

# Package ‘TKCat’

June 5, 2025

**Type** Package

**Title** Tailored Knowledge Catalog

**Version** 1.1.14

**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.

**URL** <https://patzaw.github.io/TKCat/>, <https://github.com/patzaw/TKCat/>

**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** 2025-06-05 05:00:02 UTC

## Contents

.format_bytes . . . . .	4
add_chMDB_user . . . . .	5
add_chTKCat_collection . . . . .	6
add_collection_member . . . . .	6
add_feature_def . . . . .	7
add_helpers.MDB . . . . .	7
add_km_feature . . . . .	8
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_chMDB . . . . .	13
as_fileMDB.chMDB . . . . .	13
as_KMR . . . . .	15
as_memoMDB . . . . .	16
change_chTKCat_password . . . . .	16
check_chTKCat . . . . .	17
chMDB . . . . .	17
chTKCat . . . . .	19
ch_insert . . . . .	20
collection_members.TKCat . . . . .	20
compare_MDB . . . . .	22
count_records.MDB . . . . .	22
create_chMDB . . . . .	23
create_chTKCat_user . . . . .	23
create_KMR . . . . .	24
create_POK . . . . .	25
data_files . . . . .	25
data_file_size . . . . .	26
data_model.chMDB . . . . .	26
data_tables.chMDB . . . . .	27
db_disconnect.chMDB . . . . .	28
db_info.chMDB . . . . .	28
db_reconnect.POK . . . . .	30
db_tables . . . . .	30
decode_bin . . . . .	31
dims.chMDB . . . . .	31
drop_chMDB . . . . .	33
drop_chTKCat_user . . . . .	33
empty_chMDB . . . . .	34
encode_bin . . . . .	34
explore_MDBs.TKCat . . . . .	35
filter.chMDB . . . . .	36

filter.fileMDB . . . . .	37
filter.memoMDB . . . . .	37
filter.metaMDB . . . . .	38
filter_mdb_matrix.chMDB . . . . .	38
filter_with_tables.chMDB . . . . .	39
format.chTKCat . . . . .	40
get_chMDB_metadata . . . . .	41
get_chMDB_timestamps . . . . .	41
get_chTKCat_collection . . . . .	42
get_collection_mapper . . . . .	42
get_confrontation_report . . . . .	43
get_hosts.DBICConnection . . . . .	43
get_KMR . . . . .	44
get_km_spec . . . . .	44
get_local_collection . . . . .	45
get_MDB.TKCat . . . . .	45
get_POK . . . . .	46
get_query.chMDB . . . . .	46
get_R_helpers.MDB . . . . .	47
get_shared_collections . . . . .	48
has_km_spec . . . . .	48
heads.chMDB . . . . .	49
import_collection_mapper . . . . .	50
import_local_collection . . . . .	50
init_chTKCat . . . . .	51
is.chMDB . . . . .	51
is.chTKCat . . . . .	52
is.fileMDB . . . . .	52
is.KMR . . . . .	53
is.MDB . . . . .	53
is.memoMDB . . . . .	54
is.metaMDB . . . . .	54
is.POK . . . . .	55
is.TKCat . . . . .	55
is_chMDB_public . . . . .	56
is_current_chMDB . . . . .	56
join_mdb_tables . . . . .	57
list_chMDB_timestamps . . . . .	57
list_chMDB_users . . . . .	58
list_chTKCat_collections . . . . .	59
list_chTKCat_users . . . . .	59
list_feature_properties . . . . .	60
list_local_collections . . . . .	60
list_MDBs.TKCat . . . . .	61
list_measurements . . . . .	61
list_measurement_units . . . . .	62
list_POKs . . . . .	62
list_property_values . . . . .	63

list_tables.DBIconnection . . . . .	63
list_table_features . . . . .	64
list_table_types . . . . .	64
manage_chTKCat_users . . . . .	65
map_collection_members . . . . .	65
MDB . . . . .	66
MDBs . . . . .	68
memoMDB . . . . .	69
mergeTrees_from_RelDataModel . . . . .	71
mergeTree_from_RelTableModel . . . . .	71
metaMDB . . . . .	72
parse_R_helpers . . . . .	73
read_collection_members . . . . .	75
read_fileMDB . . . . .	75
read_KMR . . . . .	76
relational_tables . . . . .	77
remove_chMDB_user . . . . .	77
remove_chTKCat_collection . . . . .	78
rm_km_feature . . . . .	78
rm_km_table . . . . .	79
scan_fileMDBs . . . . .	79
search_MDB_fields.TKCat . . . . .	80
search_MDB_tables.TKCat . . . . .	81
set_chMDB_access . . . . .	81
set_chMDB_timestamp . . . . .	82
show_collection_def . . . . .	82
slice.chMDB . . . . .	83
slice.fileMDB . . . . .	83
slice.memoMDB . . . . .	84
slice.metaMDB . . . . .	85
TKCat . . . . .	85
unarchive_chMDB . . . . .	86
update_chMDB_grants . . . . .	87
update_chTKCat_user . . . . .	87
write_collection_members . . . . .	88
write_MergeTree . . . . .	88
\$.chMDB . . . . .	89

