SparkseePython

6.0.2

Generated by Doxygen 1.8.11

Contents

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 Namespace Documentation

4.1 sparksee Namespace Reference

Embedded Graph Database.

Classes

- class Attribute

  Attribute data class.
- class AttributeKind

- class ExportType

  Export type.
- class Graph

  Graph class.
- class GraphExport

  Stores the graph exporting values.
- class Int32List

  Sparksee 32-bit signed integer list.
- class Int32ListIterator

  Int32List iterator class.
- class KeyValue
- class KeyValues

  Value set class.
- class KOpt

  KOpt class.
- class LogLevel

  Log level enumeration.
- class MissingEndpoint

  The policy to follow whenever and edge endpoint is missing during a loading.
- class NodeExport

  Stores the node exporting values.
- class NodeShape

  Node shape.
- class NodeTypeExporter

  NodeTypeExporter class.
- class NodeTypeLoader

  NodeTypeLoader class.
- class Objects

  Object identifier set class.
- class ObjectsIterator

  ObjectsIterator class.
- class ObjectType

  Object type enumeration.
- class OIDList

  Sparksee object identifier list.
- class OIDListIterator

  OIDList iterator class.
- class Order

  Order enumeration.
- class PageRank

  PageRank class.
- class Platform

  Platform class.
- class PlatformStatistics

  Platform data and statistics.
- class Query

  Query class.
- class QueryContext

  Query context interface.
- class QueryLanguage

  The supported query languages.

Type data class.

- class TypeExporter

  Base TypeExporter class.
- class TypeExporterEvent

  Provides information about the progress of an TypeExproter instance.
- class TypeExporterListener

  Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.
- class TypeList

  Sparksee type identifier list.
- class TypeListIterator

  TypeList iterator class.
- class TypeLoader

  Base TypeLoader class.
- class TypeLoaderEvent

  Provides information about the progress of a TypeLoader instance.
- class TypeLoaderListener
- class Value

  Value class.
- class ValueArray

  ValueArray class.
- class ValueList

  Value list.
- class ValueListIterator

  ValueList iterator class.
- class Values

  Value set class.
- class ValuesIterator

  Values iterator class.
- class WeakConnectivity

  WeakConnectivity class.
- class WeakConnectivityDFS

  WeakConnectivityDFS class.

Functions

- def load_nodes_csv (self, graph, file_name, node_type, separator, header, columns, attr_names, data_types, attr_kinds)

  Loads nodes from a CSV file.
- def load_edges_csv (self, graph, file_name, edge_type, tail_node_type, head_node_type, tail, head, separator, directed, header, on_missing_tail, on_missing_head, columns, attr_names, data_types, attr_kinds)

  Loads edges from a CSV file.

4.1.1 Detailed Description

Embedded Graph Database.

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

4.1.2 Function Documentation

4.1.2.1 def sparksee.load_edges_csv (*self, graph, file_name, edge_type, tail_node_type, head_node_type, tail, head, separator, directed, header, on_missing_tail, on_missing_head, columns, attr_names, data_types, attr_kinds*)

Loads edges from a CSV file.

nodeType[in] The type of the edge to load. If it does not exist, it creates it

Parameters

graph	[in] The graph to load the edges into
file_name	[in] The name of the file
edge_type	null
tail_node_type	[in] The type of the tail nodes. If it does not exist and onMissingTail is set to "Create", it creates it. Otherwise, an exception is thrown
head_node_type	[in] The type of the head nodes. It it does not exist and onMissingHead is set to "Create", it creates it. Otherwise, an exception is thrown
tail	[in] The tail column index. Default: 0
head	[in] The head column index. Default: 1
separator	[in] The column separator. Default: ","
directed	[in] True if this edge is directed or not. False otherwise. Default: True
header	[in] True if the CSV contains a header. False otherwise. Default: True
on_missing_tail	[in] The policy to follow when a tail is missing. Default: "IsError"
on_missing_head	[in] The policy to follow when a head is missing. Default: "IsError"
columns	[in] The list of columns to load. tail and head columns must be in this list if this list is specified. If the list is empty, all columns are loaded. Default: empty
attr_names	[in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty
data_types	[in] The dataTypes of the attributes if this do not already exist. If this list is empty, the method tries to infer the data type from the first non-header row. Default: empty
attr_kinds	[in] The attributeKinds of the attributes if these do not already exist. If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty

4.1.2.2 def sparksee.load_nodes_csv (*self, graph, file_name, node_type, separator, header, columns, attr_names, data_types, attr_kinds*)

Loads nodes from a CSV file.

Parameters

graph	[in] The graph to load the nodes into
file_name	[in] The name of the file
node_type	[in] The type of the node to load. If it does not exist, it creates it
separator	[in] The separator of the CSV file. Default: ","
header	[in] True if the CSV contains a header. False otherwise. Default: True
columns	[in] The list of columns to load. tail and head columns mus be in this list if this list is specified. Default: all columns
attr_names	[in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty
data_types	[in] The dataTypes of the attributes if this do not already exist. Default: empty If this list is empty, the method tries to infer the data type from the first non-header row

Parameters

attr_kinds	[in] The attributeKinds of the attributes if these do not already exist. Default: empty If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty

5 Class Documentation

5.1 sparksee.Attribute Class Reference

Attribute data class.

Public Member Functions

- def get_size (self)
  Gets the number of different values.
- def get_id (self)
  Gets the Sparksee attribute identifier.
- def get_type_id (self)
  Gets the Sparksee type identifier.
- def is_array_attribute (self)
  Check if it's an array attribute.
- def get_count (self)
  Gets the number of non-NULL values.
- def is_session_attribute (self)
  Check if it's a session attribute or a persistent one.
- def get_data_type (self)
  Gets the data type.
- def get_kind (self)
  Gets the attribute kind.
- def get_array_size (self)
  Gets the number of elements in the array.
- def get_name (self)
  Gets the unique attribute name.

Static Public Attributes

- int INVALID_ATTRIBUTE = 0
  Invalid attribute identifier constant.

5.1.1 Detailed Description

Attribute data class.

It contains information about an attribute.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.1.2 Member Function Documentation

5.1.2.1 def sparksee.Attribute.get_array_size (*self*)

Gets the number of elements in the array.

Returns

The size of the array

5.1.2.2 def sparksee.Attribute.get_count (*self*)

Gets the number of non-NULL values.

Returns

The number of non-NULL values.

5.1.2.3 def sparksee.Attribute.get_data_type (*self*)

Gets the data type.

Returns

The DataType.

5.1.2.4 def sparksee.Attribute.get_id (*self*)

Gets the Sparksee attribute identifier.

Returns

The Sparksee attribute identifier.

5.1.2.5 def sparksee.Attribute.get_kind (*self*)

Gets the attribute kind.

Returns

The AttributeKind.

5.1.2.6 def sparksee.Attribute.get_name (*self*)

Gets the unique attribute name.

Returns

The unique attribute name.

5.1.2.7 def sparksee.Attribute.get_size (*self*)

Gets the number of different values.

Returns

The number of different values.

5.1.2.8 def sparksee.Attribute.get_type_id (*self*)

Gets the Sparksee type identifier.

Returns

The Sparksee type identifier.

5.1.2.9 def sparksee.Attribute.is_array_attribute (*self*)

Check if it's an array attribute.

Returns

True if it's an array attribute, or false otherwise.

5.1.2.10 def sparksee.Attribute.is_session_attribute (*self*)

Check if it's a session attribute or a persistent one.

Returns

True if it's a session attribute, or false otherwise.

5.2 sparksee.AttributeKind Class Reference

Attribute kind enumeration.

Static Public Attributes

- int BASIC = 0

  Basic attribute (non indexed attribute).
- int INDEXED = 1

  Indexed attribute.
- int UNIQUE = 2

  Unique attribute (indexed + unique restriction).

5.2.1 Detailed Description

Attribute kind enumeration.

It determines the indexing-capabilities of an attribute.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.2.2 Member Data Documentation

5.2.2.1 int sparksee.AttributeKind.UNIQUE = 2 [static]

Unique attribute (indexed + unique restriction).

Unique restriction sets two objects cannot have the same value for an attribute but NULL.

5.3 sparksee.AttributeList Class Reference

Sparksee attribute identifier list.

Public Member Functions

- def add (self, attr)

  Adds a Sparksee attribute identifier at the end of the list.
- def clear (self)

  Clears the list.
- def __iter__ (self)

  Gets a new TypeListIterator.
- def iterator (self)

  Gets a new AttributeListIterator.
- def __init__ (self)

  Constructor.
- def count (self)

  Number of elements in the list.

5.3.1 Detailed Description

Sparksee attribute identifier list.

It stores a Sparksee attribute identifier list.

Use AttributeListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.3.2 Constructor & Destructor Documentation

5.3.2.1 def sparksee.AttributeList.__init__ (*self*)

Constructor.

This creates an empty list.

5.3.3 Member Function Documentation

5.3.3.1 def sparksee.AttributeList.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.3.3.2 def sparksee.AttributeList.add (*self, attr*)

Adds a Sparksee attribute identifier at the end of the list.

Parameters

attr	[in] Sparksee attribute identifier.

5.3.3.3 def sparksee.AttributeList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.3.3.4 def sparksee.AttributeList.iterator (*self*)

Gets a new AttributeListIterator.

Returns

AttributeListIterator instance.

5.4 sparksee.AttributeListIterator Class Reference

AttributeList iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.4.1 Detailed Description

AttributeList iterator class.

Iterator to traverse all the Sparksee attribute identifier into a AttributeList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.4.2 Member Function Documentation

5.4.2.1 def sparksee.AttributeListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.4.2.2 def sparksee.AttributeListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.4.2.3 def sparksee.AttributeListIterator.next (*self*)

Moves to the next element.

Returns

The next element.

5.5 sparksee.AttributeStatistics Class Reference

Attribute statistics class.

Public Member Functions

- def get_min_length_string (self)
  Gets the minimum length.
- def get_mode_count (self)
  Gets the number of objects with a Value equal to the mode.
- def get_variance (self)
  Gets the variance.
- def get_mode (self)
  Gets the mode.
- def get_min (self)
  Gets the minimum existing value (BASIC statistics).
- def get_median (self)
  Gets the median.
- def get_total (self)
  Gets the number of objects with a non-NULL Value (BASIC statistic).
- def get_max_length_string (self)
  Gets the maximum length.
- def get_mean (self)
  Gets the mean or average.
- def get_null (self)
  Gets the number of objects NULL a Value (BASIC statistics).
- def get_distinct (self)
  Gets the number of distinct values (BASIC statistics).
- def get_avg_length_string (self)
  Gets the average length.
- def get_max (self)
  Gets the maximum existing value (BASIC statistics).

5.5.1 Detailed Description

Attribute statistics class.

It contains statistic data about an attribute.

Some fields are valid just for numerical attributes and others just for string attributes. Also, some statistics are considered BASIC because computing them do not require to traverse all the different values of the attribute. For each getter method the documentation tells if the statistic is BASIC or not. See the Graph class method get↩ AttributeStatistics or check out the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.5.2 Member Function Documentation

5.5.2.1 def sparksee.AttributeStatistics.get_avg_length_string (*self*)

Gets the average length.

If the attribute is not an string attribute, it just returns 0.

Returns

The average length.

5.5.2.2 def sparksee.AttributeStatistics.get_distinct (*self*)

Gets the number of distinct values (BASIC statistics).

Returns

The number of distinct values.

5.5.2.3 def sparksee.AttributeStatistics.get_max (*self*)

Gets the maximum existing value (BASIC statistics).

Returns

The maximum existing value.

5.5.2.4 def sparksee.AttributeStatistics.get_max_length_string (*self*)

Gets the maximum length.

If the attribute is not an string attribute, it just returns 0.

Returns

The maximum length.

5.5.2.5 def sparksee.AttributeStatistics.get_mean (*self*)

Gets the mean or average.

Mean or average: Sum of all Values divided by the number of observations.

It is computed just for numerical attributes.

Returns

The mean.

5.5.2.6 def sparksee.AttributeStatistics.get_median (*self*)

Gets the median.

Median: Middle value that separates the higher half from the lower.

If a $<$ b $<$ c, then the median of the list {a, b, c} is b, and if a $<$ b $<$ c $<$ d, then the median of the list {a, b, c, d} is the mean of b and c, i.e. it is (b + c)/2

It is computed just for numerical attributes.

Returns

The median.

5.5.2.7 def sparksee.AttributeStatistics.get_min (*self*)

Gets the minimum existing value (BASIC statistics).

Returns

The minimum existing value.

5.5.2.8 def sparksee.AttributeStatistics.get_min_length_string (*self*)

Gets the minimum length.

If the attribute is not an string attribute, it just returns 0.

Returns

The minimum length.

5.5.2.9 def sparksee.AttributeStatistics.get_mode (*self*)

Gets the mode.

Mode: Most frequent Value.

Returns

The mode.

5.5.2.10 def sparksee.AttributeStatistics.get_mode_count (*self*)

Gets the number of objects with a Value equal to the mode.

Returns

The number of objects with a Value equal to the mode.

5.5.2.11 def sparksee.AttributeStatistics.get_null (*self*)

Gets the number of objects NULL a Value (BASIC statistics).

Returns

The number of objects NULL a Value.

5.5.2.12 def sparksee.AttributeStatistics.get_total (*self*)

Gets the number of objects with a non-NULL Value (BASIC statistic).

Returns

The number of objects with a non-NULL Value.

5.5.2.13 def sparksee.AttributeStatistics.get_variance (*self*)

Gets the variance.

It is computed just for numerical attributes.

Returns

The variance.

5.6 sparksee.BooleanList Class Reference

Boolean list.

Public Member Functions

- def clear (self)
  Clears the list.
- def __init__ (self)
  Constructor.
- def __iter__ (self)
  Gets a new TypeListIterator.
- def iterator (self)
  Gets a new BooleanListIterator.
- def add (self, value)
  Adds a Boolean at the end of the list.
- def count (self)
  Number of elements in the list.

5.6.1 Detailed Description

Boolean list.

It stores a Boolean list.

Use BooleanListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.6.2 Constructor & Destructor Documentation

5.6.2.1 def sparksee.BooleanList.__init__ (*self*)

Constructor.

This creates an empty list.

5.6.3 Member Function Documentation

5.6.3.1 def sparksee.BooleanList.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.6.3.2 def sparksee.BooleanList.add (*self, value*)

Adds a Boolean at the end of the list.

Parameters

value	[in] Boolean.

5.6.3.3 def sparksee.BooleanList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.6.3.4 def sparksee.BooleanList.iterator (*self*)

Gets a new BooleanListIterator.

Returns

BooleanListIterator instance.

5.7 sparksee.BooleanListIterator Class Reference

BooleanList iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.7.1 Detailed Description

BooleanList iterator class.

Iterator to traverse all the strings into a BooleanList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.7.2 Member Function Documentation

5.7.2.1 def sparksee.BooleanListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.7.2.2 def sparksee.BooleanListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.7.2.3 def sparksee.BooleanListIterator.next (*self*)

Moves to the next element.

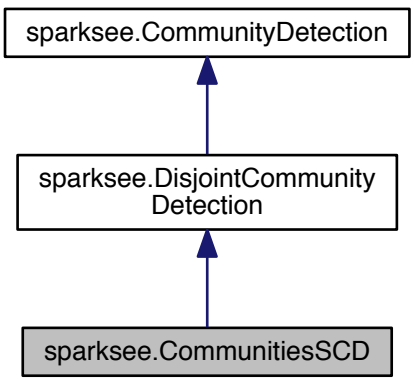
Returns

The next element.

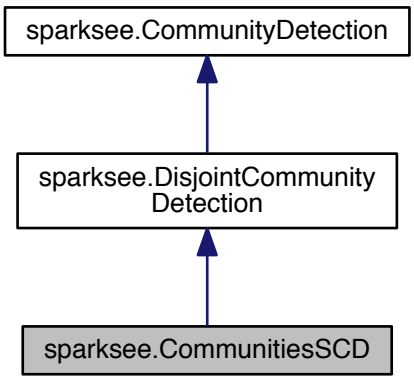
5.8 sparksee.CommunitiesSCD Class Reference

CommunitiesSCD class.

Inheritance diagram for sparksee.CommunitiesSCD:



Collaboration diagram for sparksee.CommunitiesSCD:



Public Member Functions

- def __init__ (self, session)

  Creates a new instance of CommunitiesSCD.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self)

  Allows connectivity through all edge types of the graph.
- def get_communities (self)

  Returns the results generated by the execution of the algorithm.
- def run (self)

  Executes the algorithm.
- def add_edge_type (self, type)

  Allows connectivity through edges of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.
- def add_node_type (self, type)

  Allows connectivity through nodes of the given type.
- def include_nodes (self, nodes)

  Set additional nodes that can be used.
- def set_look_ahead (self, lookahead)

  Sets the size of the lookahead iterations to look (5 by default).
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def include_edges (self, edges)

  Set additional edges that can be used.
- def close (self)

  Closes the CommunityDetection instance.
- def is_closed (self)

  Gets if CommunityDetection has been closed or not.

5.8.1 Detailed Description

CommunitiesSCD class.

Implementation of the community detection algorithm "Scalable Community Detection" based on the paper "High quality, scalable and parallel community detection for large real graphs" by Arnau Prat-Perez, David Dominguez-Sal, Josep-Lluis Larriba-Pey - WWW 2014.

The purpose of this algorithm is to find disjoint communities in an undirected graph or in a directed graph which will be considered as an undirected one.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the DisjointCommunities class using the getCommunities method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.8.2 Constructor & Destructor Documentation

5.8.2.1 def sparksee.CommunitiesSCD.__init__ (*self, session*)

Creates a new instance of CommunitiesSCD.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the communities.

Parameters

session	[in] Session to get the graph from and calculate the communities

5.8.3 Member Function Documentation

5.8.3.1 def sparksee.CommunitiesSCD.add_all_edge_types (*self*)

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

5.8.3.2 def sparksee.CommunitiesSCD.add_edge_type (*self, type*)

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters

type	[in] Edge type.

5.8.3.3 def sparksee.CommunitiesSCD.add_node_type (*self, type*)

Allows connectivity through nodes of the given type.

Parameters

type	null

5.8.3.4 def sparksee.CommunityDetection.close (*self*) [inherited]

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

5.8.3.5 def sparksee.CommunitiesSCD.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.8.3.6 def sparksee.CommunitiesSCD.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.8.3.7 def sparksee.CommunitiesSCD.get_communities (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class DisjointCommunities which contain information related to the disjoint communities found.

5.8.3.8 def sparksee.CommunitiesSCD.include_edges (*self, edges*)

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.8.3.9 def sparksee.CommunitiesSCD.include_nodes (*self, nodes*)

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.8.3.10 def sparksee.CommunityDetection.is_closed (*self*) [inherited]

Gets if CommunityDetection has been closed or not.

See also

close()

Returns

TRUE if the CommunityDetection instance has been closed, FALSE otherwise.

5.8.3.11 def sparksee.CommunitiesSCD.set_look_ahead (*self, lookahead*)

Sets the size of the lookahead iterations to look (5 by default).

Parameters

lookahead	[in] Number of iterations. It must be positive or zero.

5.8.3.12 def sparksee.CommunitiesSCD.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the disjoint communities found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

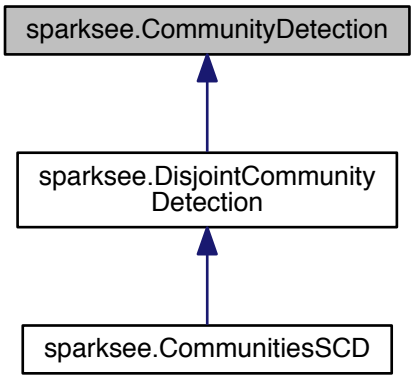
Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.9 sparksee.CommunityDetection Class Reference

CommunityDetection class.

Inheritance diagram for sparksee.CommunityDetection:



Public Member Functions

- def run (self)

  Runs the algorithm in order to find the connected components.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_node_type (self, type)

  Allows connectivity through nodes of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def close (self)

  Closes the CommunityDetection instance.
- def include_nodes (self, nodes)

Set additional nodes that can be used.

- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.

- def include_edges (self, edges)

  Set additional edges that can be used.

- def is_closed (self)

  Gets if CommunityDetection has been closed or not.

5.9.1 Detailed Description

CommunityDetection class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.9.2 Member Function Documentation

5.9.2.1 def sparksee.CommunityDetection.add_node_type (*self, type*)

Allows connectivity through nodes of the given type.

Parameters

type	null

5.9.2.2 def sparksee.CommunityDetection.close (*self*)

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

5.9.2.3 def sparksee.CommunityDetection.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.9.2.4 def sparksee.CommunityDetection.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.9.2.5 def sparksee.CommunityDetection.include_edges (*self, edges*)

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.9.2.6 def sparksee.CommunityDetection.include_nodes (*self, nodes*)

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.9.2.7 def sparksee.CommunityDetection.is_closed (*self*)

Gets if CommunityDetection has been closed or not.

See also

close()

Returns

TRUE if the CommunityDetection instance has been closed, FALSE otherwise.

5.9.2.8 def sparksee.CommunityDetection.run (*self*)

Runs the algorithm in order to find the connected components.

This method can be called only once.

5.10 sparksee.Condition Class Reference

Condition operators enumeration.

Static Public Attributes

- int EQUAL = 0

  Equal condition (==).
- int GREATER_EQUAL = 1

  Greater or equal condition (>=).
- int GREATER_THAN = 2

  Greater than condition (>).
- int LESS_EQUAL = 3

  Less or equal condition (<=).
- int LESS_THAN = 4

  Less than condition (<).
- int NOT_EQUAL = 5

  Not equal condition (!=).
- int LIKE = 6

  Substring condition.
- int LIKE_NO_CASE = 7

  Substring (no case sensitive) condition.
- int BETWEEN = 8

  In range operator condition ([x,y]).
- int REG_EXP = 9

  Regular expression condition.

5.10.1 Detailed Description

Condition operators enumeration.

It is mainly used in the attribute-based graph select operations.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.10.2 Member Data Documentation

5.10.2.1 int sparksee.Condition.BETWEEN = 8 [static]

In range operator condition ([x,y]).

Null values cannot be used together with this condition.

5.10.2.2 int sparksee.Condition.EQUAL = 0 [static]

Equal condition (==).

Null values can be used together with this condition to retrieve all objects having a null value for an attribute.

5.10.2.3 int sparksee.Condition.GREATER_EQUAL = 1 [static]

Greater or equal condition (>=).

Null values cannot be used together with this condition.

5.10.2.4 int sparksee.Condition.GREATER_THAN = 2 [static]

Greater than condition (>).

Null values cannot be used together with this condition.

5.10.2.5 int sparksee.Condition.LESS_EQUAL = 3 [static]

Less or equal condition (<=).

Null values cannot be used together with this condition.

5.10.2.6 int sparksee.Condition.LESS_THAN = 4 [static]

Less than condition (<).

Null values cannot be used together with this condition.

5.10.2.7 int sparksee.Condition.LIKE = 6 [static]

Substring condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (case sensitive). Ex:

'AAABBBCCCD' Like 'BBB' returns TRUE

'AAABBBCCCD' Like 'bbb' returns FALSE

'AAABBBCCCD' Like 'E' returns FALSE

5.10.2.8 int sparksee.Condition.LIKE_NO_CASE = 7 [static]

Substring (no case sensitive) condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:

'AAABBBCCCD' LikeNoCase 'BBB' returns TRUE

'AAABBBCCCD' LikeNoCase 'bbb' returns TRUE

'AAABBBCCCD' LikeNoCase 'E' returns FALSE

5.10.2.9 int sparksee.Condition.NOT_EQUAL = 5 [static]

Not equal condition (!=).

Null values can be used together with this condition to retrieve all objects having a non-null value for an attribute.

5.10.2.10 int sparksee.Condition.REG_EXP = 9 [static]

Regular expression condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values.

Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.

See the 'Regular expressions' section in the 'SPARKSEE User Manual' for details.

5.11 sparksee.ConnectedComponents Class Reference

ConnectedComponents class.

Public Member Functions

- def __init__ (self, s, materializedattribute)

  Creates a new instance of ConnectedComponents.
- def get_size (self, id_connected_component)

  Returns the number of nodes contained in the given connected component.
- def get_count (self)

  Returns the number of connected components found in the graph.
- def get_connected_component (self, id_node)

  Returns the connected component where the given node belongs to.
- def get_nodes (self, id_connected_component)

  Returns the collection of nodes contained in the given connected component.
- def close (self)

  Closes the ConnectedComponents instance.
- def is_closed (self)

  Gets if ConnectedComponents has been closed or not.

5.11.1 Detailed Description

ConnectedComponents class.

This class contains the results processed on a Connectivity algorithm.

These results contain information related to the connected components found. We must consider that each connected component has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different connected components found.

When executing any implementation of the Connectivity, it is possible to indicate whether the results of the execution must be stored persistently using the class Connectivity setMaterializedAttribute method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.11.2 Constructor & Destructor Documentation

5.11.2.1 def sparksee.ConnectedComponents.__init__ (*self, s, materializedattribute*)

Creates a new instance of ConnectedComponents.

This constructor method can only be called when a previous execution of any implementation of the Connectivity class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any Connectivity execution see the documentation of the Connectivity::SetMaterializedAttribute method.

Parameters

s	[in] Session to get the graph Graph on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
materializedattribute	[in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the connected components found in the graph.

5.11.3 Member Function Documentation

5.11.3.1 def sparksee.ConnectedComponents.close (*self*)

Closes the ConnectedComponents instance.

It must be called to ensure the integrity of all data.

5.11.3.2 def sparksee.ConnectedComponents.get_connected_component (*self, id_node*)

Returns the connected component where the given node belongs to.

Parameters

id_node	[in] The node identifier for which the connected component identifier where it belongs will be returned.

Returns

The connected component identifier where the given node identifier belongs to.

5.11.3.3 def sparksee.ConnectedComponents.get_count (*self*)

Returns the number of connected components found in the graph.

Returns

The number of connected components found in the graph.

5.11.3.4 def sparksee.ConnectedComponents.get_nodes (*self, id_connected_component*)

Returns the collection of nodes contained in the given connected component.

Parameters

id_connected_component	The connected component for which the collection of nodes contained in it will be returned.

Returns

The collection of node identifiers contained in the given connected component.

5.11.3.5 def sparksee.ConnectedComponents.get_size (*self, id_connected_component*)

Returns the number of nodes contained in the given connected component.

Parameters

id_connected_component	The connected component for which the number of nodes contained in it will be returned.

Returns

The number of nodes contained in the given connected component.

5.11.3.6 def sparksee.ConnectedComponents.is_closed (*self*)

Gets if ConnectedComponents has been closed or not.

See also

close()

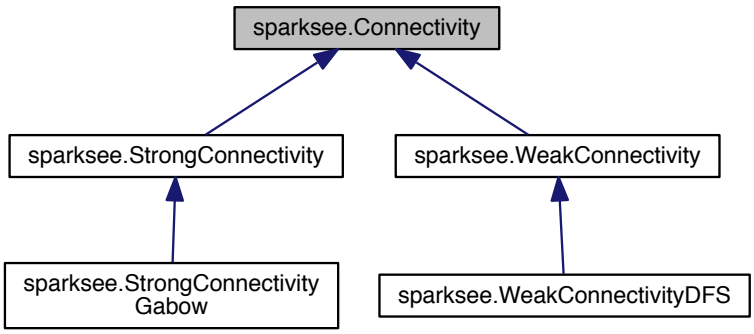
Returns

TRUE if the ConnectedComponents instance has been closed, FALSE otherwise.

5.12 sparksee.Connectivity Class Reference

Connectivity class.

