2::roxygen2-package tags and only functions
with the '@export' tag are returned.

Value
A list of functions from code plus an "help" function used to get function documentation.
Examples

```r
code <- "
  a <- 2
  # Set the 'a' value to use in the add_a function
  #
  # @param v a numeric value
  #
  # @return Does not return anything
  #
  # @export
  set_a <- function(v) a <<- v

  # Add a 'a' value defined separately
  #
  # @param x a numeric value
  #
  # @return x + a
  #
  # @export
  add_a <- function(x) x + a

  # Add a 'a' value defined separately to a b value made available
  # in environment
  #
  # @return b + a
  #
  # @export
  add_a_to_b <- function() b + a
"

helpers <- parse_R_helpers(code, b=3)
helpers$help()
helpers$help("add_a")
helpers$add_a(3.5)
helpers$set_a(4)
helpers$add_a(3.5)
helpers$add_a_to_b()
helpers <- parse_R_helpers(code, b=6)
helpers$add_a_to_b()
```

read_collection_members

Read a collection member JSON file
read_fileMDB

Description
Read a collection member JSON file

Usage
read_collection_members(txt)

Arguments
txt a JSON string or file

Value
A tibble with the description of the collection members of a resource

read_fileMDB
Read a fileMDB from a path

Description
Read a fileMDB from a path

Usage
read_fileMDB(
  path,
  dbInfo = NULL,
  dataModel = NULL,
  collectionMembers = NULL,
  check = TRUE,
  n_max = 10,
  verbose = TRUE
)

Arguments
path the path to a folder with data or with the following structure:
  - data: a folder with the data
  - DESCRIPTION.json: a file with db information
  - model: a folder with the data model json file with the same name as the one
given in the DESCRIPTION.json file

dbInfo a list or a json file with DB information: "name" (only mandatory field), "title",
"description", "url" (or "reference URL"), "version", "maintainer". If NULL
(default), the DESCRIPTION.json file found in path. This file should also con-
tains relevant parameters for the readr::read_delim() function. For example:
  - delim delimiter (default: '\w')
• **quoted_na**: Should missing values inside quotes be treated as missing values or as strings or strings. **WARNING**: THIS PARAMETER IS NOT TAKEN INTO ACCOUNT WITH readr>=2.0.0.

• **na**: String used for missing values. The default value for reading a fileMDB is "NA". But the default value for writing a fileMDB is "<NA>". This value is written in the DESCRIPTION.json file to avoid ambiguity when reading the fileMDB.

**dataModel**

a ReDaMoR::RelDataModel object or json file. If NULL (default), the model json file found in path/model.

**collectionMembers**

the members of collections as provided to the collection_members<- function. If NULL (default), the members are taken from json files found in path/model/Collections

**check**

logical: if TRUE (default) the data are confronted to the data model

**n_max**

maximum number of records to read for checks purpose (default: 10). See also ReDaMoR::confront_data().

**verbose**

if TRUE (default) display the data confrontation report

---

**Value**

A fileMDB object

**See Also**

get_confrontation_report, ReDaMoR::format_confrontation_report and ReDaMoR::format_confrontation_report_md for getting and formatting the report confronting the data to the model.

---

**read_KMR**

*Read KMR from a path*

---

**Description**

Read KMR from a path

**Usage**

read_KMR(...)
relational_tables

Get a list of relational tables

Description
Get a list of relational tables

Usage
relational_tables(x, recursive = FALSE)

Arguments
x
    a metaMDB object
recursive
    if TRUE, function returns also the relational tables from embedded metaMDBs.

Value
A list of relational tables (tibbles)

remove_chMDB_user
Drop a user of an MDB of a chTKCat object

Description
Drop a user of an MDB of a chTKCat object

Usage
remove_chMDB_user(x, mdb, login)

Arguments
x
    a chTKCat object
mdb
    name of the modeled database
login
    login of the user to drop

Value
No return value, called for side effects
remove_chTKCat_collection

Remove a collection from a chTKCat database

Description

Remove a collection from a chTKCat database

Usage

remove_chTKCat_collection(x, title)

Arguments

x           a chTKCat object

title       the title of the collection to remove

Value

No return value, called for side effects

rm_km_feature

Remove KM feature specifications from an MDB object

Description

Remove KM feature specifications from an MDB object

Usage

rm_km_feature(x, kmr, table, feature)

