Package 'kuzuR'

May 8, 2026

Title Interface to 'kuzu' Graph Database

Version 0.2.3

Maintainer Manuel Wick-Eckl <manuel.wick@gmail.com>

Description Provides a high-performance 'R' interface to the 'kuzu' graph database. It uses the 'reticulate' package to wrap the official 'Python' client ('kuzu', 'pandas', and 'networkx'), allowing users to interact with 'kuzu' seamlessly from within 'R'. Key features include managing database connections, executing 'Cypher' queries, and efficiently loading data from 'R' data frames. It also provides seamless integration with the 'R' ecosystem by converting query results directly into popular 'R' data structures, including 'tibble', 'igraph', 'tidygraph', and 'g6R' objects, making 'kuzu's powerful graph computation capabilities readily available for data analysis and visualization workflows in 'R'.
The 'kuzu' documentation can be found at <https://kuzudb.github.io/docs/>.

URL https://github.com/WickM/kuzuR, https://wickm.github.io/kuzuR/

BugReports https://github.com/WickM/kuzuR/issues

License MIT + file LICENSE

Imports reticulate, igraph, tibble, tidygraph

Suggests g6R, jsonlite, testthat (>= 3.0.0), knitr, rmarkdown, spelling, arrow

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Config/usethis/last-upkeep 2025-10-18

Language en-US

NeedsCompilation no

Author Manuel Wick-Eckl [aut, cre]

Repository CRAN

Date/Publication 2025-11-26 20:30:07 UTC

Contents

`as.data.frame.kuzu.query_result.QueryResult`
Convert a Kuzu Query Result to a Data Frame

Description

Provides an S3 method to seamlessly convert a Kuzu query result object into a standard R `data.frame`.

Usage

```
## S3 method for class 'kuzu.query_result.QueryResult'
as.data.frame(x, ...)
```

Arguments

`x`	A Kuzu query result object.
`...`	Additional arguments passed to `as.data.frame`.

Value

An R `data.frame` containing the query results.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")

# Convert the result to a data.frame
df <- as.data.frame(result)
print(df)
```

---

as_igraph *Convert a Kuzu Query Result to an igraph Object*

Description

Converts a Kuzu query result into an `igraph` graph object.

Usage

```
as_igraph(query_result)
```

Arguments

`query_result`	A `kuzu_query_result` object from `kuzu_execute()` that contains a graph.

Details

This function takes a `kuzu_query_result` object, converts it to a `networkx` graph in Python, extracts the nodes and edges into R data frames, and then constructs an `igraph` object. It is the final step in the `kuzu_execute -> as_igraph` workflow.

Value

An `igraph` object.

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
  conn <- kuzu_connection(":memory:")
  kuzu_execute(conn, "CREATE NODE TABLE Person(name STRING,
  PRIMARY KEY (name))")
  kuzu_execute(conn, "CREATE REL TABLE Knows(FROM Person TO Person)")
  kuzu_execute(conn, "CREATE (p:Person {name: 'Alice'}),
  (q:Person {name: 'Bob'})")
  kuzu_execute(conn, "MATCH (a:Person), (b:Person) WHERE
                                                  a.name='Alice' AND
                                                  b.name='Bob'
```

```
                                                    CREATE (a)-[:Knows]->(b)"
)

  res <- kuzu_execute(conn, "MATCH (p:Person)-[k:Knows]->(q:Person)
  RETURN p, k, q")
  g <- as_igraph(res)
  print(g)
  rm(conn, res, g)
}
```

`as_tibble.kuzu.query_result.QueryResult` *Convert a Kuzu Query Result to a Tibble*

Description

Provides an S3 method to convert a Kuzu query result object into a `tibble`. This requires the `tibble` package to be installed.

Usage

```
## S3 method for class 'kuzu.query_result.QueryResult'
as_tibble(x, ...)
```

Arguments

`x`	A Kuzu query result object.
`...`	Additional arguments passed to `as_tibble`.

Value

A `tibble` containing the query results.

Examples

```
if (requireNamespace("tibble", quietly = TRUE)) {
  conn <- kuzu_connection(":memory:")
  kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
  PRIMARY KEY (name))")
  kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
  result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")

  # Convert the result to a tibble
  tbl <- tibble::as_tibble(result)
  print(tbl)
}
```

as_tidygraph *Convert a Kuzu Query Result to a tidygraph Object*

Description

Converts a Kuzu query result into a `tidygraph` `tbl_graph` object.

