Package 'postGIStools'

May 3, 2019

Type Package

Title Tools for Interacting with 'PostgreSQL' / 'PostGIS' Databases

Description Functions to convert geometry and 'hstore' data types from 'PostgreSQL' into standard R objects, as well as to simplify the import of R data frames (including spatial data frames) into 'PostgreSQL'. Note: This package is deprecated. For new projects, we recommend using the 'sf' package to interface with geodatabases.

Version 0.2.3

License GPL-3

URL https://github.com/SESYNC-ci/postGIStools

BugReports https://github.com/SESYNC-ci/postGIStools/issues

Depends R (>= 3.2.3), sp (>= 1.2)

Imports DBI (>= 0.5), jsonlite (>= 0.9), methods, rgdal (>= 0.8), rgeos (>= 0.3), RPostgreSQL (>= 0.4), stringr (>= 1.0)

Suggests testthat, knitr, rmarkdown

LazyData TRUE

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Philippe Marchand [aut, cre], Richard Ellison [aut]

Maintainer Philippe Marchand <marchand.philippe@gmail.com>

Repository CRAN

Date/Publication 2019-05-03 19:50:28 UTC

R topics documented:

`postGIStools-package` *postGIStools: Tools for interacting with PostgreSQL / PostGIS Databases*

Description

postGIStools facilitates the import/export of data tables between R and PostgreSQL, in particular those with associated geometries (PostGIS extension) and hstore (key-value pairs) type columns.

Key Functions

`get_postgis_query` works like `dbGetQuery`, with the additional benefit of parsing hstore types (as a list-column in the resulting R data frame) and geometry types (producing a spatial data frame in R).

`%->%` reproduces the behavior of the PostgreSQL `hstore -> key` operator.

`postgis_insert` and `postgis_update` respectively insert new rows or update existing rows in a PostgreSQL table based on the contents of a R data frame. For spatial data and hstore columns, they performs the same conversions as `get_postgis_query`, in reverse.

`get_postgis_query` *Send SELECT query and parse geometry, hstore columns*

Description

This function is an extension of `dbGetQuery` that is useful in cases where selected columns include a PostgreSQL hstore, which is parsed as a list-column, and/or a PostGIS geometry, in which case the output is a spatial data frame (from the `sp` package).

Usage

```
get_postgis_query(conn, statement, geom_name = NA_character_,
  hstore_name = NA_character_)
```

Arguments

`conn`	A `PostgreSQLConnection-class` object, such as the output of `dbConnect`.
`statement`	Character string for a SQL SELECT query.
`geom_name`	Name of the geometry column (`NA` if none).
`hstore_name`	Name of the hstore column (`NA` if none).

Details

Conversion to spatial data frame objects will fail if there are NULL values in the geometry column, so these should be filtered out in the provided query statement.

Value

Either a data frame (if geom_name = NA) or a Spatial[Points/Lines/Polygons]DataFrame containing the query result. If a hstore column is present, it appears as a list-column in the data frame, i.e. each cell is a named list of key-value pairs.

References

The code for importing geom fields is based on a blog post by Lee Hachadoorian: Load PostGIS geometries in R without rgdal.

See Also

The %->% operator for working with hstore columns; postgis_insert and postgis_update for writing to a PostgreSQL connection.

Examples

```
## Not run:
library(RPostgreSQL)
con <- dbConnect(PostgreSQL(), dbname = "my_db")

# If geom column holds points, returns a SpatialPointsDataFrame
cities <- get_postgis_query(con, "SELECT name, geom, datalist FROM city",
                            geom_name = "geom", hstore_name = "datalist")

# Get the populations (part of datalist hstore) as a vector
pop <- cities@data$datalist %->% "population"

## End(Not run)
```

---

new_hstore *Create a empty hstore*

Description

This function creates an empty list of lists, which can be appended to a data frame as a hstore column.

Usage

```
new_hstore(nr)
```

Arguments

nr	Number of records.

Value

A empty hstore (list of lists) of length nr.

See Also

%->% to read or edit the resulting data structure.

Examples

```
contacts <- data.frame(name = c("Anne", "Bert", "Chris"))
contacts$phone <- new_hstore(3)
contacts$phone %->% "home" <- c("555-123-4567", "555-923-9134", "555-276-1123")
contacts$phone[2] %->% "cell" <- "555-889-9134"
str(contacts)
```

---

postgis_insert *Insert or update records in a PostgreSQL table from a R data frame.*

Description

These functions produce INSERT (postgis_insert) or UPDATE (postgis_update) queries to write data from a R data frame to a PostgreSQL table, with options to include a geometry layer and a list-column of key-value pairs (as a PostgreSQL hstore). The queries are passed to the database with dbSendQuery.