Arguments

x           an MDB object to update with specification tables

kmr         an MDB object with KM requirements

table       the name of an existing table in x

feature     the name of a feature with specification in x table

Value

An MDB object with relevant KM table feature specification removed
**rm_km_table**  

*Remove KM table specifications from an MDB object*

**Description**  
Remove KM table specifications from an **MDB** object

**Usage**  

```
rm_km_table(x, kmr, table)
```

**Arguments**  

- **x**: an **MDB** object to update with specification tables
- **kmr**: an **MDB** object with KM requirements
- **table**: the name of an existing table in x specification

**Value**  
An **MDB** object with relevant KM table specification removed

---

**scan_fileMDBs**  

*Scan a catalog of fileMDB*

**Description**  
Scan a catalog of **fileMDB**

**Usage**  

```
scan_fileMDBs(path, subdirs = NULL, check = TRUE, n_max = 10)
```

**Arguments**  

- **path**: directory from which all the **fileMDB** should be read
- **subdirs**: the sub directories (relative to path) to take into account. If NULL (default) all the sub directories are considered.
- **check**: logical: if TRUE (default) the data are confronted to the data model
- **n_max**: maximum number of records to read for checks purpose (default: 10). See also **ReDaMoR::confront_data()**.

**Value**  
A **TKCat** object
See Also

read_fileMDB

search_MDB_fields.TKCat

Search fields in a TKCat related object

Description

Search fields in a TKCat related object

Usage

```r
## S3 method for class 'TKCat'
search_MDB_fields(x, searchTerm)
```

```r
## S3 method for class 'chTKCat'
search_MDB_fields(x, searchTerm)
```

```r
search_MDB_fields(x, searchTerm)
```

Arguments

- `x` a TKCat related object (e.g. chTKCat)
- `searchTerm` a single character with the term to search

Value

An MDB object

search_MDB_tables.TKCat

Search tables in a TKCat related object

Description

Search tables in a TKCat related object

Usage

```r
## S3 method for class 'TKCat'
search_MDB_tables(x, searchTerm)
```

```r
## S3 method for class 'chTKCat'
search_MDB_tables(x, searchTerm)
```

```r
search_MDB_tables(x, searchTerm)
```
set_chMDB_access

**Arguments**

- **x**: a TKCat related object (e.g. chTKCat)
- **searchTerm**: a single character with the term to search

**Value**

An **MDB** object

---

**Description**

Set chMDB access

**Usage**

```r
set_chMDB_access(x, mdb, public)
```

**Arguments**

- **x**: a chTKCat object
- **mdb**: name of the modeled database
- **public**: if access is public

**Value**

No return value, called for side effects

---

set_chMDB_timestamp

**Set timestamp of the current version of an MDB in chTKCat**

**Description**

Set timestamp of the current version of an MDB in chTKCat

**Usage**

```r
set_chMDB_timestamp(x, name, timestamp)
```

**Arguments**

- **x**: a chTKCat object
- **name**: the name of the database to affect
- **timestamp**: a single POSIXct value as a timestamp for the chMDB instance.
show_collection_def

Value

No return value, called for side effects

Description

This function prints details regarding a collection: title, description and arguments information. These arguments are those that can be used to document collection members within an MDB using the `add_collection_member()` function.

Usage

`show_collection_def(collection, silent = FALSE)`

Arguments

- `collection` a json string with the collection definition as returned by `get_local_collection()`
- `silent` a logical indicating if the definition should be written (TRUE by default) or not.

Value

A list with:

- collection `title`
- collection `description`
- a list of arguments for defining collection members as a list of elements with:
  - the `type` of the argument element
  - `allowed` values if any

Examples

`get_local_collection("BE") %>% show_collection_def()`
slice.chMDB  
Subset a chMDB object according to row position in one table and return a memoMDB

Description
Subset a chMDB object according to row position in one table and return a memoMDB

Usage
```r
## S3 method for class 'chMDB'
slice(.data, ..., by = 10^5, .preserve = FALSE)
```

Arguments
- `.data`: a chMDB object
- `...`: a single argument. The name of this argument should be a table name of x and the value of this argument should be vector of integers corresponding to row indexes.
- `by`: the size of the batch: number of records to slice together (default: 10^5)
- `.preserve`: not used

