Package 'neo2R'

July 22, 2025

Type Package

Title Neo4j to R

Version 2.4.2

Maintainer Patrice Godard <patrice.godard@gmail.com>

Description The aim of neo2R is to provide simple and low level connectors for querying neo4j graph databases (<https://neo4j.com/>).
The objects returned by the query functions are either lists or data.frames with very few post-processing.
It allows fast processing of queries returning many records.
And it let the user handle post-processing according to the data model and his needs.

URL https://github.com/patzaw/neo2r

BugReports https://github.com/patzaw/neo2r/issues

Depends R (>= 3.6)

Imports base64enc, jsonlite, httr, utils

SystemRequirements neo4j (>=3 AND <=5) <https://neo4j.com/>

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Patrice Godard [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-6257-9730>),
Eusebiu Marcu [ctb]

Repository CRAN

Date/Publication 2024-01-18 10:50:02 UTC

Contents

cypher 2
graphRequest 3

import_from_df 4
multicypher 4
prepCql 6
readCql 6
startGraph 7

Index **9**

`cypher` *Run a cypher query*

Description

Run a cypher query

Usage

```
cypher(
  graph,
  query,
  parameters = NULL,
  result = c("row", "graph"),
  arraysAsStrings = TRUE,
  eltSep = " || "
)
```

Arguments

`graph`	the neo4j connection
`query`	the cypher query
`parameters`	parameters for the cypher query.
`result`	the way to return results. "row" will return a data frame and "graph" will return a list of nodes, a list of relationships and a list of paths (vectors of relationships identifiers).
`arraysAsStrings`	if result="row" and arraysAsStrings is TRUE (default) array from neo4j are converted to strings and array elements are separated by eltSep.
`eltSep`	if result="row" and arraysAsStrings is TRUE (default) array from neo4j are converted to strings and array elementes are separated by eltSep.

Value

The "result" of the query (invisible). See the "result" param.

See Also

`multicypher()`, `startGraph()`, `prepCql()`, `readCql()` and `graphRequest()`

Examples

```
## Not run:
# 2 identical queries
result <- cypher(
   graph=graph,
   query='match (n {value:$value}) return n',
   parameters=list(value="100"),
   result="graph"
)
result <- cypher(
   graph=graph,
   query='match (n {value:"100"}) return n',
   result="graph"
)

## End(Not run)
```

`graphRequest` *Run a curl request on a neo4j graph*

Description

Run a curl request on a neo4j graph

Usage

```
graphRequest(graph, endpoint, customrequest = c("POST", "GET"), postText)
```

Arguments

`graph`	the neo4j connection
`endpoint`	the endpoint for the request. To list all the available endpoints: `graphRequest(graph, endpoint="", customrequest="GET", postText="")$result`
`customrequest`	the type of request: "POST" (default) or "GET"
`postText`	the request body

Value

A list with the "header" and the "result" of the request (invisible)

See Also

`startGraph()` and `cypher()`

import_from_df *Imports a data.frame in the neo4j graph database*

Description

This function only works with localhost Neo4j instances.

Usage

```
import_from_df(graph, cql, toImport, periodicCommit = 1000, by = Inf, ...)
```

Arguments

graph	the neo4j connection
cql	the CQL query to be applied on each row of toImport. Use the 'row' prefix to refer to the data.frame column.
toImport	the data.frame to be imported as "row". Use "row.FIELD" in the cql query to refer to one FIELD of the toImport data.frame
periodicCommit	use periodic commit when loading the data (default: 10000).
by	number of rows to send by batch (default: Inf). Can be an alternative to periodic commit.
...	further parameters for cypher()

See Also

cypher()

multicypher *Run a multiple cypher queriers*

Description

Run a multiple cypher queriers

Usage

```
multicypher(
  graph,
  queries,
  parameters = NULL,
  result = c("row", "graph"),
  arraysAsStrings = TRUE,
  eltSep = " || "
)
```

