Package: bolt4jr (via r-universe)

February 25, 2025

Title Interface for the 'Neo4j Bolt' Protocol

Version 1.4.0

Description Querying, extracting, and processing large-scale network data from Neo4j databases using the 'Neo4j Bolt' <https://neo4j.com/docs/bolt/current/bolt/> protocol. This interface supports efficient data retrieval, batch processing for large datasets, and seamless conversion of query results into R data frames, making it ideal for bioinformatics, computational biology, and other graph-based applications.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports data.table, glue, purrr, reticulate

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Wanjun Gu [aut, cre] (<https://orcid.org/0000-0002-7342-7000>)

Maintainer Wanjun Gu <wanjun.gu@ucsf.edu>

Repository CRAN

Date/Publication 2025-02-25 11:50:02 UTC

Config/pak/sysreqs libpng-dev python3

Contents

convert_df *Convert a Query Result into a Data Frame*

Description

This function takes a query result object and transforms it into a data frame with specified field names. For each entry in the query result, it attempts to extract values corresponding to the given field names. If a particular field does not exist in the entry, it is replaced with NA.

Usage

```
convert_df(
  query_result,
  field_names = c("node_id", "n.identifier", "n.name", "n.source")
)
```

Arguments

query_result	A list (or similar structure) representing the query result, typically containing entries from which fields can be extracted.
field_names	A character vector of field names to be extracted from each entry in query_result. Defaults to c("node_id", "n_identifier", "n.name", "n.source").

Value

A data frame with one row per entry in query_result, and columns corresponding to the specified field_names. Missing fields are filled with NA.

Examples

```
# Suppose query_result is a list of named lists:
query_result = list(
  list(node_id = 1, n = list(identifier = 1, name = "some node", source = "internet")),
  list(node_id = 2, n = list(identifier = 2, name = "some other node", source = "library"))
)

query_result_df = convert_df(
  query_result,
  field_names = c("node_id", "n.identifier", "n.name", "n.source")
)
```

run_batch_query *Batch Query and Save Data from Neo4j*

Description

This function performs batch queries to a Neo4j database and appends the results to a TSV file.

Usage

```
run_batch_query(
  uri,
  user,
  password,
  query,
  field_names,
  filename = NULL,
  batch_size = 1000
)
```

Arguments

uri	A string specifying the URI for the Neo4j database connection.
user	A string specifying the username for the Neo4j database.
password	A string specifying the password for the Neo4j database.
query	A string containing the Cypher query to execute. The query should not include SKIP or LIMIT, as these are appended for batching.
field_names	A character vector specifying the column names to use for the resulting data.
filename	A string specifying the name of the TSV file to save the results. If NULL, a temporary file will be used.
batch_size	An integer specifying the number of records to fetch per batch. Default is 1000.

Value

No return value, called for side effects.

Examples

```
## Not run:
run_batch_query(
  uri = "bolt://localhost:7687",
  user = "<Username for Neo4j>",
  password = "<Password for Neo4j>",
  query = "MATCH (n) RETURN n LIMIT 10",
  field_names = c("id", "name"),
  filename = NULL,  # Writes to a temp file by default
  batch_size = 1000
```

```
)

## End(Not run)
```

`run_query` *Connect to Neo4j and Run a Simple Query*

Description

This function demonstrates connecting to a Neo4j database via the Python neo4j driver and using pandas to manipulate the returned data.

Usage

```
run_query(uri, user, password, query)
```

Arguments

`uri`	Neo4j URI, e.g., "bolt://localhost:7687"
`user`	Username for Neo4j
`password`	Password for Neo4j
`query`	A Cypher query to execute, e.g. "MATCH (n) RETURN n LIMIT 5"

Value

A data.frame containing the query results.

`setup_bolt4jr` *Set up the Conda environment for bolt4jr*

Description

This function initializes the Conda environment required for the `bolt4jr` package. If no Conda binary is found, it installs Miniconda. If the required Conda environment (`bolt4jr`) is not found, it creates the environment and installs the necessary dependencies.

Usage

```
setup_bolt4jr()
```

Details

The function ensures that:

- A Conda binary is available.
- A Conda environment named `bolt4jr` exists.
- The `neo4j` Python package is installed in the `bolt4jr` environment.

Call this function manually before using any functionality that relies on Python.

Value

No return value, called for side effects.

Index