

Package ‘simplegraphdb’

July 23, 2025

Title A Simple Graph Database

Version 2021.03.10

Description

This is a graph database in 'SQLite'. It is inspired by Denis Papathanasiou's Python simple-graph project on 'GitHub'.

License MIT + file LICENSE

URL <https://github.com/mikeasilva/simplegraphdb>

BugReports <https://github.com/mikeasilva/simplegraphdb/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports DBI, RSQLite, rjson, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Michael Silva [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-7344-660X>>)

Maintainer Michael Silva <mike.a.silva@gmail.com>

Repository CRAN

Date/Publication 2021-03-12 10:30:02 UTC

Contents

add_node	2
atomic	3
connect_nodes	3
find_inbound_neighbors	4
find_neighbors	5
find_node	5
find_nodes	6

find_outbound_neighbors	6
get_connections	7
initialize	7
remove_node	8
set_id	9
traverse	9
upsert_node	11
visualize	12

Index	14
--------------	-----------

add_node	<i>Generates the SQL to add a node to the database</i>
----------	--

Description

Generates the SQL to add a node to the database

Usage

```
add_node(data, identifier = NA)
```

Arguments

data	Data to be added to the node in a list format
identifier	The identifier for the node

Value

A SQL statement to add a node to a database

Examples

```
## Not run:
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)

# Add nodes with data
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
```

```

    "name" = "Ronald Wayne",
    "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
    "name" = "Mike Markkula",
    "type" = c("person", "investor")), 5))

## End(Not run)

```

atomic	<i>An atomic transaction wrapper function</i>
--------	---

Description

An atomic transaction wrapper function

Usage

```
atomic(db_file, sql_statement)
```

Arguments

db_file	The name of the SQLite database
sql_statement	The SQL statement to execute

Value

Either the query results or NA for executed SQL statements

connect_nodes	<i>Add an edge to the database</i>
---------------	------------------------------------

Description

Add an edge to the database

Usage

```
connect_nodes(source_id, target_id, properties = list())
```

Arguments

source_id	Source node's id
target_id	Target node's id
properties	Edge properties (optional)

Value

A SQL statement to insert an edge into the database

Examples

```
## Not run:
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
  "name" = "Ronald Wayne",
  "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
  "name" = "Mike Markkula",
  "type" = c("person", "investor")), 5))

# Add in some edges to the graph
atomic(apple, connect_nodes(2, 1, list("action" = "founded")))
atomic(apple, connect_nodes(3, 1, list("action" = "founded")))
atomic(apple, connect_nodes(4, 1, list("action" = "founded")))
atomic(apple, connect_nodes(5, 1, list(
  "action" = "invested",
  "equity" = 80000,
  "debt" = 170000)))
atomic(apple, connect_nodes(1, 4, list(
  "action" = "divested",
  "amount" = 800,
  "date" = "April 12, 1976")))
atomic(apple, connect_nodes(2, 3))

## End(Not run)
```

find_inbound_neighbors

Generates the SQL to find the inbound neighbors for a node in the database

Description

Generates the SQL to find the inbound neighbors for a node in the database

Usage

```
find_inbound_neighbors(identifier)
```

Arguments

identifier The identifier for the node

Value

A SQL statement to find the inbound neighbors

find_neighbors	<i>Generates the SQL to find the neighbors for a node in the database</i>
----------------	---

Description

Generates the SQL to find the neighbors for a node in the database

Usage

```
find_neighbors(identifier)
```

Arguments

identifier The identifier for the node

Value

A SQL statement to find the neighbors

find_node	<i>Generates the SQL to find a node from the database</i>
-----------	---

Description

Generates the SQL to find a node from the database

Usage

```
find_node(identifier)
```

Arguments

identifier The identifier for the node

Value

A SQL statement to find a node

find_nodes	<i>Generate SQL to find nodes matching a criteria</i>
------------	---

Description

Generate SQL to find nodes matching a criteria

Usage

```
find_nodes(data, where_fn = "search_where", search_fn = "search_equals")
```

Arguments

data	A list of data that are the search criteria
where_fn	The function to use in the SQL WHERE clause. Valid values are: search_where (default) or search_like
search_fn	The function to use in the search. Valid values are: search_equals (default), search_starts_with, or search_contains

