Package 'RODBCext'

July 12, 2015

Version 0.2.5

Title Parameterized Queries Extension for RODBC

Description An extension for RODBC package adding support for parameterized
queries.

SystemRequirements An ODBC3 driver manager and drivers.

Depends R (>= 3.0.0), RODBC (>= 1.3.0)

Suggests knitr

LazyLoad yes

Biarch yes

License GPL-2 | GPL-3

Author Mateusz Zoltak [aut, cre],
Brian Ripley [aut],
Michael Lapsley [aut]

Maintainer Mateusz Zoltak <zozlak@zozlak.org>

NeedsCompilation yes

VignetteBuilder knitr

Repository CRAN

Date/Publication 2015-07-12 11:06:21

R topics documented:

---

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 emmitting an error in such a case.

Usage

```
odbcFetchRows(channel, ...)
```

Arguments

channel	ODBC connection obtained by [odbcConnect](#)
...	other parametrs passed to [odbcFetchRows](#)

Value

see [odbcFetchRows](#)

---

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

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

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)
...	parameters to pass to sqlGetResults (if fetch=TRUE)

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 datails

Examples

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

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

## End(Not run)
```

---

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 emmitting an error in such a case.

Usage

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

Arguments

channel	ODBC connection obtained by [odbcConnect](#)
...	other parametrs passed to [sqlFetchMore](#)

Value

see [sqlFetchMore](#)

---

sqlPrepare *Prepares a query for execution*

Description

Prepares a query for execution.

Usage

```
sqlPrepare(channel, query, errors = TRUE)
```

Arguments

channel	ODBC connection obtained by [odbcConnect](#)
query	query string
errors	whether to display errors

Value

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

Examples

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

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

## End(Not run)
```

Index