link.graph Documentation

Release 0.0.1

David Delassus

September 02, 2016

Contents

1 Installation **3**

2 Features **5**

3 Contents **7**
- 3.1 Command Line Interface . . . 7
- 3.2 Graph Querying Language . . . 7
- 3.3 API documentation . . . 10

4 Donating **15**

Python Module Index **17**

link.graph is a pure-python graph database.

Check out the source code on Github.

CHAPTER 1

Installation

```
pip install link.graph
```

CHAPTER 2

Features

- API is fully implemented with Map/Reduce algorithms (see link.parallel)
- storage backend is based on KVStore and Fulltext
- command line interface

CHAPTER 3

Contents

3.1 Command Line Interface

The command line interface can be started with the command `graphcli`:

Use `Ctrl+D` to exit:

Commands history is saved in a **SQLite3** database (if available), located at: `~/.cache/graphcli/history.db`.

The configuration is stored in `$B3J0F_CONF_DIR/link/graph/cli.conf`:

```
[GRAPHCLI]

color_scheme = default  # syntax highliting in command line (see Pygments documentation)
history_size = 200  # number of commands to be loaded when graphcli is started
```

3.2 Graph Querying Language

A query to the graph is composed of two distinct parts:

- the `walk through` part, used to walk through the graph and select sets of elements
- the `operations` part, used to execute operations on the selected sets

3.2.1 Walk through

A walk-through is composed of:

- `FROM` statements, used to define sets of nodes
- `THROUGH` statements, used to walk through aliased relations
- `TO` statements, used to alias the destination nodes

The first `FROM` statement is used to selects elements from the graph, creating a sub-graph:

```
FROM <set> <node filter> [ AS <alias> ]

THROUGH <set> <relation filter> [ AS <alias> ] [ <walk mode> ]

TO <node filter> <alias>
```

A second `FROM` statements will select another sub-graph, from the alias created by the previous statement.

```
FROM NODES () {} AS nodes0
FROM nodes0 (n1 n2) { foo = "bar" } AS nodes1
FROM nodes1 (n1) { bar = "baz" AND baz = "biz" } AS nodes2

THROUGH
    RELS () {} AS rels0
    DEPTH BACKWARD 5 10
THROUGH
    rels0 (r1 r2) { weight > 2 } AS rels1
    BREADTH FORWARD 2 *
THROUGH
    rels1 (r1) {}

TO
    (n3) {}
    nodes3

TO
    (n4) {}
    nodes4
```

3.2.2 Operations

There is 4 types of operations:

- `SELECT` used to fetch aliased elements
- `INSERT` used to create new elements (may be aliased)
- `UPDATE` used to update aliased elements
- `DELETE` used to delete aliased elements

Read operations

A `SELECT` statement expects a list of alias to be returned

```
SELECT alias1, alias2, alias3
```

Create operations

An `INSERT` statement expects one of two kinds of element definition that can be aliased for further use:

- node definition
- relationship definition, which expects a set of source nodes and a set of target nodes

```
INSERT
    NODE(<new node types>) { <new node assignations> } AS alias0

INSERT
    REL(<new relationship types) { <new relationship assignations> }

    SOURCE
        <alias or node filter>
```

```
    TARGET
        <alias or node filter>
```

For example:

```
INSERT
    NODE(n2) {
        ADDTOSET foo "bar"
    } AS elt18
INSERT
    REL(r3) {
        SET weight 2
    }
    SOURCE
        (n4) { foo = "buzz" }
    TARGET
        elt18
```

Update operations

An `UPDATE` statement expects a set of new assignations on aliased properties:

```
UPDATE ( <assignations> )
```

For example:

```
UPDATE (
    SET alias2.weight 17
    ADDTOSET alias0.bar "baz"
    UNSET alias1.foo
    DELFROMSET alias0.bar "biz"
)
```

Delete operations

A `DELETE` statement have exactly the same syntax as a `SELECT` statement:

```
DELETE alias1, alias2, alias3
```