Value

A SQL statement to find nodes matching a criteria

find_outbound_neighbors	<i>Generates the SQL to find the outbound neighbors for a node in the database</i>
-------------------------	--

Description

Generates the SQL to find the outbound neighbors for a node in the database

Usage

```
find_outbound_neighbors(identifier)
```

Arguments

identifier	The identifier for the node
------------	-----------------------------

Value

A SQL statement to find outbound neighbors

get_connections	<i>Generates the SQL to find the connections for a node in the database</i>
-----------------	---

Description

Generates the SQL to find the connections for a node in the database

Usage

```
get_connections(source_id, target_id)
```

Arguments

source_id	The identifier for the source node
target_id	The identifier for the target node

Value

A SQL statement to find the edge connecting two nodes

initialize	<i>Initialize a new graph database</i>
------------	--

Description

Initialize a new graph database

Usage

```
initialize(db_file, schema_file = "./tests/schema.sql")
```

Arguments

db_file	The name of the SQLite database
schema_file	The SQL schema file (optional)

Value

No return value. It creates the database.

Examples

```
## Not run:
library(simplegraphdb)
initialize("network.sqlite")

## End(Not run)
```

remove_node	<i>Generates the SQL to remove a node from the database</i>
-------------	---

Description

Generates the SQL to remove a node from the database

Usage

```
remove_node(identifier)
```

Arguments

identifier The identifier for the node

Value

A SQL statement to delete a node

Examples

```
## Not run:
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
  "name" = "Ronald Wayne",
  "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
  "name" = "Mike Markkula",
  "type" = c("person", "investor")), 5))
atomic(apple, connect_nodes(2, 1, list("action" = "founded")))
atomic(apple, connect_nodes(3, 1, list("action" = "founded")))
atomic(apple, connect_nodes(4, 1, list("action" = "founded")))
atomic(apple, connect_nodes(5, 1, list(
  "action" = "invested",
  "equity" = 80000,
  "debt" = 170000)))
atomic(apple, connect_nodes(1, 4, list(
  "action" = "divested",
```



```

    "amount" = 800,
    "date" = "April 12, 1976"))
atomic(apple, connect_nodes(2, 3))
atomic(apple, upsert_node(2, list("nickname" = "Woz"), apple))

# Remove node 1 from the data
atomic(apple, remove_node(1))

## End(Not run)

```

set_id	<i>Sets the id attribute in JSON data</i>
--------	---

Description

Sets the id attribute in JSON data

Usage

```
set_id(identifier = NA, data)
```

Arguments

identifier	The id
data	The JSON data

Value

JSON ecoded data

traverse	<i>Finds the path as you traverse the graph</i>
----------	---

Description

Finds the path as you traverse the graph

Usage

```
traverse(db_file, src, tgt = NA, neighbors_fn = "find_neighbors")
```

Arguments

db_file	The name of the SQLite database
src	The id of the source node
tgt	The id of the target node (optional)
neighbors_fn	The neighbor function to employ. Valid options are find_neighbors, find_inbound_neighbors or find_outbound_neighbors (optional)

Value

A JSON object containing the id of the nodes in the path

Examples

```
## Not run:
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
  "name" = "Ronald Wayne",
  "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
  "name" = "Mike Markkula",
  "type" = c("person", "investor")), 5))
atomic(apple, connect_nodes(2, 1, list("action" = "founded")))
atomic(apple, connect_nodes(3, 1, list("action" = "founded")))
atomic(apple, connect_nodes(4, 1, list("action" = "founded")))
atomic(apple, connect_nodes(5, 1, list(
  "action" = "invested",
  "equity" = 80000,
  "debt" = 170000)))
atomic(apple, connect_nodes(1, 4, list(
  "action" = "divested",
  "amount" = 800,
  "date" = "April 12, 1976")))
atomic(apple, connect_nodes(2, 3))
atomic(apple, upsert_node(2, list("nickname" = "Woz"), apple))

# Traverse the data
traverse(apple, 4, 5)

# Get the inbound neighbors
traverse(apple, 5, "find_inbound_neighbors")

# Get the outbound neighbors
traverse(apple, 5, "find_outbound_neighbors")

## End(Not run)
```

upsert_node	<i>Generates the SQL to upsert a node in the database</i>
-------------	---