Value
a memoMDB object

slice.fileMDB  
Subset a fileMDB object according to row position in one table and return a memoMDB

Description
Subset a fileMDB object according to row position in one table and return a memoMDB

Usage
```r
## S3 method for class 'fileMDB'
slice(.data, ..., .preserve = FALSE)
```

Arguments
- `.data`: a fileMDB object
- `...`: a single argument. The name of this argument should be a table name of x and the value of this argument should be vector of integers corresponding to row indexes.
- `.preserve`: not used
Value

a memoMDB object

slice.memoMDB  Subset a memoMDB object according to row position in one table

Description

Subset a memoMDB object according to row position in one table

Usage

## S3 method for class 'memoMDB'
slice(.data, ..., .preserve = FALSE)

Arguments

.data  a memoMDB object
...  a single argument. The name of this argument should be a table name of x and
the value of this argument should be vector of integers corresponding to row
indexes.
.preserve  not used

Value

a memoMDB object

slice.metaMDB  Subset a metaMDB object according to row position in one table

Description

Subset a metaMDB object according to row position in one table

Usage

## S3 method for class 'metaMDB'
slice(.data, ..., .preserve = FALSE)

Arguments

.data  a metaMDB object
...  a single argument. The name of this argument should be a table name of x and
the value of this argument should be vector of integers corresponding to row
indexes.
.preserve  not used
**TKCat**

**Value**

a `memoMDB` object

---

**TKCat**: a catalog of `MDB`

**Description**

TKCat: a catalog of `MDB`

Rename a `TKCat` object

**Usage**

```r
TKCat(..., list = NULL)
```

```r
## S3 replacement method for class 'TKCat'
names(x) <- value
```

```r
## S3 method for class 'TKCat'
rename(.data, ...)
```

```r
## S3 method for class 'TKCat'
x[i]
```

```r
## S3 method for class 'TKCat'
c(...)
```

**Arguments**

- `...` TKCat objects
- `list` a list of `MDB` objects
- `x` a `TKCat` object
- `value` new `MDB` names
- `.data` a `TKCat` object
- `i` index or names of the `MDB` to take

**Value**

a `TKCat` object

**See Also**

`scan_fileMDBs`
unarchive_chMDB  
Unarchive a chMDB in a chTKCat

Description
Unarchive a chMDB in a chTKCat

Usage
unarchive_chMDB(x, name)

Arguments
x  a chTKCat object
name  the name of the database to archive

Value
No return value, called for side effects

update_chMDB_grants  
Update grants on tables in an MDB of a chTKCat object

Description
The update is done automatically based on user access.

Usage
update_chMDB_grants(x, mdb)

Arguments
x  a chTKCat object
mdb  name of the modeled database

Value
No return value, called for side effects
**update_chTKCat_user**  
*Update a chTKCat user information*

**Description**
Update a chTKCat user information

**Usage**

\[
\text{update\_chTKCat\_user}(x, \text{login}, \text{contact}, \text{admin}, \text{provider})
\]

**Arguments**

- **x**: a chTKCat object
- **login**: user login
- **contact**: contact information (can be NA)
- **admin**: a logical indicating if the user is an admin of the chTKCat instance
- **provider**: a logical indicating if the user is data provider (TRUE) or a data consumer (FALSE: default)

**Value**
No return value, called for side effects

**write_collection_members**  
*Write a collection member JSON file*

**Description**
Write a collection member JSON file

**Usage**

\[
\text{write\_collection\_members}(\text{colMembers}, \text{path} = \text{NA}, \text{collection} = \text{NULL})
\]

**Arguments**

- **colMembers**: A tibble as returned by \text{read\_collection\_members()}
- **path**: the JSON file to write. If NA (default), the JSON file is not written but returned by the function.
- **collection**: The collection definition (json string). If NULL (default), it is taken from TKCat environment (see \text{list\_local\_collections()}.}
write_MergeTree

Value

The JSON representation of collection members. If a path is provided, then the JSON is also written in it.

