

Package ‘rethinker’

October 14, 2022

Type Package

Title RethinkDB Client

Version 1.1.0

Author Miron B. Kursa

Maintainer Miron B. Kursa <M.Kursa@icm.edu.pl>

Description Simple, native 'RethinkDB' client.

URL <https://github.com/mbq/rethinker>

Suggests testthat

Imports rjson

License GPL-3

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-11-13 11:20:29 UTC

R topics documented:

close.RethinkDB_connection	2
close.RethinkDB_cursor	2
cursorNext	3
cursorToList	3
drainConnection	4
isCursorEmpty	5
isOpened	5
openConnection	6
print.RethinkDB_connection	6
print.RethinkDB_cursor	7
r	7

Index	10
--------------	-----------

close.RethinkDB_connection

Close RethinkDB connection

Description

Closes connection and stops all associated callbacks and/or sync cursor.

Usage

```
## S3 method for class 'RethinkDB_connection'  
close(con, ...)
```

Arguments

con	Connection to close.
...	Ignored.

close.RethinkDB_cursor

Close cursor

Description

Closes a given cursor and stops its associated query. Should be called on the current cursor before a new sync query is invoked on the same connection.

Usage

```
## S3 method for class 'RethinkDB_cursor'  
close(con, ...)
```

Arguments

con	Cursor to close.
...	Ignored.

cursorNext	<i>Pull next object from a cursor</i>
------------	---------------------------------------

Description

Pulls a datum from a given cursor, sending continuation queries when needed.

Usage

```
cursorNext(cursor, inBatch = FALSE)
```

Arguments

cursor	Cursor to pull from; a result of <code>r()\$. . \$.run(. . .)</code> .
inBatch	If set to TRUE, enables batch mode, i.e., returning the whole local cache (this is usually NOT the whole data available under cursor) rather than a single result. Values other than TRUE or FALSE are invalid.

Value

In a default mode, a list representing the returned response JSON, or NULL if no data is available. In a batch mode, list of such lists representing the whole cache (which may be empty, corresponding to default mode's NULL).

Note

When this function empties local cache, it may ask RethinkDB for more data and hence block. Use [isCursorEmpty](#) to decide if it makes sense to call `cursorNext`. In case you don't need any more answers for the query, close cursor with `close` method.

Author(s)

Miron B. Kursa

cursorToList	<i>Convert cursor into a list</i>
--------------	-----------------------------------

Description

Converts cursor into a list. For convenience, when given anything other than cursor returns this object unchanged; this way can be used to wrap the result of `$.run`, so that it is never a cursor.

Usage

```
cursorToList(x, maxResults = 10000)
```

Arguments

<code>x</code>	RethinkDB cursor or any object.
<code>maxResults</code>	Number of results sufficient to stop pulling from cursor.

Value

A list of elements pulled from `x` if it is a cursor, `x` otherwise.

Note

The length of a list may be larger than `maxResults` because RethinkDB transmits results in batches.

Author(s)

Miron B. Kursa

<code>drainConnection</code>	<i>Drain RethinkDB connection</i>
------------------------------	-----------------------------------

Description

Drains a given RethinkDB connection, i.e. pull query responses and both call their associated callbacks (for async queries) and/or filling sync cursor local cache. Draining ends when all async queries end; the function blocks for the entire time this is happening.

Usage

```
drainConnection(x)
```

Arguments

<code>x</code>	Connection to drain.
----------------	----------------------

Details

The async query callback will only fire during `drainConnection` or (opportunistically) `cursorNext`; consequently this function must be run to guarantee that installed callbacks will have a chance to fire.

isCursorEmpty	<i>Check if cursor is empty</i>
---------------	---------------------------------

Description

Check whether a given cursor is fully drained and will output no more datum. The function never blocks; also verifies that the underlying connection is useful.

Usage

```
isCursorEmpty(cursor)
```

Arguments

cursor	Cursor to check; a result of <code>r().\$.run(...)</code> .
--------	---

Value

TRUE if cursor has no more data to return.

Note

It is possible that `cursorNext` will return NULL just after `isCursorEmpty` returns FALSE. Change-feeds cursors (made with `r().$.changes().run(...)`) will never become empty (provided that connection won't become broken).

Author(s)

Miron B. Kursa

isOpened	<i>Check if connection is opened</i>
----------	--------------------------------------

Description

Check whether a given connection is opened. Closed connections cannot be used, and will throw errors on such an attempt; their associated callbacks and/or sync cursor are dead and won't fire/produce any more data.

Usage

```
isOpened(x)
```

Arguments

x	Connection to to check.
---	-------------------------

Value

TRUE if connection is opened and can be used for queries, FALSE otherwise.

