Package: dcmodifydb (via r-universe)

June 19, 2024

Title Modifying Rules on a DataBase

Version 0.3.1

Description Apply modification rules from R package 'dcmodify' to the database, prescribing and documenting deterministic data cleaning steps on records in a database. The rules are translated into SQL statements using R package 'dbplyr'.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Suggests testthat (>= 3.0.0), RSQLite, covr, knitr, rmarkdown

Imports dplyr, dbplyr, DBI, dcmodify (>= 0.1.9), validate, methods

URL https://github.com/data-cleaning/dcmodifydb

BugReports https://github.com/data-cleaning/dcmodifydb/issues

VignetteBuilder knitr

RoxygenNote 7.2.0

Config/testthat/edition 3

Repository https://edwindj.r-universe.dev

RemoteUrl https://github.com/data-cleaning/dcmodifydb

RemoteRef HEAD

RemoteSha dd34c40ebefd4febee335ed3805594eda8e237ff

Contents

dump_sql *Show generated sql*

Description

Writes the generated sql to a file or command line. The script contains ALTER and UPDATE statements and can be used for documentation purposes.

Usage

```
dump_sql(x, table, con = NULL, file = stdout(), ...)
```

Arguments

x	`dcmodify::modifier()` object with rules to be written
table	either a `dplyr::tbl()` object or a `character` with table name
con	optional, when `table` is a character, a dbi connection.
file	to which the sql will be written.
...	not used

Details

Note that when this script is run on the database it will change the original table. This differs from the default behavior of dcmodify which works on a (temporary) copy of the table.

Furthermore, it seems wise to wrap the generated SQL in a transaction when apply the SQL code on a database.

Value

`character` sql script with all statements.

See Also

Other sql translation: `modifier_to_sql()`

Examples

```
# load modification rules and apply:
library(dcmodify)

con <- DBI::dbConnect(RSQLite::SQLite(), dbname=system.file("db/person.db", package="dcmodifydb"))
person <- dplyr::tbl(con, "person")

rules <- modifier(.file = system.file("db/corrections.yml", package="dcmodifydb"))
print(rules)

# show sql code generated from the rules.
dump_sql(rules, person)
```

is_working_db *Rule check on the database*

Description

Get an indication of which R statement can be executed on the SQL database. dcmodifydb translates R statements into SQL statement. This works for many scenario's but not all R statements can be translated into SQL. This function checks whether a modification rule can be executed on the database.

Usage

```
is_working_db(m, tab, n = 2, warn = TRUE, sql_warn = FALSE)
```

Arguments

m	modifier() object
tab	tbl object
n	number of records to use in this check
warn	generate warnings for non-working rules
sql_warn	generate warnings with sql code for non-working rules

Value

logical with which statements are working

Examples

```
person <- dbplyr::memdb_frame(age = 12, salary = 3000)

library(dcmodify)

correction_rules <- modifier( if (age < 16) salary = 0
                            , if (retired == TRUE) salary = 0
                            )

# second rule is not working, because retired is not available
is_working_db(correction_rules, person, warn = FALSE)

# show warnings (default)
is_working_db(correction_rules, person, warn = TRUE)

# show the sql statements that are not working
is_working_db(correction_rules, person, warn = FALSE, sql_warn = TRUE)
```

modifier_to_sql *Utility function to extract SQL statements*

Description

Extract UPDATE statements from modifier object as a list of SQL statements. A user should normally be using `modify()` or `dump_sql()`, but this function may be useful.

Usage

```
modifier_to_sql(x, table, con = NULL)
```

Arguments

x	dcmodify::modifier() object
table	table object
con	optional connection

Value

list of sql UPDATE statements.

See Also

Other sql translation: `dump_sql()`

modify,ANY,modifier-method *Apply corrections/derivations to a db table*

Description

Change records in a database table using modification rules specified in a `modifier()` object. This is the main function of package dcmodifydb. For more information see the vignettes.

Usage

```
## S4 method for signature 'ANY,modifier'
modify(
  dat,
  x,
  copy = NULL,
  transaction = !isTRUE(copy),
  ignore_nw = FALSE,
  ...
)
```

Arguments

dat	tbl_sql() object, table in a SQL database
x	dcmodify::modifier() object.
copy	if TRUE (default), modify copy of table
transaction	if TRUE use one transaction for all modifications.
ignore_nw	if TRUE non-working rules are ignored
...	unused

Details

The modification rules are translated into SQL update statements and executed on the table.

Note that

- by default the updates are executed on a copy of the table.
- the default for transaction is FALSE when copy=TRUE and TRUE when copy=FALSE
- when transaction = TRUE and a modification fails, all modifications are rolled back.

Value

tbl_sql() object, referencing the modified table object.

Examples

```
library(DBI)
library(dcmodify)
library(dcmodifydb)

# silly modification rules
m <- modifier( if (cyl == 6)  gear <- 10
             , gear[cyl == 4] <- 0  # this R syntax works too :-)
             , if (gear == 3) cyl <- 2
             )

# setting up a table in the database
con <- dbConnect(RSQLite::SQLite())
dbWriteTable(con, "mtcars", mtcars[,c("cyl", "gear")])
tbl_mtcars <- dplyr::tbl(con, "mtcars")

# "Houston, we have a table"
head(tbl_mtcars)

# lets modify on a temporary copy of the table..
# this copy is only visible to the current connection
tbl_m <- modify(tbl_mtcars, m, copy=TRUE)


# and gear has changed...
head(tbl_m)
```

```
# If one certain about the changes, then you can overwrite the table with the changes
tbl_m <- modify(tbl_mtcars, m, copy=FALSE)

dbDisconnect(con)
```

---

person *Person data, income and smoking habits*

Description

A synthetic data set with person data with records to be corrected. The datasethas missing values

Usage

```
person
```

Format

A data frame with x rows and variables:

income monthly income, in US dollars

age age of a person in year

gender gender of a person

year year of measurement

smokes if a person smokes or not

cigarettes how many cigarretes a person smokes ...

The dataset is also available as a sqlite database at `system.file("db/person.db", package="dcmodifydb")`

Examples

```
# load modification rules and apply:
library(dcmodify)
rules <- modifier(.file = system.file("db/corrections.yml", package="dcmodifydb"))

con <- DBI::dbConnect(RSQLite::SQLite(), dbname=system.file("db/person.db", package="dcmodifydb"))
person <- dplyr::tbl(con, "person")
print(person)

person2 <- modify(person, rules, copy=TRUE)
print(person2)
```

Index