3.3 API documentation

3.3.1 link.graph.core package

Submodules

link.graph.algorithms.base module

link.graph.algorithms.follow module

link.graph.algorithms.backward module

link.graph.algorithms.forward module

link.graph.algorithms.filter module

link.graph.algorithms.update module

link.graph.algorithms.link module

Module contents

class link.graph.core.**GraphManager**(**args*, ***kwargs*)
Bases: object

Process request and manage access to graph storage.

mapreduce(*identifier*, *mapper*, *reducer*, *dataset*)

class link.graph.core.**GraphMiddleware**(**args*, ***kwargs*)
Bases: link.middleware.core.Middleware

link.graph.core.**getparser**(*cls*)

3.3.2 link.graph.dsl package

Submodules

link.graph.dsl.generator module

class link.graph.dsl.generator.**GraphDSLGenerator**(**args*, ***kwargs*)
Bases: object

exception Error
Bases: exceptions.Exception

GraphDSLGenerator.**MODEL_PREFIX = 'GraphDSL'**

GraphDSLGenerator.**grammar**

GraphDSLGenerator.**load_grammar**()

link.graph.dsl.generator.**single_parser_per_scope**(*_scope=None*, *_renew=False*)

link.graph.dsl.semantic module

link.graph.dsl.lexer module

class link.graph.dsl.lexer.**GraphDSLLexer**(***options*)
Bases: pygments.lexer.RegexLexer

aliases = ['graphdsl']

filenames = []

name = 'GraphDSL'

tokens = {'string_double': [('"', Token.Literal.String, '#pop'), ('\\\\([\\\\abfnrtv"\\\']|x[a-fA-F0-9]{2,4}|u[a-fA-F0-9]{4}|U

link.graph.dsl.walker.core module

class link.graph.dsl.walker.core.**GraphDSLNodeWalker**(*graphmgr*, **args*, ***kwargs*)
Bases: grako.model.DepthFirstWalker

walk_AliasNode(*node*, *children_retval*)

walk_AndFilterNode(*node*, *children_retval*)

walk_AssignAddNode(*node*, *children_retval*)

walk_AssignSetNode(*node*, *children_retval*)

walk_BooleanNode(*node*, *children_retval*)

walk_CRUDBlock(*node*, *children_retval*)

walk_CreateStatementNode(*node*, *children_retval*)

walk_DecimalNode(*node*, *children_retval*)

walk_DeleteStatementNode(*node*, *children_retval*)

walk_FromNode(*node*, *children_retval*)

walk_InnerFilterNode(*node*, *children_retval*)

walk_IntegerNode(*node*, *children_retval*)

walk_NaturalNode(*node*, *children_retval*)

walk_NodeTypeNode(*node*, *children_retval*)

walk_OrFilterNode(*node*, *children_retval*)

walk_ReadStatementNode(*node*, *children_retval*)

walk_RelationTypeNode(*node*, *children_retval*)

walk_RequestNode(*node*, *children_retval*)

walk_StringNode(*node*, *children_retval*)

walk_TermFilterNode(*node*, *children_retval*)

walk_ThroughNode(*node*, *children_retval*)

walk_ToNode(*node*, *children_retval*)

walk_TypeNode(*node*, *children_retval*)

walk_TypedFilterNode(*node*, *children_retval*)

walk_UpdateAddPropertyNode(*node*, *children_retval*)

walk_UpdateDelPropertyNode(*node*, *children_retval*)

walk_UpdateSetPropertyNode(*node*, *children_retval*)

walk_UpdateStatementNode(*node*, *children_retval*)

walk_UpdateUnsetPropertyNode(*node*, *children_retval*)

walk_ValueNode(*node*, *children_retval*)

walk_WalkModeNode(*node*, *children_retval*)

link.graph.dsl.walker.walkthrough module

class link.graph.dsl.walker.walkthrough.**Walkthrough**(*graphmgr*, **args*, ***kwargs*)