Description

Write a Clickhouse MergeTree table

Usage

```r
write_MergeTree(
  con,
  dbName,
  tableName,
  value,
  rtypes = NULL,
  nullable = NULL,
  sortKey = NULL
)
```

Arguments

- `con`: the clickhouse connection
- `dbName`: the name of the database
- `tableName`: the name of the table
- `value`: the table to import
- `rtypes`: a named character vector giving the R type of each and every columns. If NULL (default), types are guessed from value.
- `nullable`: a character vector indicating the name of the columns which are nullable (default: NULL)
- `sortKey`: a character vector indicating the name of the columns used in the sort key. If NULL (default), all the non-nullable columns are used in the key.

Value

No return value, called for side effects
Description

An `MDB` (Modeled DataBase) based on files: `fileMDB`

Rename tables of a `fileMDB` object

Usage

```r
## S3 method for class 'chMDB'
x$i

fileMDB(
  dataFiles,
  dbInfo,
  dataModel,
  readParameters = DEFAULT_READ_PARAMS,
  collectionMembers = NULL,
  check = TRUE,
  n_max = 10,
  verbose = FALSE
)

## S3 replacement method for class 'fileMDB'
names(x) <- value

## S3 method for class 'fileMDB'
rename(.data, ...)

## S3 method for class 'fileMDB'
x[i]

## S3 method for class 'fileMDB'
x[[i]]

## S3 method for class 'fileMDB'
x$i
```

Arguments

- `x` a `fileMDB` object
- `i` the index or the name of the tables to take
- `dataFiles` a named vector of path to data files with `all(names(dataFiles) %in% names(dataModel))`
- `dbInfo` a list with DB information: "name" (only mandatory field), "title", "description", "url", "version", "maintainer".
dataModel  a ReDaMoR::RelDataModel object
readParameters a list of parameters for reading the data file. (e.g. list(delim='\t', quoted_na=FALSE,))
collectionMembers the members of collections as provided to the collection_members<- function (default: NULL ==> no member).
check logical: if TRUE (default) the data are confronted to the data model
n_max maximum number of records to read for checks purpose (default: 10). See also ReDaMoR::confront_data().
verbose if TRUE display the data confrontation report (default: FALSE)
value new table names
.data a fileMDB object
...

Use new_name = old_name to rename selected tables

Value
A fileMDB object

See Also
- MDB methods: db_info, data_model, data_tables, collection_members, count_records, dims, filter_with_tables, as_fileMDB
- Additional general documentation is related to MDB.
- filter.fileMDB, slice.fileMDB

Examples

hpof <- read_fileMDB(
  path=system.file("examples/HPO-subset", package="ReDaMoR"),
  dataModel=system.file("examples/HPO-model.json", package="ReDaMoR"),
  dbInfo=list(
    "name"="HPO",
    "title"="Data extracted from the HPO database",
    "description"=paste(  
      "This is a very small subset of the HPO!",
      "Visit the reference URL for more information"  
    ),
    "url"="http://human-phenotype-ontology.github.io/"
  )
)
count_records(hpof)

## The following commands take time on fileMDB object
## Not run:
select(hpof, HPO_hp:HPO_diseases)
toTake <- "HPO_altId"
select(hpof, all_of(toTake))
hpoSlice <- slice(hpof, HPO_diseases=1:10)
count_records(hpoSlice)

if("stringr" %in% installed.packages()[,"Package"]){
  epilHP <- filter(
    hpof,
    HPO_diseases=stringr::str_detect(
      label, stringr::regex("epilepsy", ignore_case=TRUE)
    )
  )
  count_records(epilHP)
  label <- "Rolandic epilepsy"
  cn <- sym("label")
  reHP <- filter(
    hpof,
    HPO_diseases=!cn==!label
  )
}