**Index****92**

---

`.format_bytes`*Format bytes numbers in human readable values*

---

**Description**

Format bytes numbers in human readable values

**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 <a href="#">chTKCat</a> object
json	a single character indicating the collection to import. Can be: <ul style="list-style-type: none"> <li>• a path to a file</li> <li>• the name of a local collection (see <a href="#">list_local_collections()</a>)</li> <li>• the json text defining the collection</li> </ul>
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 <a href="#">MDB</a> object
collection	a collection title in <a href="#">list_local_collections()</a>
table	the table providing the collection member
...	definition of the collection fields as lists (e.g. <code>be=list(static=TRUE, value="Gene")</code> or <code>organism=list(static=TRUE, value="Homo sapiens", type="Scientific name")</code> )

---

add_feature_def	<i>Add a feature definition to Knowledge Management Requirements (KMR)</i>
-----------------	--

---

### Description

Add a feature definition to Knowledge Management Requirements (KMR)

### Usage

```
add_feature_def(kmr, name, description, properties)
```

### Arguments

kmr	a <a href="#">KMR</a> 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: <ul style="list-style-type: none"> <li>• <b>type</b>: 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).</li> <li>• <b>description</b> (optional): a description of the feature property. Useful when the feature has more than one property</li> <li>• <b>mandatory</b>: a logical indicating the property is mandatory for the feature</li> <li>• <b>measurement</b> (optional and only for "integer" and "numeric" types): the name of the measurement for checking units</li> </ul>

### Value

The modified KMR [MDB](#) object

---

add_helpers.MDB	<i>Add a set of helper functions to a compatible object</i>
-----------------	---

---

### Description

Add a set of helper functions to a compatible object

**Usage**

```
## S3 method for class 'MDB'
add_helpers(x, code, name, language, kmr, ...)

## S3 method for class 'KMR'
add_helpers(x, code, name, language, ...)

add_helpers(x, code, name, language, ...)
```

**Arguments**

x	an object accepting helpers
code	file path to the source code of helper functions
name	the name of the helper set
language	the programming language of the code (default: "R")
kmr	an <a href="#">MDB</a> object with KM requirements
...	method specific parameters

**Value**

Return x with additional helpers

---

add_km_feature	<i>Add KM feature specifications to an <a href="#">MDB</a> object</i>
----------------	---

---

**Description**

Add KM feature specifications to an [MDB](#) object

**Usage**

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

**Arguments**

x	an <a href="#">MDB</a> object to update with specification tables
kmr	an <a href="#">MDB</a> 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	<i>Add empty KM specification tables to an <a href="#">MDB</a> object</i>
-------------	---

---

**Description**

Add empty KM specification tables to an [MDB](#) object

**Usage**

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

**Arguments**

x	an <a href="#">MDB</a> object to update with specification tables
kmr	an <a href="#">MDB</a> object with KM requirements

**Value**

An [MDB](#) object with empty KM specification tables

---

add_km_table	<i>Add KM table specifications to an <a href="#">MDB</a> object</i>
--------------	---

---

**Description**

Add KM table specifications to an [MDB](#) object

**Usage**

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

**Arguments**

x	an <a href="#">MDB</a> object to update with specification tables
kmr	an <a href="#">MDB</a> 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 <a href="#">add_km_feature()</a> 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 <a href="#">KMR</a> 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  
)
```

**Arguments**

kmr	a <a href="#">KMR</a> object
name	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 <a href="#">KMR</a> 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 <a href="#">KMR</a> 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 <a href="#">chTKCat</a> 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	<i>Push an <b>MDB</b> object in a ClickHouse database</i>
----------	---

---

**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 <b>MDB</b> object
tkcon	a <b>chTKCat</b> 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	<i>Write an <b>MDB</b> object</i>
------------------	-----------------------------------

---

**Description**

Write an **MDB** object

**Usage**

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

```
)

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

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

path	the path where the MDB should be written
readParameters	The following parameters are currently supported: <ul style="list-style-type: none"> <li>• <b>delim</b>: a single character used to separate fields within a record (default: <code>'\t'</code>)</li> <li>• <b>quoted_na</b>: a single logical indicating if missing values inside quotes should be treated as missing values or strings. <b>WARNING: THIS PARAMETER IS NOT TAKEN INTO ACCOUNT WITH readr&gt;=2.0.0.</b></li> <li>• <b>na</b>: String used for missing values. The default value for reading a fileMDB is "NA". But the default value for writing a fileMDB is ""&lt;NA&gt;"". This value is written in the DESCRIPTION.json file to avoid ambiguity when reading the fileMDB.</li> </ul>
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 <sup>5</sup> )
...	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	<i>Check a <a href="#">chTKCat</a> object</i>
---------------	---

---

**Description**

Check a [chTKCat](#) object

**Usage**

```
check_chTKCat(x, verbose = FALSE)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
verbose	a logical indicating if information messages should be displayed.

**Value**

Invisible result: [chTKCat](#) object

---

chMDB	<i>An <a href="#">MDB</a> (Modeled DataBase) relying on ClickHouse: <a href="#">chMDB</a></i>
-------	---

---

**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]]

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