Usage

```
postgis_insert(conn, df, tbl, write_cols = NA,
  geom_name = NA_character_, hstore_name = NA_character_)

postgis_update(conn, df, tbl, id_cols, update_cols,
  geom_name = NA_character_, hstore_name = NA_character_,
  hstore_concat = TRUE)
```

Arguments

conn	A PostgreSQLConnection-class object, such as the output of dbConnect.
df	A data frame (if geom_name = NA) or Spatial[Points/Lines/Polygons]DataFrame.
tbl	Name of the PostgreSQL table to write to.
write_cols	A character vector, corresponding to the columns in df to insert in the database table. If NA, inserts all columns.

geom_name	Name of the geometry column in the database table (NA if none).
hstore_name	Name of the hstore column in both df and the database table (NA if none).
id_cols	A character vector, corresponding to the columns in df used to match records between df and the database table.
update_cols	A character vector, corresponding to the columns that must be updated in the database table based on values in df.
hstore_concat	If TRUE, hstore columns are updated by concatenation.

Details

All column names used in the query must match between the input data frame and the target database table (except for geom_name which only applies to the table).

postgis_update creates an *UPDATE ... SET ... FROM ...* query, which effectively joins the original table and input data frame based on matching values in id_cols, then updates the values in update_cols. The combination of id_cols must be unique in df, but they can be duplicated in the database table, in which case multiple rows are updated from a single row in df. Neither the geometry nor the hstore column can be used in id_cols.

Note that if hstore_concat = TRUE (the default), hstore columns are updated by *concatenation*, i.e. new keys are added, values associated with existing keys are updated, no keys are deleted. To overwrite whole hstore "cells", potentially deleting keys absent in df, set hstore_concat = FALSE.

Value

The result of dbSendQuery.

See Also

get_postgis_query for the inverse operation (read from database to R).

Examples

```
## Not run:
library(RPostgreSQL)
con <- dbConnect(PostgreSQL(), dbname = "my_db")

# Returns a SpatialPointsDataFrame
cities <- get_postgis_query(con, "SELECT name, geom, datalist FROM city",
                            geom_name = "geom", hstore_name = "datalist")

# Create a new field in hstore and update DB
cities@data$datalist %->% "pop_density" <-
   cities@data$datalist %->% "population" / cities@data$datalist %->% "area"
postgis_update(con, cities, "city",
               id_cols = "name", update_cols = "datalist",
               geom_name = "geom", hstore_name = "datalist")

# Add rows to DB with postgis_insert
# (new_cities is a SpatialPointsDataFrame with same columns as cities)
postgis_insert(con, new_cities, "city",
```

```
                geom_name = "geom", hstore_name = "datalist")

## End(Not run)
```

`%->%` *Extract or replace hstore values by key*

Description

Operator to get or set values corresponding to a given key for all records in a hstore.

Usage

```
hstore %->% key

hstore %->% key <- value
```

Arguments

`hstore`	A hstore (i.e. list of lists).
`key`	Character string corresponding to a key in `hstore`.
`value`	Vector of values of the same length as `hstore`.

Details

Based on the hstore "`->`" operator in PostgreSQL, the `%->%` operator returns values associated with a given key for all records in the hstore. The assignment version of the operator i.e. `hstore %->% key <- value` either creates a new key-value pair or, if they key exists, update the associated value. It can also delete a key by assigning its value to NULL.

Note that to subset the records in `hstore` to which the operator applies, you must use single brackets so that the result remains a list of lists. See below for usage examples.

Value

For the extract version, a vector of the same length as `hstore`, containing the value corresponding to `key` for each record (or `NA` if none). For the replace version, the modified hstore.

See Also

`new_hstore` to create a empty hstore.

Examples

```
contacts <- data.frame(name = c("Anne", "Bert", "Chris"))
contacts$phone <- new_hstore(3)
contacts$phone %->% "home" <- c("555-123-4567", "555-923-9134", "555-276-1123")
contacts$phone[2:3] %->% "home"

contacts$phone[2] %->% "home" <- NULL
contacts$phone %->% "home"
contacts$phone[2] %->% "cell" <- "555-889-9134"
contacts$phone %->% "cell"
```

Index