Inheritance diagram for sparksee.Connectivity:



Public Member Functions

- def run (self)

  Runs the algorithm in order to find the connected components.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def get_connected_components (self)

  Returns the results generated by the execution of the algorithm.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_node_type (self, t)

  Allows connectivity through nodes of the given type.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- def close (self)

  Closes the Connectivity instance.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def is_closed (self)

  Gets if Connectivity has been closed or not.

5.12.1 Detailed Description

Connectivity class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.12.2 Member Function Documentation

5.12.2.1 def sparksee.Connectivity.add_node_type (*self, t*)

Allows connectivity through nodes of the given type.

Parameters

t	null

5.12.2.2 def sparksee.Connectivity.close (*self*)

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

5.12.2.3 def sparksee.Connectivity.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.12.2.4 def sparksee.Connectivity.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.12.2.5 def sparksee.Connectivity.get_connected_components (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

5.12.2.6 def sparksee.Connectivity.is_closed (*self*)

Gets if Connectivity has been closed or not.

See also

close()

Returns

TRUE if the Connectivity instance has been closed, FALSE otherwise.

5.12.2.7 def sparksee.Connectivity.run (*self*)

Runs the algorithm in order to find the connected components.

This method can be called only once.

5.12.2.8 def sparksee.Connectivity.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.13 sparksee.Context Class Reference

Context class.

Public Member Functions

- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def __init__ (self, session, node)

  Creates a new instance.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def compute (self)

  Gets the resulting collection of nodes.
- def add_edge_type (self, t, d)

  Allows for traversing edges of the given type.
- def compute (self, session, node, node_types, edge_types, dir, maxhops, include)

  Helper method to easily compute a context from a node.
- def set_maximum_hops (self, maxhops, include)

  Sets the maximum hops restriction.
- def add_node_type (self, t)

  Allows for traversing nodes of the given type.
- def close (self)

  Closes the Context instance.
- def add_all_edge_types (self, d)

  Allows for traversing all edge types of the graph.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def is_closed (self)

  Gets if Context has been closed or not.

5.13.1 Detailed Description

Context class.

It provides a very similar functionality than the Traversal classes. The main difference is Context returns a resulting collection whereas Traversal provides an iterator behaviour.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.13.2 Constructor & Destructor Documentation

5.13.2.1 def sparksee.Context.__init__ (*self, session, node*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform operation.
node	[in] Node to start traversal from.

5.13.3 Member Function Documentation

5.13.3.1 def sparksee.Context.add_all_edge_types (*self, d*)

Allows for traversing all edge types of the graph.

Parameters

d	[in] Edge direction.

5.13.3.2 def sparksee.Context.add_edge_type (*self, t, d*)

Allows for traversing edges of the given type.

Parameters

t	[in] Edge type.
d	[in] Edge direction.

5.13.3.3 def sparksee.Context.add_node_type (*self, t*)

Allows for traversing nodes of the given type.

Parameters

t	null

5.13.3.4 def sparksee.Context.close (*self*)

Closes the Context instance.

It must be called to ensure the integrity of all data.

5.13.3.5 def sparksee.Context.compute (*self*)

Gets the resulting collection of nodes.

Returns

The resulting collection of nodes.

Referenced by sparksee.Context.compute().

5.13.3.6 def sparksee.Context.compute (*self, session, node, node_types, edge_types, dir, maxhops, include*)

Helper method to easily compute a context from a node.

Parameters

session	[in] Session to get the graph from and perform operation.
node	[in] Node to start traversal from.
node_types	[in] Allowed node type list. NULL means all node types are allowed.
edge_types	[in] Allowed edge type list. NULL means all edge types are allowed.
dir	[in] Allowed direction for the allowed edge types.
maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
include	[in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

Returns

Returns an Objects with the computed context of a node.

References sparksee.Context.compute().

5.13.3.7 def sparksee.Context.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.13.3.8 def sparksee.Context.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.13.3.9 def sparksee.Context.is_closed (*self*)

Gets if Context has been closed or not.

See also

close()

Returns

TRUE if the Context instance has been closed, FALSE otherwise.

5.13.3.10 def sparksee.Context.set_maximum_hops (*self, maxhops, include*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

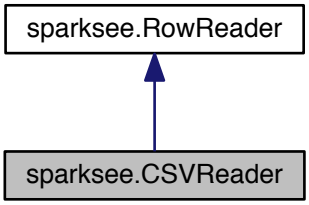
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
include	[in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

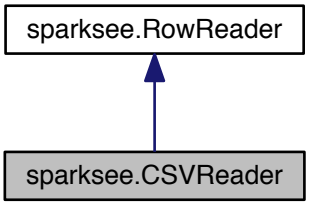
5.14 sparksee.CSVReader Class Reference

CSVReader interface.

Inheritance diagram for sparksee.CSVReader:



Collaboration diagram for sparksee.CSVReader:



Public Member Functions

- def set_separator (self, sep)

  Sets the character used to separate fields in the file.
- def set_num_lines (self, num_lines)

  Used to limit the number of lines that will be read.
- def close (self)

  Closes the reader.
- def set_locale (self, locale_str)

  Sets the locale that will be used to read the file.
- def read (self, row)

  Reads the next row as a string array.
- def set_multilines (self, num_extralines)

  Allows the use of fields with more than one line.
- def open (self, file_path)

  Opens the source file path.
- def reset (self)

  Moves the reader to the beginning.
- def set_start_line (self, start_line)

  Sets the number of lines to be skiped from the beginning.

- def set_single_line (self)

  Only allows single line fields.
- def get_row (self)

  The row number for the current row.
- def __init__ (self)

  Constructs CSVReader.
- def set_quotes (self, quotes)

  Sets the character used to quote fields.

5.14.1 Detailed Description

CSVReader interface.

A very simple CSV reader.

It works as any other RowReader, but open must be called once before the first read operation.

Using the format RFC 4180.

Except: leading and trailing spaces, adjacent to CSV separator character, are trimmed.

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (").

Fields with multiple lines can be allowed (and the maximum lines specified), but the default is a single line.

The locale string can be used to set the language, country and the file encoding. The format must be "[language↩_territory][.codeset]". But only the file encoding is being used in the current version.

The languages supported are: "en_US", "es_ES" and "ca_ES".

The file encodings supported are: "utf8" and "iso88591".

For example:

To don't change the default locales, use an empty string: "".

To read a file in utf8 with the default language settings use ".utf8".

To read a file in iso88591 with English language use: "en_US.iso88591".

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.14.2 Member Function Documentation

5.14.2.1 def sparksee.CSVReader.close (*self*)

Closes the reader.

Exceptions

IOError	If the close fails.

5.14.2.2 def sparksee.CSVReader.get_row (*self*)

The row number for the current row.

Returns

The current row number; 0 if there is no current row.

Exceptions

IOError	If it fails.

5.14.2.3 def sparksee.CSVReader.open (*self, file_path*)

Opens the source file path.

File can be optionally compressed in GZIP format.

Parameters

file_path	[in] CSV file path.

Exceptions

IOError	If bad things happen opening the file.

5.14.2.4 def sparksee.CSVReader.read (*self, row*)

Reads the next row as a string array.

Parameters

row	[out] A string list with each comma-separated element as a separate entry.

Returns

Returns true if a row had been read or false otherwise.

Exceptions

IOError	If bad things happen during the read.

5.14.2.5 def sparksee.CSVReader.reset (*self*)

Moves the reader to the beginning.

Restarts the reader.

Returns

true if the reader can be restarted, false otherwise.

Exceptions

IOError	If bad things happen during the restart.

5.14.2.6 def sparksee.CSVReader.set_locale (*self, locale_str*)

Sets the locale that will be used to read the file.

Parameters

locale_str	[in] The locale string for the file encoding.

5.14.2.7 def sparksee.CSVReader.set_multilines (*self, num_extralines*)

Allows the use of fields with more than one line.

Parameters

num_extralines	[in] Maximum number of extra lines for each column (0==unlimited, N==N+1 total rows).

5.14.2.8 def sparksee.CSVReader.set_num_lines (*self, num_lines*)

Used to limit the number of lines that will be read.

Parameters

num_lines	[in] The maximum number of lines to read (0 == unlimited)

5.14.2.9 def sparksee.CSVReader.set_quotes (*self, quotes*)

Sets the character used to quote fields.

Parameters

quotes	[in] Quote character.

Exceptions

RuntimeError	null

5.14.2.10 def sparksee.CSVReader.set_separator (*self, sep*)

Sets the character used to separate fields in the file.

Parameters

sep	[in] Separator character.

Exceptions

RuntimeError	null

5.14.2.11 def sparksee.CSVReader.set_start_line (*self, start_line*)

Sets the number of lines to be skiped from the beginning.

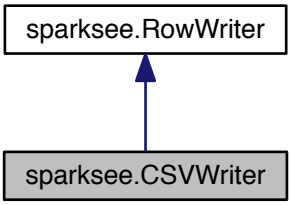
Parameters

start_line	[in] The line number to skip for start reading

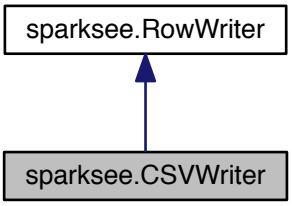
5.15 sparksee.CSVWriter Class Reference

CSVWriter interface.

Inheritance diagram for sparksee.CSVWriter:



Collaboration diagram for sparksee.CSVWriter:



Public Member Functions

- def __init__ (self)
  Creates a new instance.
- def set_separator (self, sep)
  Sets the character used to separate fields in the file.
- def set_auto_quotes (self, autoquotes)
  Sets on/off the automatic quote mode.
- def open (self, f)
  Opens the output file path.
- def write (self, row)
  Writes the next row.
- def set_forced_quotes (self, forcequotes)
  Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.
- def close (self)
  Closes the writer.
- def set_locale (self, locale_str)
  Sets the locale that will be used to write the file.
- def set_quotes (self, quotes)
  Sets the character used to quote fields.

5.15.1 Detailed Description

CSVWriter interface.

A very simple CSV writer implementing RowWriter.

It works as any other RowWriter, but open must be called once before the first write operation.

It uses the format RFC 4180: http://tools.ietf.org/html/rfc4180

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (") and autoquote is enabled.

See the CSVReader locale documentation or the SPARKSEE User Manual.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.15.2 Member Function Documentation

5.15.2.1 def sparksee.CSVWriter.close (*self*)

Closes the writer.

Exceptions

RuntimeError	null
IOError	If the close fails.

5.15.2.2 def sparksee.CSVWriter.open (*self, f*)

Opens the output file path.

Parameters

f	[in] Output file path.

Exceptions

IOError	If bad things happen opening the file.

5.15.2.3 def sparksee.CSVWriter.set_auto_quotes (*self, autoquotes*)

Sets on/off the automatic quote mode.

If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced quotes are provided, there will not be any quote.

Parameters

autoquotes	[in] If TRUE it enables the automatic quote mode, if FALSE it disables it.

5.15.2.4 def sparksee.CSVWriter.set_forced_quotes (*self, forcequotes*)

Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

Parameters

forcequotes	[in] A booleanList with the position for each column that must be quoted set to true.

5.15.2.5 def sparksee.CSVWriter.set_locale (*self, locale_str*)

Sets the locale that will be used to write the file.

Parameters

locale_str	[in] The locale string for the file encoding.

5.15.2.6 def sparksee.CSVWriter.set_quotes (*self, quotes*)

Sets the character used to quote fields.

Parameters

quotes	[in] Quote character.

Exceptions

RuntimeError	null

5.15.2.7 def sparksee.CSVWriter.set_separator (*self, sep*)

Sets the character used to separate fields in the file.

Parameters

sep	[in] Separator character.

Exceptions

RuntimeError	null

5.15.2.8 def sparksee.CSVWriter.write (*self, row*)

Writes the next row.

Parameters

row	[in] Row of data.

Exceptions

RuntimeError	null
IOError	If bad things happen during the write.

5.16 sparksee.Database Class Reference

Database class.

Public Member Functions

- def dump_schema (self, file_path)

  Dump the database schema to the given file using the Sparksee scripting format.
- def new_session (self)

  Creates a new Session.
- def get_path (self)

  Gets the path of the Database.
- def enable_rollback (self)

  Enables the rollback mechanism.
- def fix_current_cache_max_size (self)

  Sets the cache maximum size to the current cache size in use.
- def redo_precommitted (self, tx_id)

  Redo a pending precommitted transaction recovered.
- def get_statistics (self, stats)

  Gets Database statistics.
- def set_cache_max_size (self, mega_bytes)

  Sets the cache maximum size (in MB).
- def disable_rollback (self)

  Disables the rollback mechanism.
- def close (self)

  Closes the Database instance.
- def get_alias (self)

  Gets the alias of the Database.
- def get_cache_max_size (self)

  Gets the cache maximum size (in MB).
- def is_closed (self)

  Gets if Database instance has been closed or not.

5.16.1 Detailed Description

Database class.

All the data of the Database is stored into a persistent file which just can be created or open through a Sparksee instance.

Also, all the manipulation of a Database must be done by means of a Session which can be initiated from a Database instance.

Multiple Databases do not share the memory, that is there is no negotiation among them. In those cases, memory must be prefixed for each Database. To do that, use the SPARKSEEConfig.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.16.2 Member Function Documentation

5.16.2.1 def sparksee.Database.close (*self*)

Closes the Database instance.

It must be called to ensure the integrity of all data.

5.16.2.2 def sparksee.Database.dump_schema (*self, file_path*)

Dump the database schema to the given file using the Sparksee scripting format.

Parameters

file_path	null

5.16.2.3 def sparksee.Database.get_alias (*self*)

Gets the alias of the Database.

Returns

The alias of the Database.

5.16.2.4 def sparksee.Database.get_cache_max_size (*self*)

Gets the cache maximum size (in MB).

Returns

Returns the current cache max size.

5.16.2.5 def sparksee.Database.get_path (*self*)

Gets the path of the Database.

Returns

The path of the Database.

5.16.2.6 def sparksee.Database.get_statistics (*self, stats*)

Gets Database statistics.

Parameters

stats	[out] The DatabaseStatistics instance.

5.16.2.7 def sparksee.Database.is_closed (*self*)

Gets if Database instance has been closed or not.

See also

close()

Returns

TRUE if the Database instance has been closed, FALSE otherwise.

5.16.2.8 def sparksee.Database.redo_precommitted (*self, tx_id*)

Redo a pending precommitted transaction recovered.

YOU SHOULD NOT USE THIS METHOD. This method only exists for a specific service.

Parameters

tx↩ _id	null

5.16.2.9 def sparksee.Database.set_cache_max_size (*self, mega_bytes*)

Sets the cache maximum size (in MB).

0 means unlimited which is all the physical memory of the computer minus a small margin.

Parameters

mega_bytes	[in] The new cache max size.

5.17 sparksee.DatabaseStatistics Class Reference

Database statistics.

Public Member Functions

- def get_read (self)
  Gets total read data in KBytes.
- def get_data (self)
  Gets database size in KBytes.
- def get_temp (self)
  Gets temporary storage file size in KBytes.
- def get_write (self)
  Gets total written data in KBytes.
- def get_cache (self)
  Gets cache size in KBytes.
- def get_sessions (self)
  Gets the number of sessions.

5.17.1 Detailed Description

Database statistics.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.17.2 Member Function Documentation

5.17.2.1 def sparksee.DatabaseStatistics.get_cache (*self*)

Gets cache size in KBytes.

Returns

Cache size in KBytes.

5.17.2.2 def sparksee.DatabaseStatistics.get_data (*self*)

Gets database size in KBytes.

Returns

Database size in KBytes.

5.17.2.3 def sparksee.DatabaseStatistics.get_read (*self*)

Gets total read data in KBytes.

Returns

Total read data in KBytes.

5.17.2.4 def sparksee.DatabaseStatistics.get_sessions (*self*)

Gets the number of sessions.

Returns

The number of sessions.

5.17.2.5 def sparksee.DatabaseStatistics.get_temp (*self*)

Gets temporary storage file size in KBytes.

Returns

Temporary storage file size in KBytes.

5.17.2.6 def sparksee.DatabaseStatistics.get_write (*self*)

Gets total written data in KBytes.

Returns

Total read written in KBytes.

5.18 sparksee.DataType Class Reference

Data type (domain) enumeration.

Static Public Attributes

- int BOOLEAN = 0

  Boolean data type.
- int INTEGER = 1

  32-bit signed integer data type.
- int LONG = 2

  64-bit signed integer data type.
- int DOUBLE = 3

  64-bit signed double data type.
- int TIMESTAMP = 4

  Distance from Epoch (UTC) time in milliseconds precision.
- int STRING = 5

  Unicode string data type.
- int TEXT = 6

  Large unicode character object (CLOB) data type.
- int OID = 7

  Object identifier (OID) data type.

5.18.1 Detailed Description

Data type (domain) enumeration.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.18.2 Member Data Documentation

5.18.2.1 int sparksee.DataType.STRING = 5 [static]

Unicode string data type.

2048 characters maximum length.

5.18.2.2 int sparksee.DataType.TEXT = 6 [static]

Large unicode character object (CLOB) data type.

TextStream

5.18.2.3 int sparksee.DataType.TIMESTAMP = 4 [static]

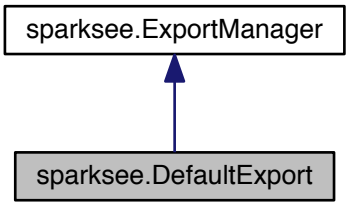
Distance from Epoch (UTC) time in milliseconds precision.

It just works properly with timestamps in the range ['1970-01-01 00:00:01' UTC, '2038-01-19 03:14:07' UTC].

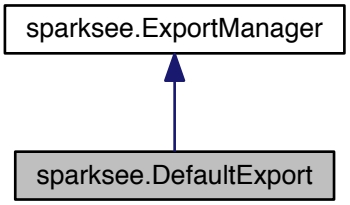
5.19 sparksee.DefaultExport Class Reference

Default implementation for ExportManager class.

Inheritance diagram for sparksee.DefaultExport:



Collaboration diagram for sparksee.DefaultExport:



Public Member Functions

- def prepare (self, graph)
  Default implementation of the ExportManager class method Prepare.
- def get_graph (self, graph_export)
  Default implementation of the ExportManager class method GetGraph.
- def release (self)
  Default implementation of the ExportManager class method Release.
- def __init__ (self)
  Creates a new instance.
- def get_node_type (self, type, node_export)
  Default implementation of the ExportManager class method GetNodeType.
- def enable_type (self, type)
  Default implementation of the ExportManager class method EnableType.
- def get_node (self, node, node_export)
  Default implementation of the ExportManager class method GetNode.
- def get_edge (self, edge, edge_export)
  Default implementation of the ExportManager class method GetEdge.
- def get_edge_type (self, type, edge_export)
  Default implementation of the ExportManager class method GetEdgeType.

5.19.1 Detailed Description

Default implementation for ExportManager class.

It uses the default values from GraphExport, NodeExport and EdgeExport to export all node and edge types.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.19.2 Member Function Documentation

5.19.2.1 def sparksee.DefaultExport.enable_type (*self, type*)

Default implementation of the ExportManager class method EnableType.

This enables all node and edge types to be exported.

Parameters

type	[in] The type to enable.

Returns

TRUE.

5.19.2.2 def sparksee.DefaultExport.get_edge (*self, edge, edge_export*)

Default implementation of the ExportManager class method GetEdge.

This sets the default EdgeExport values and sets the OID as the label. Also, it exports the edge as directed just if the edge is directed.

Parameters

edge	[in] An edge.
edge_export	[out] The EdgeExport that will store the information.

Returns

TRUE.

5.19.2.3 def sparksee.DefaultExport.get_edge_type (*self, type, edge_export*)

Default implementation of the ExportManager class method GetEdgeType.

This sets de default EdgeExport values.

Parameters

type	[in] An edge type.
edge_export	[out] The EdgeExport that will store the information.

Returns

TRUE.

5.19.2.4 def sparksee.DefaultExport.get_graph (*self, graph_export*)

Default implementation of the ExportManager class method GetGraph.

This sets the default GraphExport values and "Graph" as the label.

Parameters

graph_export	[out] The GraphExport that will store the information.

Returns

TRUE.

5.19.2.5 def sparksee.DefaultExport.get_node (*self, node, node_export*)

Default implementation of the ExportManager class method GetNode.

This sets the default NodeExport values and sets the OID as the label.

Parameters

node	[in] A node.
node_export	[out] The NodeExport that will store the information.

Returns

TRUE.

5.19.2.6 def sparksee.DefaultExport.get_node_type (*self, type, node_export*)

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

Parameters

type	[in] A node type.
node_export	[out] The NodeExport that will store the information.

Returns

TRUE.

5.19.2.7 def sparksee.DefaultExport.prepare (*self, graph*)

Default implementation of the ExportManager class method Prepare.

Parameters

graph	null

5.20 sparksee.DisjointCommunities Class Reference

DisjointCommunities class.

Public Member Functions

- def get_nodes (self, id_community)
  Returns the collection of nodes contained in the given community.
- def get_community (self, id_node)
  Returns the disjoint community where the given node belongs to.
- def get_count (self)
  Returns the number of communities found in the graph.
- def __init__ (self, session, materializedattribute)
  Creates a new instance of DisjointCommunities.
- def get_size (self, id_community)
  Returns the number of nodes contained in the given community.
- def close (self)
  Closes the DisjointCommunities instance.
- def is_closed (self)
  Gets if DisjointCommunities has been closed or not.

5.20.1 Detailed Description

DisjointCommunities class.

This class contains the results processed on a DisjointCommunityDetection algorithm.

These results contain information related to the communities found. We must consider that each community has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different communities found.

When executing any implementation of the DisjointCommunityDetection, it is possible to indicate whether the results of the execution must be stored persistently using the class DisjointCommunityDetection setMaterializedAttribute method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.20.2 Constructor & Destructor Documentation

5.20.2.1 def sparksee.DisjointCommunities.__init__ (*self, session, materializedattribute*)

Creates a new instance of DisjointCommunities.

This constructor method can only be called when a previous execution of any implementation of the DisjointCommunityDetection class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any DisjointCommunityDetection execution see the documentation of the DisjointCommunityDetection::SetMaterializedAttribute method.

Parameters

session	[in] Session to get the graph Graph on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
materializedattribute	[in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the communities found in the graph.

5.20.3 Member Function Documentation

5.20.3.1 def sparksee.DisjointCommunities.close (*self*)

Closes the DisjointCommunities instance.

It must be called to ensure the integrity of all data.

5.20.3.2 def sparksee.DisjointCommunities.get_community (*self, id_node*)

Returns the disjoint community where the given node belongs to.

Parameters

id_node	[in] The node identifier for which the disjoint community identifier where it belongs will be returned.

Returns

The disjoint community identifier where the given node identifier belongs to.

5.20.3.3 def sparksee.DisjointCommunities.get_count (*self*)

Returns the number of communities found in the graph.

Returns

The number of communities found in the graph.

5.20.3.4 def sparksee.DisjointCommunities.get_nodes (*self, id_community*)

Returns the collection of nodes contained in the given community.

Parameters

id_community	The community for which the collection of nodes contained in it will be returned.

Returns

The collection of node identifiers contained in the given community.

5.20.3.5 def sparksee.DisjointCommunities.get_size (*self, id_community*)

Returns the number of nodes contained in the given community.

Parameters

id_community	The community for which the number of nodes contained in it will be returned.

Returns

The number of nodes contained in the given community.

5.20.3.6 def sparksee.DisjointCommunities.is_closed (*self*)

Gets if DisjointCommunities has been closed or not.

See also

close()

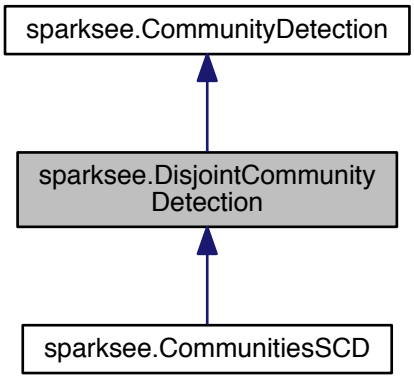
Returns

TRUE if the DisjointCommunities instance has been closed, FALSE otherwise.

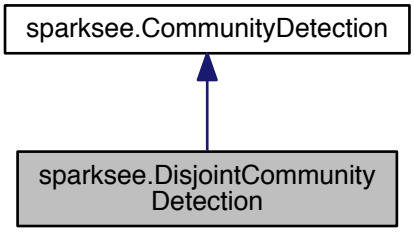
5.21 sparksee.DisjointCommunityDetection Class Reference

DisjointCommunityDetection class.

Inheritance diagram for sparksee.DisjointCommunityDetection:



Collaboration diagram for sparksee.DisjointCommunityDetection:



Public Member Functions

- def run (self)

  Runs the algorithm in order to find the communities.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self)

  Allows connectivity through all edge types of the graph.
- def get_communities (self)

  Returns the results generated by the execution of the algorithm.
- def add_edge_type (self, type)

  Allows connectivity through edges of the given type.
- def add_node_type (self, type)

  Allows connectivity through nodes of the given type.

- def exclude_edges (self, edges)

  Set which edges can't be used.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.
- def include_nodes (self, nodes)

  Set additional nodes that can be used.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def include_edges (self, edges)

  Set additional edges that can be used.
- def close (self)

  Closes the CommunityDetection instance.
- def is_closed (self)

  Gets if CommunityDetection has been closed or not.

5.21.1 Detailed Description

DisjointCommunityDetection class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.21.2 Member Function Documentation

5.21.2.1 def sparksee.DisjointCommunityDetection.add_all_edge_types (*self*)

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

5.21.2.2 def sparksee.DisjointCommunityDetection.add_edge_type (*self, type*)

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters

type	[in] Edge type.

5.21.2.3 def sparksee.DisjointCommunityDetection.add_node_type (*self, type*)

Allows connectivity through nodes of the given type.

Parameters

type	null

5.21.2.4 def sparksee.CommunityDetection.close (*self*) [inherited]

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

5.21.2.5 def sparksee.DisjointCommunityDetection.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.21.2.6 def sparksee.DisjointCommunityDetection.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.21.2.7 def sparksee.DisjointCommunityDetection.get_communities (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class DisjointCommunities which contain information related to the disjoint communities found.

5.21.2.8 def sparksee.DisjointCommunityDetection.include_edges (*self, edges*)

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.21.2.9 def sparksee.DisjointCommunityDetection.include_nodes (*self, nodes*)

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.21.2.10 def sparksee.CommunityDetection.is_closed (*self*) [inherited]

Gets if CommunityDetection has been closed or not.

See also

close()

Returns

TRUE if the CommunityDetection instance has been closed, FALSE otherwise.

5.21.2.11 def sparksee.DisjointCommunityDetection.run (*self*)

Runs the algorithm in order to find the communities.

This method can be called only once.

5.21.2.12 def sparksee.DisjointCommunityDetection.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the disjoint communities found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.22 sparksee.EdgeData Class Reference

Edge data class.

Public Member Functions

- def get_head (self)
  Gets the head of the edge.
- def get_edge (self)
  Gets the edge identifier.
- def get_tail (self)
  Gets the tail of the edge.

5.22.1 Detailed Description

Edge data class.

It stores the tail and the head of an edge instance.

In case of undirected eges, the tail and the head are just the two ends of the edge.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.22.2 Member Function Documentation

5.22.2.1 def sparksee.EdgeData.get_edge (*self*)

Gets the edge identifier.

Returns

The Sparksee edge identifier.

5.22.2.2 def sparksee.EdgeData.get_head (*self*)

Gets the head of the edge.

Returns

The Sparksee edge identifier of the head of the edge.

