Package ‘RODBCext’

February 5, 2020

Version 0.3.2
Title Parameterized Queries Extension for RODBC
Description An extension for RODBC package adding support for parameterized queries.
URL https://github.com/zozlak/RODBCext
SystemRequirements An ODBC3 driver manager and drivers.
Depends R (>= 3.0.0), RODBC (>= 1.3.0)
Suggests knitr, rmarkdown, testthat
LazyLoad yes
Biarch yes
License GPL-2 | GPL-3
Maintainer Mateusz Zoltak <zozlak@zozlak.org>
NeedsCompilation yes
VignetteBuilder knitr
RoxygenNote 6.1.1
Author Mateusz Zoltak [aut, cre],
Brian Ripley [aut],
Michael Lapsley [aut],
Will Beasley [ctb],
Juergen Altfeld [ctb]
Repository CRAN
Date/Publication 2020-02-05 19:40:02 UTC

R topics documented:

odbcFetchRows .................................................. 2
odbcGetQueryTimeout ........................................... 2
odbcSetQueryTimeout ............................................ 3
sqlExecute ....................................................... 4
sqlFetchMore ..................................................... 6
sqlPrepare ...................................................... 6
odbcGetQueryTimeout

Index 8

odbcFetchRows Overlay over odbcFetchRows

Description
RODBC::odbcFetchRows crashes if the ODBC channel is in "query prepared but already not executed" state. This function is a small overlay emitting an error in such a case.

Usage
odbcFetchRows(channel, ...)

Arguments
- channel ODBC connection obtained by odbcConnect
- ... other parameters passed to odbcFetchRows

Value
see odbcFetchRows

odbcGetQueryTimeout Gets the current query timeout of a prepared query

Description
A query has to be already prepared using SQLPrepare()
Throws an error if an error occurred

Usage
odbcGetQueryTimeout(channel)

Arguments
- channel an RODBC channel containing an open connection

Value
The current query timeout value in seconds. 0 means "no timeout"

See Also
odbcSetQueryTimeout, odbcConnect, odbcDriverConnect
odbcSetQueryTimeout

Examples

```r
## Not run:
conn = odbcConnect('MyDataSource')
sqlPrepare(conn, "SELECT * FROM myTable WHERE column = ?")
odbcGetQueryTimeout(conn)  # shows the current query timeout of the prepared statement
sqlExecute(conn, 'myValue')
sqlFetchMore(conn)

## End(Not run)
```

doctorSetQueryTimeout Sets the query timeout of a prepared query

Description

A query has to be already prepared using SQLPrepare()

Throws an error if any error occurred

Usage

```r
odbcSetQueryTimeout(channel, timeout = 0)
```

Arguments

- `channel` an open RODBC channel (connection)
- `timeout` the new query timeout value in seconds (0 means "no timeout")

Value

- `0` = success, `1` = success but with an info message,

Note

Not all drivers will support a query timeout. You may get an error then or the query timeout values remains unchanged silently.

See Also

`odbcGetQueryTimeout`, `odbcConnect`, `odbcDriverConnect`
sqlExecute

## Not run:
```r
conn = odbcConnect('MyDataSource')
sqlPrepare(conn, "SELECT * FROM myTable WHERE column = ?")
odbcSetQueryTimeout(conn, 120)  # sets the query timeout of the prepared statement
sqlExecute(conn, 'myValue')
sqlFetchMore(conn)
## End(Not run)
```

### sqlExecute

Executes an already prepared query

## Description

Executes a parameterized query.

Optionally (fetch=TRUE) fetches results using `sqlGetResults`.

Optionally (query=NULL) uses query already prepared by `sqlPrepare`.

