Package 'rpostgis'

January 9, 2014

Version 0.6

Date 2014-01-09

Title PostGIS and PostgreSQL related functions

Description This package provides additional functions to the RPostgreSQL package, mostly convenient wrappers, with some PostGIS oriented functions.

Author Mathieu Basille

Maintainer Mathieu Basille <basille@ase-research.org>

Depends RPostgreSQL,sp

License GPL (>= 3)

URL http://ase-research.org/basille/rpostgis

Collate 'rpostgis-package.r'

R topics documented:

pgAddKey *Add key*

Description

Add a primary or foreign key to a table column.

Usage

```
pgAddKey(conn, name, colname,
  type = c("primary", "foreign"), reference, colref,
  display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
colname	A character string specifying the name of the column to which the key will be assign.
type	The type of the key, either primary or foreign
reference	A character string specifying a foreign table name to which the foreign key will be associated.
colref	A character string specifying the name of the primary key in the foreign table to which the foreign key will be associated.
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-altertable.html

Examples

```
pgAddKey(name = c("fla", "bli"), colname = "id", type = "foreign",
    reference = c("flu", "bla"), colref = "id", exec = FALSE)
```

pgAsDate *Converts to timestamp*

Description

Convert a date field to a timestamp with or without time zone.

Usage

```
pgAsDate(conn, name, date = "date", tz = NULL,
  display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
date	A character string specifying the date field.
tz	A character string specifying the time zone, in "EST", "America/New_York", "EST5EDT", "-5".
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/datatype-datetime.html

Examples

```
pgAsDate(name = c("fla", "bli"), date = "date", tz = "GMT", exec = FALSE)
```

pgColumn *Add or remove a column*

Description

Add or remove a column to/from a table.

Usage

```
pgColumn(conn, name, colname, action = c("add", "drop"),
  coltype = "integer", cascade = FALSE, display = TRUE,
  exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
colname	A character string specifying the name of the column to which the key will be associated.
action	A character string specifying if the column is to be added ("add", default) or removed ("drop").
coltype	A character string indicating the type of the column, if action = "add".
cascade	Logical. Whether to drop foreign key constraints of other tables, if action = "drop".
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-altertable.html

Examples

```
## Add an integer column
pgColumn(name = c("fla", "bli"), colname = "field", exec = FALSE)
## Drop a column (with CASCADE)
pgColumn(name = c("fla", "bli"), colname = "field", action = "drop",
    cascade = TRUE, exec = FALSE)
```

pgComment *Comment table/view/schema*

Description

Comment on a table, a view or a schema.

Usage

```
pgComment(conn, name, comment,
  type = c("table", "view", "schema"), display = TRUE,
  exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table, view or schema name.
comment	A character string specifying the comment.
type	The type of the object to comment, either table or view
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-comment.html

Examples

```
pgComment(name = c("fla", "bli"), comment = "Comment on a view.",
    type = "view", exec = FALSE)
pgComment(name = "fla", comment = "Comment on a schema.", type = "schema",
    exec = FALSE)
```

pgDrop *Drop table/view/schema*

Description

Drop a table, a view or a schema.

Usage

```
pgDrop(conn, name, type = c("table", "view", "schema"),
  ifexists = FALSE, cascade = FALSE, display = TRUE,
  exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table, view or schema name.
type	The type of the object to comment, either table or view
ifexists	Do not throw an error if the table does not exist. A notice is issued in this case.
cascade	Automatically drop objects that depend on the table (such as views).
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-droptable.html, http://www.postgresql.org/docs/current/static/sql-dropview.html, http://www.postgresql.org/docs/current/static/sql-dropschema.html

Examples

```
pgDrop(name = c("fla", "bli"), type = "view", exec = FALSE)
pgDrop(name = "fla", type = "schema", cascade = "TRUE", exec = FALSE)
```

pgGetPts *Retrieve point geometries*

Description

Retrieve point geometries from a PostGIS table, and convert it to a SpatialPoints or a SpatialPointsDataFrame.

Usage

```
pgGetPts(conn, name, pts = "pts_geom", colname = NULL,
  include = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table, view or schema name.
pts	The name of the point geometry column.
colname	The name of the columns to include or exclude (defaults to NULL, i.e. no column).
include	Include or exclude colname (default TRUE). If colname = NULL and include = FALSE, all columns are retrieved.