5.22.2.3 def sparksee.EdgeData.get_tail (*self*)

Gets the tail of the edge.

Returns

The Sparksee edge identifier of the tail of the edge.

5.23 sparksee.EdgeExport Class Reference

Stores edge exporting values.

Public Member Functions

- def set_as_directed (self, directed)

  Sets if the edge should be managed as directed.
- def get_labelcolor_rgb (self)

  Gets the edge label color.
- def set_label (self, label)

  Sets the edge label.
- def set_width (self, width)

  Sets the edge width.
- def set_color_rgb (self, color)

  Sets the edge color.
- def set_font_size (self, size)

  Sets the edge label font size.
- def set_labelcolor_rgb (self, color)

  Sets the edge label color.
- def get_color_rgb (self)

  Gets the edge color.
- def get_font_size (self)

  Gets the edge label font size.
- def set_defaults (self)

  Sets to default values.
- def __init__ (self)

  Creates a new instance.
- def get_label (self)

  Gets the edge label.
- def get_width (self)

  Gets the edge width.
- def as_directed (self)

  Gets if the edge should be managed as directed.

5.23.1 Detailed Description

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

As directed: TRUE.

Color: 13882323 (OxD3D3D3, Light gray).

Label color: 0 (Ox000000, Black).

Width: 5px.

Font size: 10.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.23.2 Member Function Documentation

5.23.2.1 def sparksee.EdgeExport.as_directed (*self*)

Gets if the edge should be managed as directed.

TRUE is the default value. If TRUE, use as directed, otherwise use as undirected.

Returns

The edge direction.

5.23.2.2 def sparksee.EdgeExport.get_color_rgb (*self*)

Gets the edge color.

Returns

The edge color.

5.23.2.3 def sparksee.EdgeExport.get_font_size (*self*)

Gets the edge label font size.

Returns

The edge label font size.

5.23.2.4 def sparksee.EdgeExport.get_label (*self*)

Gets the edge label.

Returns

The edge label.

5.23.2.5 def sparksee.EdgeExport.get_labelcolor_rgb (*self*)

Gets the edge label color.

Returns

The edge label color.

5.23.2.6 def sparksee.EdgeExport.get_width (*self*)

Gets the edge width.

Returns

The edge width.

5.23.2.7 def sparksee.EdgeExport.set_as_directed (*self, directed*)

Sets if the edge should be managed as directed.

Parameters

directed	[in] If TRUE, use as directed, otherwise use as undirected.

5.23.2.8 def sparksee.EdgeExport.set_color_rgb (*self, color*)

Sets the edge color.

Parameters

color	[in] The edge color.

5.23.2.9 def sparksee.EdgeExport.set_font_size (*self, size*)

Sets the edge label font size.

Parameters

size	[in] The edge label font size.

5.23.2.10 def sparksee.EdgeExport.set_label (*self, label*)

Sets the edge label.

Parameters

label	[in] The edge label.

5.23.2.11 def sparksee.EdgeExport.set_labelcolor_rgb (*self, color*)

Sets the edge label color.

Parameters

color	[in] The edge label color.

5.23.2.12 def sparksee.EdgeExport.set_width (*self, width*)

Sets the edge width.

Parameters

width	[in] The edge width.

5.24 sparksee.EdgesDirection Class Reference

Edges direction enumeration.

Static Public Attributes

- int INGOING = 0

  In-going edges.
- int OUTGOING = 1

  Out-going edges.
- int ANY = 2

  In-going or out-going edges.

5.24.1 Detailed Description

Edges direction enumeration.

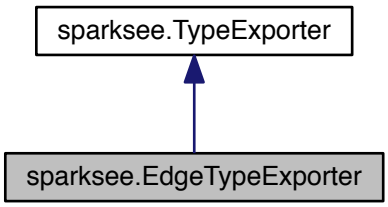
Author

Sparsity Technologies http://www.sparsity-technologies.com

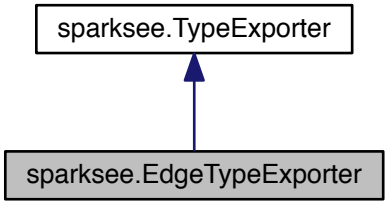
5.25 sparksee.EdgeTypeExporter Class Reference

EdgeTypeExporter class.

Inheritance diagram for sparksee.EdgeTypeExporter:



Collaboration diagram for sparksee.EdgeTypeExporter:



Public Member Functions

- def run (self)

  See the TypeExporter class Run method.
- def set_type (self, type)

  Sets the type to be exported.
- def set_head_position (self, pos)

  Sets the position (index column) of the head attribute in the exported data.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.
- def set_head_attribute (self, attr)

  Sets the attribute that will be used to get the value to be dumped for the head of the edge.
- def set_tail_attribute (self, attr)

  Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

- def set_tail_position (self, pos)

  Sets the position (index column) of the tail attribute in the exported data.
- def __init__ (self)

  Creates a new instance.
- def set_header (self, header)

  Sets the presence of a header row.
- def set_row_writer (self, rw)

  Sets the output data destination.
- def set_graph (self, graph)

  Sets the graph that will be exported.
- def __init__ (self, row_writer, graph, type, attrs, h_pos, t_pos, h_attr, t_attr)

  Creates a new instance.
- def register (self, tel)

  Registers a new listener.

5.25.1 Detailed Description

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.25.2 Constructor & Destructor Documentation

5.25.2.1 def sparksee.EdgeTypeExporter.__init__ (*self, row_writer, graph, type, attrs, h_pos, t_pos, h_attr, t_attr*)

Creates a new instance.

Parameters

row_writer	[in] Output RowWriter.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be exported.
h_pos	[in] The position (index column) for the head value.
t_pos	[in] The position (index column) for the tail value.
h_attr	[in] The attribute identifier to get the value to be dumped for the head.
t_attr	[in] The attribute identifier to get the value to be dumped for the tail.

References sparksee.EdgeTypeExporter.__init__().

5.25.3 Member Function Documentation

5.25.3.1 def sparksee.EdgeTypeExporter.register (*self, tel*)

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

5.25.3.2 def sparksee.EdgeTypeExporter.run (*self*)

See the TypeExporter class Run method.

Exceptions

RuntimeError	null
IOError	null

5.25.3.3 def sparksee.EdgeTypeExporter.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

5.25.3.4 def sparksee.EdgeTypeExporter.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.25.3.5 def sparksee.EdgeTypeExporter.set_graph (*self, graph*)

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

5.25.3.6 def sparksee.EdgeTypeExporter.set_head_attribute (*self, attr*)

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

Parameters

attr	[in] Head Attribute

5.25.3.7 def sparksee.EdgeTypeExporter.set_head_position (*self, pos*)

Sets the position (index column) of the head attribute in the exported data.

Parameters

pos	[in] Head position

5.25.3.8 def sparksee.EdgeTypeExporter.set_header (*self, header*)

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

5.25.3.9 def sparksee.EdgeTypeExporter.set_row_writer (*self, rw*)

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

5.25.3.10 def sparksee.EdgeTypeExporter.set_tail_attribute (*self, attr*)

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

Parameters

attr	[in] Tail Attribute

5.25.3.11 def sparksee.EdgeTypeExporter.set_tail_position (*self, pos*)

Sets the position (index column) of the tail attribute in the exported data.

Parameters

pos	[in] Tail position

5.25.3.12 def sparksee.EdgeTypeExporter.set_type (*self, type*)

Sets the type to be exported.

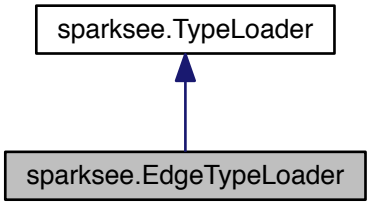
Parameters

type	[in] Type identifier.

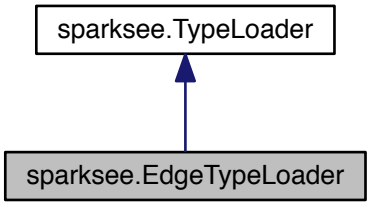
5.26 sparksee.EdgeTypeLoader Class Reference

EdgeTypeLoader class.

Inheritance diagram for sparksee.EdgeTypeLoader:



Collaboration diagram for sparksee.EdgeTypeLoader:



Public Member Functions

- def run (self)

  See the TypeLoader class Run method.
- def set_log_error (self, path)

  Sets a log error file.

- def set_type (self, type)

  Sets the type to be loaded.
- def set_head_position (self, pos)

  Sets the position of the head attribute in the source data.
- def set_locale (self, locale_str)

  Sets the locale that will be used to read the data.
- def set_timestamp_format (self, timestamp_format)

  Sets a specific timestamp format.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def __init__ (self)

  Creates a new instance.
- def register (self, tel)

  Registers a new listener.
- def __init__ (self, row_reader, graph, type, attrs, attrs_pos, h_pos, t_pos, h_attr, t_attr, head_mep, tail_mep)

  Creates a new instance.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.
- def run_two_phases (self)

  See the TypeLoader class RunTwoPhases method.
- def set_head_attribute (self, attr)

  Sets the attribute that will be used to find the head of the edge.
- def set_tail_attribute (self, attr)

  Sets the attribute that will be used to find the tail of the edge.
- def set_tail_position (self, pos)

  Sets the position of the tail attribute in the source data.
- def set_row_reader (self, rr)

  Sets the input data source.
- def set_tail_mep (self, mep)

  Sets the flag to create missing tail nodes when loading nodes or edges.
- def set_attribute_positions (self, attrs_pos)

  Sets the list of attribute positions.
- def set_graph (self, graph)

  Sets the graph where the data will be loaded.
- def run_n_phases (self, partitions)

  See the TypeLoader class RunNPhases method.
- def set_head_mep (self, mep)

  Sets the flag to create missing head nodes when loading nodes or edges.
- def set_log_off (self)

  Truns off all the error reporting.

5.26.1 Detailed Description

EdgeTypeLoader class.

Specific TypeLoader implementation for edge types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.26.2 Constructor & Destructor Documentation

5.26.2.1 def sparksee.EdgeTypeLoader.__init__ (*self, row_reader, graph, type, attrs, attrs_pos, h_pos, t_pos, h_attr, t_attr, head_mep, tail_mep*)

Creates a new instance.

Parameters

row_reader	[in] Input RowReader.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be loaded.
attrs_pos	[in] Attribute positions (column index >=0). to all listeners.
h_pos	[in] The position (index column) for the head value.
t_pos	[in] The position (index column) for the tail value.
h_attr	[in] The attribute identifier for the head.
t_attr	[in] The attribute identifier for the tail.
head_mep	[in] The MissingEndpoint policy for the head nodes
tail_mep	[in] The MissingEndpoint policy for the tail nodes

References sparksee.EdgeTypeLoader.__init__().

5.26.3 Member Function Documentation

5.26.3.1 def sparksee.EdgeTypeLoader.register (*self, tel*)

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

5.26.3.2 def sparksee.EdgeTypeLoader.run (*self*)

See the TypeLoader class Run method.

Exceptions

RuntimeError	null
IOError	null

5.26.3.3 def sparksee.EdgeTypeLoader.run_n_phases (*self, partitions*)

See the TypeLoader class RunNPhases method.

Parameters

partitions	null

Exceptions

RuntimeError	null
IOError	null

5.26.3.4 def sparksee.EdgeTypeLoader.run_two_phases (*self*)

See the TypeLoader class RunTwoPhases method.

Exceptions

RuntimeError	null
IOError	null

5.26.3.5 def sparksee.EdgeTypeLoader.set_attribute_positions (*self, attrs_pos*)

Sets the list of attribute positions.

Parameters

attrs_pos	[in] Attribute positions (column index >=0).

5.26.3.6 def sparksee.EdgeTypeLoader.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

5.26.3.7 def sparksee.EdgeTypeLoader.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.26.3.8 def sparksee.EdgeTypeLoader.set_graph (*self, graph*)

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

5.26.3.9 def sparksee.EdgeTypeLoader.set_head_attribute (*self, attr*)

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters

attr	[in] Head Attribute

5.26.3.10 def sparksee.EdgeTypeLoader.set_head_mep (*self, mep*)

Sets the flag to create missing head nodes when loading nodes or edges.

flagTrue if the nodes need to be created. False otherwise

Parameters

mep	null

5.26.3.11 def sparksee.EdgeTypeLoader.set_head_position (*self, pos*)

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters

pos	[in] Head position

5.26.3.12 def sparksee.EdgeTypeLoader.set_locale (*self, locale_str*)

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

locale_str	[in] The locale string for the read data. See CSVReader.

5.26.3.13 def sparksee.EdgeTypeLoader.set_log_error (*self, path*)

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

IOError	If bad things happen opening the file.

5.26.3.14 def sparksee.EdgeTypeLoader.set_log_off (*self*)

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.26.3.15 def sparksee.EdgeTypeLoader.set_row_reader (*self, rr*)

Sets the input data source.

Parameters

rr	[in] Input RowReader.

5.26.3.16 def sparksee.EdgeTypeLoader.set_tail_attribute (*self, attr*)

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

Parameters

attr	[in] Tail Attribute

5.26.3.17 def sparksee.EdgeTypeLoader.set_tail_mep (*self, mep*)

Sets the flag to create missing tail nodes when loading nodes or edges.

flagTrue if the nodes need to be created. False otherwise

Parameters

mep	null

5.26.3.18 def sparksee.EdgeTypeLoader.set_tail_position (*self, pos*)

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters

pos	[in] Tail position

5.26.3.19 def sparksee.EdgeTypeLoader.set_timestamp_format (*self, timestamp_format*)

Sets a specific timestamp format.

Parameters

timestamp_format	[in] A string with the timestamp format definition.

5.26.3.20 def sparksee.EdgeTypeLoader.set_type (*self, type*)

Sets the type to be loaded.

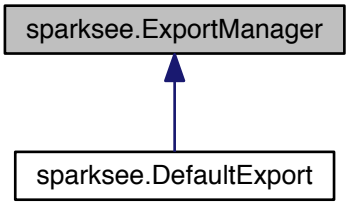
Parameters

type	[in] Type identifier.

5.27 sparksee.ExportManager Class Reference

Defines how to export a graph to an external format.

Inheritance diagram for sparksee.ExportManager:



Public Member Functions

- def get_edge (self, edge, edge_export)

  Gets the edge export definition for the given edge.
- def release (self)

  Ends the export process.
- def get_graph (self, graph_export)

  Gets the graph export definition.
- def get_node (self, node, node_export)

  Gets the node export definition for the given node.
- def get_node_type (self, type, node_export)

  Gets the default node export definition for the given node type.
- def enable_type (self, type)

  Gets whether a node or edge type must be exported or not.
- def prepare (self, graph)

  Prepares the graph for the export process.
- def get_edge_type (self, type, edge_export)

  Gets the default node export definition for the given edge type.

5.27.1 Detailed Description

Defines how to export a graph to an external format.

This is an interface which must be implemented by the user. While the export proces, a call for each node or edge type and node or edge object is done to get how to export that element.

It is possible to export a Graph to a diferent fortmats. Nowadays, available formats are defined in the ExportType enum.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.27.2 Member Function Documentation

5.27.2.1 def sparksee.ExportManager.enable_type (*self, type*)

Gets whether a node or edge type must be exported or not.

Parameters

type	Node or edge type identifier.

Returns

If TRUE all objects of the given type will be exported, otherwise they will not be exported.

5.27.2.2 def sparksee.ExportManager.get_edge (*self, edge, edge_export*)

Gets the edge export definition for the given edge.

Parameters

edge	Edge identifier.
edge_export	[out] The EdgeExport which defines how to export given edge.

Returns

TRUE if the given EdgeExport has been updated, otherwise FALSE will be returned and the default EdgeExport for the type the edge belongs to will be used.

5.27.2.3 def sparksee.ExportManager.get_edge_type (*self, type, edge_export*)

Gets the default node export definition for the given edge type.

GetEdge has a higher priority than this function. That is, only if GetEdge returns FALSE, the EdgeExport of this function will be used.

Parameters

type	[in] Edge type identifier.
edge_export	[out] The EdgeExport which defines how to export the edges of the given type.

Returns

TRUE.

5.27.2.4 def sparksee.ExportManager.get_graph (*self, graph_export*)

Gets the graph export definition.

Parameters

graph_export	[out] The GraphExport which defines how to export the graph.

Returns

TRUE.

5.27.2.5 def sparksee.ExportManager.get_node (*self, node, node_export*)

Gets the node export definition for the given node.

Parameters

node	Node identifier.
node_export	[out] The NodeExport which defines how to export given node.

Returns

TRUE if the given NodeExport has been updated, otherwise FALSE will be returned and the default NodeExport for the type the node belongs to will be used.

5.27.2.6 def sparksee.ExportManager.get_node_type (*self, type, node_export*)

Gets the default node export definition for the given node type.

GetNode has a higher priority than this function. That is, only if GetNode returns FALSE, the NodeExport of this function will be used.

Parameters

type	[in] Node type identifier.
node_export	[out] The NodeExport which defines how to export the nodes of the given type.

Returns

TRUE.

5.27.2.7 def sparksee.ExportManager.prepare (*self, graph*)

Prepares the graph for the export process.

It is called once before the export process.

Parameters

graph	Graph to be exported.

5.27.2.8 def sparksee.ExportManager.release (*self*)

Ends the export process.

It is called once after the export process.

5.28 sparksee.ExportType Class Reference

Export type.

Static Public Attributes

- int GRAPHVIZ = 0

  Export to Graphviz format.
- int GRAPHML = 1

  Export to GraphML format.
- int YGRAPHML = 2

  Export to YGRAPHML format.

5.28.1 Detailed Description

Export type.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.28.2 Member Data Documentation

5.28.2.1 int sparksee.ExportType.GRAPHML = 1 [static]

Export to GraphML format.

GraphML home page: http://graphml.graphdrawing.org/

5.28.2.2 int sparksee.ExportType.GRAPHVIZ = 0 [static]

Export to Graphviz format.

Graphviz home page: http://www.graphviz.org

5.28.2.3 int sparksee.ExportType.YGRAPHML = 2 [static]

Export to YGRAPHML format.

It is an GraphML format extended with a set of yWorks ("http://www.yworks.com") extensions. Thus, it allows for the visualization of the exported graph with the public yEd visualization tool ("http://www.yworks.com/products/yed").

5.29 sparksee.Graph Class Reference

Graph class.

Public Member Functions

- def find_or_create_edge (self, etype, tail, head)
  Gets any of the edges of the specified type between two given nodes (tail and head).
- def find_edge_types (self)
  Gets all existing Sparksee edge type identifiers.
- def get_attribute_interval_count (self, attr, lower, include_lower, higher, include_higher)
  Gets how many objects have a value into the given range for the given attribute.
- def neighbors (self, objs, etype, dir)
  Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.
- def backup (self, file)
  Dumps all the data to a backup file.
- def remove_attribute (self, attr)
  Removes the given attribute.
- def remove_type (self, type)
  Removes the given type.

- def degree (self, oid, etype, dir)

  Gets the number of edges from or to the given node OID and for the given edge type.
- def find_types (self)

  Gets all existing Sparksee node and edge type identifiers.
- def new_array_attribute (self, type, name, dt, size)

  Creates a new array attribute.
- def find_type (self, name)

  Gets the Sparksee type identifier for the given type name.
- def top_k (self, attribute, operation, lower, order, k)

  Gets a KeyValues iterator as a result of the TopK operation.
- def get_edge_peer (self, edge, node)

  Gets the other end for the given edge.
- def set_attribute_text (self, oid, attr, tstream)

  Sets the writable TextStream for the given text attribute and OID.
- def set_attribute (self, attr, value)

  Sets the value of the given attribute for all the objects of the types the attribute applies to.
- def index_neighbors (self, edge_type, neighbors)

  Creates or destroys the neighbors index of an edge type.
- def get_object_type (self, oid)

  Gets the Sparksee node or edge type identifier for the given OID.
- def index_attribute (self, attr, kind)

  Updates the index of the given attribute.
- def set_array_attribute (self, attr, value)

  Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.
- def top_k (self, attribute, order, k)

  Gets a KeyValues iterator as a result of the TopK operation.
- def get_attribute (self, oid, attr, value)

  Gets the Value for the given attribute and OID.
- def get_attribute_statistics (self, attr, basic)

  Gets statistics from the given attribute.
- def select (self, attr, cond, lower, higher, restriction)

  Selects all OIDs satisfying the given condition for the given attribute.
- def count_nodes (self)

  Gets the number of nodes.
- def set_attribute_default_value (self, attr, value)

  Sets a default value for an attribute.
- def find_object (self, attr, value)

  Finds one object having the given Value for the given attribute.
- def neighbors (self, oid, etype, dir)

  Selects all neighbor nodes from or to the given node OID and for the given edge type.
- def new_session_attribute (self, type, dt, kind)

  Creates a new Session attribute.
- def encrypted_backup (self, file, key_in_hex, iv_in_hex)

  Dumps all the data to a backup file.
- def drop (self, oid)

  Drops the given OID.
- def set_array_attribute_void (self, oid, attr, value)

  Sets all the elements of the ValueArray for the given array attribute and OID.
- def edges (self, etype, tail, head)

  Gets all the edges of the given type between two given nodes (tail and head).
- def set_attribute (self, oid, attr, value)

Sets the Value for the given attribute and OID.

- def find_edge (self, etype, tail, head)

  Gets any of the edges of the given type between two given nodes (tail and head).
- def tails (self, edges)

  Gets all the tails from the given edges collection.
- def select (self, attr, cond, lower, higher)

  Selects all OIDs satisfying the given condition for the given attribute.
- def get_values (self, attr)

  Gets the Value collection for the given attribute.
- def export (self, file, type, em)

  Exports the Graph.
- def new_restricted_edge_type (self, name, tail, head, neighbors)

  Creates a new restricted edge type.
- def get_attribute_text (self, oid, attr)

  Gets the read-only TextStream for the given text attribute and OID.
- def explode (self, objs, etype, dir)

  Selects all edges from or to each of the node OID in the given collection and for the given edge type.
- def new_edge_type (self, name, directed, neighbors)

  Creates a new edge type.
- def top_k (self, attribute, operation, lower, higher, order, k)

  Gets a KeyValues iterator as a result of the TopK operation.
- def dump_data (self, file)

  Dumps logical data to a file.
- def set_array_attribute (self, oid, attr, value)

  Sets all the elements of the ValueArray for the given array attribute and OID and returns it.
- def top_k (self, attribute, operation, lower, higher, order, k, restriction)

  Gets a KeyValues iterator as a result of the TopK operation.
- def find_attribute (self, type, name)

  Gets the Sparksee attribute identifier for the given type identifier and attribute name.
- def new_node (self, type)

  Creates a new node instance.
- def find_node_types (self)

  Gets all existing Sparksee node type identifiers.
- def new_attribute (self, type, name, dt, kind)

  Creates a new attribute.
- def top_k (self, attribute, operation, lower, order, k, restriction)

  Gets a KeyValues iterator as a result of the TopK operation.
- def new_session_array_attribute (self, type, dt, size)

  Creates a new Session array attribute.
- def new_edge (self, type, tail_attr, tail_v, head_attr, head_v)

  Creates a new edge instance.
- def get_attribute (self, attr)

  Gets information about the given attribute.
- def new_node_type (self, name)

  Creates a new node type.
- def dump_storage (self, file)

  Dumps internal storage data to a file.
- def select (self, type)

  Selects all OIDs belonging to the given type.
- def select (self, attr, cond, value, restriction)

  Selects all OIDs satisfying the given condition for the given attribute.

- def select (self, attr, cond, value)

  Selects all OIDs satisfying the given condition for the given attribute.
- def rename_attribute (self, attr, new_name)

  Renames an attribute.
- def top_k (self, attribute, order, k, restriction)

  Gets a KeyValues iterator as a result of the TopK operation.
- def count_edges (self)

  Gets the number of edges.
- def rename_type (self, type, new_name)

  Renames a type.
- def tails_and_heads (self, edges, tails, heads)

  Gets all the tails and heads from the given edges collection.
- def new_session_attribute (self, type, dt, kind, default_value)

  Creates a new Session attribute with a default value.
- def heads (self, edges)

  Gets all the heads from the given edges collection.
- def new_attribute (self, type, name, dt, kind, default_value)

  Creates a new attribute with a default value.
- def rename_type (self, old_name, new_name)

  Renames a type.
- def explode (self, oid, etype, dir)

  Selects all edges from or to the given node OID and for the given edge type.
- def get_attributes (self, oid)

  Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.
- def find_attributes (self, type)

  Gets all existing Sparksee attribute identifiers for the given type identifier.
- def get_array_attribute (self, oid, attr)

  Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.
- def get_attribute (self, oid, attr)

  Gets the Value for the given attribute and OID.
- def find_or_create_object (self, attr, value)

  Finds one object having the given Value for the attribute or it creates one does not exist any.
- def get_edge_data (self, edge)

  Gets information about an edge.
- def drop (self, objs)

  Drops all the OIDs from the given collection.
- def new_edge (self, type, tail, head)

  Creates a new edge instance.
- def get_type (self, type)

  Gets information about the given type.

5.29.1 Detailed Description

Graph class.

Each Database has a Graph associated, which is the persistent graph which contains all data stored into the graph database and is retrieved from a Session.

Check out the 'API' and the 'SPARKSEE graph database' sections in the SPARKSEE User Manual for more details on the use of this class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.29.2 Member Function Documentation

5.29.2.1 def sparksee.Graph.backup (*self, file*)

Dumps all the data to a backup file.

See the Sparksee class Restore methods.

Parameters

file	[in] Output backup file path.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created.

5.29.2.2 def sparksee.Graph.count_edges (*self*)

Gets the number of edges.

Returns

The number of edges.

5.29.2.3 def sparksee.Graph.count_nodes (*self*)

Gets the number of nodes.

Returns

The number of nodes.

5.29.2.4 def sparksee.Graph.degree (*self, oid, etype, dir*)

Gets the number of edges from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

The number of edges.

5.29.2.5 def sparksee.Graph.drop (*self, oid*)

Drops the given OID.

It also removes its egdges as well as its attribute values.

Parameters

oid	[in] Sparksee OID to be removed.

Referenced by sparksee.Graph.drop().

5.29.2.6 def sparksee.Graph.drop (*self, objs*)

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs the same operation for each object in the given set.

Parameters

objs	[in] Objects collection with the OIDs to be removed.

References sparksee.Graph.drop().

5.29.2.7 def sparksee.Graph.dump_data (*self, file*)

Dumps logical data to a file.

Parameters

file	[in] Output file path.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created.

5.29.2.8 def sparksee.Graph.dump_storage (*self, file*)

Dumps internal storage data to a file.

Parameters

file	[in] Output file path.

Exceptions

RuntimeError	null

Exceptions

IOError	If the given file cannot be created.

5.29.2.9 def sparksee.Graph.edges (*self, etype, tail, head*)

Gets all the edges of the given type between two given nodes (tail and head).

Parameters

etype	[in] Sparksee edge type identifier.
tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Objects instance.

5.29.2.10 def sparksee.Graph.encrypted_backup (*self, file, key_in_hex, iv_in_hex*)

Dumps all the data to a backup file.

See the Sparksee class RestoreEncryptedBackup methods.

Parameters

file	[in] Output backup file path.
key_in_hex	[In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes).
iv_in_hex	[In] The AES Initialization Vector as an hexadecimal string (16 bytes).

Exceptions

RuntimeError	null
IOError	If the given file cannot be created.

5.29.2.11 def sparksee.Graph.explode (*self, objs, etype, dir*)

Selects all edges from or to each of the node OID in the given collection and for the given edge type.

Parameters

objs	[in] Sparksee node OID collection.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

Referenced by sparksee.Graph.explode().