```

### Arguments

tkcon	a <a href="#">chTKCat</a> object
dbTables	a named vector of tables in tkcon\$chcon with all(names(dbTables) %in% names(dataModel))
dbInfo	a list with DB information: <b>"name"</b> (only mandatory field), "title", "description", "url", "version", "maintainer".
dataModel	a <a href="#">ReDaMoR::RelDataModel</a> object
collectionMembers	the members of collections as provided to the <a href="#">collection_members&lt;-</a> 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 <a href="#">ReDaMoR::confront_data()</a> .
verbose	if TRUE display the data confrontation report
x	a chMDB object
value	new table names
.data	a <a href="#">chMDB</a> object
...	additional parameters
i	the index or the name of the tables to take

### Value

A chMDB object

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

### 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 <a href="#">Clickhouse settings</a>
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: <a href="#">ClickHouseHTTP::ClickHouseHTTP()</a> ; other supported driver: <a href="#">RClickhouse::clickhouse()</a> )
...	additional parameters for connection (see <a href="#">ClickHouseHTTP::dbConnect</a> , <a href="#">ClickHouseHTTPDriver-method</a> for the default driver)

**Value**

a chTKCat object

**See Also**

[check\\_chTKCat\(\)](#), [db\\_disconnect\(\)](#), [db\\_reconnect\(\)](#)

---

ch_insert	<i>Insert records by batches in a Clickhouse table</i>
-----------	--

---

**Description**

Insert records by batches in a Clickhouse table

**Usage**

```
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	<i>Collection members</i>
--------------------------	---------------------------

---

**Description**

Collection members

Collection members

**Usage**

```
## 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: <ul style="list-style-type: none"> <li>• <b>collection</b> (character): The name of the collection</li> <li>• <b>cid</b> (character): Collection identifier</li> <li>• <b>resource</b> (character): The name of the resource</li> <li>• <b>mid</b> (integer): The identifier of the member</li> <li>• <b>table</b> (character): The table recording collection information</li> <li>• <b>field</b> (character): The collection field.</li> <li>• <b>static</b> (logical): TRUE if the field value is common to all elements.</li> <li>• <b>value</b> (character): The name of the table column if static is FALSE or the field value if static is TRUE.</li> <li>• <b>type</b> (character): the type of the field. (not necessarily used ==&gt; NA if not)</li> </ul>

### 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)

---

 compare\_MDB

*Compare two MDB objects*


---

### Description

Compare two MDB objects

### Usage

```
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

---

 count\_records.MDB

*Count the number of records*


---

### Description

Count the number of records

### Usage

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

count_records(x, ...)
```

**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  
)
```

**Arguments**

x	a <code>chTKCat</code> 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 <code>chTKCat</code> 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

---

create_KMR	<i>Create an <code>MDB</code> object with Knowledge Management Requirements (KMR)</i>
------------	---

---

**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	<i>Create a piece of knowledge (POK) from an <a href="#">MDB</a> and a <a href="#">KMR</a> object</i>
------------	---

---

**Description**

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

**Usage**

```
create_POK(mdb, kmr, tkcat = NULL)
```

**Arguments**

mdb	a <a href="#">MDB</a> object with KM specifications
kmr	a <a href="#">KMR</a> object with KM requirements
tkcat	A <a href="#">TKCat</a> or <a href="#">chTKCat</a> 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	<i>Get the data files from a <a href="#">fileMDB</a> object</i>
------------	---

---

**Description**

Get the data files from a [fileMDB](#) object

**Usage**

```
data_files(x)
```

**Arguments**

x	a <a href="#">fileMDB</a> object
---	----------------------------------

**Value**

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

---

data_file_size	<i>Get the size of data files from a <a href="#">fileMDB</a> object</i>
----------------	---

---

**Description**

Get the size of data files from a [fileMDB](#) object

**Usage**

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

**Arguments**

x	a <a href="#">fileMDB</a> 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.chMDB	<i>Get object data model</i>
------------------	------------------------------

---

**Description**

Get object data model

**Usage**

```
## 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, ...)

## 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	<i>Get object data tables</i>
-------------------	-------------------------------

---

**Description**

Get object data tables

**Usage**

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

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

data_tables(x, ..., skip = 0, n_max = Inf)

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

## 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**

```
## S3 method for class 'chMDB'  
db_disconnect(x)  
  
## S3 method for class 'chTKCat'  
db_disconnect(x)  
  
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  
DB information

**Usage**

```
## 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, ...)  
  
db_info(x) <- value  
  
## 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: <ul style="list-style-type: none"><li>• <b>name</b>: a single character</li><li>• <b>title</b>: a single character</li><li>• <b>description</b>: a single character</li><li>• <b>url</b>: a single character</li><li>• <b>version</b>: a single character</li><li>• <b>maintainer</b>: a single character vector</li><li>• <b>size</b>: a numeric vector providing the size of the DB in bytes</li></ul>