Usage

```
as_tidygraph(query_result)
```

Arguments

`query_result`	A `kuzu_query_result` object from `kuzu_execute()` that contains a graph.

Value

A `tbl_graph` object.

Examples

```
if (requireNamespace("tidygraph", quietly = TRUE)) {
  conn <- kuzu_connection(":memory:")
  kuzu_execute(conn, "CREATE NODE TABLE Person(name STRING,
  PRIMARY KEY (name))")
  kuzu_execute(conn, "CREATE (p:Person {name: 'Alice'})")
  res <- kuzu_execute(conn, "MATCH (p:Person) RETURN p")
  g_tidy <- as_tidygraph(res)
  print(g_tidy)
  rm(conn, res, g_tidy)
}
```

check_kuzu_installation *Check for Kuzu Python Dependencies*

Description

This function checks if the required Python packages (`kuzu`, `pandas`, `networkx`) are available in the user's `reticulate` environment. If any packages are missing, it provides a clear, actionable message guiding the user on how to install them manually.

Usage

```
check_kuzu_installation()
```

Value

NULL invisibly. The function is called for its side effect of checking dependencies and printing messages.

Examples

```
check_kuzu_installation()
```

---

kuzu_connection *Create a Connection to a Kuzu Database*

Description

Establishes a connection to a Kuzu database. If the database does not exist at the specified path, it will be created. This function combines the database initialization and connection steps into a single call.

Usage

```
kuzu_connection(path)
```

Arguments

path	A string specifying the file path for the database. For an in-memory database, use ":memory:".

Value

A Python object representing the connection to the Kuzu database.

Examples

```
# Create an in-memory database and connection
conn <- kuzu_connection(":memory:")

# Create or connect to an on-disk database
temp_db_dir <- file.path(tempdir(), "kuzu_disk_example_db")
db_path <- file.path(temp_db_dir, "kuzu_db")
dir.create(temp_db_dir, recursive = TRUE, showWarnings = FALSE)

# Establish connection
conn_disk <- kuzu_connection(db_path)

# Ensure the database is shut down and removed on exit
on.exit({
  # Access the 'db' object from the reticulate main module
  main <- reticulate::import_main()
  if (!is.null(main$db)) {
```

```
    main$db$shutdown()
  }
  unlink(temp_db_dir, recursive = TRUE)
})
```

---

kuzu_copy_from_csv *Load Data from a CSV File into a Kuzu Table*

Description

Loads data from a CSV file into a specified table in the Kuzu database.

Usage

```
kuzu_copy_from_csv(conn, file_path, table_name, optional_csv_parameter = NULL)
```

Arguments

conn	A Kuzu connection object.
file_path	A string specifying the path to the CSV file.
table_name	A string specifying the name of the destination table in Kuzu.
optional_csv_parameter	An optional parameter for CSV-specific configurations (e.g., delimiter, header). Refer to Kuzu documentation for available options.

Value

This function is called for its side effect of loading data and does not return a value.

See Also

Kuzu CSV Import

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE City(name STRING, population INT64,
PRIMARY KEY (name))")

# Create a temporary CSV file
csv_file <- tempfile(fileext = ".csv")
write.csv(data.frame(name = c("Berlin", "London"),
population = c(3645000, 8982000)),
          csv_file, row.names = FALSE)

# Load data from CSV
kuzu_copy_from_csv(conn, csv_file, "City")
```

```
# Verify the data
result <- kuzu_execute(conn, "MATCH (c:City) RETURN c.name, c.population")
print(as.data.frame(result))

# Clean up the temporary file
unlink(csv_file)
```

`kuzu_copy_from_df` *Load Data from a Data Frame or Tibble into a Kuzu Table*

Description

Efficiently copies data from an R `data.frame` or `tibble` into a specified table in the Kuzu database.

Usage