## End(Not run)
Index

.format_bytes, 4
[TKCat (TKCat), 85
.chMDB (chMDB), 17
.fileMDB ($ chMDB), 89
.memoMDB (metaMDB), 69
.metaMDB (metaMDB), 72
.chMDB (chMDB), 17
.fileMDB ($ chMDB), 89
.memoMDB (metaMDB), 69
.metaMDB (metaMDB), 72
$.chMDB, 89
$.fileMDB ($ chMDB), 89
$.memoMDB (metaMDB), 69
$.metaMDB (metaMDB), 72
add_chMDB_user, 5
add_chTKCat_collection, 6
add_collection_member, 6
add_collection_member(), 82
add_feature_def, 7
add_helpers (add_helpers.MDB), 7
addHelpers.MDB, 7
add_km_feature, 8
add_km_feature(), 9
add_km_spec, 9
add_km_table, 9
add_property_values, 10
add_table_def, 10
add_table_features, 11
add_unit_def, 12
archive_chMDB, 12
as.list.MDB (MDB), 66
as_chMDB, 13
as_fileMDB, 18, 68, 70, 73, 90
as_fileMDB (as_fileMDB.chMDB), 13
as_KMR, 15
as_memoMDB, 16
c.MDB (MDB), 66
c.TKCat (TKCat), 85
ch_insert, 20
change_chTKCat_password, 16
check_chTKCat, 17
check_chTKCat(), 19
chMDB, 13, 17, 18, 30, 31, 36, 51, 56, 83
chTKCat, 5, 6, 12, 13, 16–18, 19, 23–25, 33–35, 40–42, 44–47, 51, 52, 56–59, 61, 62, 65, 77, 78, 80, 81, 86, 87
ClickHouseHTTP::ClickHouseHTTP(), 19
ClickHouseHTTP::dbConnect, ClickHouseHTTPDriver-method, 19
collection_members, 18, 68, 70, 73, 90
collection_members
(collection_members.TKCat), 20
collection_members.TKCat, 20
collection_members<-, 18, 69, 76, 90
collection_members<-
(collection_members.TKCat), 20
collection_members, 22
count_records, 18, 68, 70, 73, 90
count_records (count_records.MDB), 22
count_records.MDB, 22
create_chMDB, 23
create_chTKCat_user, 23
create_KMR, 24
create_POK, 25
data_file_size, 26
data_files, 25
data_model, 18, 68, 70, 73, 90
data_model (data_model.chMDB), 26
data_model.chMDB, 26
data_tables, 18, 68, 70, 73, 90
data_tables (data_tables.chMDB), 27
data_tables.chMDB, 27
db_disconnect (db_disconnect.chMDB), 28
db_disconnect(), 18, 19
db_disconnect.chMDB, 28
db_info, 18, 61, 68, 70, 73, 90
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_info(db_info.chMDB)</td>
<td>28</td>
</tr>
<tr>
<td>db_info.chMDB</td>
<td>28</td>
</tr>
<tr>
<td>db_info&lt;- (db_info.chMDB)</td>
<td>28</td>
</tr>
<tr>
<td>db_reconnect(db_reconnect.POK)</td>
<td>30</td>
</tr>
<tr>
<td>db_reconnect()</td>
<td>18, 19</td>
</tr>
<tr>
<td>db_reconnect.POK</td>
<td>30</td>
</tr>
<tr>
<td>db_tables</td>
<td>30</td>
</tr>
<tr>
<td>decode_bin</td>
<td>31</td>
</tr>
<tr>
<td>db_info.chMDB</td>
<td>28</td>
</tr>
<tr>
<td>db_reconnect()</td>
<td>18, 19</td>
</tr>
<tr>
<td>db_reconnect.POK</td>
<td>30</td>
</tr>
<tr>
<td>db_tables</td>
<td>30</td>
</tr>
<tr>
<td>decode_bin</td>
<td>31</td>
</tr>
<tr>
<td>dplyr::pull</td>
<td>67</td>
</tr>
<tr>
<td>dplyr::tibble</td>
<td>21, 27, 49, 60, 62–64</td>
</tr>
<tr>
<td>drop_chMDB</td>
<td>33</td>
</tr>
<tr>
<td>drop_chTKCat_user</td>
<td>33</td>
</tr>
<tr>
<td>empty_chMDB</td>
<td>34</td>
</tr>
<tr>
<td>encode_bin</td>
<td>34</td>