### 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	<i>Reconnect an object to a database</i>
------------------	--

---

**Description**

Reconnect an object to a database

**Usage**

```
## 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	<i>Get the DB tables from a <a href="#">chMDB</a> or <a href="#">metaMDB</a> object</i>
-----------	---

---

**Description**

Get the DB tables from a [chMDB](#) or [metaMDB](#) object

**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`).

---

decode_bin	<i>Decode base64 string</i>
------------	-----------------------------

---

**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)

---

dims.chMDB	<i>Detailed information about the format of the tables</i>
------------	--

---

**Description**

Detailed information about the format of the tables

**Usage**

```
## 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 <sup>6</sup> )
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	<i>Drop a database from a <a href="#">chTKCat</a></i>
------------	---

---

**Description**

Drop a database from a [chTKCat](#)

**Usage**

```
drop_chMDB(x, name)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
name	the name of the database to remove

**Value**

No return value, called for side effects

---

drop_chTKCat_user	<i>Drop a user from a <a href="#">chTKCat</a> object</i>
-------------------	--

---

**Description**

Drop a user from a [chTKCat](#) object

**Usage**

```
drop_chTKCat_user(x, login)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
login	login of the user to drop

**Value**

No return value, called for side effects

---

empty_chMDB	<i>Empty a chMDB in a chTKCat</i>
-------------	-----------------------------------

---

**Description**

Empty a chMDB in a [chTKCat](#)

**Usage**

```
empty_chMDB(x, name, timestamp = NA)
```

**Arguments**

x	a <a href="#">chTKCat</a> 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	<i>Encode a binary file in a base64 string</i>
------------	--

---

**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 <a href="#">shiny::div</a> 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 <a href="#">shiny::addResourcePath</a>
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 <a href="#">manage_chTKCat_users()</a> ). If NULL (default), the functionality is not added.

**Value**

No return value, called for side effects

---

filter.chMDB	<i>Filter a <a href="#">chMDB</a> object and return a <a href="#">memoMDB</a></i>
--------------	---

---

**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 <a href="#">chMDB</a> object
...	each argument should have the name of one of the tables of the <a href="#">chMDB</a> 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	<i>Filter a <a href="#">fileMDB</a> object and return a <a href="#">memoMDB</a></i>
----------------	---

---

**Description**

Filter a [fileMDB](#) object and return a [memoMDB](#)

**Usage**

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

**Arguments**

.data	a <a href="#">fileMDB</a> object
...	each argument should have the name of one of the tables of the <a href="#">fileMDB</a> object and contain a simple logical expression involving the names of the corresponding table.
.preserve	not used

**Value**

a [memoMDB](#) object

---

filter.memoMDB	<i>Filter a <a href="#">memoMDB</a> object</i>
----------------	--

---

**Description**

Filter a [memoMDB](#) object

**Usage**

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

**Arguments**

.data	a <a href="#">memoMDB</a> object
...	each argument should have the name of one of the tables of the <a href="#">memoMDB</a> object and contain a simple logical expression involving the names of the corresponding table.
.preserve	not used

**Value**

a filtered [memoMDB](#) object

---

filter.metaMDB	<i>Filter a metaMDB object</i>
----------------	--------------------------------

---

**Description**

Filter a [metaMDB](#) object

**Usage**

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

**Arguments**

.data	a <a href="#">metaMDB</a> object
...	each argument should have the name of one of the tables of the <a href="#">metaMDB</a> 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	<i>Filter a matrix stored in an MDB</i>
-------------------------	---

---

**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 <a href="#">MDB</a> 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, ...)

filter_with_tables(x, tables, checkTables = TRUE, ...)

## 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 <a href="#">MDB</a> 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	<i>Format a <a href="#">chTKCat</a> object for printing</i>
----------------	---

---

**Description**

Format a [chTKCat](#) object for printing

**Usage**

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

**Arguments**

x	a <a href="#">chTKCat</a> 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**

```
get_chMDB_metadata(x, dbName, timestamp = NA)
```

**Arguments**

x	a <a href="#">chTKCat</a> 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**

```
get_chMDB_timestamps(x, name)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
name	the name of the database

**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 <a href="#">chTKCat</a> 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 <a href="#">list_local_collections()</a> ).
------------	---

**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**

```
get_confrontation_report()
```

**Value**

A confrontation report generated by `ReDaMoR::confront_data()`

---

`get_hosts.DBIconnection`*Get database hosts*

---

**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, ...)
```

```
get_hosts(x, ...)
```

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

**Arguments**

<code>x</code>	an object with database connection(s)
<code>...</code>	additional parameters for methods.

**Value**

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

---

get_KMR	<i>Get a <b>KMR</b> object from a <b>TKCat</b> or a <b>chTKCat</b> object</i>
---------	---

---

**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	<i>Get <b>KM</b> specifications from an <b>MDB</b> object</i>
-------------	---

---

**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

```
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

```
## 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\(\)](#).

**Value**

An [MDB](#) 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.