Bases: object

backward_breadth_nodes(*nodes*, *rel_ids*)

backward_depth_nodes(*nodes*, *rel_ids*, *begin*, *end*, *iteration=0*)

breadth_nodes(*nodes*, *rel_ids*, *begin*, *end*, *func*)

depth_nodes(*nodes*, *rel_ids*, *begin*, *end*, *func*)

filter_nodes(*nodes*, *to*)

forward_breadth_nodes(*nodes*, *rel_ids*)

forward_depth_nodes(*nodes*, *rel_ids*, *begin*, *end*, *iteration=0*)

select_nodes(*fromstmt*, *aliased_sets*)

select_relationships(*throughnode*, *aliased_sets*)

uniq_elts(*aliased_set*)

walk_nodes(*startnodes*, *rels*, *walkmode*)

link.graph.dsl.walker.walkthrough.**getmapfunc**(*key*, *match*)

link.graph.dsl.walker.walkthrough.**reducefunc**(*reducer*, *key*, *values*)

link.graph.dsl.walker.crud module

class link.graph.dsl.walker.crud.**CRUDOperations**(*graphmgr*, **args*, ***kwargs*)

Bases: object

compute_result(*data_id*, *ids*, *result*)

create_element(*store*, *statement*, *aliased_sets*)

do_CreateStatementNode(*statement*, *aliased_sets*)

do_DeleteStatementNode(*statement*, *aliased_sets*)

do_NodeTypeNode(*statement*, *aliased_sets*)

do_ReadStatementNode(*statement*, *aliased_sets*)

do_RelationTypeNode(*statement*, *aliased_sets*)

do_UpdateAddPropertyNode(*statement*, *aliased_sets*)

do_UpdateDelPropertyNode(*statement*, *aliased_sets*)

do_UpdateSetPropertyNode(*statement*, *aliased_sets*)

do_UpdateStatementNode(*statement*, *aliased_sets*)

do_UpdateUnsetPropertyNode(*statement*, *aliased_sets*)

get_links(*statement*, *aliased_sets*)

link.graph.dsl.walker.filter module

Module contents

3.3.3 link.graph.cli package

Submodules

link.graph.cli.core module

class link.graph.cli.core.**GraphCLI**(*graphuri*, **args*, ***kwargs*)
Bases: object

cancel_input(*event*)

continuation_tokens(*cli*, *width*)

get_title()

newline_or_execute(*event*)

register_shortcuts()

tab_or_complete(*event*)

link.graph.cli.history module

class link.graph.cli.history.**HistoryManager**(**args*, ***kwargs*)
Bases: object

add_to_history(*cmd*)

close()

history

read_history()

link.graph.cli.completion.core module

link.graph.cli.completion.semantic module

Module contents

CHAPTER 4

Donating

l

link.graph.cli, 13
link.graph.cli.core, 13
link.graph.cli.history, 13
link.graph.core, 10
link.graph.dsl, 13
link.graph.dsl.generator, 10
link.graph.dsl.lexer, 11
link.graph.dsl.walker.core, 11
link.graph.dsl.walker.crud, 12
link.graph.dsl.walker.walkthrough, 12

A

add_to_history() (link.graph.cli.history.HistoryManager method), 13
aliases (link.graph.dsl.lexer.GraphDSLLexer attribute), 11

B

backward_breadth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
backward_depth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
breadth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12

C

cancel_input() (link.graph.cli.core.GraphCLI method), 13
close() (link.graph.cli.history.HistoryManager method), 13
compute_result() (link.graph.dsl.walker.crud.CRUDOperations method), 12
continuation_tokens() (link.graph.cli.core.GraphCLI method), 13
create_element() (link.graph.dsl.walker.crud.CRUDOperations method), 12
CRUDOperations (class in link.graph.dsl.walker.crud), 12