5.29.2.12 def sparksee.Graph.explode (*self, oid, etype, dir*)

Selects all edges from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

References sparksee.Graph.explode().

5.29.2.13 def sparksee.Graph.export (*self, file, type, em*)

Exports the Graph.

Parameters

file	[in] Output file.
type	[in] Export type.
em	[in] Defines how to do the export for each graph object.

Exceptions

IOError	null

5.29.2.14 def sparksee.Graph.find_attribute (*self, type, name*)

Gets the Sparksee attribute identifier for the given type identifier and attribute name.

Parameters

type	[in] Sparksee type identifier.
name	[in] Unique attribute name.

Returns

The Sparksee attribute identifier for the given type and attribute name or InvalidAttribute if there is no attribute with the given name for the given type.

5.29.2.15 def sparksee.Graph.find_attributes (*self, type*)

Gets all existing Sparksee attribute identifiers for the given type identifier.

Parameters

type	[in] Sparksee type identifier.

Returns

Sparksee attribute identifier list.

5.29.2.16 def sparksee.Graph.find_edge (*self, etype, tail, head*)

Gets any of the edges of the given type between two given nodes (tail and head).

If there are more than one, then any of them will be returned. And in case there are no edge between the given tail and head, the Objects InvalidOID will be returned.

Parameters

etype	[in] Sparksee edge type identifier.
tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Any of the edges or the Objects InvalidOID.

5.29.2.17 def sparksee.Graph.find_edge_types (*self*)

Gets all existing Sparksee edge type identifiers.

Returns

Sparksee edge type identifier list.

5.29.2.18 def sparksee.Graph.find_node_types (*self*)

Gets all existing Sparksee node type identifiers.

Returns

Sparksee node type identifier list.

5.29.2.19 def sparksee.Graph.find_object (*self, attr, value*)

Finds one object having the given Value for the given attribute.

If there are more than one, then any of them will be returned. And in case there are no object having this Value, the Objects InvalidOID will be returned.

Parameters

attr	[in] Sparksee attribute identifier.
value	[in] Value.

Returns

Sparksee OID or the Objects InvalidOID.

5.29.2.20 def sparksee.Graph.find_or_create_edge (*self, etype, tail, head*)

Gets any of the edges of the specified type between two given nodes (tail and head).

If it can not find any edge of this type between them it tries to create a new one.

Parameters

etype	[in] Sparksee edge type identifier.
tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Any of the edges or the Objects InvalidOID.

5.29.2.21 def sparksee.Graph.find_or_create_object (*self, attr, value*)

Finds one object having the given Value for the attribute or it creates one does not exist any.

If the attribute is a node or edge attribute and at least one node/edge with that value is found, it returns one of them. But if it does not exist, then: If it's a node attribute it will create it and set the attribute. If it's an edge attribute it will return the InvalidOID.

Using this method with a global attribute will return the InvalidOID.

Parameters

attr	[in] Sparksee attribute identifier.
value	[in] Value.

Returns

Sparksee OID or the Objects InvalidOID.

5.29.2.22 def sparksee.Graph.find_type (*self, name*)

Gets the Sparksee type identifier for the given type name.

Parameters

name	[in] Unique type name.

Returns

The Sparksee type identifier for the given type name or the Type InvalidType if there is no type with the given name.

5.29.2.23 def sparksee.Graph.find_types (*self*)

Gets all existing Sparksee node and edge type identifiers.

Returns

Sparksee node and edge type identifier list.

5.29.2.24 def sparksee.Graph.get_array_attribute (*self, oid, attr*)

Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.

Returns

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

5.29.2.25 def sparksee.Graph.get_attribute (*self, oid, attr, value*)

Gets the Value for the given attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
value	[in|out] Value for the given attribute and for the given OID.

Referenced by sparksee.Graph.get_attribute().

5.29.2.26 def sparksee.Graph.get_attribute (*self, attr*)

Gets information about the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

Returns

The Attribute for the given Sparksee attribute identifier.

References sparksee.Graph.get_attribute().

5.29.2.27 def sparksee.Graph.get_attribute (*self, oid, attr*)

Gets the Value for the given attribute and OID.

The other version of this call, where the Value is an output parameter instead of the return, is better because it allows the user to reuse an existing Value instance, whereas this call always creates a new Value instance to be returned.

It never returns NULL. Thus, in case the OID has a NULL value for the attribute it returns a NULL Value instance.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.

Returns

A new Value instance having the attribute value for the given OID.

References sparksee.Graph.get_attribute().

5.29.2.28 def sparksee.Graph.get_attribute_interval_count (*self, attr, lower, include_lower, higher, include_higher*)

Gets how many objects have a value into the given range for the given attribute.

This only works for the attributes with the AttributeKind Indexed or Unique.

Given values must belong to the same DataType than the attribute.

Parameters

attr	[in] Sparksee attribute identifier.
lower	[in] Lower bound Value of the range.
include_lower	[in] If TRUE, include lower bound Value of the range.
higher	[in] Higher bound Value of the range.
include_higher	[in] If TRUE, include higher bound Value of the range.

Returns

Number of objects having a value into the given range.

5.29.2.29 def sparksee.Graph.get_attribute_statistics (*self, attr, basic*)

Gets statistics from the given attribute.

The statistics can only be obtained from Indexed or Unique attributes.

Parameters

attr	[in] Sparksee attribute identifier.
basic	[in] If FALSE all statistics are computed, if TRUE just those statistics marked as basic will be computed (see description of the AttributeStatistics class). Of course, computing just basic statistics will be faster than computing all of them.

Returns

An AttributeStatistics instace.

5.29.2.30 def sparksee.Graph.get_attribute_text (*self, oid, attr*)

Gets the read-only TextStream for the given text attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.

Returns

A TextStream. This returns a TextStream to read.

5.29.2.31 def sparksee.Graph.get_attributes (*self, oid*)

Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.

Parameters

oid	[in] Sparksee OID.

Returns

Sparksee attribute identifier list.

5.29.2.32 def sparksee.Graph.get_edge_data (*self, edge*)

Gets information about an edge.

Parameters

edge	[in] Sparksee edge identifier.

Returns

An EdgeData instance.

5.29.2.33 def sparksee.Graph.get_edge_peer (*self, edge, node*)

Gets the other end for the given edge.

Parameters

edge	[in] Sparksee edge identifier.
node	[in] Sparksee node identifier. It must be one of the ends of the edge.

Returns

The other end of the edge.

5.29.2.34 def sparksee.Graph.get_object_type (*self, oid*)

Gets the Sparksee node or edge type identifier for the given OID.

Parameters

oid	[in] Sparksee OID.

Returns

Sparksee node or edge type identifier.

5.29.2.35 def sparksee.Graph.get_type (*self, type*)

Gets information about the given type.

Parameters

type	[in] Sparksee type identifier.

Returns

The Type for the given Sparksee type identifier.

5.29.2.36 def sparksee.Graph.get_values (*self, attr*)

Gets the Value collection for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

Returns

Returns a Values object.

5.29.2.37 def sparksee.Graph.heads (*self, edges*)

Gets all the heads from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.

Returns

The heads collection.

5.29.2.38 def sparksee.Graph.index_attribute (*self, attr, kind*)

Updates the index of the given attribute.

This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one is Indexed or Unique.

Parameters

attr	[in] Sparksee attribute identifier.
kind	[in] Attribute kind.

5.29.2.39 def sparksee.Graph.index_neighbors (*self, edge_type, neighbors*)

Creates or destroys the neighbors index of an edge type.

Parameters

edge_type	[in] Sparksee Edge type identifier.
neighbors	[in] If TRUE, it indexes the neighbor nodes, otherwise it removes the index.

5.29.2.40 def sparksee.Graph.neighbors (*self, objs, etype, dir*)

Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

Parameters

objs	[in] Sparksee node OID collection.

Parameters

etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

Referenced by sparksee.Graph.neighbors().

5.29.2.41 def sparksee.Graph.neighbors (*self, oid, etype, dir*)

Selects all neighbor nodes from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

References sparksee.Graph.neighbors().

5.29.2.42 def sparksee.Graph.new_array_attribute (*self, type, name, dt, size*)

Creates a new array attribute.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Base Data type for the new array attribute elements.
size	[in] The array size.

Returns

Unique Sparksee attribute identifier.

5.29.2.43 def sparksee.Graph.new_attribute (*self, type, name, dt, kind*)

Creates a new attribute.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.

Returns

Unique Sparksee attribute identifier.

Referenced by sparksee.Graph.new_attribute().

5.29.2.44 def sparksee.Graph.new_attribute (*self, type, name, dt, kind, default_value*)

Creates a new attribute with a default value.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.
default_value	[in] The default value to use in each new node/edge.

Returns

Unique Sparksee attribute identifier.

References sparksee.Graph.new_attribute().

5.29.2.45 def sparksee.Graph.new_edge (*self, type, tail_attr, tail_v, head_attr, head_v*)

Creates a new edge instance.

The tail of the edge will be any node having the given tailV Value for the given tailAttr attribute identifier, and the head of the edge will be any node having the given headV Value for the given headAttr attribute identifier.

Parameters

type	[in] Sparksee type identifier.
tail_attr	[in] Sparksee attribute identifier.
tail_v	[in] Value.
head_attr	[in] Sparksee attribute identifier.
head_v	[in] Value.

Returns

Unique OID of the new edge instance.

Referenced by sparksee.Graph.new_edge().

5.29.2.46 def sparksee.Graph.new_edge (*self, type, tail, head*)

Creates a new edge instance.

Parameters

type	[in] Sparksee type identifier.
tail	[in] Source Sparksee OID.
head	[in] Target Sparksee OID.

Returns

Unique OID of the new edge instance.

References sparksee.Graph.new_edge().

5.29.2.47 def sparksee.Graph.new_edge_type (*self, name, directed, neighbors*)

Creates a new edge type.

Parameters

name	[in] Unique name for the new edge type.
directed	[in] If TRUE, this creates a directed edge type, otherwise this creates a undirected edge type.
neighbors	[in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns

Unique Sparksee type identifier.

5.29.2.48 def sparksee.Graph.new_node (*self, type*)

Creates a new node instance.

Parameters

type	[in] Sparksee type identifier.

Returns

Unique OID of the new node instance.

5.29.2.49 def sparksee.Graph.new_node_type (*self, name*)

Creates a new node type.

Parameters

name	[in] Unique name for the new node type.

Returns

Unique Sparksee type identifier.

5.29.2.50 def sparksee.Graph.new_restricted_edge_type (*self, name, tail, head, neighbors*)

Creates a new restricted edge type.

Parameters

name	[in] Unique name for the new edge type.
tail	[in] Tail Sparksee node type identifier.
head	[in] Head Sparksee node type identifier.
neighbors	[in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns

Unique Sparksee type identifier.

5.29.2.51 def sparksee.Graph.new_session_array_attribute (*self, type, dt, size*)

Creates a new Session array attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Base Data type for the new array attribute elements.
size	[in] The array size.

Returns

Unique Sparksee attribute identifier.

5.29.2.52 def sparksee.Graph.new_session_attribute (*self, type, dt, kind*)

Creates a new Session attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.

Returns

Unique Sparksee attribute identifier.

Referenced by sparksee.Graph.new_session_attribute().

5.29.2.53 def sparksee.Graph.new_session_attribute (*self, type, dt, kind, default_value*)

Creates a new Session attribute with a default value.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.
default_value	[in] The default value to use in each new node/edge.

Returns

Unique Sparksee attribute identifier.

References sparksee.Graph.new_session_attribute().

5.29.2.54 def sparksee.Graph.remove_attribute (*self, attr*)

Removes the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

5.29.2.55 def sparksee.Graph.remove_type (*self, type*)

Removes the given type.

This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

Parameters

type	[in] Sparksee type identifier.

5.29.2.56 def sparksee.Graph.rename_attribute (*self, attr, new_name*)

Renames an attribute.

The new name must be available.

Parameters

attr	[in] Sparksee attribute identifier.
new_name	[in] The new name for the attribute.

5.29.2.57 def sparksee.Graph.rename_type (*self, type, new_name*)

Renames a type.

The new name must be available.

Parameters

type	[in] The type to be renamed.
new_name	[in] The new name for the type.

Referenced by sparksee.Graph.rename_type().

5.29.2.58 def sparksee.Graph.rename_type (*self, old_name, new_name*)

Renames a type.

The new name must be available.

Parameters

old_name	[in] The current name of the type to be renamed.
new_name	[in] The new name for the type.

References sparksee.Graph.rename_type().

5.29.2.59 def sparksee.Graph.select (*self, attr, cond, lower, higher, restriction*)

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two Value arguments.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied. It must be the Between Condition.
lower	[in] Lower-bound Value to be satisfied.
higher	[in] Higher-bound Value to be satisfied.
restriction	[in] Objects to limit the select in this set of objects.

Returns

Objects instance.

Referenced by sparksee.Graph.select().

5.29.2.60 def sparksee.Graph.select (*self, attr, cond, lower, higher*)

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two Value arguments.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied. It must be the Between Condition.
lower	[in] Lower-bound Value to be satisfied.
higher	[in] Higher-bound Value to be satisfied.

Returns

Objects instance.

References sparksee.Graph.select().

5.29.2.61 def sparksee.Graph.select (*self, type*)

Selects all OIDs belonging to the given type.

Parameters

type	[in] Sparksee type identifier.

Returns

Objects instance.

References sparksee.Graph.select().

5.29.2.62 def sparksee.Graph.select (*self, attr, cond, value, restriction*)

Selects all OIDs satisfying the given condition for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied.
value	[in] Value to be satisfied.
restriction	[in] Objects to limit the select in this set of objects.

Returns

Objects instance.

References sparksee.Graph.select().

5.29.2.63 def sparksee.Graph.select (*self, attr, cond, value*)

Selects all OIDs satisfying the given condition for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied.
value	[in] Value to be satisfied.

Returns

Objects instance.

References sparksee.Graph.select().

5.29.2.64 def sparksee.Graph.set_array_attribute (*self, attr, value*)

Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.

Parameters

attr	[in] Sparksee array attribute identifier.
value	[in] Value for all the array elements of the arrays.

Referenced by sparksee.Graph.set_array_attribute().

5.29.2.65 def sparksee.Graph.set_array_attribute (*self, oid, attr, value*)

Sets all the elements of the ValueArray for the given array attribute and OID and returns it.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray. And you can get it again with the GetArrayAttribute operation.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.
value	[in] Value for all the array elements of the given attribute and OID.

Returns

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

References sparksee.Graph.set_array_attribute().

5.29.2.66 def sparksee.Graph.set_array_attribute_void (*self, oid, attr, value*)

Sets all the elements of the ValueArray for the given array attribute and OID.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray which can be obtained using the GetArrayAttribute operation.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.
value	[in] Value for all the array elements of the given attribute and OID.

5.29.2.67 def sparksee.Graph.set_attribute (*self, attr, value*)

Sets the value of the given attribute for all the objects of the types the attribute applies to.

Does not work for TEXT attribute types

txThe Transaction this operation belongs to attributeThe attribute to initialize

Parameters

attr	null
value	The value to initialize the attribute to

Referenced by sparksee.Graph.set_attribute().

5.29.2.68 def sparksee.Graph.set_attribute (*self, oid, attr, value*)

Sets the Value for the given attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
value	[in] Value for the given attribute and for the given OID.

References sparksee.Graph.set_attribute().

5.29.2.69 def sparksee.Graph.set_attribute_default_value (*self, attr, value*)

Sets a default value for an attribute.

The default value will be applied to all the new nodes or edges.

The given value must have the same DataType as the attribute or be a NULL value to remove the current default value.

Parameters

attr	[in] The attribute.
value	[in] The default value to use for this attribute.

5.29.2.70 def sparksee.Graph.set_attribute_text (*self, oid, attr, tstream*)

Sets the writable TextStream for the given text attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
tstream	[in] New Text value. This corresponds to a TextStream to write.

5.29.2.71 def sparksee.Graph.tails (*self, edges*)

Gets all the tails from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.

Returns

The tails collection.

5.29.2.72 def sparksee.Graph.tails_and_heads (*self, edges, tails, heads*)

Gets all the tails and heads from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.
tails	[in|out] If not NULL, all the tails will be stored here.
heads	[in|out] If not NULL, all the heads will be stored here.

5.29.2.73 def sparksee.Graph.top_k (*self, attribute, operation, lower, order, k*)

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

Referenced by sparksee.Graph.top_k().

5.29.2.74 def sparksee.Graph.top_k (*self, attribute, order, k*)

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

References sparksee.Graph.top_k().

5.29.2.75 def sparksee.Graph.top_k (*self, attribute, operation, lower, higher, order, k*)

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
higher	The upper bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

References sparksee.Graph.top_k().

5.29.2.76 def sparksee.Graph.top_k (*self, attribute, operation, lower, higher, order, k, restriction*)

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
higher	The upper bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK
restriction	null

Returns

A KeyValues instance

References sparksee.Graph.top_k().

5.29.2.77 def sparksee.Graph.top_k (*self, attribute, operation, lower, order, k, restriction*)

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK
restriction	null

Returns

A KeyValues instance

References sparksee.Graph.top_k().

5.29.2.78 def sparksee.Graph.top_k (*self, attribute, order, k, restriction*)

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters

attribute	The attribute to perform the TopK on
order	The ordering semantincs of the top-k
k	The size of the TopK
restriction	null

Returns

A KeyValues instance

References sparksee.Graph.top_k().

5.30 sparksee.GraphExport Class Reference

Stores the graph exporting values.

Public Member Functions

- def set_defaults (self)
  Sets to default values.
- def set_label (self, label)
  Sets the graph label.
- def get_label (self)
  Gets the graph label.
- def __init__ (self)
  Creates a new GraphExport instance.

5.30.1 Detailed Description

Stores the graph exporting values.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.30.2 Member Function Documentation

5.30.2.1 def sparksee.GraphExport.get_label (*self*)

Gets the graph label.

Returns

The graph label.

5.30.2.2 def sparksee.GraphExport.set_label (*self, label*)

Sets the graph label.

Parameters

label	[in] The graph label.

5.31 sparksee.Int32List Class Reference

Sparksee 32-bit signed integer list.

Public Member Functions

- def clear (self)
  Clears the list.
- def __init__ (self)
  Constructor.
- def __iter__ (self)
  Gets a new TypeListIterator.
- def iterator (self)
  Gets a new Int32ListIterator.
- def add (self, value)
  Adds an 32-bit signed integer at the end of the list.
- def count (self)
  Number of elements in the list.

5.31.1 Detailed Description

Sparksee 32-bit signed integer list.

It stores a 32-bit signed integer list.

Use Int32ListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.31.2 Constructor & Destructor Documentation

5.31.2.1 def sparksee.Int32List.__init__ (*self*)

Constructor.

This creates an empty list.

5.31.3 Member Function Documentation

5.31.3.1 def sparksee.Int32List.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.31.3.2 def sparksee.Int32List.add (*self, value*)

Adds an 32-bit signed integer at the end of the list.

Parameters

value	[in] The integer.

5.31.3.3 def sparksee.Int32List.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.31.3.4 def sparksee.Int32List.iterator (*self*)

Gets a new Int32ListIterator.

Returns

Int32ListIterator instance.

5.32 sparksee.Int32ListIterator Class Reference

Int32List iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.32.1 Detailed Description

Int32List iterator class.

Iterator to traverse all the integer into a Int32List instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.32.2 Member Function Documentation

5.32.2.1 def sparksee.Int32ListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.32.2.2 def sparksee.Int32ListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.32.2.3 def sparksee.Int32ListIterator.next (*self*)

Moves to the next element.

Returns

The next element.

5.33 sparksee.KeyValue Class Reference

Public Member Functions

- def to_string (self)

  Returns a String representation of the KeyValue, used in ***unicode*** *and* ***str***

5.33.1 Member Function Documentation

5.33.1.1 def sparksee.KeyValue.to_string (*self*)

Returns a String representation of the KeyValue, used in **unicode** and **str**

5.34 sparksee.KeyValues Class Reference

Value set class.

Public Member Functions

- def next (self)

  Gets the next KeyValue pair.
- def next (self, kv)

  Gets the next KeyValue pair.
- def has_next (self)

  Checks if the KeyValues has more KeyValue pairs.
- def close (self)

  Closes the KeyValues instance.
- def __next__ (self)

  Used in next()
- def is_closed (self)

  Gets if KeyValues instance has been closed or not.

5.34.1 Detailed Description

Value set class.

This is a set of Value instances, that is there is no duplicated elements.

Use a ValuesIterator to traverse all the elements into the set.

When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.34.2 Member Function Documentation

5.34.2.1 def sparksee.KeyValues.__next__ (*self*)

Used in next()

Returns

The next element

5.34.2.2 def sparksee.KeyValues.close (*self*)

Closes the KeyValues instance.

It must be called to ensure the integrity of all data.

5.34.2.3 def sparksee.KeyValues.has_next (*self*)

Checks if the KeyValues has more KeyValue pairs.

Returns

Returns ture if there are more KeyValue pairs

5.34.2.4 def sparksee.KeyValues.is_closed (*self*)

Gets if KeyValues instance has been closed or not.

See also

close()

Returns

TRUE if the Values instance has been closed, FALSE otherwise.

5.34.2.5 def sparksee.KeyValues.next (*self*)

Gets the next KeyValue pair.

Returns

Returns the next KeyValue pair

Referenced by sparksee.KeyValues.next().

5.34.2.6 def sparksee.KeyValues.next (*self, kv*)

Gets the next KeyValue pair.

Parameters

kv	Returns the next KeyValue pair

References sparksee.KeyValues.next().

5.35 sparksee.KOpt Class Reference

KOpt class.

Public Member Functions

- def __init__ (self, session, tour)
  Creates a new instance.
- def set_current_tour (self, tour)
  Sets current tour as a list of nodes.
- def get_current_cost (self)
  Returns tour cost.
- def add_all_edge_types (self, dir)
  Allows for traversing all edge types of the graph.
- def get_current_tour (self)

Returns tour as a list of nodes.

- def set_edge_weight_attribute_type (self, attr)

  Sets the attribute to use as edge weight.
- def run_two_opt (self)

  Runs 2-Opt local search.
- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def run_three_opt (self)

  Runs 3-Opt local search.
- def set_max_iterations (self, max_iterations)

  Sets maximum number of iterations.
- def set_time_limit (self, max_time)

  Limits execution time.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def __init__ (self, session)

  Creates a new instance.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.

5.35.1 Detailed Description

KOpt class.

Implements the 2-Opt and 3-Opt algorithms

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.35.2 Constructor & Destructor Documentation

5.35.2.1 def sparksee.KOpt.__init__ (*self, session, tour*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform algorithm.
tour	[in] Initial tour that needs to be improved.

Referenced by sparksee.KOpt.__init__().

5.35.2.2 def sparksee.KOpt.__init__ (*self, session*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform algorithm.

References sparksee.KOpt.__init__().

5.35.3 Member Function Documentation

5.35.3.1 def sparksee.KOpt.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.35.3.2 def sparksee.KOpt.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Exceptions

RuntimeError	null

5.35.3.3 def sparksee.KOpt.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

sparksee::gdb::Error

Parameters

type	[in] Node type.

Exceptions

RuntimeError	null

5.35.3.4 def sparksee.KOpt.set_current_tour (*self, tour*)

Sets current tour as a list of nodes.

Parameters

tour	[in] Initial tour that needs to be improved.

5.35.3.5 def sparksee.KOpt.set_edge_weight_attribute_type (*self, attr*)

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double.

sparksee::gdb::Error

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

Exceptions

RuntimeError	null

5.35.3.6 def sparksee.KOpt.set_max_iterations (*self, max_iterations*)

Sets maximum number of iterations.

By default the algorithm will run until no improvement can be made in the current tour.

Parameters

max_iterations	[in] Maximum number of iterations

5.35.3.7 def sparksee.KOpt.set_time_limit (*self, max_time*)

Limits execution time.

Parameters

max_time	[in] Time limit in miliseconds

5.36 sparksee.LogLevel Class Reference

Log level enumeration.

Static Public Attributes

- int OFF = 0

  Disable log.
- int SEVERE = 1

  Severe log level.
- int WARNING = 2

  Warning log level.
- int INFO = 3

  Info log level.
- int CONFIG = 4

  Config log level.
- int FINE = 5

  Fine log level.
- int DEBUG = 6

  Debug log level.

5.36.1 Detailed Description

Log level enumeration.

Log level priority order is as follows, from minimum to maximum log information: Off (log is disabled), Severe, Warning, Info, Config, Fine, Debug.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.36.2 Member Data Documentation

5.36.2.1 int sparksee.LogLevel.CONFIG = 4 [static]

Config log level.

Errors, warnings, information messages and configuration details of the different components are shown.

5.36.2.2 int sparksee.LogLevel.DEBUG = 6 [static]

Debug log level.

This is for Sparksee development purposes and just works with debug versions of the library.

5.36.2.3 int sparksee.LogLevel.FINE = 5 [static]

Fine log level.

This is the higher and finest public log level, everything is dumped to the log.

5.36.2.4 int sparksee.LogLevel.INFO = 3 [static]

Info log level.

Errors, warnings and information messages are shown.

5.36.2.5 int sparksee.LogLevel.SEVERE = 1 [static]

Severe log level.

This is the lower log level, just errors are shown.

5.36.2.6 int sparksee.LogLevel.WARNING = 2 [static]

Warning log level.

Errors and warnings are shown.

5.37 sparksee.MissingEndpoint Class Reference

The policy to follow whenever and edge endpoint is missing during a loading.

Static Public Attributes

- int IS_ERROR = 0

  Throw an error.
- int CREATE = 1

  Create the endpoint.
- int IGNORE = 2

  Ignore the edge.

5.37.1 Detailed Description

The policy to follow whenever and edge endpoint is missing during a loading.

5.38 sparksee.NodeExport Class Reference

Stores the node exporting values.

Public Member Functions

- def get_labelcolor_rgb (self)
  Gets the node label color.
- def set_label (self, label)
  Sets the node label.
- def set_width (self, width)
  Gets the node width.
- def set_color_rgb (self, color)
  Sets the node color.
- def set_font_size (self, size)
  Sets the node label font size.
- def set_height (self, height)
  Sets the node height.
- def get_shape (self)
  Gets the node shape.
- def set_labelcolor_rgb (self, color)
  Sets the node label color.
- def is_fit (self)
  Gets whether the node size is fitted to the label or not.
- def get_color_rgb (self)
  Gets the node color.
- def set_fit (self, fit)
  Sets the node fit property.
- def __init__ (self)
  Creates a new instance.
- def get_font_size (self)
  Gets the node label font size.
- def set_defaults (self)
  Sets to default values.
- def get_label (self)
  Gets the node label.
- def get_height (self)
  Gets the node height.
- def get_width (self)
  Gets the node width.
- def set_shape (self, shape)
  Sets the node shape.

5.38.1 Detailed Description

Stores the node exporting values.

When 'fit' is set to TRUE, then 'height' and 'width' will be ignored.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

Shape: Box.

Color: 10863606 (0xa5c3f6).

Label color: 0 (Ox000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.38.2 Member Function Documentation

5.38.2.1 def sparksee.NodeExport.get_color_rgb (*self*)

Gets the node color.

Returns

The node color.

5.38.2.2 def sparksee.NodeExport.get_font_size (*self*)

Gets the node label font size.

Returns

The node label font size.

5.38.2.3 def sparksee.NodeExport.get_height (*self*)

Gets the node height.

Returns

The node height in pixels.

5.38.2.4 def sparksee.NodeExport.get_label (*self*)

Gets the node label.

Returns

The node label.