```
kuzu_copy_from_df(conn, df, table_name)
```

Arguments

`conn`	A Kuzu connection object.
`df`	A `data.frame` or `tibble` containing the data to load. Column names in the data frame should match the property names in the Kuzu table.
`table_name`	A string specifying the name of the destination table in Kuzu.

Details

When loading into a relationship table, Kuzu assumes the first two columns in the file are: FROM Node Column: The primary key of the FROM nodes. TO Node Column: The primary key of the TO nodes.

Value

This function is called for its side effect of loading data and does not return a value.

See Also

Kuzu Copy from DataFrame

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE REL TABLE Knows(FROM User TO User)")

# Load from a data.frame
users_df <- data.frame(name = c("Carol", "Dan"), age = c(35, 40))
kuzu_copy_from_df(conn, users_df, "User")

# Load from a tibble (requires pre-existing nodes)
kuzu_execute(conn, "CREATE (u:User {name: 'Alice'}), (v:User {name: 'Bob'})")
knows_df <- data.frame(from_person = c("Alice", "Bob"),
to_person = c("Bob", "Carol"))
kuzu_copy_from_df(conn, knows_df, "Knows")

result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
print(as.data.frame(result))

result_rel <- kuzu_execute(conn, "MATCH (a:User)-[k:Knows]->(b:User)
RETURN a.name, b.name")
print(as.data.frame(result_rel))
```

---

kuzu_copy_from_json *Load Data from a JSON File into a Kuzu Table*

Description

Loads data from a JSON file into a specified table in the Kuzu database. This function also ensures the JSON extension is loaded and available.

Usage

```
kuzu_copy_from_json(conn, file_path, table_name)
```

Arguments

conn	A Kuzu connection object.
file_path	A string specifying the path to the JSON file.
table_name	A string specifying the name of the destination table in Kuzu.

Value

This function is called for its side effect of loading data and does not return a value.

See Also

Kuzu JSON Import, Kuzu JSON Extension

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE Product(id INT64, name STRING,
PRIMARY KEY (id))")

# Create a temporary JSON file
json_file <- tempfile(fileext = ".json")
json_data <- '[{"id": 1, "name": "Laptop"}, {"id": 2, "name": "Mouse"}]'
writeLines(json_data, json_file)

# Load data from JSON
kuzu_copy_from_json(conn, json_file, "Product")

# Verify the data
result <- kuzu_execute(conn, "MATCH (p:Product) RETURN p.id, p.name")
print(as.data.frame(result))

# Clean up the temporary file
unlink(json_file)
```

`kuzu_copy_from_parquet` *Load Data from a Parquet File into a Kuzu Table*

Description

Loads data from a Parquet file into a specified table in the Kuzu database.

Usage

```
kuzu_copy_from_parquet(conn, file_path, table_name)
```

Arguments

`conn`	A Kuzu connection object.
`file_path`	A string specifying the path to the Parquet file.
`table_name`	A string specifying the name of the destination table in Kuzu.

Value

This function is called for its side effect of loading data and does not return a value.

See Also

Kuzu Parquet Import

Examples

```
if (requireNamespace("arrow", quietly = TRUE)) {
  conn <- kuzu_connection(":memory:")
  kuzu_execute(conn, "CREATE NODE TABLE Country(name STRING, code STRING,
  PRIMARY KEY (name))")

  # Create a temporary Parquet file
  parquet_file <- tempfile(fileext = ".parquet")
  country_df <- data.frame(name = c("USA", "Canada"), code = c("US", "CA"))
  arrow::write_parquet(country_df, parquet_file)

  # Load data from Parquet
  kuzu_copy_from_parquet(conn, parquet_file, "Country")

  # Verify the data
  result <- kuzu_execute(conn, "MATCH (c:Country) RETURN c.name, c.code")
  print(as.data.frame(result))

  # Clean up the temporary file
  unlink(parquet_file)
}
```

`kuzu_execute` *Execute a Cypher Query*

Description

Submits a Cypher query to the Kuzu database for execution. This function is used for all database operations, including schema definition (DDL), data manipulation (DML), and querying (MATCH).

Usage

```
kuzu_execute(conn, query)
```

Arguments

`conn`	A Kuzu connection object, as returned by `kuzu_connection()`.
`query`	A string containing the Cypher query to be executed.

Value

A Python object representing the query result.

Examples

```
conn <- kuzu_connection(":memory:")

# Create a node table
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")

# Insert data
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")

# Query data
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
```

---

kuzu_get_all *Retrieve All Rows from a Query Result*

Description

Fetches all rows from a Kuzu query result and returns them as a list of lists.

Usage

```
kuzu_get_all(result)
```

Arguments

result	A Kuzu query result object.

Value

A list where each element is a list representing a row of results.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
all_results <- kuzu_get_all(result)
```

kuzu_get_column_data_types *Get Column Data Types from a Query Result*

Description

Retrieves the data types of the columns in a Kuzu query result.