---

get\_POK

*Get a [POK](#) from a [chTKCat](#) connection or a [TKCat](#) object*

---

**Description**

Get a [POK](#) from a [chTKCat](#) connection or a [TKCat](#) object

**Usage**

```
get_POK(x, mdb, kmr)
```

**Arguments**

x	a <a href="#">chTKCat</a> or a <a href="#">TKCat</a> object
mdb	<a href="#">MDB</a> object with KM specifications or its name in tkcat
kmr	<a href="#">KMR</a> object with KM requirements or its name in tkcat

**Value**

A [POK](#) object

---

get\_query.chMDB

*Get SQL query*

---

**Description**

Get SQL query

**Usage**

```
## S3 method for class 'chMDB'
get_query(x, query, autoalias = TRUE, ...)

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

get_query(x, query, ...)
```

**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.
...	method specific parameters

**Value**

A tibble with query results

---

get_R_helpers.MDB	<i>Get a set of helper functions from an object</i>
-------------------	---

---

**Description**

Get a set of helper functions from an object

**Usage**

```
## 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 <a href="#">KMR</a> object
tkcat	A <a href="#">TKCat</a> or <a href="#">chTKCat</a> object to make available in helper environment
...	method specific parameters
mdb	An <a href="#">MDB</a> 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.

**Value**

Return a list of functions

---

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

---

has\_km\_spec

*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 <a href="#">MDB</a> object with specification tables
kmr	an <a href="#">MDB</a> object with KM requirements



**Value**

A logical: TRUE if the MDB has KM specifications

---

heads.chMDB	<i>Get the first records of each object data tables</i>
-------------	---

---

**Description**

Get the first records of each object data tables

**Usage**

```
## 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**

<code>collection</code>	the name of the targeted collection (it should belong to local collections: see <a href="#">list_local_collections()</a> ).
<code>fun</code>	a function which takes 2 data.frames (x and 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**

<code>txt</code>	a JSON string or file
<code>overwrite</code>	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	<i>Initialize a chTKCat database</i>
--------------	--------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x	a <a href="#">chTKCat</a> 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	<i>Check if the object is a <a href="#">chMDB</a> object</i>
----------	--

---

**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	<i>Check if an object represents Knowledge Management Requirements (KMR)</i>
--------	--

---

**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	<i>Check if the object is an <b>MDB</b> object</i>
--------	--

---

**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**

```
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**

```
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	<i>Is a chMDB public</i>
-----------------	--------------------------

---

**Description**

Is a chMDB public

**Usage**

```
is_chMDB_public(x, mdb)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
mdb	name of the modeled database

**Value**

A logical indicating if the chMDB is public or not.

---

is_current_chMDB	<i>Check if the <a href="#">chMDB</a> object refers to the current instance of the MDB</i>
------------------	--

---

**Description**

Check if the [chMDB](#) object refers to the current instance of the MDB

**Usage**

```
is_current_chMDB(x)
```

**Arguments**

x	a <a href="#">chMDB</a> object
---	--------------------------------

**Value**

A single logical: TRUE if x refers to the current instance of the MDB.



---

join_mdb_tables	<i>Join connected tables</i>
-----------------	------------------------------

---

**Description**

Join connected tables

**Usage**

```
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: <ul style="list-style-type: none"> <li>• "left": includes all rows of the first provided table</li> <li>• "right": includes all rows of the last provided table</li> <li>• "inner": includes all rows in all provided tables</li> <li>• "full": includes all rows in at least one provide table</li> </ul>
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	<i>List instance timestamps of an MDB in <a href="#">chTKCat</a></i>
-----------------------	--

---

**Description**

List instance timestamps of an MDB in [chTKCat](#)

**Usage**

```
list_chMDB_timestamps(x, name)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
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	<i>List users of an MDB of a <a href="#">chTKCat</a> object</i>
------------------	---

---

**Description**

List users of an MDB of a [chTKCat](#) object

**Usage**

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

**Arguments**

x	a <a href="#">chTKCat</a> object
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**

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

**Arguments**

<code>kmr</code>	a <a href="#">KMR</a> object
<code>feature</code>	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**

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

**Arguments**

<code>withJson</code>	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 <i>MDB</i>
-----------------	---------------------------

---

**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 <i>TKCat</i> related object (e.g. <i>chTKCat</i> )
withInfo	if TRUE (default), the function returns a table with <i>db_info</i> . If FALSE, it returns only <i>MDB</i> 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 <i>KMR</i> 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**

<code>kmr</code>	a <a href="#">KMR</a> object
<code>measurement</code>	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**

<code>x</code>	a <a href="#">chTKCat</a> or <a href="#">TKCat</a> object
<code>kmr</code>	<a href="#">KMR</a> object with KM requirements or its name in <code>tkcat</code>

**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 <a href="#">KMR</a> 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.DBIconnection  
*List tables in a clickhouse database*

---

**Description**

List tables in a clickhouse database

**Usage**

```
## S3 method for class 'DBIconnection'
list_tables(x, dbName = NULL, ...)