5.38.2.5 def sparksee.NodeExport.get_labelcolor_rgb (*self*)

Gets the node label color.

Returns

The node label color.

5.38.2.6 def sparksee.NodeExport.get_shape (*self*)

Gets the node shape.

Returns

The node shape.

5.38.2.7 def sparksee.NodeExport.get_width (*self*)

Gets the node width.

Returns

The node width in pixels.

5.38.2.8 def sparksee.NodeExport.is_fit (*self*)

Gets whether the node size is fitted to the label or not.

Returns

If TRUE, then the node size is fitted to the label, otherwise the size is fixed with the values of 'height' and 'width'.

5.38.2.9 def sparksee.NodeExport.set_color_rgb (*self, color*)

Sets the node color.

Parameters

color	The node color.

5.38.2.10 def sparksee.NodeExport.set_fit (*self, fit*)

Sets the node fit property.

Parameters

fit	[in] If TRUE, then the node size is fitted to the label ('height' and 'width' will be ignored), otherwise the size is fixed with the values of 'height' and 'width'.

5.38.2.11 def sparksee.NodeExport.set_font_size (*self, size*)

Sets the node label font size.

Parameters

size	[in] The node label font size.

5.38.2.12 def sparksee.NodeExport.set_height (*self, height*)

Sets the node height.

Parameters

height	[in] The node height in pixels.

5.38.2.13 def sparksee.NodeExport.set_label (*self, label*)

Sets the node label.

Parameters

label	[in] The node label.

5.38.2.14 def sparksee.NodeExport.set_labelcolor_rgb (*self, color*)

Sets the node label color.

Parameters

color	[in] The node label color.

5.38.2.15 def sparksee.NodeExport.set_shape (*self, shape*)

Sets the node shape.

Parameters

shape	[in] The node shape.

5.38.2.16 def sparksee.NodeExport.set_width (*self, width*)

Gets the node width.

Parameters

width	The node width in pixels.

5.39 sparksee.NodeShape Class Reference

Node shape.

Static Public Attributes

- int BOX = 0
  Box shape.
- int ROUND = 1
  Round shape.

5.39.1 Detailed Description

Node shape.

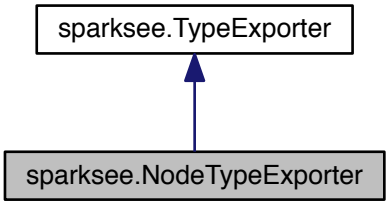
Author

Sparsity Technologies http://www.sparsity-technologies.com

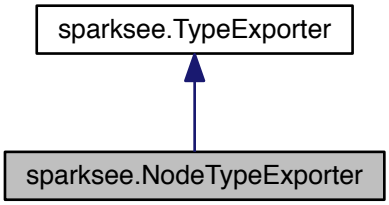
5.40 sparksee.NodeTypeExporter Class Reference

NodeTypeExporter class.

Inheritance diagram for sparksee.NodeTypeExporter:



Collaboration diagram for sparksee.NodeTypeExporter:



Public Member Functions

- def __init__ (self)

  Creates a new instance.
- def run (self)

  See the TypeExporter class Run method.
- def __init__ (self, row_writer, graph, type, attrs)

  Creates a new instance.
- def set_type (self, type)

  Sets the type to be exported.
- def set_header (self, header)

  Sets the presence of a header row.
- def set_row_writer (self, rw)

  Sets the output data destination.
- def set_graph (self, graph)

  Sets the graph that will be exported.
- def register (self, tel)

  Registers a new listener.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.

5.40.1 Detailed Description

NodeTypeExporter class.

Specific TypeExporter implementation for node types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.40.2 Constructor & Destructor Documentation

5.40.2.1 def sparksee.NodeTypeExporter.__init__ (*self, row_writer, graph, type, attrs*)

Creates a new instance.

Parameters

row_writer	[in] Output RowWriter.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be exported.

References sparksee.NodeTypeExporter.__init__().

5.40.3 Member Function Documentation

5.40.3.1 def sparksee.NodeTypeExporter.register (*self, tel*)

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

5.40.3.2 def sparksee.NodeTypeExporter.run (*self*)

See the TypeExporter class Run method.

Exceptions

RuntimeError	null
IOError	null

5.40.3.3 def sparksee.NodeTypeExporter.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

5.40.3.4 def sparksee.NodeTypeExporter.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.40.3.5 def sparksee.NodeTypeExporter.set_graph (*self, graph*)

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

5.40.3.6 def sparksee.NodeTypeExporter.set_header (*self, header*)

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

5.40.3.7 def sparksee.NodeTypeExporter.set_row_writer (*self, rw*)

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

5.40.3.8 def sparksee.NodeTypeExporter.set_type (*self, type*)

Sets the type to be exported.

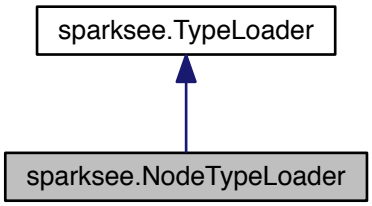
Parameters

type	[in] Type identifier.

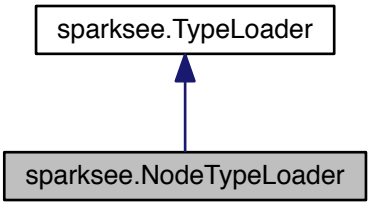
5.41 sparksee.NodeTypeLoader Class Reference

NodeTypeLoader class.

Inheritance diagram for sparksee.NodeTypeLoader:



Collaboration diagram for sparksee.NodeTypeLoader:



Public Member Functions

- def run (self)

  See the TypeLoader class Run method.
- def set_log_error (self, path)

  Sets a log error file.
- def set_type (self, type)

  Sets the type to be loaded.
- def set_locale (self, locale_str)

  Sets the locale that will be used to read the data.
- def set_timestamp_format (self, timestamp_format)

  Sets a specific timestamp format.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def __init__ (self)

  Creates a new instance.
- def register (self, tel)

  Registers a new listener.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.
- def __init__ (self, row_reader, graph, type, attrs, attrs_pos)

  Creates a new instance.
- def run_two_phases (self)

  See the TypeLoader class RunTwoPhases method.
- def set_row_reader (self, rr)

  Sets the input data source.
- def set_attribute_positions (self, attrs_pos)

  Sets the list of attribute positions.
- def set_graph (self, graph)

  Sets the graph where the data will be loaded.
- def run_n_phases (self, partitions)

  See the TypeLoader class RunNPhases method.
- def set_log_off (self)

  Truns off all the error reporting.

5.41.1 Detailed Description

NodeTypeLoader class.

Specific TypeLoader implementation for node types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.41.2 Constructor & Destructor Documentation

5.41.2.1 def sparksee.NodeTypeLoader.__init__ (*self, row_reader, graph, type, attrs, attrs_pos*)

Creates a new instance.

Parameters

row_reader	[in] Input RowReader.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be loaded.
attrs_pos	[in] Attribute positions (column index >=0).

References sparksee.NodeTypeLoader.__init__().

5.41.3 Member Function Documentation

5.41.3.1 def sparksee.NodeTypeLoader.register (*self, tel*)

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

5.41.3.2 def sparksee.NodeTypeLoader.run (*self*)

See the TypeLoader class Run method.

Exceptions

RuntimeError	null
IOError	null

5.41.3.3 def sparksee.NodeTypeLoader.run_n_phases (*self, partitions*)

See the TypeLoader class RunNPhases method.

Parameters

partitions	null

Exceptions

RuntimeError	null
IOError	null

5.41.3.4 def sparksee.NodeTypeLoader.run_two_phases (*self*)

See the TypeLoader class RunTwoPhases method.

Exceptions

RuntimeError	null
IOError	null

5.41.3.5 def sparksee.NodeTypeLoader.set_attribute_positions (*self, attrs_pos*)

Sets the list of attribute positions.

Parameters

attrs_pos	[in] Attribute positions (column index >=0).

5.41.3.6 def sparksee.NodeTypeLoader.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

5.41.3.7 def sparksee.NodeTypeLoader.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.41.3.8 def sparksee.NodeTypeLoader.set_graph (*self, graph*)

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

5.41.3.9 def sparksee.NodeTypeLoader.set_locale (*self, locale_str*)

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

locale_str	[in] The locale string for the read data. See CSVReader.

5.41.3.10 def sparksee.NodeTypeLoader.set_log_error (*self, path*)

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

IOError	If bad things happen opening the file.

5.41.3.11 def sparksee.NodeTypeLoader.set_log_off (*self*)

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.41.3.12 def sparksee.NodeTypeLoader.set_row_reader (*self, rr*)

Sets the input data source.

Parameters

rr	[in] Input RowReader.

5.41.3.13 def sparksee.NodeTypeLoader.set_timestamp_format (*self, timestamp_format*)

Sets a specific timestamp format.

Parameters

timestamp_format	[in] A string with the timestamp format definition.

5.41.3.14 def sparksee.NodeTypeLoader.set_type (*self, type*)

Sets the type to be loaded.

Parameters

type	[in] Type identifier.

5.42 sparksee.Objects Class Reference

Object identifier set class.

Public Member Functions

- def copy (self, objs)

  Performs the copy operation.
- def count (self)

  Gets the number of elements into the collection.
- def equals (self, objs)

  Checks if the given Objects contains the same information.
- def any (self)

  Gets an element from the collection.
- def contains (self, objs)

  Check if this objects contains the other one.
- def combine_intersection (self, objs1, objs2)

  Creates a new Objects instance which is the intersection of the two given.
- def __iter__ (self)

  Gets a new ObjectsIterator.
- def iterator (self)

  Gets an ObjectsIterator.
- def union (self, objs)

  Performs the union operation.
- def combine_union (self, objs1, objs2)

  Creates a new Objects instance which is the union of the two given.
- def intersection (self, objs)

  Performs the intersection operation.
- def clear (self)

  Clears the collection removing all its elements.
- def iterator_from_index (self, index)

Gets an ObjectsIterator skipping index elements.

- def exists (self, e)

  Gets if the given element exists into the collection.
- def copy (self)

  Creates a new Objects instance as a copy of the given one.
- def difference (self, objs)

  Performs the difference operation.
- def combine_difference (self, objs1, objs2)

  Creates a new Objects instance which is the difference of the two given.
- def sample (self, exclude, samples)

  Creates a new Objects instance which is a sample of the calling one.
- def close (self)

  Closes the Objects instance.
- def add (self, e)

  Adds an element into the collection.
- def remove (self, e)

  Removes an element from the collection.
- def iterator_from_element (self, e)

  Gets an ObjectsIterator starting from the given element.
- def is_closed (self)

  Gets if Objects instance has been closed or not.

Static Public Attributes

- int INVALID_OID = 0

  Invalid OID constant.

5.42.1 Detailed Description

Object identifier set class.

It stores a collection of Sparksee object identifiers as a set. As a set, there is no order and no duplicated elements.

This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.

This class is not thread-safe.

ObjectsIterator must be used to traverse all the elements into the set.

When the Objects instance is closed, it closes all existing and non-closed ObjectsIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.42.2 Member Function Documentation

5.42.2.1 def sparksee.Objects.__iter__ (*self*)

Gets a new ObjectsIterator.

Returns

ObjectsIterator instance

5.42.2.2 def sparksee.Objects.add (*self, e*)

Adds an element into the collection.

Parameters

e	[in] Element to be added.

Returns

TRUE if the element is added, FALSE if the element was already into the collection.

5.42.2.3 def sparksee.Objects.any (*self*)

Gets an element from the collection.

Returns

Any element from the collection.

Exceptions

RuntimeError	null
RuntimeError	whether the collection is empty.

5.42.2.4 def sparksee.Objects.close (*self*)

Closes the Objects instance.

It must be called to ensure the integrity of all data.

5.42.2.5 def sparksee.Objects.combine_difference (*self, objs1, objs2*)

Creates a new Objects instance which is the difference of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

5.42.2.6 def sparksee.Objects.combine_intersection (*self, objs1, objs2*)

Creates a new Objects instance which is the intersection of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

5.42.2.7 def sparksee.Objects.combine_union (*self, objs1, objs2*)

Creates a new Objects instance which is the union of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

5.42.2.8 def sparksee.Objects.contains (*self, objs*)

Check if this objects contains the other one.

Parameters

objs	Objects collection.

Returns

True if it contains the given object.

5.42.2.9 def sparksee.Objects.copy (*self, objs*)

Performs the copy operation.

This updates the Objects calling instance and copies the given Objects instance.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

Referenced by sparksee.Objects.copy().

5.42.2.10 def sparksee.Objects.copy (*self*)

Creates a new Objects instance as a copy of the given one.

Returns

The new Objects instance.

References sparksee.Objects.copy().

5.42.2.11 def sparksee.Objects.count (*self*)

Gets the number of elements into the collection.

Returns

The number of elements into the collection.

5.42.2.12 def sparksee.Objects.difference (*self, objs*)

Performs the difference operation.

This updates the Objects calling instance removing those existing elements at the given Objects instance.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

5.42.2.13 def sparksee.Objects.equals (*self, objs*)

Checks if the given Objects contains the same information.

Parameters

objs	[in] Objects instance.

Returns

True if the objects are equal or false otherwise.

5.42.2.14 def sparksee.Objects.exists (*self, e*)

Gets if the given element exists into the collection.

Parameters

e	[in] Element.

Returns

TRUE if the element exists into the collection, FALSE otherwise.

5.42.2.15 def sparksee.Objects.intersection (*self, objs*)

Performs the intersection operation.

Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

5.42.2.16 def sparksee.Objects.is_closed (*self*)

Gets if Objects instance has been closed or not.

See also

close()

Returns

TRUE if the Objects instance has been closed, FALSE otherwise.

5.42.2.17 def sparksee.Objects.iterator (*self*)

Gets an ObjectsIterator.

Returns

ObjectsIterator instance.

5.42.2.18 def sparksee.Objects.iterator_from_element (*self, e*)

Gets an ObjectsIterator starting from the given element.

Objects collection has no order, so this method is implementation-dependent. e[in] The first element to traverse in the resulting

Parameters

e	[in] The first element to traverse in the resulting ObjectsIterator instance.

Returns

ObjectsIterator instance.

5.42.2.19 def sparksee.Objects.iterator_from_index (*self, index*)

Gets an ObjectsIterator skipping index elements.

Objects collection has no order, so this method is implementation-dependent.

Parameters

index	[in] The number of elements to skip from the beginning. It must be in the range [0..Size).

Returns

ObjectsIterator instance.

5.42.2.20 def sparksee.Objects.remove (*self, e*)

Removes an element from the collection.

Parameters

e	[in] Element to be removed.

Returns

TRUE if the element is removed, FALSE if the element was not into the collection.

5.42.2.21 def sparksee.Objects.sample (*self, exclude, samples*)

Creates a new Objects instance which is a sample of the calling one.

Parameters

exclude	[in] If not NULL, elements into this collection will be excluded from the resulting one.
samples	[in] Number of elements into the resulting collection.

Returns

Sample collection.

5.42.2.22 def sparksee.Objects.union (*self, objs*)

Performs the union operation.

This adds all existing elements of the given Objects instance to the Objects calling instance

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

5.43 sparksee.ObjectsIterator Class Reference

ObjectsIterator class.

Public Member Functions

- def next (self)
  Gets the next element to traverse.
- def has_next (self)
  Gets if there are more elements to traverse.
- def close (self)
  Closes the ObjectsIterator instance.
- def __next__ (self)
  Used in next()
- def is_closed (self)
  Gets if ObjectsIterator instance has been closed or not.

5.43.1 Detailed Description

ObjectsIterator class.

Iterator to traverse all the object identifiers from an Objects instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.43.2 Member Function Documentation

5.43.2.1 def sparksee.ObjectsIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.43.2.2 def sparksee.ObjectsIterator.close (*self*)

Closes the ObjectsIterator instance.

It must be called to ensure the integrity of all data.

5.43.2.3 def sparksee.ObjectsIterator.has_next (*self*)

Gets if there are more elements to traverse.

Returns

TRUE if there are more elements to traverse, FALSE otherwise.

5.43.2.4 def sparksee.ObjectsIterator.is_closed (*self*)

Gets if ObjectsIterator instance has been closed or not.

See also

close()

Returns

TRUE if the ObjectsIterator instance has been closed, FALSE otherwise.

5.43.2.5 def sparksee.ObjectsIterator.next (*self*)

Gets the next element to traverse.

Returns

The next element.

5.44 sparksee.ObjectType Class Reference

Object type enumeration.

Static Public Attributes

- int NODE = 0

  Node object type.
- int EDGE = 1

  Edge object type.

5.44.1 Detailed Description

Object type enumeration.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.45 sparksee.OIDList Class Reference

Sparksee object identifier list.

Public Member Functions

- def clear (self)
  Clears the list.
- def __init__ (self, num_invalid_oids)
  Constructor.
- def __init__ (self)
  Constructor.
- def __iter__ (self)
  Gets a new TypeListIterator.
- def set (self, pos, oid)
  Sets a Sparksee object identifier at the specified position of the list.
- def iterator (self)
  Gets a new OIDListIterator.
- def add (self, attr)
  Adds a Sparksee object identifier at the end of the list.
- def count (self)
  Number of elements in the list.

5.45.1 Detailed Description

Sparksee object identifier list.

It stores a Sparksee object identifier list.

Use OIDListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.45.2 Constructor & Destructor Documentation

5.45.2.1 def sparksee.OIDList.__init__ (*self, num_invalid_oids*)

Constructor.

This creates a list with N invalid oids.

Parameters

num_invalid_oids	[in] The number of invalid oids added to the list.

Referenced by sparksee.OIDList.__init__().

5.45.2.2 def sparksee.OIDList.__init__ (*self*)

Constructor.

This creates an empty list.

References sparksee.OIDList.__init__().

5.45.3 Member Function Documentation

5.45.3.1 def sparksee.OIDList.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.45.3.2 def sparksee.OIDList.add (*self, attr*)

Adds a Sparksee object identifier at the end of the list.

Parameters

attr	[in] Sparksee object identifier.

5.45.3.3 def sparksee.OIDList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.45.3.4 def sparksee.OIDList.iterator (*self*)

Gets a new OIDListIterator.

Returns

OIDListIterator instance.

5.45.3.5 def sparksee.OIDList.set (*self, pos, oid*)

Sets a Sparksee object identifier at the specified position of the list.

Parameters

pos	[in] List position [0..Count()-1].
oid	[in] Sparksee object identifier.

5.46 sparksee.OIDListIterator Class Reference

OIDList iterator class.

Public Member Functions

- def next (self)
  Moves to the next element.
- def has_next (self)
  Gets if there are more elements.
- def __next__ (self)
  Used in next()

5.46.1 Detailed Description

OIDList iterator class.

Iterator to traverse all the Sparksee object identifier into a OIDList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.46.2 Member Function Documentation

5.46.2.1 def sparksee.OIDListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.46.2.2 def sparksee.OIDListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.46.2.3 def sparksee.OIDListIterator.next (*self*)

Moves to the next element.

Returns

The next element.

5.47 sparksee.Order Class Reference

Order enumeration.

Static Public Attributes

- int ASCENDENT = 0
  From lower to higher.
- int DESCENDENT = 1
  From higher to lower.

5.47.1 Detailed Description

Order enumeration.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.48 sparksee.PageRank Class Reference

PageRank class.

Public Member Functions

- def set_damping (self, damping)
  Sets the damping value for the PageRank.
- def set_default_weight (self, weight)
  Sets the default weight for those cases when a given edge does not have a weight attribute set.
- def run (self)
  Runs the algorithm.
- def add_all_edge_types (self, dir)
  Allows for traversing all edge types of the graph.
- def set_output_attribute_type (self, attr)
  Sets the output attribute type.
- def set_starting_node (self, start_node)
  Sets the starting node of the page rank to compute the Personalized PageRank variant.
- def set_initial_page_rank_value (self, start_value)
  Sets the initial PageRank value.
- def set_edge_weight_attribute_type (self, attr)

Sets the attribute to use as edge weight.

- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def set_num_iterations (self, num_iterations)

  Sets the number of iterations to run the PageRank for.
- def set_tolerance (self, tolerance)

  Sets the tolerance threashold to continue computing the PageRank after each iteration.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def __init__ (self, session)

  Builds the PageRank.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.

5.48.1 Detailed Description

PageRank class.

Implements the PageRank algorithm

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.48.2 Constructor & Destructor Documentation

5.48.2.1 def sparksee.PageRank.__init__ (*self, session*)

Builds the PageRank.

Parameters

session	[in] The session to use

5.48.3 Member Function Documentation

5.48.3.1 def sparksee.PageRank.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters

dir	[in] Edge direction.

5.48.3.2 def sparksee.PageRank.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Exceptions

RuntimeError	null

5.48.3.3 def sparksee.PageRank.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	null

Exceptions

RuntimeError	null

5.48.3.4 def sparksee.PageRank.run (*self*)

Runs the algorithm.

sparksee::gdb::Error

Exceptions

RuntimeError	null

5.48.3.5 def sparksee.PageRank.set_damping (*self, damping*)

Sets the damping value for the PageRank.

Parameters

damping	[in] The damping value. Default: 0.85

5.48.3.6 def sparksee.PageRank.set_default_weight (*self, weight*)

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters

weight	[in] The default weight

5.48.3.7 def sparksee.PageRank.set_edge_weight_attribute_type (*self, attr*)

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

sparksee::gdb::Error

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

Exceptions

RuntimeError	null

5.48.3.8 def sparksee.PageRank.set_initial_page_rank_value (*self, start_value*)

Sets the initial PageRank value.

If a starting node is set, this initial value is only set for the starting node and the rest of nodes are set to 0.0

Parameters

start_value	[in] The initial value to set. Default: 0.0

5.48.3.9 def sparksee.PageRank.set_num_iterations (*self, num_iterations*)

Sets the number of iterations to run the PageRank for.

Parameters

num_iterations	[in] The number of interations to set. Default: 20

5.48.3.10 def sparksee.PageRank.set_output_attribute_type (*self, attr*)

Sets the output attribute type.

If the PageRank will run on more than one node type, then the output attribute must be of type GLOBAL_TYPE or NODES_TYPE. Otherwise, it must be a valid attribute for the used node type.

Parameters

attr	[in] The attribute to store the result. Default: InvalidAttribute

Exceptions

RuntimeError	null

5.48.3.11 def sparksee.PageRank.set_starting_node (*self, start_node*)

Sets the starting node of the page rank to compute the Personalized PageRank variant.

sparksee::gdb::Error

Parameters

start_node	null

Exceptions

RuntimeError	null

5.48.3.12 def sparksee.PageRank.set_tolerance (*self, tolerance*)

Sets the tolerance threashold to continue computing the PageRank after each iteration.

If all the changes to any PPR value after an iteration are below that tolerance threshold, the algorithm finishes.

Parameters

tolerance	[in] The tolerance to use normalized between 0 and 1. Default: 0.000001

5.49 sparksee.Platform Class Reference

Platform class.

Public Member Functions

- def get_statistics (self, stats)

  Gets platform data and statistics.

5.49.1 Detailed Description

Platform class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.49.2 Member Function Documentation

5.49.2.1 def sparksee.Platform.get_statistics (*self, stats*)

Gets platform data and statistics.

Parameters

stats	[in|out] This updates the given PlatformStatistics.

5.50 sparksee.PlatformStatistics Class Reference

Platform data and statistics.

Public Member Functions

- def __init__ (self)
  Creates a new instance setting all values to 0.
- def get_total_mem (self)
  Gets physical memory size in Bytes.
- def get_available_mem (self)
  Gets avialable (free) memory size in Bytes.
- def get_real_time (self)
  Gets time in microseconds (since epoch).
- def get_system_time (self)
  Gets CPU system time.
- def get_user_time (self)
  Gets CPU user time.
- def get_num_c_p_us (self)
  Gets the number of CPUs.

5.50.1 Detailed Description

Platform data and statistics.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.50.2 Member Function Documentation

5.50.2.1 def sparksee.PlatformStatistics.get_available_mem (*self*)

Gets avialable (free) memory size in Bytes.

Returns

Avialable (free) memory size in Bytes.

5.50.2.2 def sparksee.PlatformStatistics.get_num_c_p_us (*self*)

Gets the number of CPUs.

Returns

The number of CPUs.

5.50.2.3 def sparksee.PlatformStatistics.get_real_time (*self*)

Gets time in microseconds (since epoch).

Returns

Time in microseconds (since epoch).

5.50.2.4 def sparksee.PlatformStatistics.get_system_time (*self*)

Gets CPU system time.

Returns

CPU system time.

5.50.2.5 def sparksee.PlatformStatistics.get_total_mem (*self*)

Gets physical memory size in Bytes.

Returns

Physical memory size in Bytes.

5.50.2.6 def sparksee.PlatformStatistics.get_user_time (*self*)

Gets CPU user time.

Returns

CPU user time.

5.51 sparksee.Query Class Reference

Query class.

Public Member Functions

- def set_stream (self, stream, handler)

  Sets a query stream handler.
- def set_dynamic (self, name, value)

  Sets the value for a dynamic paramater.
- def execute (self, stmt, reiterable)

  Executes the given statement.
- def close (self)

  Closes the Query instance.
- def is_closed (self)

  Gets if the Query instance has been closed or not.

5.51.1 Detailed Description

Query class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.51.2 Member Function Documentation

5.51.2.1 def sparksee.Query.close (*self*)

Closes the Query instance.

It must be called to ensure the integrity of all data.

5.51.2.2 def sparksee.Query.execute (*self, stmt, reiterable*)

Executes the given statement.

Parameters

stmt	[in] Query statement.
reiterable	[in] Whether we want the resultset to be reiterable or not

Returns

A ResultSet instance with the contents of the result of the query.

5.51.2.3 def sparksee.Query.is_closed (*self*)

Gets if the Query instance has been closed or not.

See also

close()

Returns

TRUE if the Query instance has been closed, FALSE otherwise.

5.51.2.4 def sparksee.Query.set_dynamic (*self, name, value*)

Sets the value for a dynamic paramater.

Parameters

name	[in] Parameter name
value	[in] Parameter value

5.51.2.5 def sparksee.Query.set_stream (*self, stream, handler*)

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.

Parameters

stream	[in] The stream name
handler	[in] Query stream handler

Returns

The previous handler, or NULL if it does not exists

5.52 sparksee.QueryContext Class Reference

Query context interface.

Public Member Functions

- def new_query (self)

  Creates a new Query.
- def __init__ (self)

  Default constructor.

5.52.1 Detailed Description

Query context interface.

A QueryContext contains and manages the resources required to run a Query. A Session is one example of a QueryContext connected to a Sparksee database. The applications can implement their own contexts to run queries out of Sparksee.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.53 sparksee.QueryLanguage Class Reference

The supported query languages.

Static Public Attributes

- int SPARKSEE_ALGEBRA = 0

  The internal Sparksee Algebra.
- int SPARKSEE_CYPHER = 1

  The Sparksee Cypher Language inspired by the OpenCypher Query Language.

5.53.1 Detailed Description

The supported query languages.

5.54 sparksee.QueryStream Class Reference

Query stream interface.

Public Member Functions

- def fetch (self, list)

  Gets the next row and moves the iterator forward.
- def prepare (self, list)

  Prepares the stream before it is started.
- def start (self, list)

  Starts the stream.

5.54.1 Detailed Description

Query stream interface.

A QueryStream is the interface between the application and the STREAM operator. When the operator starts inside a Query, the method is prepared with query-defined arguments. Then, if there are input operations, the STREAM operator builds the ResultSets and starts the iteration. Finally, the operator fetches rows until no more are available.

Application exceptions must be cached by the subclass that implements the interface.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.54.2 Member Function Documentation

5.54.2.1 def sparksee.QueryStream.fetch (*self, list*)

Gets the next row and moves the iterator forward.