D

depth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
do_CreateStatementNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_DeleteStatementNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_NodeTypeNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_ReadStatementNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_RelationTypeNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_UpdateAddPropertyNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_UpdateDelPropertyNode() (link.graph.dsl.walker.crud.CRUDOperations method), 12
do_UpdateSetPropertyNode() (link.graph.dsl.walker.crud.CRUDOperations method), 13
do_UpdateStatementNode() (link.graph.dsl.walker.crud.CRUDOperations method), 13
do_UpdateUnsetPropertyNode() (link.graph.dsl.walker.crud.CRUDOperations method), 13

F

filenames (link.graph.dsl.lexer.GraphDSLLexer attribute), 11
filter_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
forward_breadth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
forward_depth_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12

G

get_links() (link.graph.dsl.walker.crud.CRUDOperations method), 13
get_title() (link.graph.cli.core.GraphCLI method), 13
getmapfunc() (in module link.graph.dsl.walker.walkthrough), 12
getparser() (in module link.graph.core), 10
grammar (link.graph.dsl.generator.GraphDSLGenerator attribute), 10

GraphCLI (class in link.graph.cli.core), 13
GraphDSLGenerator (class in link.graph.dsl.generator), 10
GraphDSLGenerator.Error, 10
GraphDSLLexer (class in link.graph.dsl.lexer), 11
GraphDSLNodeWalker (class in link.graph.dsl.walker.core), 11
GraphManager (class in link.graph.core), 10
GraphMiddleware (class in link.graph.core), 10

H

history (link.graph.cli.history.HistoryManager attribute), 13
HistoryManager (class in link.graph.cli.history), 13

L

link.graph.cli (module), 13
link.graph.cli.core (module), 13
link.graph.cli.history (module), 13
link.graph.core (module), 10
link.graph.dsl (module), 13
link.graph.dsl.generator (module), 10
link.graph.dsl.lexer (module), 11
link.graph.dsl.walker.core (module), 11
link.graph.dsl.walker.crud (module), 12
link.graph.dsl.walker.walkthrough (module), 12
load_grammar() (link.graph.dsl.generator.GraphDSLGenerator method), 10

M

mapreduce() (link.graph.core.GraphManager method), 10
MODEL_PREFIX (link.graph.dsl.generator.GraphDSLGenerator attribute), 10

N

name (link.graph.dsl.lexer.GraphDSLLexer attribute), 11
newline_or_execute() (link.graph.cli.core.GraphCLI method), 13

R

read_history() (link.graph.cli.history.HistoryManager method), 13
reducefunc() (in module link.graph.dsl.walker.walkthrough), 12
register_shortcuts() (link.graph.cli.core.GraphCLI method), 13

S

select_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
select_relationships() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
single_parser_per_scope() (in module link.graph.dsl.generator), 10

T

tab_or_complete() (link.graph.cli.core.GraphCLI method), 13
tokens (link.graph.dsl.lexer.GraphDSLLexer attribute), 11

U

uniq_elts() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12

W

walk_AliasNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_AndFilterNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_AssignAddNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_AssignSetNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_BooleanNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_CreateStatementNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_CRUDBlock() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_DecimalNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_DeleteStatementNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_FromNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_InnerFilterNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_IntegerNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_NaturalNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_nodes() (link.graph.dsl.walker.walkthrough.Walkthrough method), 12
walk_NodeTypeNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_OrFilterNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_ReadStatementNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_RelationTypeNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11

walk_RequestNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_StringNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_TermFilterNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_ThroughNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_ToNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_TypedFilterNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_TypeNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_UpdateAddPropertyNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 11
walk_UpdateDelPropertyNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
walk_UpdateSetPropertyNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
walk_UpdateStatementNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
walk_UpdateUnsetPropertyNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
walk_ValueNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
walk_WalkModeNode() (link.graph.dsl.walker.core.GraphDSLNodeWalker method), 12
Walkthrough (class in link.graph.dsl.walker.walkthrough), 12