## S3 method for class 'chTKCat'
list_tables(x, dbName = NULL, ...)

list_tables(x, ...)
```

**Arguments**

x	an object with a clickhouse connection
dbName	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 <a href="#">KMR</a> 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 <a href="#">KMR</a> 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**

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

**Arguments**

x	a <a href="#">chTKCat</a> 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**

```
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.

<code>xm</code>	collection member x: a data.frame with the fields "field", "static", "value", "type" as returned by the <code>read_collection_members()</code> function.
<code>ym</code>	collection member y: a data.frame with the fields "field", "static", "value", "type" as returned by the <code>read_collection_members()</code> function.
<code>suffix</code>	the suffix to append to field names from x and y tables. Default: <code>c("_x", "_y")</code>
<code>fun</code>	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.
<code>...</code>	additional parameters for the fun function.

### Details

`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.

### Value

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.

---

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

```
## 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

<code>x</code>	an MDB object
<code>use.names</code>	return the names of the tables
<code>...</code>	additional parameters
<code>.data</code>	an MDB object
<code>var</code>	a variable specified as in <a href="#">dplyr::pull</a>
<code>name</code>	not used but kept for compatibility with the generic function
<code>y</code>	an MDB object
<code>by</code>	a tibble as returned by the <a href="#">get_shared_collections()</a> 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 <a href="#">map_collection_members()</a> function.
<code>dbInfo</code>	a list with DB information: <b>"name"</b> (only mandatory field), "title", "description", "url", "version", "maintainer".
<code>dmAutoLayout</code>	if TRUE (default) the layout of the merged data model is automatically adjusted.
<code>rtColor</code>	the color of the relational tables in the merged data model (default: "yellow")
<code>funs</code>	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 <a href="#">get_collection_mapper()</a> 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 `funcs` 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

memoMDB

*An **MDB** (Modeled DataBase) in memory: memoMDB***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 <b>ReDaMoR::RelDataModel</b> object
dbInfo	a list with DB information: " <b>name</b> " (only mandatory field), "title", "description", "url", "version", "maintainer".
collectionMembers	the members of collections as provided to the <b>collection_members&lt;-</b> 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 <a href="#">memoMDB</a> object
value	new table names
.data	a <a href="#">memoMDB</a> 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

```
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()
count_records(hpo)

## Too long on win-builder.r-project.org
## Not run:

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

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

```

    )
  )
  count_records(epiIHP)
}

## End(Not run)

```

---

```

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 <a href="#">ReDaMoR::RelDataModel</a> 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	<i>A metaMDB object</i>
---------	-------------------------

---

**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
```

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

**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: " <b>name</b> " (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 <a href="#">metaMDB</a> object
...	additional parameters
i	the index or the name of the tables to take

**Value**

A metaMDB object

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

**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.

---

parse\_R\_helpers

*Parse source code to get R helpers*

---

**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**

```
{  
  
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*

---

**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: <ul style="list-style-type: none"> <li>• <b>data</b>: a folder with the data</li> <li>• <b>DESCRIPTION.json</b>: a file with db information</li> <li>• <b>model</b>: a folder with the data model json file with the same name as the one given in the DESCRIPTION.json file</li> </ul>
dbInfo	a list or a json file with DB information: " <b>name</b> " (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 contains relevant parameters for the <code>readr::read_delim()</code> function. For example: <ul style="list-style-type: none"> <li>• <b>delim delimiter</b> (default: <code>'\t'</code>)</li> <li>• <b>quoted_na</b>: Should missing values inside quotes be treated as missing values or as strings or strings. <b>WARNING: THIS PARAMETER IS NOT TAKEN INTO ACCOUNT WITH readr&gt;=2.0.0.</b></li> <li>• <b>na</b>: String used for missing values. The default value for reading a fileMDB is "NA". But the default value for writing a fileMDB is "&lt;NA&gt;". This value is written in the DESCRIPTION.json file to avoid ambiguity when reading the fileMDB.</li> </ul>
dataModel	a <code>ReDaMoR::RelDataModel</code> object or json file. If NULL (default), the model json file found in path/model.
collectionMembers	the members of collections as provided to the <code>collection_members&lt;-</code> 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 <code>ReDaMoR::confront_data()</code> .
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(...)
```

**Arguments**

```
...           parameters for read\_fileMDB\(\)
```

**Value**

A [KMR](#) object

---

```
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 <a href="#">chTKCat</a> 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 <a href="#">chTKCat</a> 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 <a href="#">MDB</a> object to update with specification tables
kmr	an <a href="#">MDB</a> 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	<i>Remove KM table specifications from an <a href="#">MDB</a> object</i>
-------------	--

---

**Description**

Remove KM table specifications from an [MDB](#) object

**Usage**

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

**Arguments**

x	an <a href="#">MDB</a> object to update with specification tables
kmr	an <a href="#">MDB</a> 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	<i>Scan a catalog of <a href="#">fileMDB</a></i>
---------------	--

---

**Description**

Scan a catalog of [fileMDB](#)

**Usage**

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

**Arguments**

path	directory from which all the <a href="#">fileMDB</a> 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 <a href="#">ReDaMoR::confront_data()</a> .

**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**

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

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

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

**Arguments**

x	a <a href="#">TKCat</a> related object (e.g. <a href="#">chTKCat</a> )
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

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

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