The end of sequence is denoted by returning TRUE with an empty row. A valid row must contain as many values (even NULL) as expected by the query.

Parameters

list	[out] Storage for the new rows

Returns

TRUE if there is a row or end of sequence, FALSE on error

5.54.2.2 def sparksee.QueryStream.prepare (*self, list*)

Prepares the stream before it is started.

Parameters

list	[in] Optional list of arguments

Returns

FALSE on error

5.54.2.3 def sparksee.QueryStream.start (*self, list*)

Starts the stream.

Parameters

list	[in] Optional list of input ResultSets

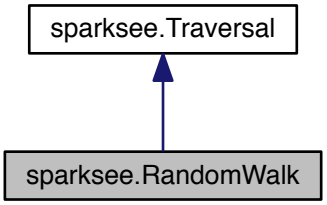
Returns

FALSE on error

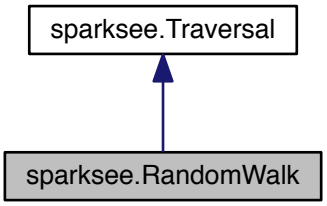
5.55 sparksee.RandomWalk Class Reference

RandomWalk class.

Inheritance diagram for sparksee.RandomWalk:



Collaboration diagram for sparksee.RandomWalk:



Public Member Functions

- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def set_default_weight (self, weight)

  Sets the default weight for those cases when a given edge does not have a weight attribute set.
- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def set_seed (self, seed)

  Sets the seed of the random walk.
- def set_edge_weight_attribute_type (self, attr)

  Sets the attribute to use as edge weight.
- def set_in_out_parameter (self, val)

  Sets the In-Out parameter of the RandomWalk.
- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def __init__ (self, session, node)

  Builds the RandomWalk.

- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def set_return_parameter (self, val)

  Sets the return parameter of the RandomWalk.
- def next (self)

  Gets the next object of the traversal.
- def has_next (self)

  Gets if there are more objects to be traversed.
- def get_current_depth (self)

  Returns the depth of the current node.
- def reset (self, start_node)

  Sets the starting node of the RandomWalk.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def __iter__ (self)

  Gets a new TraversalIterator.
- def __next__ (self)

  Used in next()
- def close (self)

  Closes the Traversal instance.
- def is_closed (self)

  Gets if Traversal has been closed or not.

5.55.1 Detailed Description

RandomWalk class.

Implements the RandomWalk algorithm

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.55.2 Constructor & Destructor Documentation

5.55.2.1 def sparksee.RandomWalk.__init__ (*self, session, node*)

Builds the RandomWalk.

Parameters

session	[in] The session to use
node	[in] The starting node of the traversal

5.55.3 Member Function Documentation

5.55.3.1 def sparksee.Traversal.__iter__ (*self*) [inherited]

Gets a new TraversalIterator.

Returns

TraversalIterator instance

5.55.3.2 def sparksee.Traversal.__next__ (*self*) [inherited]

Used in next()

Returns

The next element

5.55.3.3 def sparksee.RandomWalk.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.55.3.4 def sparksee.RandomWalk.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.55.3.5 def sparksee.RandomWalk.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

5.55.3.6 def sparksee.Traversal.close (*self*) [inherited]

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

5.55.3.7 def sparksee.RandomWalk.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.55.3.8 def sparksee.RandomWalk.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.55.3.9 def sparksee.RandomWalk.get_current_depth (*self*)

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

5.55.3.10 def sparksee.RandomWalk.has_next (*self*)

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

5.55.3.11 def sparksee.Traversal.is_closed (*self*) [inherited]

Gets if Traversal has been closed or not.

See also

close()

Returns

TRUE if the Traversal instance has been closed, FALSE otherwise.

5.55.3.12 def sparksee.RandomWalk.next (*self*)

Gets the next object of the traversal.

Returns

A node or edge identifier.

5.55.3.13 def sparksee.RandomWalk.reset (*self, start_node*)

Sets the starting node of the RandomWalk.

This method resets the RandomWalk.

sparksee::gdb::Error

Parameters

start_node	null

5.55.3.14 def sparksee.RandomWalk.set_default_weight (*self, weight*)

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters

weight	[in] The default weight

5.55.3.15 def sparksee.RandomWalk.set_edge_weight_attribute_type (*self, attr*)

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

5.55.3.16 def sparksee.RandomWalk.set_in_out_parameter (*self, val*)

Sets the In-Out parameter of the RandomWalk.

Parameters

val	The In-Out parameter to set. Default: 1.0

5.55.3.17 def sparksee.RandomWalk.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.55.3.18 def sparksee.RandomWalk.set_return_parameter (*self, val*)

Sets the return parameter of the RandomWalk.

Parameters

val	The return parameter to set. Default: 1.0

5.55.3.19 def sparksee.RandomWalk.set_seed (*self, seed*)

Sets the seed of the random walk.

Parameters

seed	The seed to generate the random numbers that drive the random walk

5.56 sparksee.ResultSet Class Reference

ResultSet class.

Public Member Functions

- def rewind (self)

  Positions the cursor before the first row.
- def get_column_data_type (self, index)

  Gets the datatype for the given column.
- def get_column (self, index)

  Gets the value for the given column.
- def get_num_columns (self)

  Gets the number of columns.
- def get_column_name (self, index)

  Gets the name for the given column.
- def next (self)

  Fetches the next row.
- def get_j_s_o_n (self, rows)

Returns rows in JSON format.

- def get_column (self, index, value)

  Gets the value for the given column.

- def get_column_index (self, name)

  Gets the column index for the given column name.

- def is_closed (self)

  Gets if the ResultSet instance has been closed or not.

5.56.1 Detailed Description

ResultSet class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.56.2 Member Function Documentation

5.56.2.1 def sparksee.ResultSet.get_column (*self, index*)

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

Parameters

index	[in] Column index.

Returns

The Value of the given column.

Referenced by sparksee.ResultSet.get_column().

5.56.2.2 def sparksee.ResultSet.get_column (*self, index, value*)

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

Parameters

index	[in] Column index.
value	[in|out] Value.

References sparksee.ResultSet.get_column().

5.56.2.3 def sparksee.ResultSet.get_column_data_type (*self, index*)

Gets the datatype for the given column.

Parameters

index	[in] Column index.

Returns

DataType for the given column.

5.56.2.4 def sparksee.ResultSet.get_column_index (*self, name*)

Gets the column index for the given column name.

Parameters

name	[in] Column name.

Returns

Column index.

5.56.2.5 def sparksee.ResultSet.get_column_name (*self, index*)

Gets the name for the given column.

Parameters

index	[in] Column index.

Returns

Column name.

5.56.2.6 def sparksee.ResultSet.get_j_s_o_n (*self, rows*)

Returns rows in JSON format.

Rows are returned from the current position.

Parameters

rows	[in] Maximum number of rows

Returns

JSON representation of the next <rows> rows in the resultset

5.56.2.7 def sparksee.ResultSet.get_num_columns (*self*)

Gets the number of columns.

Columns are in the range [0...COLUMNS).

Returns

The number of columns.

5.56.2.8 def sparksee.ResultSet.is_closed (*self*)

Gets if the ResultSet instance has been closed or not.

See also

#close()

Returns

TRUE if the ResultSet instance has been closed, FALSE otherwise.

5.56.2.9 def sparksee.ResultSet.next (*self*)

Fetches the next row.

A ResultSet cursor is initially positioned before the first row; the first call to the method "Next" makes the first row the current row; the second call makes the second row the current row, and so on.

QueryExceptionIf a database access error occurs.

Returns

TRUE if the next row has been successfully fetched, FALSE otherwise.

5.57 sparksee.ResultSetList Class Reference

ResultSet list.

Public Member Functions

- def clear (self)
  Clears the list.
- def get (self, index)
  Returns the ResultSet at the specified position in the list.
- def __iter__ (self)
  Gets a new ResultSetListIterator.
- def iterator (self)
  Gets a new ResultSetListIterator.
- def __init__ (self)
  Constructor.
- def count (self)
  Number of elements in the list.

5.57.1 Detailed Description

ResultSet list.

It stores a ResultSet list.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.57.2 Constructor & Destructor Documentation

5.57.2.1 def sparksee.ResultSetList.__init__ (*self*)

Constructor.

This creates an empty list.

5.57.3 Member Function Documentation

5.57.3.1 def sparksee.ResultSetList.__iter__ (*self*)

Gets a new ResultSetListIterator.

Returns

ResultSetListIterator instance

5.57.3.2 def sparksee.ResultSetList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.57.3.3 def sparksee.ResultSetList.get (*self, index*)

Returns the ResultSet at the specified position in the list.

Parameters

index	[in] Index of the element to return, starting at 0.

5.57.3.4 def sparksee.ResultSetList.iterator (*self*)

Gets a new ResultSetListIterator.

Returns

ResultSetListIterator instance.

5.58 sparksee.ResultSetListIterator Class Reference

ResultSetList iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.58.1 Detailed Description

ResultSetList iterator class.

Iterator to traverse all the values into a ResultSetList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.58.2 Member Function Documentation

5.58.2.1 def sparksee.ResultSetListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.58.2.2 def sparksee.ResultSetListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.58.2.3 def sparksee.ResultSetListIterator.next (*self*)

Moves to the next element.

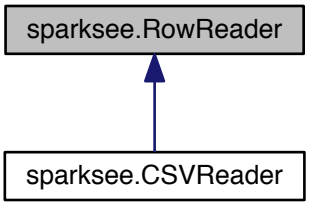
Returns

The next element.

5.59 sparksee.RowReader Class Reference

RowReader interface.

Inheritance diagram for sparksee.RowReader:



Public Member Functions

- def close (self)
  Closes the reader.
- def reset (self)
  Moves the reader to the beginning.
- def read (self, row)
  Reads the next row as a string array.
- def get_row (self)
  The row number for the current row.

5.59.1 Detailed Description

RowReader interface.

Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once close is called no more read operations can be executed.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.59.2 Member Function Documentation

5.59.2.1 def sparksee.RowReader.close (*self*)

Closes the reader.

Exceptions

IOError	If the close fails.

5.59.2.2 def sparksee.RowReader.get_row (*self*)

The row number for the current row.

Returns

The current row number; 0 if there is no current row.

Exceptions

IOError	If it fails.

5.59.2.3 def sparksee.RowReader.read (*self, row*)

Reads the next row as a string array.

Parameters

row	[out] A string list with each comma-separated element as a separate entry.

Returns

Returns true if a row had been read or false otherwise.

Exceptions

IOError	If bad things happen during the read.

5.59.2.4 def sparksee.RowReader.reset (*self*)

Moves the reader to the beginning.

Restarts the reader.

Returns

true if the reader can be restarted, false otherwise.

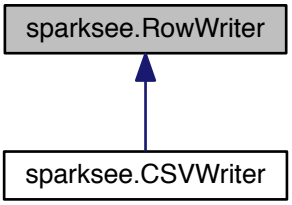
Exceptions

IOError	If bad things happen during the restart.

5.60 sparksee.RowWriter Class Reference

RowWriter interface.

Inheritance diagram for sparksee.RowWriter:



Public Member Functions

- def close (self)

  Closes the writer.
- def write (self, row)

  Writes the next row.

5.60.1 Detailed Description

RowWriter interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more write operations can be executed.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.60.2 Member Function Documentation

5.60.2.1 def sparksee.RowWriter.close (*self*)

Closes the writer.

Exceptions

RuntimeError	null
IOError	If the close fails.

5.60.2.2 def sparksee.RowWriter.write (*self, row*)

Writes the next row.

Parameters

row	[in] Row of data.

Exceptions

RuntimeError	null
IOError	If bad things happen during the write.

5.61 sparksee.ScriptParser Class Reference

ScriptParser.

Public Member Functions

- def __init__ (self)
  Constructor.
- def set_error_log (self, path)
  Sets the error log.
- def generate_schema_script (self, path, db)
  Writes an script with the schema definition for the given database.
- def parse (self, path, execute, locale_str)
  Parses the given input file.
- def set_output_log (self, path)
  Sets the output log.

5.61.1 Detailed Description

ScriptParser.

The ScriptParser can create schemas and load data from a set of commands in a sparksee script.

A SPARKSEE script contains an ordered list of commands. ScriptParser will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined SPARKSEE schema.

Check out the 'Scripting' chapter in the SPARKSEE User Manual for a comprehensive explanation on the grammar of the SPARKSEE commands and how they work.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.61.2 Member Function Documentation

5.61.2.1 def sparksee.ScriptParser.generate_schema_script (*self, path, db*)

Writes an script with the schema definition for the given database.

Parameters

path	[in] Filename of the script to be writen.
db	[in] Database.

Exceptions

IOError	If bad things happen opening or writing the file.

5.61.2.2 def sparksee.ScriptParser.parse (*self, path, execute, locale_str*)

Parses the given input file.

Parameters

path	[in] Input file path.
execute	[in] If TRUE the script is executed, if FALSE it is just parsed.
locale_str	[in] The locale string for reading the input file. See CSVReader.

Returns

TRUE if ok, FALSE in case of error.

Exceptions

IOError	If bad things happen opening the file.

5.61.2.3 def sparksee.ScriptParser.set_error_log (*self, path*)

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters

path	[in] Path of the error log.

Exceptions

IOError	If bad things happen opening the file.

5.61.2.4 def sparksee.ScriptParser.set_output_log (*self, path*)

Sets the output log.

If not set, output log corresponds to standard output.

Parameters

path	[in] Path of the output log.

Exceptions

IOError	If bad things happen opening the file.

5.62 sparksee.Session Class Reference

Session class.

Public Member Functions

- def rollback (self)
  Rollbacks a transaction.
- def pre_commit (self)
  PreCommits a transaction.
- def begin (self)
  Begins a transaction.
- def begin_update (self)
  Begins an update transaction.
- def new_query (self, lang)
  Creates a new Query.
- def commit (self)
  Commits a transaction.
- def get_graph (self)
  Gets the Graph instance.
- def close (self)
  Closes the Session instance.
- def new_objects (self)
  Creates a new Objects instance.
- def get_in_memory_pool_capacity (self)
  Gets the capacity of the in-memory pool.
- def is_closed (self)
  Gets if Session instance has been closed or not.

5.62.1 Detailed Description

Session class.

A Session is a stateful period of activity of a user with the Database.

All the manipulation of a Database must be enclosed into a Session. A Session can be initiated from a Database instance and allows for getting a Graph instance which represents the persistent graph (the graph database).

Also, temporary data is associated to the Session, thus when a Session is closed, all the temporary data associated to the Session is removed too. Objects or Values instances or even session attributes are an example of temporary data.

Moreover, a Session is exclusive for a thread, thus if it is shared among threads results may be fatal or unexpected.

Check out the 'Processing' and 'Transactions' sections in the SPARKSEE User Manual for details about how Sessions work and the use of transactions.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.62.2 Member Function Documentation

5.62.2.1 def sparksee.Session.close (*self*)

Closes the Session instance.

It must be called to ensure the integrity of all data.

5.62.2.2 def sparksee.Session.get_graph (*self*)

Gets the Graph instance.

Returns

The Graph instance.

5.62.2.3 def sparksee.Session.get_in_memory_pool_capacity (*self*)

Gets the capacity of the in-memory pool.

Returns

Returns the capacity of the in-memory pool

5.62.2.4 def sparksee.Session.is_closed (*self*)

Gets if Session instance has been closed or not.

See also

close()

Returns

TRUE if the Session instance has been closed, FALSE otherwise.

5.62.2.5 def sparksee.Session.new_objects (*self*)

Creates a new Objects instance.

Returns

The new Objects instance.

5.62.2.6 def sparksee.Session.new_query (*self, lang*)

Creates a new Query.

Parameters

lang	The query language to create the query for

5.62.2.7 def sparksee.Session.pre_commit (*self*)

PreCommits a transaction.

YOU SHOULD NOT USE THIS METHOD. This method exists for a specific service and it is used to force that the transaction is written in the recovery log even though it had not been committed yet (and may never be).

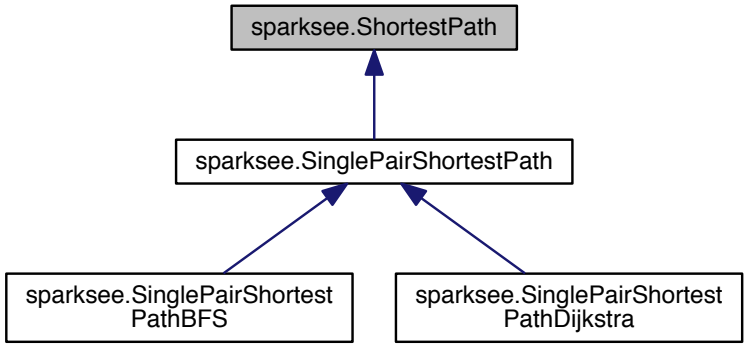
Returns

The transaction id. This has to be used in RedoPrecommited in case of a recovery process.

5.63 sparksee.ShortestPath Class Reference

ShortestPath class.

Inheritance diagram for sparksee.ShortestPath:



Public Member Functions

- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def run (self)

  Runs the algorithm.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.

- def exclude_edges (self, edges)

  Set which edges can't be used.
- def close (self)

  Closes the ShortestPath instance.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def is_closed (self)

  Gets if ShortestPath has been closed or not.

5.63.1 Detailed Description

ShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph.

The user must set which node and edge types can be used for the traversal.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.63.2 Member Function Documentation

5.63.2.1 def sparksee.ShortestPath.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.63.2.2 def sparksee.ShortestPath.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.63.2.3 def sparksee.ShortestPath.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	null

5.63.2.4 def sparksee.ShortestPath.close (*self*)

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

5.63.2.5 def sparksee.ShortestPath.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.63.2.6 def sparksee.ShortestPath.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.63.2.7 def sparksee.ShortestPath.is_closed (*self*)

Gets if ShortestPath has been closed or not.

See also

close()

Returns

TRUE if the ShortestPath instance has been closed, FALSE otherwise.

5.63.2.8 def sparksee.ShortestPath.run (*self*)

Runs the algorithm.

This method can only be called once.

5.63.2.9 def sparksee.ShortestPath.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

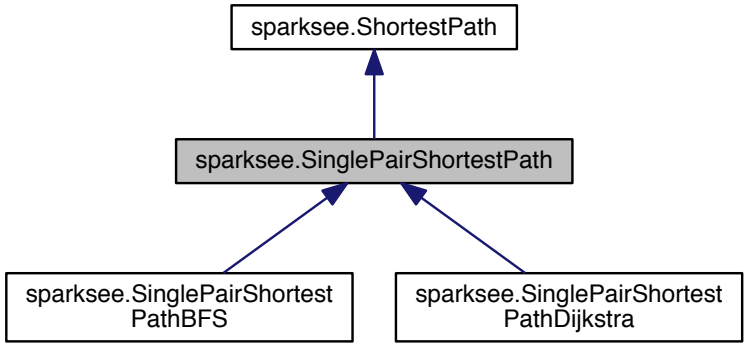
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

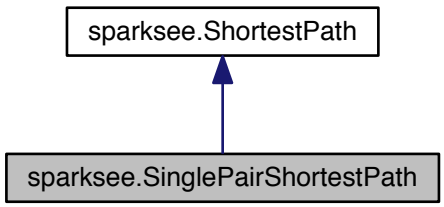
5.64 sparksee.SinglePairShortestPath Class Reference

SinglePairShortestPath class.

Inheritance diagram for sparksee.SinglePairShortestPath:



Collaboration diagram for sparksee.SinglePairShortestPath:



Public Member Functions

- def run (self)

  Runs the algorithm.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.

- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def get_path_as_nodes (self)

  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- def get_path_as_edges (self)

  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def exists (self)

  Returns TRUE If a path exists or FALSE otherwise.
- def get_cost (self)

  Gets the cost of the shortest path.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def close (self)

  Closes the ShortestPath instance.
- def is_closed (self)

  Gets if ShortestPath has been closed or not.

5.64.1 Detailed Description

SinglePairShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph from a given source node and to a given destination node.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.64.2 Member Function Documentation

5.64.2.1 def sparksee.SinglePairShortestPath.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.64.2.2 def sparksee.SinglePairShortestPath.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.64.2.3 def sparksee.SinglePairShortestPath.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	null

5.64.2.4 def sparksee.ShortestPath.close (*self*) [inherited]

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

5.64.2.5 def sparksee.SinglePairShortestPath.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.64.2.6 def sparksee.SinglePairShortestPath.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.64.2.7 def sparksee.SinglePairShortestPath.get_cost (*self*)

Gets the cost of the shortest path.

The cost for unweighted algorithms is the number of hops of the shortest path. For weighted algoritms, the cost is the sum of the costs of the edges of the shortest path.

Returns

The cost of the shortest path.

5.64.2.8 def sparksee.SinglePairShortestPath.get_path_as_edges (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

5.64.2.9 def sparksee.SinglePairShortestPath.get_path_as_nodes (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

5.64.2.10 def sparksee.ShortestPath.is_closed (*self*) [inherited]

Gets if ShortestPath has been closed or not.

See also

close()

Returns

TRUE if the ShortestPath instance has been closed, FALSE otherwise.

5.64.2.11 def sparksee.SinglePairShortestPath.run (*self*)

Runs the algorithm.

This method can only be called once.

5.64.2.12 def sparksee.SinglePairShortestPath.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

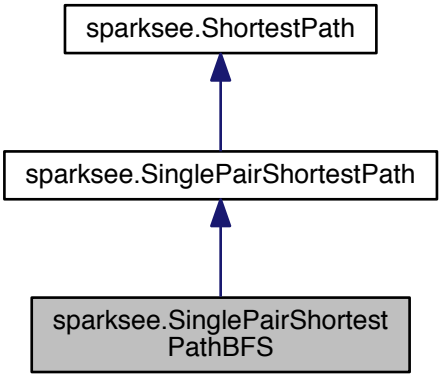
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

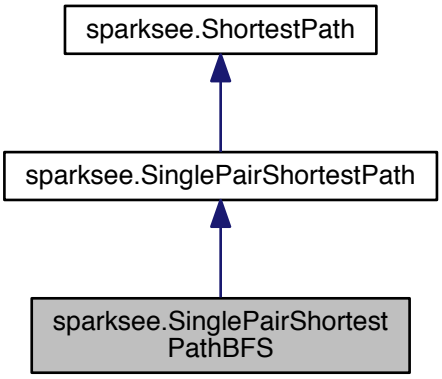
5.65 sparksee.SinglePairShortestPathBFS Class Reference

SinglePairShortestPathBFS class.

Inheritance diagram for sparksee.SinglePairShortestPathBFS:



Collaboration diagram for sparksee.SinglePairShortestPathBFS:



Public Member Functions

- def exclude_nodes (self, nodes)
  Set which nodes can't be used.
- def get_cost (self)
  Gets the cost of the shortest path.
- def run (self)
  Executes the algorithm.
- def add_all_edge_types (self, dir)
  Allows for traversing all edge types of the graph.
- def exclude_edges (self, edges)
  Set which edges can't be used.
- def __init__ (self, session, src, dst)
  Creates a new instance.
- def add_edge_type (self, type, dir)
  Allows for traversing edges of the given type.
- def get_path_as_nodes (self)
  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- def add_node_type (self, type)
  Allows for traversing nodes of the given type.
- def exists (self)
  Returns TRUE If a path exists or FALSE otherwise.
- def check_only_existence (self)
  Set that only the path existence must be calculated and not the path itself.
- def get_path_as_edges (self)
  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- def set_maximum_hops (self, maxhops)
  Sets the maximum hops restriction.
- def add_all_node_types (self)
  Allows for traversing all node types of the graph.
- def close (self)
  Closes the ShortestPath instance.
- def is_closed (self)
  Gets if ShortestPath has been closed or not.

5.65.1 Detailed Description

SinglePairShortestPathBFS class.

It solves the single-pair shortest path problem using a BFS-based implementation.

It is a unweighted algorithm, that is it assumes all edges have the same cost.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.65.2 Constructor & Destructor Documentation

5.65.2.1 def sparksee.SinglePairShortestPathBFS.__init__ (*self, session, src, dst*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
src	[in] Source node.
dst	[dst] Destination node.

5.65.3 Member Function Documentation

5.65.3.1 def sparksee.SinglePairShortestPathBFS.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.65.3.2 def sparksee.SinglePairShortestPathBFS.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.65.3.3 def sparksee.SinglePairShortestPathBFS.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	null

5.65.3.4 def sparksee.SinglePairShortestPathBFS.check_only_existence (*self*)

Set that only the path existence must be calculated and not the path itself.

That method should improve the performance of the algorithm, but a call to GetPathAsNodes or GetPathAsEdges will generate an exception even if the path exists.

5.65.3.5 def sparksee.ShortestPath.close (*self*) [inherited]

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

5.65.3.6 def sparksee.SinglePairShortestPathBFS.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.65.3.7 def sparksee.SinglePairShortestPathBFS.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.65.3.8 def sparksee.SinglePairShortestPathBFS.get_cost (*self*)

Gets the cost of the shortest path.

The cost is the number of hops of the shortest path.

Returns

The cost of the shortest path.

5.65.3.9 def sparksee.SinglePairShortestPathBFS.get_path_as_edges (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

5.65.3.10 def sparksee.SinglePairShortestPathBFS.get_path_as_nodes (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

5.65.3.11 def sparksee.ShortestPath.is_closed (*self*) [inherited]

Gets if ShortestPath has been closed or not.

See also

close()

Returns

TRUE if the ShortestPath instance has been closed, FALSE otherwise.

5.65.3.12 def sparksee.SinglePairShortestPathBFS.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

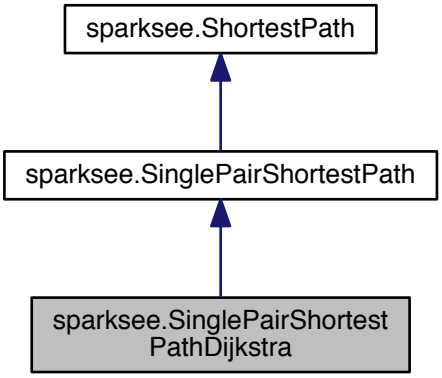
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

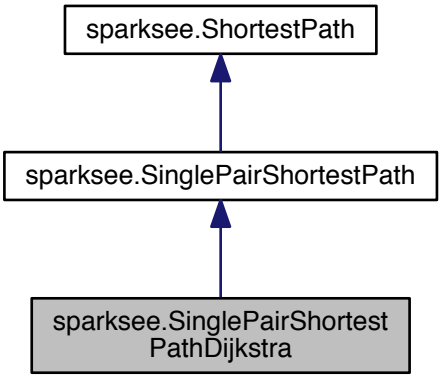
5.66 sparksee.SinglePairShortestPathDijkstra Class Reference

SinglePairShortestPathDijkstra class.

Inheritance diagram for sparksee.SinglePairShortestPathDijkstra:



Collaboration diagram for sparksee.SinglePairShortestPathDijkstra:



Public Member Functions

- def __init__ (self, session, src, dst)
  Creates a new instance.
- def exclude_nodes (self, nodes)
  Set which nodes can't be used.
- def get_cost (self)
  Gets the cost of the shortest path.
- def run (self)
  Executes the algorithm.
- def add_all_edge_types (self, dir)
  Allows for traversing all edge types of the graph.
- def exclude_edges (self, edges)
  Set which edges can't be used.
- def add_edge_type (self, type, dir)
  Allows for traversing edges of the given type.
- def set_unweighted_edge_cost (self, weight)
  Sets the weight assigned to the unweighted edges.
- def get_path_as_nodes (self)
  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- def set_dynamic_edge_cost_callback (self, dyn_cost_calculator)
  Set a class callback to dynamically calculate the cost of the edges.
- def add_node_type (self, type)
  Allows for traversing nodes of the given type.
- def exists (self)
  Returns TRUE If a path exists or FALSE otherwise.
- def get_path_as_edges (self)
  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- def add_weighted_edge_type (self, type, dir, attr)
  Allows for traversing edges of the given type using the given attribute as the weight.
- def set_maximum_hops (self, maxhops)
  Sets the maximum hops restriction.
- def add_all_node_types (self)
  Allows for traversing all node types of the graph.
- def close (self)
  Closes the ShortestPath instance.
- def is_closed (self)
  Gets if ShortestPath has been closed or not.