Value

A SpatialPoints or a SpatialPointsDataFrame

Author(s)

Mathieu Basille <basille@ase-research.org>

Examples

```
## Not run:
## Retrieve only the points in the column 'pts_geom'
pgGetPts(conn, c("fla", "bli"))
## Return a SpatialPointsDataFrame with columns c1 & c2 as data
pgGetPts(conn, c("fla", "bli"), colname = c("c1", "c2"))
## Return a SpatialPointsDataFrame with every column except c1 & c2 as
## data
pgGetPts(conn, c("fla", "bli"), colname = c("c1", "c2"), include = FALSE)
## Return a SpatialPointsDataFrame with every column as data
pgGetPts(conn, c("fla", "bli"), include = FALSE)
## End(Not run)
```

pgIndex *CREATE INDEX*

Description

Defines a new index.

Usage

```
pgIndex(conn, name, colname, idxname, unique = FALSE,
  method = c("btree", "hash", "rtree", "gist"),
  display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
colname	A character string specifying the name of the column to which the key will be associated.
idxname	A character string specifying the name of the index to be created. By default, this is the name of the table (without the schema) suffixed by _idx.
unique	Logical. Causes the system to check for duplicate values in the table when the index is created (if data already exist) and each time data is added. Attempts to insert or update data which would result in duplicate entries will generate an error.
method	The name of the method to be used for the index. Choices are "btree", "hash", "rtree", and "gist". The default method is btree.
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-createindex.html; the PostGIS documentation for GiST indexes: http://postgis.net/docs/using_postgis_dbmanagement.html#id541286

Examples

```
pgIndex(name = c("fla", "bli"), colname = "wkb_geometry", method = "gist",
    exec = FALSE)
```

pgMakePts *Add a POINT or LINESTRING geometry field.*

Description

Add a new POINT or LINESTRING geometry field.

Usage

```
pgMakePts(conn, name, colname = "pts_geom", x = "x",
  y = "y", srid, index = TRUE, display = TRUE,
  exec = TRUE)

pgMakeStp(conn, name, colname = "stp_geom", x = "x",
  y = "y", dx = "dx", dy = "dy", srid, index = TRUE,
  display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
colname	A character string specifying the name of the new geometry column.
x	The name of the x/longitude field.
y	The name of the y/latitude field.
dx	The name of the dx field (i.e. increment in x direction).
dy	The name of the dy field (i.e. increment in y direction).
srid	A valid SRID for the new geometry.
index	Logical. Whether to create an index on the new geometry.
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostGIS documentation for ST_MakePoint: http://postgis.net/docs/ST_MakePoint.html, and for ST_MakeLine: http://postgis.net/docs/ST_MakeLine.html, which are the main functions of the call.

Examples

```
## Create a new POINT field called "pts_geom"
pgMakePts(name = c("fla", "bli"), x = "longitude", y = "latitude",
    srid = 4326, exec = FALSE)

## Create a new LINESTRING field called "stp_geom"
pgMakeStp(name = c("fla", "bli"), x = "longitude", y = "latitude",
    dx = "xdiff", dy = "ydiff", srid = 4326, exec = FALSE)
```

---

pgSchema *Create schema*

Description

Create a schema.

Usage

```
pgSchema(conn, name, display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL schema name.
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-createschema.html

Examples

```
pgSchema(name = "schema", exec = FALSE)
```

pgVacuum *VACUUM*

Description

Performs a VACUUM (garbage-collect and optionally analyze) on a table.

Usage

```
pgVacuum(conn, name, full = FALSE, verbose = FALSE,
  analyze = TRUE, display = TRUE, exec = TRUE)
```

Arguments

conn	A connection object.
name	A character string specifying a PostgreSQL table name.
full	Logical. Whether to perform a "full" vacuum, which can reclaim more space, but takes much longer and exclusively locks the table.
verbose	Logical. Whether to print a detailed vacuum activity report for each table.
analyze	Logical. Whether to update statistics used by the planner to determine the most efficient way to execute a query (default to TRUE).
display	Logical. Whether to display the query (defaults to TRUE).
exec	Logical. Whether to execute the query (defaults to TRUE).

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

The PostgreSQL documentation: http://www.postgresql.org/docs/current/static/sql-vacuum.html

Examples

```
pgVacuum(name = c("fla", "bli"), full = TRUE, exec = FALSE)
```

rpostgis *PostGIS and PostgreSQL functions*

Description

rpostgis

Details

This package provides additional functions to the RPostgreSQL package, mostly convenient wrappers to PostgreSQL queries, with some PostGIS oriented functions. For a list of documented functions, use library(help = "rpostgis")

Author(s)

Mathieu Basille <basille@ase-research.org>

Index