Arguments

graph	the neo4j connection
queries	queries to submit. It can be either a character vector for which each element corresponds to a cypher query. Or it can be a list of lists with the following slots: • **query** (mandatory): A single character corresponding to the cypher query. • **parameters** (optional): A set of parameters specific for this query. If not provided, the *parameters* parameter of the function is used (see below). • **result** (optional): The specific way to return the results of this query. If not provided, the *result* parameter of the function is used (see below).
parameters	default parameters for the cypher queries.
result	default way to return results. "row" will return a data frame and "graph" will return a list of nodes, a list of relationships and a list of paths (vectors of relationships identifiers).
arraysAsStrings	if result="row" and arraysAsStrings is TRUE (default) array from neo4j are converted to strings and array elements are separated by eltSep.
eltSep	if result="row" and arraysAsStrings is TRUE (default) array from neo4j are converted to strings and array elementes are separated by eltSep.

Value

A list of "result" of the queries (invisible). See the "result" param.

See Also

cypher(), startGraph(), prepCql(), readCql() and graphRequest()

Examples

```
## Not run:
result <- multicypher(
   graph,
   queries=list(
      q1="match (n) return n.value limit 5",
      q2=list(
         query="match (f {value:$val})-[r]->(t) return f, r, t limit 5",
         result="graph",
         parameters=list(val=100)
      )
   )
)

## End(Not run)
```

prepCql *Prepares a CQL query from a character vector*

Description

Prepares a CQL query from a character vector

Usage

```
prepCql(...)
```

Arguments

...	character vectors with cQL commands

Value

A well formated CQL query

See Also

`cypher()` and `readCql()`

Examples

```
prepCql(c(
 "MATCH (n)",
 "RETURN n"
))
```

readCql *Parse a CQL file and returned the prepared queries*

Description

Parse a CQL file and returned the prepared queries

Usage

```
readCql(file)
```

Arguments

file	the name of the file to be parsed

Value

A character vector of well formated CQL queries

See Also

`cypher()` and `prepCql()`

`startGraph` *Prepare connection to neo4j database*

Description

Prepare connection to neo4j database

Usage

```
startGraph(
  url,
  database = NA,
  username = NA,
  password = NA,
  importPath = NA,
  .opts = list(),
  check = TRUE
)
```

Arguments

`url`	the DB url
`database`	the name of the database. If NA (default) it will use "data" with versions 3.. of Neo4j and "neo4j" with versions 4..
`username`	the neo4j user name (default: NA; works only if authentication has been disabled in neo4j by setting NEO4J.AUTH=none)
`password`	the neo4j user password (default: NA; works only if authentication has been disabled in neo4j by setting NEO4J.AUTH=none)
`importPath`	path to the import directory (default: NA => no import directory). Import only works with local neo4j instance.
`.opts`	a named list identifying the curl options for the handle (see `httr::config()` and `httr::httr_options()` for a complete list of available options; for example: `.opts = list(ssl_verifypeer = 0)`). Moreover, this parameter can be used to pass additional headers to the graph requests as "extendedHeaders": it is useful, for example, for OAuth access delegation (see details).
`check`	check the connection before returning it (default: TRUE). Set to false when connection to the "system" database

Details

The "ssl.verifypeer" logical option available in the RCurl package used in former versions of neo2R (<= 2.2.0) is not recognized by `httr::config()`. However, for backward compatibility, if it used, it is translated into "ssl_verifypeer" integer option recognized by the httr package with a warning message.

Headers in `.opts$extendedHeaders` are added to, or overwrite, the default Neo4j headers. If there is a `.opts$extendedHeaders[["Authorization"]]` value, the default Neo4j "Authorization" header (user credentials) is provided automaticaly as "X-Authorization". This mechanism is used for OAuth access delegation.

Value

A connection to the graph DB: a list with the url and necessary headers

Index

cypher, 2
cypher(), *3–7*

graphRequest, 3
graphRequest(), *2*, *5*

httr::config(), *7*, *8*
httr::httr_options(), *7*

import_from_df, 4

multicypher, 4
multicypher(), *2*

prepCql, 6
prepCql(), *2*, *5*, *7*

readCql, 6
readCql(), *2*, *5*, *6*

startGraph, 7
startGraph(), *2*, *3*, *5*