```
search_MDB_tables(x, searchTerm)
```

### Arguments

x	a <a href="#">TKCat</a> related object (e.g. <a href="#">chTKCat</a> )
searchTerm	a single character with the term to search

### Value

An [MDB](#) object

---

```
set_chMDB_access
```

*Set chMDB access*

---

### Description

Set chMDB access

### Usage

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

### Arguments

x	a <a href="#">chTKCat</a> 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

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

### Arguments

x	a <a href="#">chTKCat</a> object
name	the name of the database to affect
timestamp	a single POSIXct value as a timestamp for the chMDB instance.

### Value

No return value, called for side effects

---

show\_collection\_def    *Show the definition of a collection*

---

### 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 <a href="#">get_local_collection()</a>
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	<i>Subset a <a href="#">chMDB</a> object according to row position in one table and return a <a href="#">memoMDB</a></i>
-------------	--

---

**Description**

Subset a [chMDB](#) object according to row position in one table and return a [memoMDB](#)

**Usage**

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

**Arguments**

.data	a <a href="#">chMDB</a> 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 <sup>5</sup> )
.preserve	not used

**Value**

a [memoMDB](#) object

---

slice.fileMDB	<i>Subset a <a href="#">fileMDB</a> object according to row position in one table and return a <a href="#">memoMDB</a></i>
---------------	--

---

**Description**

Subset a [fileMDB](#) object according to row position in one table and return a [memoMDB](#)

**Usage**

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

**Arguments**

.data	a <a href="#">fileMDB</a> 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	<i>Subset a <a href="#">memoMDB</a> object according to row position in one table</i>
---------------	---

---

**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 <a href="#">memoMDB</a> 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 <i>metaMDB</i> 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 <i>metaMDB</i> 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

---

TKCat	<i>TKCat</i> : a catalog of <i>MDB</i>
-------	--

---

**Description**

TKCat: a catalog of *MDB*

Rename a *TKCat* object

**Usage**

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

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

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

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

## 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 <a href="#">chTKCat</a> 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**

```
update_chTKCat_user(x, login, contact, admin, provider)
```

**Arguments**

x	a <a href="#">chTKCat</a> object
login	user login
contact	contact information (can be NA)
admin	a logical indicating if the user is an admin of the <a href="#">chTKCat</a> 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

```
write_collection_members(colMembers, path = NA, collection = NULL)
```

### Arguments

colMembers	A tibble as returned by <a href="#">read_collection_members()</a>
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 <a href="#">list_local_collections()</a> ).

### Value

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

---

```
write_MergeTree
```

*Write a Clickhouse [MergeTree](https://clickhouse.com/docs/en/engines/table-engines/mergetree-family/mergetree/MergeTree) table*

---

### Description

Write a Clickhouse **MergeTree** table

### Usage

```
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

---

\$.chMDB	<i>An <b>MDB</b> (Modeled DataBase) based on files: fileMDB</i>
----------	---

---

**Description**

An **MDB** (Modeled DataBase) based on files: fileMDB  
 Rename tables of a **fileMDB** object

**Usage**

```
## 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

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

```

### Arguments

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

### Value

A fileMDB object

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

### 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 (memoMDB), 69
- [.metaMDB (metaMDB), 72
- [[.chMDB (chMDB), 17
- [[.fileMDB (\$.chMDB), 89
- [[.memoMDB (memoMDB), 69
- [[.metaMDB (metaMDB), 72
- \$.chMDB, 89
- \$.fileMDB (\$.chMDB), 89
- \$.memoMDB (memoMDB), 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
- add\_helpers.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.chMDB (chMDB), 17
- as.list.fileMDB (\$.chMDB), 89
- as.list.MDB (MDB), 66
- as.list.metaMDB (metaMDB), 72
- as\_chMDB, 13
- as\_fileMDB, 18, 68, 70, 73, 90
- as\_fileMDB (as\_fileMDB.chMDB), 13
- 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, 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, 80–82, 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
- compare\_MDB, 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](#)
- db\_info(db\_info.chMDB), [28](#)
- db\_info.chMDB, [28](#)
- db\_info<- (db\_info.chMDB), [28](#)
- db\_reconnect (db\_reconnect.POK), [30](#)
- db\_reconnect(), [18](#), [19](#)
- db\_reconnect.POK, [30](#)
- db\_tables, [30](#)
- decode\_bin, [31](#)
- dims, [18](#), [70](#), [73](#), [90](#)
- dims (dims.chMDB), [31](#)
- dims.chMDB, [31](#)
- dplyr::pull, [67](#)
- dplyr::tibble, [21](#), [27](#), [49](#), [60](#), [62–64](#)
- drop\_chMDB, [33](#)
- drop\_chTKCat\_user, [33](#)
  
- empty\_chMDB, [34](#)
- encode\_bin, [34](#)
- explore\_MDBs (explore\_MDBs.TKCat), [35](#)
- explore\_MDBs.TKCat, [35](#)
  