Description

Generates the SQL to upsert a node in the database

Usage

```
upsert_node(identifier, data, db_file)
```

Arguments

identifier	The identifier for the node
data	Data to be added to the node in a list format
db_file	The name of the 'SQLite' database

Value

A SQL statement to upsert a node

Examples

```
## Not run:
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
  "name" = "Ronald Wayne",
  "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
  "name" = "Mike Markkula",
  "type" = c("person", "investor")), 5))
atomic(apple, connect_nodes(2, 1, list("action" = "founded")))
atomic(apple, connect_nodes(3, 1, list("action" = "founded")))
atomic(apple, connect_nodes(4, 1, list("action" = "founded")))
atomic(apple, connect_nodes(5, 1, list(
  "action" = "invested",
  "equity" = 80000,
```

```

    "debt" = 170000)))
atomic(apple, connect_nodes(1, 4, list(
  "action" = "divested",
  "amount" = 800,
  "date" = "April 12, 1976")))
atomic(apple, connect_nodes(2, 3))

#Upsert some data
atomic(apple, upsert_node(2, list("nickname" = "Woz"), apple))

## End(Not run)

```

visualize

Generates dot files for visualization of the graph

Description

Generates dot files for visualization of the graph

Usage

```

visualize(
  db_file,
  dot_file = "file.dot",
  path = c(),
  exclude_node_keys = c(),
  hide_node_key = FALSE,
  node_kv = " ",
  exclude_edge_keys = c(),
  hide_edge_key = FALSE,
  edge_kv = " "
)

```

Arguments

db_file	The name of the SQLite database
dot_file	The name of the file
path	The path to include in the visualization
exclude_node_keys	The node keys to exclude from the visualization
hide_node_key	Boolean if the node key is hidden
node_kv	The node key values
exclude_edge_keys	The key of edges to exclude
hide_edge_key	Boolean if the edge key is hidden
edge_kv	The edge key values

Value

No return value. It creates a file.

Examples

```
## Not run:
library(simplegraphdb)
library(simplegraphdb)
apple <- "apple_test.sqlite"
initialize(apple)
atomic(apple, add_node(list(
  "name" = "Apple Computer Company",
  "type" = c("company", "start-up"),
  "founded" = "April 1, 1976"), 1))
atomic(apple, add_node(list(
  "name" = "Steve Wozniak",
  "type" = c("person", "engineer", "founder")), 2))
atomic(apple, add_node(list(
  "name" = "Steve Jobs",
  "type" = c("person", "designer", "founder")), 3))
atomic(apple, add_node(list(
  "name" = "Ronald Wayne",
  "type" = c("person", "administrator", "founder")), 4))
atomic(apple, add_node(list(
  "name" = "Mike Markkula",
  "type" = c("person", "investor")), 5))
atomic(apple, connect_nodes(2, 1, list("action" = "founded")))
atomic(apple, connect_nodes(3, 1, list("action" = "founded")))
atomic(apple, connect_nodes(4, 1, list("action" = "founded")))
atomic(apple, connect_nodes(5, 1, list(
  "action" = "invested",
  "equity" = 80000,
  "debt" = 170000)))
atomic(apple, connect_nodes(1, 4, list(
  "action" = "divested",
  "amount" = 800,
  "date" = "April 12, 1976")))
atomic(apple, connect_nodes(2, 3))
atomic(apple, upsert_node(2, list("nickname" = "Woz"), apple))

# Visualize the data
visualize(apple, dot_file = "apple.dot", path = c(4, 1, 5))

## End(Not run)
```

Index

`add_node`, [2](#)

`atomic`, [3](#)

`connect_nodes`, [3](#)

`find_inbound_neighbors`, [4](#)

`find_neighbors`, [5](#)

`find_node`, [5](#)

`find_nodes`, [6](#)

`find_outbound_neighbors`, [6](#)

`get_connections`, [7](#)

`initialize`, [7](#)

`remove_node`, [8](#)

`set_id`, [9](#)

`traverse`, [9](#)

`upsert_node`, [11](#)

`visualize`, [12](#)