Usage

```
kuzu_get_column_data_types(result)
```

Arguments

result	A Kuzu query result object.

Value

A character vector of column data types.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
kuzu_get_column_data_types(result)
```

kuzu_get_column_names *Get Column Names from a Query Result*

Description

Retrieves the names of the columns in a Kuzu query result.

Usage

```
kuzu_get_column_names(result)
```

Arguments

result	A Kuzu query result object.

Value

A character vector of column names.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
kuzu_get_column_names(result)
```

---

kuzu_get_n *Retrieve the First N Rows from a Query Result*

Description

Fetches the first n rows from a Kuzu query result.

Usage

```
kuzu_get_n(result, n)
```

Arguments

result	A Kuzu query result object.
n	The number of rows to retrieve.

Value

A list of the first n rows.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
kuzu_execute(conn, "CREATE (:User {name: 'Bob', age: 30})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
first_row <- kuzu_get_n(result, 1)
```

kuzu_get_next *Retrieve the Next Row from a Query Result*

Description

Fetches the next available row from a Kuzu query result. This function can be called repeatedly to iterate through results one by one.

Usage

```
kuzu_get_next(result)
```

Arguments

result	A Kuzu query result object.

Value

A list representing the next row, or NULL if no more rows are available.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
kuzu_execute(conn, "CREATE (:User {name: 'Bob', age: 30})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
row1 <- kuzu_get_next(result)
row2 <- kuzu_get_next(result)
```

kuzu_get_schema *Get Schema from a Query Result*

Description

Retrieves the schema (column names and data types) of a Kuzu query result.

Usage

```
kuzu_get_schema(result)
```

Arguments

result	A Kuzu query result object.

Value

A named list where names are column names and values are data types.

Examples

```
conn <- kuzu_connection(":memory:")
kuzu_execute(conn, "CREATE NODE TABLE User(name STRING, age INT64,
PRIMARY KEY (name))")
kuzu_execute(conn, "CREATE (:User {name: 'Alice', age: 25})")
result <- kuzu_execute(conn, "MATCH (a:User) RETURN a.name, a.age")
kuzu_get_schema(result)
```

---

kuzu_merge_df *Merge Data from a Data Frame into Kuzu using a Merge Query*

Description

This function is intended for merging data from an R `data.frame` into Kuzu using a specified merge query. It leverages Python's reticulate to interact with Kuzu's Python API.

Usage

```
kuzu_merge_df(conn, df, merge_query)
```

Arguments

`conn`	A Kuzu connection object.
`df`	A `data.frame` or `tibble` containing the data to merge.
`merge_query`	A string representing the Kuzu query for merging data.

Value

This function is called for its side effect of merging data and does not return a value.

See Also

Kuzu Copy from DataFrame

Examples

```
## Not run:
my_data <- data.frame(
   name = c("Alice", "Bob"),
   item = c("Book", "Pen"),
   current_city = c("New York", "London")
 )
```

```
 merge_statement <- "MERGE (p:Person {name: df.name})
 MERGE (i:Item {name: df.item})
 MERGE (p)-[:PURCHASED]->(i)
 ON MATCH SET p.current_city = df.current_city
 ON CREATE SET p.current_city = df.current_city"

 # Note: 'conn' would need to be a valid Kuzu connection object
 # and the schema (Person, Item, PURCHASED tables) would need to be created
 # before running this example.
 # kuzu_merge_df(conn, my_data, merge_statement)

 # Example with a different merge query structure:
 my_data_2 <- data.frame(
   person_name = c("Charlie"),
   purchased_item = c("Laptop"),
   city = c("Paris")
 )
#
 merge_statement_2 <- "MERGE (p:Person {name: person_name})
 MERGE (i:Item {name: purchased_item})
 MERGE (p)-[:PURCHASED]->(i)
 ON MATCH SET p.current_city = city
 ON CREATE SET p.current_city = city"

 # kuzu_merge_df(conn, my_data_2, merge_statement_2)

## End(Not run)
```

Index