5.66.1 Detailed Description

SinglePairShortestPathDijkstra class.

It solves the single-pair shortest path problem using a Dijkstra-based implementation.

It is a weighted algorithm, so it takes into account the cost of the edges to compute a minimum-cost shortest path. That is, the user may set for each edge type which attribute should be used to retrieve the cost of the edge. If no attribute is given for an edge type, this will assume the edge has a fixed cost (the default is 1). Only numerical attribute can be set as weight attributes (that is Long, Integer or Double attributes are allowed).

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.66.2 Constructor & Destructor Documentation

5.66.2.1 def sparksee.SinglePairShortestPathDijkstra.__init__ (*self, session, src, dst*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
src	[in] Source node.
dst	[dst] Destination node.

5.66.3 Member Function Documentation

5.66.3.1 def sparksee.SinglePairShortestPathDijkstra.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.66.3.2 def sparksee.SinglePairShortestPathDijkstra.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.66.3.3 def sparksee.SinglePairShortestPathDijkstra.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	null

5.66.3.4 def sparksee.SinglePairShortestPathDijkstra.add_weighted_edge_type (*self, type, dir, attr*)

Allows for traversing edges of the given type using the given attribute as the weight.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.
attr	[in] Attribute to be used as the weight. It must be a global attribute or an attribute of the given edge type.

5.66.3.5 def sparksee.ShortestPath.close (*self*) [inherited]

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

5.66.3.6 def sparksee.SinglePairShortestPathDijkstra.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.66.3.7 def sparksee.SinglePairShortestPathDijkstra.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.66.3.8 def sparksee.SinglePairShortestPathDijkstra.get_cost (*self*)

Gets the cost of the shortest path.

The cost is the sum of the weights of the edges in the shortest path.

Returns

The cost of the shortest path.

5.66.3.9 def sparksee.SinglePairShortestPathDijkstra.get_path_as_edges (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

5.66.3.10 def sparksee.SinglePairShortestPathDijkstra.get_path_as_nodes (*self*)

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

5.66.3.11 def sparksee.ShortestPath.is_closed (*self*) [inherited]

Gets if ShortestPath has been closed or not.

See also

close()

Returns

TRUE if the ShortestPath instance has been closed, FALSE otherwise.

5.66.3.12 def sparksee.SinglePairShortestPathDijkstra.set_dynamic_edge_cost_callback (*self, dyn_cost_calculator*)

Set a class callback to dynamically calculate the cost of the edges.

The callback can be set to NULL (the default) to use the normal attribute based cost weights. The given class must be kept alive by the user for as long as the algorithm is running.

Parameters

dyn_cost_calculator	[in] Class callback to calculate the edge costs

5.66.3.13 def sparksee.SinglePairShortestPathDijkstra.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.66.3.14 def sparksee.SinglePairShortestPathDijkstra.set_unweighted_edge_cost (*self, weight*)

Sets the weight assigned to the unweighted edges.

All the edges from the types added without an explicit weight attribute will get this weight. The default weight for this edges is 1.

Parameters

weight	[in] The weight value for unweighted edges.

5.67 sparksee.SinglePairShortestPathDijkstraDynamicCost Class Reference

Defines how to calculate an edge weight.

Public Member Functions

- def calculate_edge_cost (self, source_node, source_cost, source_level, target_node, edge, edge_weight_↩ attr)

  Dynamic calculation of an edge weight.

5.67.1 Detailed Description

Defines how to calculate an edge weight.

This is an interface which must be implemented by the user. While the algorithm is running, one or more calls for each edge that can be in the shortest path will be issued to get the weight of the edge that could change based on the predecessor node and the current minimum path from the source.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.67.2 Member Function Documentation

5.67.2.1 def sparksee.SinglePairShortestPathDijkstraDynamicCost.calculate_edge_cost (*self, source_node, source_cost, source_level, target_node, edge, edge_weight_attr*)

Dynamic calculation of an edge weight.

This method will be called several times during the algorithm execution. It can be called for the same edge multiple times because the path to the source node (and as a consequence the source weight and levels) might change while the algorithm is running. This method will have the normal default implementation that only uses the specified edge attribute to get the weight or the unweighted edge cost if the edge doesn't have the specified cost attribute. But it can be overriden in a derived user class to implement a dinamic edge weight, that may need the sourceCost and sourceLevels to calculate the weight.

Parameters

source_node	[in] The current node
source_cost	[in] Current total cost up to the source node
source_level	[in] Current path size up to the source node
target_node	[in] The next node
edge	[in] The edge to the next node that this methos must weight
edge_weight_attr	[in] The attribute that stores the edge weight or InvalidAttribute

Returns

Returns the current weight of the edge (>=0) or <0 if the edge can not be used.

5.68 sparksee.Sparksee Class Reference

Sparksee class.

Public Member Functions

- def restore_encrypted_backup (self, path, backup_file, config, key_in_hex, iv_in_hex)

  Restores a Database from an encrypted backup file.
- def resize_in_memory_pool (self, new_capacity)

  Resizes the in memory pool allocator.
- def create (self, path, alias, config)

  Creates a new Database instance.
- def verify_checksums (self, path)

  Verifies the Checksum integrity of the given image file and it's recovery log file.
- def remove_checksums (self, path)

  Converts a database WITH checksums into a database WITHOUT checksums.
- def restore (self, path, backup_file)

  Restores a Database from a backup file.
- def add_checksums (self, path)

  Converts a database WITHOUT checksums into a database WITH checksums.
- def restore_encrypted_backup (self, path, backup_file, key_in_hex, iv_in_hex)

  Restores a Database from an encrypted backup file.
- def decrypt (self, path)

  Converts a database WITH encryption into a database WITHOUT encryption.
- def restore (self, path, backup_file, config)

  Restores a Database from a backup file.
- def create (self, path, alias)

  Creates a new Database instance.
- def open (self, path, read_only)

  Opens an existing Database instance.
- def __init__ (self, config)

  Creates a new instance.
- def open (self, path, read_only, config)

  Opens an existing Database instance.
- def set_unrecoverable_error_callback (self, callback)

  Sets the unrecoverable error callback.
- def encrypt (self, path)

  Converts a database WITHOUT encryption into an encrypted database.
- def close (self)

  Closes the Sparksee instance.
- def get_in_memory_pool_capacity (self)

  Gets the capacity of the in-memory pool (in Megabytes)

Static Public Attributes

- float VERSION = 5.0

  Sparksee version.

5.68.1 Detailed Description

Sparksee class.

All Sparksee programs must have one single Sparksee instance to manage one or more Database instances.

This class allows for the creation of new Databases or open an existing one.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.68.2 Constructor & Destructor Documentation

5.68.2.1 def sparksee.Sparksee.__init__ (*self, config*)

Creates a new instance.

Parameters

config	[in/out] Sparksee configuration (may be updated).

5.68.3 Member Function Documentation

5.68.3.1 def sparksee.Sparksee.add_checksums (*self, path*)

Converts a database WITHOUT checksums into a database WITH checksums.

Any error found is logged. The database (and it's recovery log if present) does NOT have checksums. sparksee↩
::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the checksum could be added or false otherwise.

Exceptions

RuntimeError	null
IOError	null

5.68.3.2 def sparksee.Sparksee.close (*self*)

Closes the Sparksee instance.

It must be called to ensure the integrity of all data.

5.68.3.3 def sparksee.Sparksee.create (*self, path, alias, config*)

Creates a new Database instance.

Parameters

path	[in] Database storage file.
alias	[in] Database alias name.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created.

Referenced by sparksee.Sparksee.create().

5.68.3.4 def sparksee.Sparksee.create (*self, path, alias*)

Creates a new Database instance.

Parameters

path	[in] Database storage file.
alias	[in] Database alias name.

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created.

References sparksee.Sparksee.create().

5.68.3.5 def sparksee.Sparksee.decrypt (*self, path*)

Converts a database WITH encryption into a database WITHOUT encryption.

Any error found is logged. The database (and it's recovery log if present) is encrypted. The current encryption had been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the database could be unencrypted. The errors are returned with an exception.

Exceptions

RuntimeError	null
IOError	null

5.68.3.6 def sparksee.Sparksee.encrypt (*self, path*)

Converts a database WITHOUT encryption into an encrypted database.

Any error found is logged. The database (and it's recovery log if present) does is NOT encrypted. The encryption had already been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the database could be encrypted. The errors are returned with an exception.

Exceptions

RuntimeError	null
IOError	null

5.68.3.7 def sparksee.Sparksee.get_in_memory_pool_capacity (*self*)

Gets the capacity of the in-memory pool (in Megabytes)

Returns

Returns the capacity of the in memory pool

5.68.3.8 def sparksee.Sparksee.open (*self, path, read_only*)

Opens an existing Database instance.

Parameters

path	[in] Database storage file.
read_only	[in] If TRUE, open Database in read-only mode.

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file does not exist.

Referenced by sparksee.Sparksee.open().

5.68.3.9 def sparksee.Sparksee.open (*self, path, read_only, config*)

Opens an existing Database instance.

The provided configuration will be used for all the database settings but not for the license, which must already be properly setup in this class.

Parameters

path	[in] Database storage file.
read_only	[in] If TRUE, open Database in read-only mode.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file does not exist.

References sparksee.Sparksee.open().

5.68.3.10 def sparksee.Sparksee.remove_checksums (*self, path*)

Converts a database WITH checksums into a database WITHOUT checksums.

Any error found is logged. The database (and it's recovery log if present) have checksums. sparksee::gdb::File↩ NotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the checksum could be removed. The errors are returned with an exception.

Exceptions

RuntimeError	null
IOError	null

5.68.3.11 def sparksee.Sparksee.resize_in_memory_pool (*self, new_capacity*)

Resizes the in memory pool allocator.

Parameters

new_capacity	The new capacity of the in memory pool allocator

Returns

Returns true if was successful

5.68.3.12 def sparksee.Sparksee.restore (*self, path, backup_file*)

Restores a Database from a backup file.

See the Graph class Backup method.

Parameters

path	[in] Database storage file.
backup_file	[in] The Backup file to be restored.

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created, or the exported data file does not exists.

Referenced by sparksee.Sparksee.restore().

5.68.3.13 def sparksee.Sparksee.restore (*self, path, backup_file, config*)

Restores a Database from a backup file.

See the Graph class Backup method.

Parameters

path	[in] Database storage file.
backup_file	[in] The Backup file to be restored.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created, or the exported data file does not exists.

References sparksee.Sparksee.restore().

5.68.3.14 def sparksee.Sparksee.restore_encrypted_backup (*self, path, backup_file, config, key_in_hex, iv_in_hex*)

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters

path	[in] Database storage file.
backup_file	[in] The Backup file to be restored.
config	[in] Use a specific configuration instead of the one in this Sparksee
key_in_hex	[In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
iv_in_hex	[In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created, or the exported data file does not exists.

Referenced by sparksee.Sparksee.restore_encrypted_backup().

5.68.3.15 def sparksee.Sparksee.restore_encrypted_backup (*self, path, backup_file, key_in_hex, iv_in_hex*)

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters

path	[in] Database storage file.
backup_file	[in] The Backup file to be restored.
key_in_hex	[In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
iv_in_hex	[In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns

A Database instance.

Exceptions

RuntimeError	null
IOError	If the given file cannot be created, or the exported data file does not exists.

References sparksee.Sparksee.restore_encrypted_backup().

5.68.3.16 def sparksee.Sparksee.set_unrecoverable_error_callback (*self, callback*)

Sets the unrecoverable error callback.

Parameters

callback	The unrecoverable error callback to set

5.68.3.17 def sparksee.Sparksee.verify_checksums (*self, path*)

Verifies the Checksum integrity of the given image file and it's recovery log file.

The main database file checksums are always checked.

When the recovery log file exists, the recovery log file is also checked for checksum and format errors.

When the checksums of the main file or the recovery file are incorrect, the funcion returns false, in any other error it throws an exception.

Any error found is logged as a WARNING.

sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the image file

Returns

Returns true if the checksum verification succeeded. Returns false if an invalid checksum was detected.

Exceptions

RuntimeError	null
IOError	null

5.69 sparksee.SparkseeConfig Class Reference

Sparksee configuration class.

Public Member Functions

- def set_recovery_enabled (self, status)

  Enables or disables the recovery.
- def set_license_pre_download_days (self, days)

  Start trying to automatically download a new license at the specified number of days before expiration.
- def get_extent_pages (self)

  Gets the number of pages per extent.
- def save_all (self)

  Save all the current configuration settings in the specified config file.
- def get_a_e_s_key (self)

  Get the AES encryption key in a hexadecimal encoded string.
- def get_pool_persistent_max_size (self)

  Gets the maximum size for the persistent pool in number of frames.
- def get_pool_temporary_min_size (self)

  Gets the minimum size for the temporary pool in number of frames.
- def get_call_stack_dump (self)

  Gets whether the signals will be captured to dump the call stack or not.
- def set_recovery_cache_max_size (self, extents)

  Sets the maximum size for the recovery log cache in extents.
- def set_a_e_s_encryption_enabled (self, key_in_hex, iv_in_hex)

  Enables storage encryption using AES with the given key and iv.
- def get_high_availability_enabled (self)

  Gets whether high availability mode is enabled or disabled.
- def get_pool_persistent_min_size (self)

  Gets the minimum size for the persistent pool in number of frames.
- def set_checksum_enabled (self, status)

  Enables or disables the storage checksum usage.
- def set_pool_persistent_min_size (self, frames)

  Sets the minimum size for the persistent pool in number of frames.
- def set_cache_statistics_enabled (self, status)

  Enables or disables cache statistics.
- def set_recovery_checkpoint_time (self, micro_seconds)

  Sets the delay time (in microseconds) between automatic checkpoints.
- def set_encryption_disabled (self)

  Disables storage encryption.
- def set_recovery_log_file (self, file_path)

  Sets the recovery log file.
- def set_pool_frame_size (self, extents)

  Sets the size of a pool frame in number of extents.
- def get_recovery_log_file (self)

  Gets the recovery log file.
- def set_high_availability_enabled (self, status)

  Enables or disables high availability mode.
- def get_cache_statistics_snapshot_time (self)

  Gets the cache statistics snapshot time in microseconds.
- def get_extent_size (self)

  Gets the size of a extent.
- def get_sparksee_config_file (self)

  Gets the config file path.
- def set_extent_size (self, k_bytes)

Sets the size of the extents in KB.

- def set_high_availability_i_p (self, ip)

  Sets the IP address and port of the instance.
- def get_pool_temporary_max_size (self)

  Gets the maximum size for the temporary pool in number of frames.
- def set_in_mem_alloc_size (self, size)

  Sets the in-memory allocator size.
- def __init__ (self, path)

  Creates a new instance with a specific config file.
- def set_high_availability_master_history (self, file_path)

  Sets the master's history log.
- def set_tmp_enabled (self, enabled)

  Sets whether to use temporary storage for computations or not.
- def get_log_file (self)

  Gets the log file.
- def get_tmp_enabled (self)

  Gets whether using temporary storage for computations is enabled.
- def get_pool_frame_size (self)

  Gets the size of a pool frame in number of extents.
- def get_cache_statistics_enabled (self)

  Gets whether cache statistics are enabled or disabled.
- def set_high_availability_coordinators (self, ip)

  Sets the coordinators address and port list.
- def set_high_availability_synchronization (self, micro_seconds)

  Sets the synchronization polling time.
- def get_client_id (self)

  Gets the client identifier.
- def get_cache_statistics_file (self)

  Gets the cache statistics log file.
- def set_sparksee_config_file (self, path)

  Sets the config file path.
- def save (self)

  Save the current configuration in the specified config file.
- def set_pool_persistent_max_size (self, frames)

  Sets the maximum size for the persistent pool in number of frames.
- def get_log_level (self)

  Gets the log level.
- def set_license (self, key)

  Sets the license key.
- def get_download_status (self)

  Gets a message with the license download result.
- def set_license_id (self, license_id)

  Set the license identifier.
- def get_license (self)

  Gets the license key.
- def get_high_availability_i_p (self)

  Gets the IP address and port of the instance.
- def set_a_e_s_encryption_enabled (self, key_size)

  Enables storage encryption using AES and GENARATES a key and an iv.
- def get_in_mem_alloc_size (self)

  Gets the in-memory allocator size.

- def get_license_pre_download_days (self)

  Get the number of days before expiration when a new license will be downloaded.
- def set_call_stack_dump (self, status)

  Sets whether the signals will be captured to dump the call stack or not.
- def download_license (self)

  Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).
- def get_recovery_enabled (self)

  Gets whether the recovery is enabled or disabled.
- def get_license_request (self)

  Get information useful to manually request a license.
- def set_cache_statistics_file (self, file_path)

  Sets the cache statistics log file.
- def get_tmp_folder (self)

  Gets the temporary folder used for temporary staging.
- def get_pool_partitions (self)

  Gets the number of partitions in each PartitionedPool.
- def get_high_availability_coordinators (self)

  Gets the coordinators address and port list.
- def get_recovery_checkpoint_time (self)

  Gets the delay time (in microseconds) between automatic checkpoints.
- def set_log_file (self, file_path)

  Sets the log file.
- def set_extent_pages (self, pages)

  Sets the number of pages per extent.
- def set_client_id (self, client_id)

  Set the client identifier.
- def set_log_level (self, level)

  Sets the log level.
- def get_recovery_cache_max_size (self)

  Gets the maximum size for the recovery log cache in extents.
- def get_checksum_enabled (self)

  Gets whether the storage checksum usage is enabled or disabled.
- def get_high_availability_synchronization (self)

  Gets the synchronization polling time.
- def get_high_availability_master_history (self)

  Gets the master's history log.
- def get_license_id (self)

  Gets the license identifier.
- def __init__ (self, path, client_id, license_id)

  Creates a new instance with a specific config file and IDs.
- def get_a_e_s_i_v (self)

  Get the AES initialization vector in a hexadecimal encoded string.
- def get_encryption_enabled (self)

  Gets whether the storage encryption is enabled or disabled.
- def set_pool_partitions (self, pools)

  Sets the number of pools in each PartitionedPool.
- def get_rollback_enabled (self)

  Gets whether the rollback is enabled or disabled.
- def set_pool_temporary_max_size (self, frames)

  Sets the maximum size for the temporary pool in number of frames.

- def get_cache_max_size (self)

  Gets the maximum size for the cache (all pools) in MB.
- def set_rollback_enabled (self, status)

  Enables or disables the rollback.
- def set_tmp_folder (self, tmp_folder)

  Sets the temporary folder used for temporary staging.
- def set_cache_statistics_snapshot_time (self, micro_seconds)

  Sets the cache statistics snapshot time.
- def set_cache_max_size (self, mega_bytes)

  Sets the maximum size for the cache (all pools) in MB.
- def __init__ (self)

  Creates a new instance.
- def set_pool_temporary_min_size (self, frames)

  Sets the minimum size for the temporary pool in number of frames.
- def download_expected (self)

  Check if a new license will be automatically downloaded with the current settings.

5.69.1 Detailed Description

Sparksee configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overrided with values from a properties file (see SparkseeProperties class). Also, this settings can be overrided calling a specific setter.

For each field, it is shown its default value and the property to override this value:

Extent size: 4KB ('sparksee.storage.extentsize' at SparkseeProperties).

Pages per extent: 1 page ('sparksee.storage.extentpages' at SparkseeProperties).

Checksums enabled: true ('sparksee.storage.checksum' at SparkseeProperties).

Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at SparkseeProperties).

Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at SparkseeProperties).

Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at SparkseeProperties).

Minimum size for the temporary pool: 16 frames ('sparksee.io.pool.temporal.minsize' at SparkseeProperties).

Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at SparkseeProperties).

Number of pools in the pool cluster: 0 pools ('sparksee.io.pool.clustersize' at SparkseeProperties). 0 or 1 means the clustering is disabled.

Maximum size for the cache (all pools): 0 MB ('sparksee.io.cache.maxsize' at SparkseeProperties).

License code: "" ('sparksee.license' at SparkseeProperties). No license code means evaluation license.

Log level: Info ('sparksee.log.level' at SparkseeProperties).

Log file: "sparksee.log" ('sparksee.log.file' at SparkseeProperties).

Cache statistics: false (disabled) ('sparksee.cache.statistics' at SparkseeProperties).

Cache statistics log file: "statistics.log" ('sparksee.cache.statisticsFile' at SparkseeProperties).

Cache statistics snapshot time: 1000 msecs [TimeUnit] ('sparksee.cache.statisticsSnapshotTime' at Sparksee↩ Properties).

Recovery enabled: false ('sparksee.io.recovery' at SparkseeProperties).

Recovery log file: "" ('sparksee.io.recovery.logfile' at SparkseeProperties).

Recovery cache max size: 1MB ('sparksee.io.recovery.cachesize' at SparkseeProperties).

Recovery checkpoint time: 60 seconds [TimeUnit] ('sparksee.io.recovery.checkpointTime' at SparkseeProperties).

High-availability: false (disabled) ('sparksee.ha' at SparkseeProperties).

High-availability coordinators: "" ('sparksee.ha.coordinators' at SparkseeProperties).

High-availability IP: "" ('sparksee.ha.ip' at SparkseeProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] ('sparksee.ha.sync' at SparkseeProperties).

High-availability master history: 1D (1 day) [TimeUnit] ('sparksee.ha.master.history' at SparkseeProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

<X>[D|H|M|S|s|m|u]

where <X> is a number followed by an optional character which represents the unit: D for days, H for hours, M for minutes, S or s for seconds, m for milliseconds and u for microseconds. If no unit character is given, seconds are assumed.

Capture abort signals to dump the call stack ('sparksee.callstackdump' at SparkseeProperties) is enabled by default on most platforms.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.69.2 Constructor & Destructor Documentation

5.69.2.1 def sparksee.SparkseeConfig.__init__ (*self, path*)

Creates a new instance with a specific config file.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.

Referenced by sparksee.SparkseeConfig.__init__().

5.69.2.2 def sparksee.SparkseeConfig.__init__ (*self, path, client_id, license_id*)

Creates a new instance with a specific config file and IDs.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the config file already exists, the client and license ids should have the same values as the given arguments. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.
client_id	[in] The client identifier.
license_id	[in] The license identifier.

References sparksee.SparkseeConfig.__init__().

5.69.2.3 def sparksee.SparkseeConfig.__init__ (*self*)

Creates a new instance.

Values are set with default values.

References sparksee.SparkseeConfig.__init__().

5.69.3 Member Function Documentation

5.69.3.1 def sparksee.SparkseeConfig.download_license (*self*)

Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).

Returns

Returns -1 if a valid license key couldn't be downloaded, 0 if the same license was downloaded or 1 if a different license is downloaded.

5.69.3.2 def sparksee.SparkseeConfig.get_a_e_s_i_v (*self*)

Get the AES initialization vector in a hexadecimal encoded string.

Returns

The AES initialization vector as an hexadecimal string.

5.69.3.3 def sparksee.SparkseeConfig.get_a_e_s_key (*self*)

Get the AES encryption key in a hexadecimal encoded string.

Returns

The AES encryption Key as an hexadecimal string.

5.69.3.4 def sparksee.SparkseeConfig.get_cache_max_size (*self*)

Gets the maximum size for the cache (all pools) in MB.

Returns

The maximum size for the cache (all pools) in MB.

5.69.3.5 def sparksee.SparkseeConfig.get_cache_statistics_enabled (*self*)

Gets whether cache statistics are enabled or disabled.

Returns

TRUE if cache statistics are enabled, FALSE otherwise.

5.69.3.6 def sparksee.SparkseeConfig.get_cache_statistics_file (*self*)

Gets the cache statistics log file.

Useless if cache statistics are disabled.

Returns

The cache statistics log file.

5.69.3.7 def sparksee.SparkseeConfig.get_cache_statistics_snapshot_time (*self*)

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

Returns

The cache statistics snapshot time in microseconds.

5.69.3.8 def sparksee.SparkseeConfig.get_call_stack_dump (*self*)

Gets whether the signals will be captured to dump the call stack or not.

Returns

TRUE if the signals must be captured, FALSE otherwise.

5.69.3.9 def sparksee.SparkseeConfig.get_checksum_enabled (*self*)

Gets whether the storage checksum usage is enabled or disabled.

Returns

TRUE if the checksum is enabled, FALSE otherwise.

5.69.3.10 def sparksee.SparkseeConfig.get_client_id (*self*)

Gets the client identifier.

Returns

The client identifier.

5.69.3.11 def sparksee.SparkseeConfig.get_download_status (*self*)

Gets a message with the license download result.

It can be used when the DownloadLicense call failed.

Returns

The log download result.

5.69.3.12 def sparksee.SparkseeConfig.get_encryption_enabled (*self*)

Gets whether the storage encryption is enabled or disabled.

Returns

TRUE if the encryption is enabled, FALSE otherwise.

5.69.3.13 def sparksee.SparkseeConfig.get_extent_pages (*self*)

Gets the number of pages per extent.

Returns

The number of pages per extent.

5.69.3.14 def sparksee.SparkseeConfig.get_extent_size (*self*)

Gets the size of a extent.

Returns

The size of a extent in KB.

5.69.3.15 def sparksee.SparkseeConfig.get_high_availability_coordinators (*self*)

Gets the coordinators address and port list.

Returns

The coordinators address and port list.

5.69.3.16 def sparksee.SparkseeConfig.get_high_availability_enabled (*self*)

Gets whether high availability mode is enabled or disabled.

Returns

TRUE if high availability mode is enabled, FALSE otherwise.

5.69.3.17 def sparksee.SparkseeConfig.get_high_availability_i_p (*self*)

Gets the IP address and port of the instance.

Returns

The IP address and port of the instance.

5.69.3.18 def sparksee.SparkseeConfig.get_high_availability_master_history (*self*)

Gets the master's history log.

Returns

The master's history log.

5.69.3.19 def sparksee.SparkseeConfig.get_high_availability_synchronization (*self*)

Gets the synchronization polling time.

Returns

The Synchronization polling time.

5.69.3.20 def sparksee.SparkseeConfig.get_in_mem_alloc_size (*self*)

Gets the in-memory allocator size.

Returns

Returns the in-memory allocator size

5.69.3.21 def sparksee.SparkseeConfig.get_license (*self*)

Gets the license key.

Returns

The license key.

5.69.3.22 def sparksee.SparkseeConfig.get_license_id (*self*)

Gets the license identifier.

Returns

The license identifier.

5.69.3.23 def sparksee.SparkseeConfig.get_license_pre_download_days (*self*)

Get the number of days before expiration when a new license will be downloaded.

Returns

Returns the number of days or -1 if it will never be automatically downloaded.

5.69.3.24 def sparksee.SparkseeConfig.get_log_file (*self*)

Gets the log file.

Returns

The log file.

5.69.3.25 def sparksee.SparkseeConfig.get_log_level (*self*)

Gets the log level.

Returns

The LogLevel.

5.69.3.26 def sparksee.SparkseeConfig.get_pool_frame_size (*self*)

Gets the size of a pool frame in number of extents.