- fileMDB, [15](#), [25](#), [26](#), [37](#), [52](#), [66](#), [68](#), [75](#), [76](#), [79](#), [80](#), [83](#), [84](#), [89](#), [90](#)
- fileMDB (\$.chMDB), [89](#)
- filter.chMDB, [18](#), [36](#)
- filter.fileMDB, [37](#), [90](#)
- filter.memoMDB, [37](#), [70](#)
- filter.metaMDB, [38](#), [73](#)
- filter\_mdb\_matrix  
     (filter\_mdb\_matrix.chMDB), [38](#)
- filter\_mdb\_matrix.chMDB, [38](#)
- filter\_with\_tables, [18](#), [68](#), [70](#), [73](#), [90](#)
- filter\_with\_tables  
     (filter\_with\_tables.chMDB), [39](#)
- filter\_with\_tables.chMDB, [39](#)
- format.chTKCat, [40](#)
  
- get\_chMDB\_metadata, [41](#)
- get\_chMDB\_timestamps, [41](#)
- get\_chTKCat\_collection, [42](#)
- get\_collection\_mapper, [42](#)
- get\_collection\_mapper(), [67](#)
- get\_confrontation\_report, [16](#), [43](#), [46](#), [73](#), [76](#)
- get\_hosts (get\_hosts.DBIconnection), [43](#)
- get\_hosts.DBIconnection, [43](#)
  
- get\_km\_spec, [44](#)
- get\_KMR, [44](#)
- get\_local\_collection, [45](#)
- get\_local\_collection(), [82](#)
- get\_MDB, [41](#)
- get\_MDB (get\_MDB.TKCat), [45](#)
- get\_MDB(), [44](#)
- get\_MDB.TKCat, [45](#)
- get\_POK, [46](#)
- get\_query (get\_query.chMDB), [46](#)
- get\_query.chMDB, [46](#)
- get\_R\_helpers (get\_R\_helpers.MDB), [47](#)
- get\_R\_helpers.MDB, [47](#)
- get\_shared\_collections, [48](#)
- get\_shared\_collections(), [67](#)
  
- has\_km\_spec, [48](#)
- heads (heads.chMDB), [49](#)
- heads.chMDB, [49](#)
  
- import\_collection\_mapper, [50](#)
- import\_local\_collection, [50](#)
- init\_chTKCat, [51](#)
- is.chMDB, [51](#)
- is.chTKCat, [52](#)
- is.fileMDB, [52](#)
- is.KMR, [53](#)
- is.MDB, [53](#)
- is.memoMDB, [54](#)
- is.metaMDB, [54](#)
- is.POK, [55](#)
- is.TKCat, [55](#)
- is\_chMDB\_public, [56](#)
- is\_current\_chMDB, [56](#)
  
- join\_mdb\_tables, [57](#)
  
- KMR, [7](#), [10–12](#), [15](#), [25](#), [44](#), [46](#), [47](#), [60–64](#), [76](#), [77](#)
- KMR (create\_KMR), [24](#)
  
- length.MDB (MDB), [66](#)
- lengths.MDB (MDB), [66](#)
- list\_chMDB\_timestamps, [57](#)
- list\_chMDB\_users, [58](#)
- list\_chTKCat\_collections, [59](#)
- list\_chTKCat\_users, [59](#)
- list\_feature\_properties, [60](#)
- list\_local\_collections, [60](#)
- list\_local\_collections(), [6](#), [42](#), [50](#), [88](#)

- list\_MDBs (list\_MDBs.TKCat), 61
- list\_MDBs.TKCat, 61
- list\_measurement\_units, 62
- 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, 86, 89, 90
- MDBs, 68
- memoMDB, 16, 36–38, 40, 54, 66, 68, 69, 69, 70, 83–85
- memoMDB(), 16
- merge.MDB (MDB), 66
- mergeTree\_from\_RelTableModel, 71
- mergeTrees\_from\_RelDataModel, 71
- metaMDB, 30, 31, 38, 54, 57, 66, 68, 72, 73, 77, 85
  
- 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, 75
- read\_collection\_members(), 66, 88
- read\_fileMDB, 75, 80
- read\_fileMDB(), 77
- read\_KMR, 76
- readr::read\_delim(), 76
  
- ReDaMoR::confront\_data(), 18, 43, 45, 76, 80, 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, 86
- search\_MDB\_fields
  - (search\_MDB\_fields.TKCat), 80
- search\_MDB\_fields.TKCat, 80
- search\_MDB\_tables
  - (search\_MDB\_tables.TKCat), 81
- search\_MDB\_tables.TKCat, 81
- select.MDB (MDB), 66
- set\_chMDB\_access, 81
- set\_chMDB\_timestamp, 82
- 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, 85
  
- TKCat, 25, 35, 44–47, 55, 61, 62, 80, 81, 85, 85, 86
  
- unarchive\_chMDB, 86
- update\_chMDB\_grants, 87
- update\_chTKCat\_user, 87
  
- write\_collection\_members, 88
- write\_MergeTree, 88