Author(s)

Miron B. Kursa

openConnection	<i>Open connection to a RethinkDB server</i>
----------------	--

Description

Opens connection to a given RethinkDB server.

Usage

```
openConnection(host = "localhost", port = 28015, authKey = NULL,
               v = "V0_4")
```

Arguments

host	Host to connect to.
port	Port to connect to.
authKey	Authentication key. Not supported yet.
v	Protocol version; "V0_3" and "V0_4" supported, the default should be used.

Value

Object of a class `RethinkDB_connection`, which can be passed to `r()$.run` and `r()$.runAsync` functions.

print.RethinkDB_connection	<i>Print RethinkDB connection</i>
----------------------------	-----------------------------------

Description

Prints a RethinkDB connection details, including a number of pending callbacks.

Usage

```
## S3 method for class 'RethinkDB_connection'
print(x, ...)
```

Arguments

x	Connection to print.
...	Ignored.

Note

Never blocks.

`print.RethinkDB_cursor`
Print cursor

Description

Prints a given cursor's status.

Usage

```
## S3 method for class 'RethinkDB_cursor'  
print(x, ...)
```

Arguments

x	Cursor to print.
...	Ignored.

Note

Never blocks; also checks whether the underlying connection is alive.

`r` *ReQL root*

Description

Creates ReQL root for building a query.

Usage

```
r(db, table)
```

Arguments

db	DB name; this is optional, and is just a syntax sugar for <code>r()\$db(db)</code> .
table	Table name; this is optional, requires db to be given, and is just a syntax sugar for <code>r()\$db(db)\$table(table)</code>

Value

ReQL root; use `$` (or `[[[]]]`) to chain query terms (like `r()$db("test")$table("test")`). In general, anonymous attributes are passed as attributes while named as term options. In context of term arguments, named lists are treated as JSON objects (following `rjson` package heuristics), unnamed lists and simple vectors as JSON arrays; classes and attributes are ignored. Term options should be called in the snake case form (for instance `return_changes` not `returnChanges`), as documented for the original Python driver. To finalise, use `$run` or `$runAsync`. For a comprehensive description of all terms, see RethinkDB API reference; here we give an overview of some:

`run(connection, ...)`

Evaluate the query; the function will block until first response from RethinkDB to this query will be received. May return cursor, an object representing a stream of data on which `cursorNext` and `cursorToList` can be used to extract actual information. `...` may be used to specify run options, like `profile`, `durability` or `read_mode`.

`runAsync(connection, callback, ...)`

Evaluate the query; for each datum received `x`, run `callback(x)`. Callback should return `TRUE` to be re-evaluated on proceeding data; any other response will cause the query to be dropped immediately. This function returns immediately; to ask R to start evaluating async queries, run `drainConnection`. Note that callbacks can be also called while `$run()` blocks waiting for other query to execute.

`bracket(...)` Implementation of the JavaScript `(...)` and Python `[...]` operation.

`funcall(function,atts)`

Implementation of the JavaScript `.do()`; note that the order of arguments is different.

Note

ReQL is implemented as an environment, thus is mutable unlike most R objects. To this end, you can use variables for chaining like this `r()->query; query$db("a"); query$table("b");` but consequently you can't use variables to make a re-usable stub, i.e., this is invalid: `r()->query; query$db("a")$table("aa")$run(...) query$db("b")$table("bb")$run(...);`

If you get "trying to apply non-function" error, you likely have misspelled term name or trying to use a non-existent one.

To view raw AST (at any depth), use `$query`.

Author(s)

Miron B. Kursa

Examples

```
## Not run:
#Connect to the RethinkDB instance
cn<-openConnection()

#Get document count in some_db's some_table
```



```
r()$db("some_db")$table("some_table")$count()$run(cn)
#...same can be done shorter
r("some_db","some_table")$count()$run(cn)

#Fetch 5 random docs from some_db's some_table...
r("some_db","some_table")$sample(5)$run(cn)->cursor
#...and present as a list`
cursorToList(cursor)

#Insert an element
r("some_db","some_table")$insert(list(id="new",a=1:10,b=list(c=1,d=2)))$run(cn)

#Close connection
close(cn)

## End(Not run)
```

Index

close.RethinkDB_connection, 2
close.RethinkDB_cursor, 2
cursorNext, 3, 4, 5, 8
cursorToList, 3, 8

drainConnection, 4, 8

isCursorEmpty, 3, 5
isOpened, 5

openConnection, 6

print.RethinkDB_connection, 6
print.RethinkDB_cursor, 7

r, 7