</tr>
<tr>
<td>explore_MDBs(explore_MDBs.TKCat)</td>
<td>35</td>
</tr>
<tr>
<td>explore_MDBs.TKCat</td>
<td>35</td>
</tr>
<tr>
<td>fileMDB</td>
<td>15, 25, 26, 37, 52, 66, 75, 76, 79, 83, 89, 90</td>
</tr>
<tr>
<td>fileMDB($chMDB)</td>
<td>89</td>
</tr>
<tr>
<td>filter.chMDB</td>
<td>18, 36</td>
</tr>
<tr>
<td>filter.fileMDB</td>
<td>37, 90</td>
</tr>
<tr>
<td>filter.memoMDB</td>
<td>37, 70</td>
</tr>
<tr>
<td>filter.metaMDB</td>
<td>38, 73</td>
</tr>
<tr>
<td>filter_mdb_matrix</td>
<td>38</td>
</tr>
<tr>
<td>filter_mdb_matrix.chMDB</td>
<td>38</td>
</tr>
<tr>
<td>filter_with_tables</td>
<td>18, 68, 70, 73, 90</td>
</tr>
<tr>
<td>filter_with_tables</td>
<td>39</td>
</tr>
<tr>
<td>filter_with_tables.chMDB</td>
<td>39</td>
</tr>
<tr>
<td>format.chTKCat</td>
<td>40</td>
</tr>
<tr>
<td>get_chMDB_metadata</td>
<td>41</td>
</tr>
<tr>
<td>get_chMDB_timestamps</td>
<td>41</td>
</tr>
<tr>
<td>get_chTKCat_collection</td>
<td>42</td>
</tr>
<tr>
<td>get_collection_mapper</td>
<td>42</td>
</tr>
<tr>
<td>get_collection_mapper()</td>
<td>67</td>
</tr>
<tr>
<td>get_confrontation_report</td>
<td>16, 43, 46, 73, 76</td>
</tr>
<tr>
<td>get_hosts(get_hosts.DBIConnection)</td>
<td>43</td>
</tr>
<tr>
<td>get_hosts.DBIConnection</td>
<td>43</td>
</tr>
<tr>
<td>get_km_spec</td>
<td>44</td>
</tr>
<tr>
<td>KMR</td>
<td>7, 10–12, 15, 25, 44, 46, 47, 60–64, 76</td>
</tr>
<tr>
<td>KMR</td>
<td>24</td>
</tr>
<tr>
<td>get_local_collection()</td>
<td>82</td>
</tr>
<tr>
<td>get_MDB</td>
<td>41</td>
</tr>
<tr>
<td>get_MDB(get_MDB.TKCat)</td>
<td>45</td>
</tr>
<tr>
<td>get_MDB()</td>
<td>44</td>
</tr>
<tr>
<td>get_MDB.TKCat</td>
<td>45</td>
</tr>
<tr>
<td>get_POK</td>
<td>46</td>
</tr>
<tr>
<td>get_query(get_query.chMDB)</td>
<td>46</td>
</tr>
<tr>
<td>get_query.chMDB</td>
<td>46</td>
</tr>
<tr>
<td>get_R_helpers(get_R_helpers.MDB)</td>
<td>47</td>
</tr>
<tr>
<td>get_R_helpers.MDB</td>
<td>47</td>
</tr>
<tr>
<td>get_shared_collections</td>
<td>48</td>
</tr>
<tr>
<td>get_shared_collections()</td>
<td>67</td>
</tr>
<tr>
<td>has_km_spec</td>
<td>48</td>
</tr>
<tr>
<td>heads(heads.chMDB)</td>
<td>49</td>
</tr>
<tr>
<td>heads.chMDB</td>
<td>49</td>
</tr>
<tr>
<td>import_collection_mapper</td>
<td>50</td>
</tr>
<tr>
<td>import_local_collection</td>
<td>50</td>
</tr>
<tr>
<td>init_chTKCat</td>
<td>51</td>
</tr>
<tr>
<td>is.chMDB</td>
<td>51</td>
</tr>
<tr>
<td>is.chTKCat</td>
<td>52</td>
</tr>
<tr>
<td>is.fileMDB</td>
<td>52</td>
</tr>
<tr>
<td>is.KMR</td>
<td>53</td>
</tr>
<tr>
<td>is.MDB</td>
<td>53</td>
</tr>
<tr>
<td>is.memoMDB</td>
<td>54</td>
</tr>
<tr>
<td>is.metaMDB</td>
<td>54</td>
</tr>
<tr>
<td>is.POK</td>
<td>55</td>
</tr>
<tr>
<td>is.TKCat</td>
<td>55</td>
</tr>
<tr>
<td>is_chMDB_public</td>
<td>56</td>
</tr>
<tr>
<td>is_current_chMDB</td>
<td>56</td>
</tr>
<tr>
<td>join_mdb_tables</td>
<td>57</td>
</tr>
<tr>
<td>KMR</td>
