Package: lazysql (via r-universe)

October 23, 2024

Type Package

Title Lazy SQL Programming

Version 0.1.3.9000

Date 2016-03-11

Description Helper functions to build SQL statements under program control for dbGetQuery, dbSendQuery, sqldf, etc. They are intended to increase speed of coding and to reduce coding errors. Arguments are carefully checked, in particular SQL identifiers such as names of tables or columns. More patterns will be added as required.

URL https://github.com/UweBlock/lazysql

BugReports https://github.com/UweBlock/lazysql/issues

License MIT + file LICENSE

LazyData TRUE

Imports checkmate (>= 1.7.2), magrittr, plyr

Suggests testthat

RoxygenNote 5.0.1

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

RemoteUrl https://github.com/uweblock/lazysql

RemoteRef HEAD

RemoteSha c6622f65d2a12ef434e6e5955f56f105b78956a7

Contents

`date_between` *Create SQL string to select date between two given dates*

Description

Create string with SQL `BETWEEN` expression for `WHERE` clause to select dates within the given range.

Usage

```
date_between(column_name, date_range)
```

Arguments

`column_name`	[character(1)] Name of data base column to select dates from.
`date_range`	[Date(1:2)] One or two dates giving the date range in which the dates should be enclosed (closed interval). If only one date is given, it is taken for both upper and lower limits.

Details

`column_name` must be a valid SQL identifier. It is validated to conform to the regular expression returned by `valid_identifier_regex`.

Value

Character string to be used in SQL statement.

Author(s)

Uwe Block

See Also

`valid_identifier_regex`.

Examples

```
date1 <- as.Date("2016-02-22")
date2 <- as.Date("2016-02-11")

# SQL expression for a date range
(sql_expr1 <- lazysql::date_between("STD_1", c(date1, date2)))

# SQL expression for a single date
(sql_expr2 <- lazysql::date_between("STD_1", date1))
```

```
# sample SQL statements
paste("select * from TEST_TABLE where", sql_expr1)

paste("select * from TEST_TABLE where", sql_expr2)
```

`in_condition` *Create SQL string to select values included in a set of given values*

Description

Create string with SQL `IN` expression for `WHERE` clause to select values included in a set of given values.

Usage

```
in_condition(column_name, choices, negation = c("", "not"))
```

Arguments

`column_name`	[character(1)] Name of data base column to select values from.
`choices`	[character(1:Inf)] or [integer(1:Inf)] The values which must be matched. Character values must not contain any single or double quotes to avoid problems with SQL syntax and for safety reasons.
`negation`	[character(1)] If `"not"` the selection is inverted to a `NOT IN` expression.

Details

`column_name` must be a valid SQL identifier. It is validated to conform to the regular expression returned by `valid_identifier_regex`.

Value

Character string to be used in SQL statement.

Author(s)

Uwe Block

See Also

`valid_identifier_regex`.

Examples

```
# SQL expressions
lazysql::in_condition("COL_1", 1:3)

lazysql::in_condition("COL_1", 1:3, "not")

lazysql::in_condition("COL_1", LETTERS[2:3])

lazysql::in_condition("COL_1", LETTERS[2:3], "not")
```

---

lazysql *Lazy SQL programming*

Description

Helper functions to build SQL statements under program control for dbGetQuery, dbSendQuery, sqldf, etc. They are intended to increase speed of coding and to reduce coding errors. Arguments are carefully checked, in particular SQL identifiers such as names of tables or columns. More patterns will be added as required.

Author(s)

Uwe Block

See Also

date_between, in_condition, natural_key

---

natural_key *Create SQL string for joining on matching natural keys*

Description

Create string with SQL expressions for WHERE clause to join two tables on the given columns.

Usage

```
natural_key(table_names, key_columns)
```

Arguments

table_names	[character(2)] Name of data base tables to be joined.
key_columns	[character(1:Inf)] Names of key columns in both tables.

Details

The names of tables and key columns must be valid SQL identifiers. They are validated to conform to the regular expression returned by valid_identifier_regex.

The SQL string is created in 3 steps:

1. Combine table names with key names, eg, "PRL.FLIGHT_NR".
2. Create logical expressions, eg, "PRL.FLIGHT_NR = PRL_SSR.FLIGHT_NR"
3. Concatenate logical expressions by "and" to form final SQL esxpression.

Value

Character string to be used in SQL statement.

Note

The current implementation assumes that key columns have the same names in both tables.

Author(s)

Uwe Block

See Also

valid_identifier_regex.

Examples

```
# SQL expression
(sql_expr <- lazysql::natural_key(c("TAB1", "tab_2"),c("COL1", "col_2")))

# sample SQL JOIN statement
paste("select * from TAB1, TAB2 where", sql_expr)
```

---

valid_identifier_regex

Regex pattern to validate SQL identifier names

Description

Returns a regular expression to validate unquoted SQL identifiers.

Usage

```
valid_identifier_regex()
```

Details

Valid SQL identifiers must begin with an alphabetic character followed by alphanumeric characters or underscores "_".

Value

Character string with regular expression.

Note

The current implementation doesn't allow any other special characters in SQL identfiers or quoted SQL identifiers for safety reasons. In future releases, valid SQL identifiers might be defined depending on the target database system.

Author(s)

Uwe Block

References

ORACLE Database SQL Language Reference.

Examples

```
lazysql::valid_identifier_regex()
```

Index