## Usage

```r
sqlExecute(channel, query = NULL, data = NULL, fetch = FALSE,
errors = TRUE, rows_at_time = attr(channel, "rows_at_time"),
force_loop = FALSE, query_timeout = NULL, ...)
```

## Arguments

- `channel` ODBC connection obtained by `odbcConnect`
- `query` a query string (NULL if query already prepared using `sqlPrepare`)
- `data` data to pass to `sqlExecute` (as data.frame)
- `fetch` whether to automatically fetch results (if data provided)
- `errors` whether to display errors
- `rows_at_time` number of rows to fetch at one time - see details of `sqlQuery`
- `force_loop` whether to execute queries in the explicit loop with separate query planing for each iteration (usefull if executing a query invalidates its plan, e.g. EXEC queries on Ms SQL Server)
- `query_timeout` the query timeout value in seconds (0 means "no timeout", NULL does not change the default value)
- `...` parameters to pass to `sqlGetResults` (if fetch=TRUE)
sqlExecute

Details
Return value depends on the combination of parameters:

- if there were errors during query preparation or execution or fetching results return value depends on errors parameter - if errors=TRUE error is thrown, otherwise -1 will be returned
- if fetch=FALSE and there were no errors invisible(1) will be returned
- if fetch=TRUE and there were no errors a data.frame with results will be returned

Value
see details

Examples
```r
## Not run:
conn = odbcConnect('MyDataSource')

# prepare, execute and fetch results separately
sqlPrepare(conn, "SELECT * FROM myTable WHERE column = ?")
sqlExecute(conn, NULL, 'myValue')
sqlGetResults(conn)

# prepare and execute at one time, fetch results separately
sqlExecute(conn, "SELECT * FROM myTable WHERE column = ?", 'myValue')
sqlGetResults(conn)

# prepare, execute and fetch at one time
sqlExecute(conn, "SELECT * FROM myTable WHERE column = ?", 'myValue', TRUE)

# prepare, execute and fetch at one time using multiple wildcards for data passthrough
sqlExecute(
  conn,
  query="SELECT * FROM table WHERE column1 = ? AND column2 = ?",
  data=data.frame('column1value', 'column2value'),
  fetch=TRUE
)

# prepare, execute and fetch at one time, pass additional parameters to sqlFetch()
sqlExecute(
  conn,
  "SELECT * FROM myTable WHERE column = ?",
  'myValue',
  fetch = TRUE,
  stringsAsFactors = FALSE
)

# prepare, execute and fetch at one time using a query timeout value
sqlExecute(conn, "SELECT * FROM myTable WHERE column = ?", 'myValue', TRUE, query_timeout=45)

# execute a simple statement without parameters using a query timeout value
sqlExecute(conn, "SELECT * FROM myTable", fetch = TRUE, query_timeout = 60)
```
## sqlPrepare

### Description

Prepares a query for execution.

### Usage

```r
sqlPrepare(channel, query, errors = TRUE, query_timeout = NULL)
```

### Arguments

- `channel`: ODBC connection obtained by `odbcConnect`
- `query`: query string
- `errors`: whether to display errors
- `query_timeout`: the query timeout value in seconds (0 means "no timeout", NULL does not change the default value)

### Value

`invisible(1)` on success, `-1` or an error (depending on `errors` parameter) on error

## sqlFetchMore

### Overlay over sqlFetchMore

### Description

RODBC::sqlFetchMore crashes if the ODBC channel is in "query prepared but already not executed" state. This function is a small overlay emitting an error in such a case.

### Usage

```r
sqlFetchMore(channel, ...)
```

### Arguments

- `channel`: ODBC connection obtained by `odbcConnect`
- `...`: other parameters passed to `sqlFetchMore`

### Value

`see sqlFetchMore`
Examples

```r
## Not run:
  conn = odbcConnect('MyDataSource')

  sqlPrepare(conn, "SELECT * FROM myTable WHERE column = ?")
  sqlExecute(conn, NULL, 'myValue')
  sqlFetchMore(conn)

  # with a query timeout
  sqlPrepare(conn, "SELECT * FROM myTable WHERE column = ?", query_timeout=60)
  sqlExecute(conn, data='myValue', fetch=TRUE)

## End(Not run)
```
Index

odbcConnect, 2–4, 6
odbcDriverConnect, 2, 3
odbcFetchRows, 2, 2
odbcGetQueryTimeout, 2, 3
odbcSetQueryTimeout, 2, 3

sqlExecute, 4
sqlFetchMore, 6, 6
sqlGetResults, 4
sqlPrepare, 4, 6
sqlQuery, 4