<td>7, 10–12, 15, 25, 44, 46, 47, 60–64, 76</td>
</tr>
<tr>
<td>KMR</td>
<td>24</td>
</tr>
<tr>
<td>length.MDB</td>
<td>66</td>
</tr>
<tr>
<td>lengths.MDB</td>
<td>66</td>
</tr>
<tr>
<td>list_chMDB_timestamps</td>
<td>57</td>
</tr>
<tr>
<td>list_chMDB_users</td>
<td>58</td>
</tr>
<tr>
<td>list_chTKCat_collections</td>
<td>59</td>
</tr>
<tr>
<td>list_chTKCat_users</td>
<td>59</td>
</tr>
<tr>
<td>list_feature_properties</td>
<td>60</td>
</tr>
<tr>
<td>list_local_collections</td>
<td>60</td>
</tr>
<tr>
<td>list_local_collections()</td>
<td>6, 42, 50, 87</td>
</tr>
<tr>
<td>list_MDBs(list_MDBs.TKCat)</td>
<td>61</td>
</tr>
<tr>
<td>list_MDBs.TKCat</td>
<td>61</td>
</tr>
<tr>
<td>list_measurement_units</td>
<td>62</td>
</tr>
</tbody>
</table>
list_measurements, 61
list_POKs, 62
list_property_values, 63
list_table_features, 64
list_table_types, 64
list_tables
(list_tables.DBIConnection), 63
list_tables.DBIConnection, 63
manage_chTKCat_users, 65
manage_chTKCat_users(), 36
map_collection_members, 65
map_collection_members(), 67
matrix, 27, 49
MDB, 6–11, 13, 17, 18, 24, 25, 35, 39, 40,
44–48, 53, 61, 66, 69, 70, 72, 73,
78–82, 85, 89, 90
MDBs, 68
memoMDB, 16, 36–38, 40, 54, 66, 68, 69, 70,
83–85
memoMDB(), 16
merge.MDB (MDB), 66
mergeTrees_from_RelTableModel, 71
mergeTrees_from_RelDataModel, 71
metaMDB, 30, 31, 38, 54, 57, 66, 68, 72, 73, 77, 84
names.MDB (MDB), 66
names<-chMDB (chMDB), 17
names<-.fileMDB ($chMDB), 89
names<-.memoMDB (memoMDB), 69
names<-.metaMDB (metaMDB), 72
names<-.TKCat (TKCat), 85
parse_R_helpers, 73
POK, 46, 55, 62
POK (create_POK), 25
pull.MDB (MDB), 66
RClickhouse::clickhouse(), 19
read_collection_members, 74
read_collection_members(), 66, 87
read_fileMDB, 75, 80
read_fileMDB(), 76
read_KMR, 76
readr::read_delim(), 75
ReDaMoR::confront_data(), 18, 43, 45, 76,
79, 90
ReDaMoR::format_confrontation_report,
16, 46, 73, 76
ReDaMoR::format_confrontation_report_md,
16, 46, 73, 76
ReDaMoR::RelDataModel, 18, 27, 69, 71, 72,
76, 90
ReDaMoR::RelTableModel, 71, 72
relational_tables, 77
remove_chMDB_user, 77
remove_chTKCat_collection, 78
rename.chMDB (chMDB), 17
rename.fileMDB ($chMDB), 89
rename.memoMDB (memoMDB), 69
rename.metaMDB (metaMDB), 72
rename.TKCat (TKCat), 85
rm_km_feature, 78
rm_km_table, 79
roxygen2::roxygen2-package, 73
scan_fileMDBs, 79, 85
search_MDB_fields
(search_MDB_fields.TKCat), 80
search_MDB_fields.TKCat, 80
search_MDB_tables
(search_MDB_tables.TKCat), 80
search_MDB_tables.TKCat, 80
select.MDB (MDB), 66
set_chMDB_access, 81
set_chMDB_timestamp, 81
shiny::addResourcePath, 36
shiny::div, 36
show_collection_def, 82
slice.chMDB, 18, 83
slice.fileMDB, 83, 90
slice.memoMDB, 70, 84
slice.metaMDB, 73, 84
TKCat, 25, 35, 44–47, 55, 61, 62, 80, 81, 85, 85
unarchive_chMDB, 86
update_chMDB_grants, 86
update_chTKCat_user, 87
write_collection_members, 87
write_MergeTree, 88