Returns

The size of a pool frame in number of extents.

5.69.3.27 def sparksee.SparkseeConfig.get_pool_partitions (*self*)

Gets the number of partitions in each PartitionedPool.

Returns

The number of partitions in each PartitionedPool.

5.69.3.28 def sparksee.SparkseeConfig.get_pool_persistent_max_size (*self*)

Gets the maximum size for the persistent pool in number of frames.

Returns

The maximum size for the persistent pool in number of frames.

5.69.3.29 def sparksee.SparkseeConfig.get_pool_persistent_min_size (*self*)

Gets the minimum size for the persistent pool in number of frames.

Returns

The minimum size for the persistent pool in number of frames.

5.69.3.30 def sparksee.SparkseeConfig.get_pool_temporary_max_size (*self*)

Gets the maximum size for the temporary pool in number of frames.

Returns

The maximum size for the temporary pool in number of frames.

5.69.3.31 def sparksee.SparkseeConfig.get_pool_temporary_min_size (*self*)

Gets the minimum size for the temporary pool in number of frames.

Returns

The minimum size for the temporary pool in number of frames.

5.69.3.32 def sparksee.SparkseeConfig.get_recovery_cache_max_size (*self*)

Gets the maximum size for the recovery log cache in extents.

Returns

The maximum size for the recovery log cache in extents.

5.69.3.33 def sparksee.SparkseeConfig.get_recovery_checkpoint_time (*self*)

Gets the delay time (in microseconds) between automatic checkpoints.

Returns

The delay time (in microseconds) between automatic checkpoints.

5.69.3.34 def sparksee.SparkseeConfig.get_recovery_enabled (*self*)

Gets whether the recovery is enabled or disabled.

Returns

TRUE if the recovery is enabled, FALSE otherwise.

5.69.3.35 def sparksee.SparkseeConfig.get_recovery_log_file (*self*)

Gets the recovery log file.

Returns

The recovery log file.

5.69.3.36 def sparksee.SparkseeConfig.get_rollback_enabled (*self*)

Gets whether the rollback is enabled or disabled.

Returns

TRUE if the rollback is enabled, FALSE otherwise.

5.69.3.37 def sparksee.SparkseeConfig.get_sparksee_config_file (*self*)

Gets the config file path.

Returns

Returns the configured path or an empty string

5.69.3.38 def sparksee.SparkseeConfig.get_tmp_enabled (*self*)

Gets whether using temporary storage for computations is enabled.

Returns

True if enabled

5.69.3.39 def sparksee.SparkseeConfig.get_tmp_folder (*self*)

Gets the temporary folder used for temporary staging.

Returns

The temporary folder path

5.69.3.40 def sparksee.SparkseeConfig.save (*self*)

Save the current configuration in the specified config file.

It will try to save the current configuration only if a config file was specified. This method will be automatically used when a Sparksee class downloads a new License.

Returns

Returns true if the config file is successfully written or false otherwise.

5.69.3.41 def sparksee.SparkseeConfig.save_all (*self*)

Save all the current configuration settings in the specified config file.

It will try to save the current configuration only if a config file was specified. The saved config file WILL CONATAIN all the modified settings including the secret ENCRYPTION KEYS. You should usually just use the Save method instead.

Returns

Returns true if the config file is successfully written or false otherwise.

5.69.3.42 def sparksee.SparkseeConfig.set_a_e_s_encryption_enabled (*self, key_in_hex, iv_in_hex*)

Enables storage encryption using AES with the given key and iv.

The key and initialization vector will not be saved in the config file.

Parameters

key_in_hex	[In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes).
iv_in_hex	[In] The AES Initialization Vector as an hexadecimal string (16 bytes).

Returns

Returns true if the encryption had been enabled with the given arguments or false otherwise.

Referenced by sparksee.SparkseeConfig.set_a_e_s_encryption_enabled().

5.69.3.43 def sparksee.SparkseeConfig.set_a_e_s_encryption_enabled (*self, key_size*)

Enables storage encryption using AES and GENARATES a key and an iv.

YOU MUST GET AND KEEP SAFE THE GENERATED KEY AND IV! The key and initialization vector will not be saved in the config file.

Parameters

key_size	[In] The key size in bytes (8, 16 or 32).

Returns

Returns true if the encryption had been enabled.

References sparksee.SparkseeConfig.set_a_e_s_encryption_enabled().

5.69.3.44 def sparksee.SparkseeConfig.set_cache_max_size (*self, mega_bytes*)

Sets the maximum size for the cache (all pools) in MB.

Parameters

mega_bytes	[in] The maximum size for the cache (all pools) in MB. It must be non-negative.

5.69.3.45 def sparksee.SparkseeConfig.set_cache_statistics_enabled (*self, status*)

Enables or disables cache statistics.

Parameters

status	[in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

5.69.3.46 def sparksee.SparkseeConfig.set_cache_statistics_file (*self, file_path*)

Sets the cache statistics log file.

Useless if cache statistics are disabled.

Parameters

file_path	[in] The cache statistics log file.

5.69.3.47 def sparksee.SparkseeConfig.set_cache_statistics_snapshot_time (*self, micro_seconds*)

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters

micro_seconds	[in] The cache statistics snapshot time in microseconds.

5.69.3.48 def sparksee.SparkseeConfig.set_call_stack_dump (*self, status*)

Sets whether the signals will be captured to dump the call stack or not.

Parameters

status	[in] If TRUE signals must be captured.

5.69.3.49 def sparksee.SparkseeConfig.set_checksum_enabled (*self, status*)

Enables or disables the storage checksum usage.

Parameters

status	[in] If TRUE this enables the checksum, if FALSE then disables it.

5.69.3.50 def sparksee.SparkseeConfig.set_client_id (*self, client_id*)

Set the client identifier.

If you don't have a client identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

Parameters

client↩*_id*	[in] The client identifier.

5.69.3.51 def sparksee.SparkseeConfig.set_extent_pages (*self, pages*)

Sets the number of pages per extent.

Parameters

pages	[in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

5.69.3.52 def sparksee.SparkseeConfig.set_extent_size (*self, k_bytes*)

Sets the size of the extents in KB.

Parameters

k_bytes	[in] The size of an extent in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

5.69.3.53 def sparksee.SparkseeConfig.set_high_availability_coordinators (*self, ip*)

Sets the coordinators address and port list.

Parameters

ip	[in] The coordinators address and port list.

5.69.3.54 def sparksee.SparkseeConfig.set_high_availability_enabled (*self, status*)

Enables or disables high availability mode.

Parameters

status	[in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.

5.69.3.55 def sparksee.SparkseeConfig.set_high_availability_i_p (*self, ip*)

Sets the IP address and port of the instance.

Parameters

ip	[in] The IP address and port of the instance.

5.69.3.56 def sparksee.SparkseeConfig.set_high_availability_master_history (*self, file_path*)

Sets the master's history log.

Parameters

file_path	[in] The master's history log.

5.69.3.57 def sparksee.SparkseeConfig.set_high_availability_synchronization (*self, micro_seconds*)

Sets the synchronization polling time.

Parameters

micro_seconds	[in] The synchronization polling time.

5.69.3.58 def sparksee.SparkseeConfig.set_in_mem_alloc_size (*self, size*)

Sets the in-memory allocator size.

Parameters

size	The size to set the in-memory allocator to

5.69.3.59 def sparksee.SparkseeConfig.set_license (*self, key*)

Sets the license key.

Parameters

key	[in] The license key. Returns -1 if the new key can't be used, 0 if it's the same license or 1 if the new license is applied.

5.69.3.60 def sparksee.SparkseeConfig.set_license_id (*self, license_id*)

Set the license identifier.

If you don't have a license identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

Parameters

license↩*_id*	[in] The license identifier.

5.69.3.61 def sparksee.SparkseeConfig.set_license_pre_download_days (*self, days*)

Start trying to automatically download a new license at the specified number of days before expiration.

Parameters

days	[in] Number of days before expiration or -1 if the license should never be downloaded.

5.69.3.62 def sparksee.SparkseeConfig.set_log_file (*self, file_path*)

Sets the log file.

Parameters

file_path	[in] The log file.

5.69.3.63 def sparksee.SparkseeConfig.set_log_level (*self, level*)

Sets the log level.

Parameters

level	[in] The LogLevel.

5.69.3.64 def sparksee.SparkseeConfig.set_pool_frame_size (*self, extents*)

Sets the size of a pool frame in number of extents.

Parameters

extents	[in] The size of a pool frame in number of extents. It must be non-negative.

5.69.3.65 def sparksee.SparkseeConfig.set_pool_partitions (*self, pools*)

Sets the number of pools in each PartitionedPool.

Parameters

pools	[in] The number of pools in each PartitionedPool. It must be non-negative.

5.69.3.66 def sparksee.SparkseeConfig.set_pool_persistent_max_size (*self, frames*)

Sets the maximum size for the persistent pool in number of frames.

Parameters

frames	[in] The maximum size for the persistent pool in number of frames. It must be non-negative.

5.69.3.67 def sparksee.SparkseeConfig.set_pool_persistent_min_size (*self, frames*)

Sets the minimum size for the persistent pool in number of frames.

Parameters

frames	[in] The minimum size for the persistent pool in number of frames. It must be non-negative.

5.69.3.68 def sparksee.SparkseeConfig.set_pool_temporary_max_size (*self, frames*)

Sets the maximum size for the temporary pool in number of frames.

Parameters

frames	[in] The maximum size for the temporary pool in number of frames. It must be non-negative.

5.69.3.69 def sparksee.SparkseeConfig.set_pool_temporary_min_size (*self, frames*)

Sets the minimum size for the temporary pool in number of frames.

Parameters

frames	[in] The minimum size for the temporary pool in number of frames. It must be non-negative.

5.69.3.70 def sparksee.SparkseeConfig.set_recovery_cache_max_size (*self, extents*)

Sets the maximum size for the recovery log cache in extents.

Parameters

extents	[in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

5.69.3.71 def sparksee.SparkseeConfig.set_recovery_checkpoint_time (*self, micro_seconds*)

Sets the delay time (in microseconds) between automatic checkpoints.

Parameters

micro_seconds	[in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each commited transaction.

5.69.3.72 def sparksee.SparkseeConfig.set_recovery_enabled (*self, status*)

Enables or disables the recovery.

Parameters

status	[in] If TRUE this enables the recovery, if FALSE then disables it.

5.69.3.73 def sparksee.SparkseeConfig.set_recovery_log_file (*self, file_path*)

Sets the recovery log file.

Parameters

file_path	[in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log)

5.69.3.74 def sparksee.SparkseeConfig.set_rollback_enabled (*self, status*)

Enables or disables the rollback.

Parameters

status	[in] If TRUE this enables the rollback, if FALSE then disables it.

5.69.3.75 def sparksee.SparkseeConfig.set_sparksee_config_file (*self, path*)

Sets the config file path.

The file is not loaded in this operation, see the constructor options if you need to load the file. The config file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.

5.69.3.76 def sparksee.SparkseeConfig.set_tmp_enabled (*self, enabled*)

Sets whether to use temporary storage for computations or not.

Parameters

enabled	True to use temporary storage for computation

5.69.3.77 def sparksee.SparkseeConfig.set_tmp_folder (*self, tmp_folder*)

Sets the temporary folder used for temporary staging.

Parameters

tmp_folder	The temporary folder to set

5.70 sparksee.SparkseeProperties Class Reference

Sparksee properties file.

Public Member Functions

- def clear (self)

  Remove all the properties.
- def get (self, key, def)

  Gets a property.
- def get_integer (self, key, def)

  Gets a property as an integer.
- def set_h_i (self, value)

  YOU SHOULD NOT USE THIS METHOD.
- def load (self, path)

Loads properties from the given file path.

- def get_time_unit (self, key, def)

  Gets a property as a time unit.

- def get_boolean (self, key, def)

  Gets a property as a boolean.

5.70.1 Detailed Description

Sparksee properties file.

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: key=value

By default, this loads properties from the file './sparksee.cfg'. The user may choose to load a different file by calling the method Load().

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

5.70.2 Member Function Documentation

5.70.2.1 def sparksee.SparkseeProperties.get (*self, key, def*)

Gets a property.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The value of the property, or def if the key is not found.

5.70.2.2 def sparksee.SparkseeProperties.get_boolean (*self, key, def*)

Gets a property as a boolean.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The property value, or def if the key is not found or in case of error.

5.70.2.3 def sparksee.SparkseeProperties.get_integer (*self, key, def*)

Gets a property as an integer.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The property value, or def if the key is not found or in case of error.

5.70.2.4 def sparksee.SparkseeProperties.get_time_unit (*self, key, def*)

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Valid format for the string representation: Blanks at the begining or at the end are ignored. No blanks are allowed between the time duration and the unit time.

Allowed time units: 'D' for days, 'H' for hours, 'M' for minutes, 'S' o 's' for seconds, 'm' for milliseconds and 'u' for microseconds.

There is a special case: If no time unit is given, seconds is the default. So, '10' means 10 seconds.

Parameters

key	[in] The name of the property to lookup.
def	[in] The default value (in microseconds) to be returned in case there is no property with the name key.

Returns

The time duration in microseconds, or def if the key is not found or in case of error.

5.70.2.5 def sparksee.SparkseeProperties.load (*self, path*)

Loads properties from the given file path.

Parameters

path	[in] File path to load properties from.

5.70.2.6 def sparksee.SparkseeProperties.set_h_i (*self, value*)

YOU SHOULD NOT USE THIS METHOD.

This method exists only for a specific service.

Parameters

value	null

5.71 sparksee.StringList Class Reference

String list.

Public Member Functions

- def clear (self)
  Clears the list.
- def __init__ (self)
  Constructor.
- def __iter__ (self)
  Gets a new TypeListIterator.
- def iterator (self)
  Gets a new StringListIterator.
- def add (self, str)
  Adds a String at the end of the list.
- def count (self)
  Number of elements in the list.

5.71.1 Detailed Description

String list.

It stores a String (unicode) list.

Use StringListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.71.2 Constructor & Destructor Documentation

5.71.2.1 def sparksee.StringList.__init__ (*self*)

Constructor.

This creates an empty list.

5.71.3 Member Function Documentation

5.71.3.1 def sparksee.StringList.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.71.3.2 def sparksee.StringList.add (*self, str*)

Adds a String at the end of the list.

Parameters

str	[in] String.

5.71.3.3 def sparksee.StringList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.71.3.4 def sparksee.StringList.iterator (*self*)

Gets a new StringListIterator.

Returns

StringListIterator instance.

5.72 sparksee.StringListIterator Class Reference

StringList iterator class.

Public Member Functions

- def next (self)
  Moves to the next element.
- def has_next (self)
  Gets if there are more elements.
- def __next__ (self)
  Used in next()

5.72.1 Detailed Description

StringList iterator class.

Iterator to traverse all the strings into a StringList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.72.2 Member Function Documentation

5.72.2.1 def sparksee.StringListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.72.2.2 def sparksee.StringListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.72.2.3 def sparksee.StringListIterator.next (*self*)

Moves to the next element.

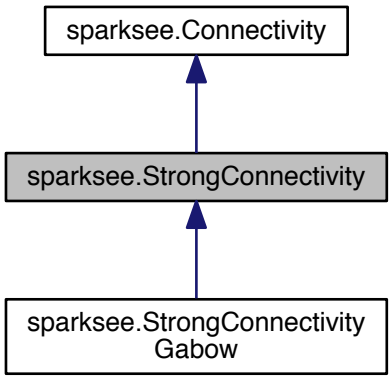
Returns

The next element.

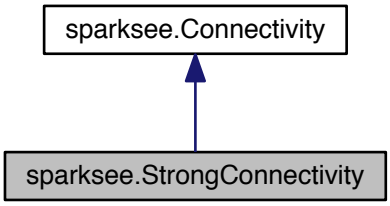
5.73 sparksee.StrongConnectivity Class Reference

StrongConnectivity class.

Inheritance diagram for sparksee.StrongConnectivity:



Collaboration diagram for sparksee.StrongConnectivity:



Public Member Functions

- def add_edge_type (self, type, dir)

  Allows connectivity through edges of the given type.
- def run (self)

  Runs the algorithm in order to find the connected components.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def get_connected_components (self)

  Returns the results generated by the execution of the algorithm.
- def add_all_edge_types (self, dir)

  Allows connectivity through all edge types of the graph.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_node_type (self, t)

  Allows connectivity through nodes of the given type.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def close (self)

  Closes the Connectivity instance.
- def is_closed (self)

  Gets if Connectivity has been closed or not.

5.73.1 Detailed Description

StrongConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.73.2 Member Function Documentation

5.73.2.1 def sparksee.StrongConnectivity.add_all_edge_types (*self, dir*)

Allows connectivity through all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.73.2.2 def sparksee.StrongConnectivity.add_edge_type (*self, type, dir*)

Allows connectivity through edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.73.2.3 def sparksee.StrongConnectivity.add_node_type (*self, t*)

Allows connectivity through nodes of the given type.

Parameters

t	null

5.73.2.4 def sparksee.Connectivity.close (*self*) [inherited]

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

5.73.2.5 def sparksee.StrongConnectivity.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.73.2.6 def sparksee.StrongConnectivity.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.73.2.7 def sparksee.StrongConnectivity.get_connected_components (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

5.73.2.8 def sparksee.Connectivity.is_closed (*self*) [inherited]

Gets if Connectivity has been closed or not.

See also

close()

Returns

TRUE if the Connectivity instance has been closed, FALSE otherwise.

5.73.2.9 def sparksee.StrongConnectivity.run (*self*)

Runs the algorithm in order to find the connected components.

This method can be called only once.

5.73.2.10 def sparksee.StrongConnectivity.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

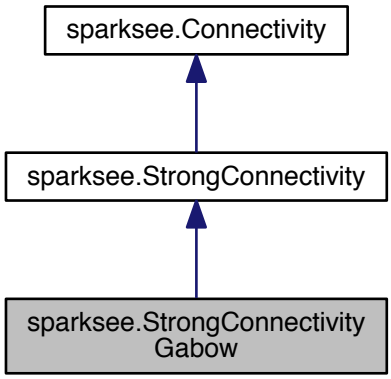
Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

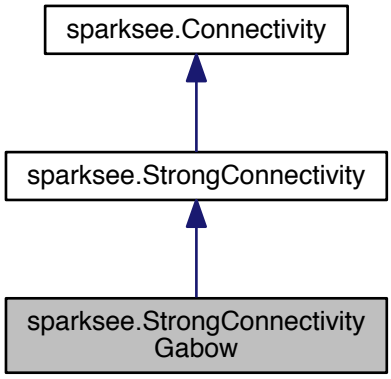
5.74 sparksee.StrongConnectivityGabow Class Reference

This class can be used to solve the problem of finding strongly connected components in a directed graph.

Inheritance diagram for sparksee.StrongConnectivityGabow:



Collaboration diagram for sparksee.StrongConnectivityGabow:



Public Member Functions

- def add_edge_type (self, type, dir)

  Allows connectivity through edges of the given type.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def get_connected_components (self)

  Returns the results generated by the execution of the algorithm.
- def run (self)

  Executes the algorithm.
- def add_all_edge_types (self, dir)

  Allows connectivity through all edge types of the graph.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_node_type (self, t)

  Allows connectivity through nodes of the given type.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- def __init__ (self, session)

  Creates a new instance of StrongConnectivityGabow.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def close (self)

  Closes the Connectivity instance.
- def is_closed (self)

  Gets if Connectivity has been closed or not.

5.74.1 Detailed Description

This class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type. This implementation is based on the Gabow algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.74.2 Constructor & Destructor Documentation

5.74.2.1 def sparksee.StrongConnectivityGabow.__init__ (*self, session*)

Creates a new instance of StrongConnectivityGabow.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the strong connected components.

Parameters

session	[in] Session to get the graph from and calculate the connectivity

5.74.3 Member Function Documentation

5.74.3.1 def sparksee.StrongConnectivityGabow.add_all_edge_types (*self, dir*)

Allows connectivity through all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.74.3.2 def sparksee.StrongConnectivityGabow.add_edge_type (*self, type, dir*)

Allows connectivity through edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.74.3.3 def sparksee.StrongConnectivityGabow.add_node_type (*self, t*)

Allows connectivity through nodes of the given type.

Parameters

t	null

5.74.3.4 def sparksee.Connectivity.close (*self*) [inherited]

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

5.74.3.5 def sparksee.StrongConnectivityGabow.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.74.3.6 def sparksee.StrongConnectivityGabow.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.74.3.7 def sparksee.StrongConnectivityGabow.get_connected_components (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

5.74.3.8 def sparksee.Connectivity.is_closed (*self*) [inherited]

Gets if Connectivity has been closed or not.

See also

close()

Returns

TRUE if the Connectivity instance has been closed, FALSE otherwise.

5.74.3.9 def sparksee.StrongConnectivityGabow.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.75 sparksee.TextStream Class Reference

TextStream class.

Public Member Functions

- def write (self, data_i_n, length)
  Write data to the stream.
- def close (self)
  Closes the stream.
- def read (self, length)
  Read data from the stream.
- def is_null (self)
  Returns TRUE if the stream is not available.
- def __init__ (self, append)
  Creates a new instance.

5.75.1 Detailed Description

TextStream class.

It allows for reading and writting Text attribute values.

It is very important to close the stream once no more reading or writting operations will be performed to ensure data is successfully stored.

Whereas string attributes are set and got using the Value class, text attributes are operated using a stream pattern.

Use of TextStream for writing: (i) Create a TextStream instance and (ii) set the stream for a text attribute of a node or edge instance with the graph SetAttributeText method. Once the set attribute text has been done, (iii) perform as many write operations as you need to the TextStream instance. Lastly, (iv) exeucte Close to flush and close the stream.

Use of TextStream for reading: (i) Get the stream of a text attribute of a node or edge instance with the Get↩ AttributeText graph method. Once you have the TextStream instance, (ii) you can execute Read operations to read from the stream. (iii) The end of the stream is reached when Read returns 0. Finally, (iv) execute Close to close stream resources.

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.75.2 Constructor & Destructor Documentation

5.75.2.1 def sparksee.TextStream.__init__ (*self, append*)

Creates a new instance.

A TextStream only can be created by the user to write data.

Parameters

append	[in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the begining of the stream.

5.75.3 Member Function Documentation

5.75.3.1 def sparksee.TextStream.close (*self*)

Closes the stream.

Once the Stream is closed, it cannot be used again.

Closing the stream is mandatory when the stream is not null and strongly recommended when it's null to avoid deallocation problems in some platforms.

5.75.3.2 def sparksee.TextStream.is_null (*self*)

Returns TRUE if the stream is not available.

It returns for reading or writing data.

Returns

FALSE if the stream is ready

5.75.3.3 def sparksee.TextStream.read (*self, length*)

Read data from the stream.

Parameters

length	[in] Length of the given data buffer. It must be > 0.

Returns

The read data.

5.75.3.4 def sparksee.TextStream.write (*self, data_i_n, length*)

Write data to the stream.

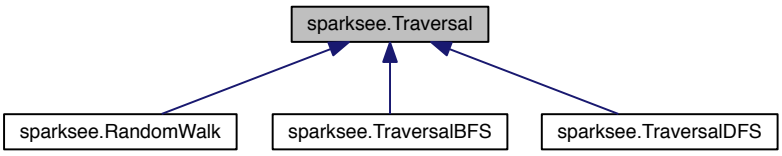
Parameters

data_i↩ _n	The string to write data from.
length	Number of characters to write. It must be > 0.

5.76 sparksee.Traversal Class Reference

Traversal class.

Inheritance diagram for sparksee.Traversal:



Public Member Functions

- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def has_next (self)

  Gets if there are more objects to be traversed.
- def get_current_depth (self)

  Returns the depth of the current node.
- def __iter__ (self)

  Gets a new TraversalIterator.
- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def __next__ (self)

  Used in next()
- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def next (self)

  Gets the next object of the traversal.
- def close (self)

  Closes the Traversal instance.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def is_closed (self)

  Gets if Traversal has been closed or not.

5.76.1 Detailed Description

Traversal class.

Any class implementing this abstract class can be used to traverse a graph.

Once the instance has been created and the allowed node and edge types has been set, it can be used as an iterator, retrieving the next object identifier of the traversal until there are no more.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.76.2 Member Function Documentation

5.76.2.1 def sparksee.Traversal.__iter__ (*self*)

Gets a new TraversalIterator.

Returns

TraversalIterator instance

5.76.2.2 def sparksee.Traversal.__next__ (*self*)

Used in next()

Returns

The next element

5.76.2.3 def sparksee.Traversal.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.76.2.4 def sparksee.Traversal.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.76.2.5 def sparksee.Traversal.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

5.76.2.6 def sparksee.Traversal.close (*self*)

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

5.76.2.7 def sparksee.Traversal.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.76.2.8 def sparksee.Traversal.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.76.2.9 def sparksee.Traversal.get_current_depth (*self*)

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

5.76.2.10 def sparksee.Traversal.has_next (*self*)

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

5.76.2.11 def sparksee.Traversal.is_closed (*self*)

Gets if Traversal has been closed or not.

See also

close()

Returns

TRUE if the Traversal instance has been closed, FALSE otherwise.

5.76.2.12 def sparksee.Traversal.next (*self*)

Gets the next object of the traversal.

Returns

A node or edge identifier.

5.76.2.13 def sparksee.Traversal.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

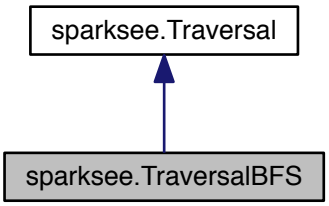
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

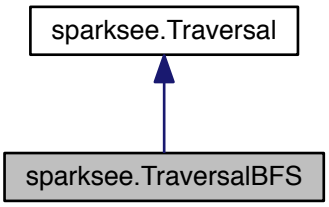
5.77 sparksee.TraversalBFS Class Reference

Breadth-First Search implementation of Traversal.

Inheritance diagram for sparksee.TraversalBFS:



Collaboration diagram for sparksee.TraversalBFS:



Public Member Functions

- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def next (self)

  Gets the next object of the traversal.
- def has_next (self)

  Gets if there are more objects to be traversed.
- def get_current_depth (self)

  Returns the depth of the current node.
- def __init__ (self, session, node)

  Creates a new instance.
- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def __iter__ (self)

  Gets a new TraversalIterator.
- def __next__ (self)

  Used in next()
- def close (self)

  Closes the Traversal instance.
- def is_closed (self)

  Gets if Traversal has been closed or not.

5.77.1 Detailed Description

Breadth-First Search implementation of Traversal.

Starting from a source node, it visits all its neighbors at distance 1, then all its neighbors at distance 2, and so on.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.77.2 Constructor & Destructor Documentation

5.77.2.1 def sparksee.TraversalBFS.__init__ (*self, session, node*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
node	[in] Node to start traversal from.

5.77.3 Member Function Documentation

5.77.3.1 def sparksee.Traversal.__iter__ (*self*) [inherited]

Gets a new TraversalIterator.

Returns

TraversalIterator instance

5.77.3.2 def sparksee.Traversal.__next__ (*self*) [inherited]

Used in next()

Returns

The next element

5.77.3.3 def sparksee.TraversalBFS.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.77.3.4 def sparksee.TraversalBFS.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.77.3.5 def sparksee.TraversalBFS.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

5.77.3.6 def sparksee.Traversal.close (*self*) [inherited]

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

5.77.3.7 def sparksee.TraversalBFS.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.77.3.8 def sparksee.TraversalBFS.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.77.3.9 def sparksee.TraversalBFS.get_current_depth (*self*)

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

5.77.3.10 def sparksee.TraversalBFS.has_next (*self*)

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

5.77.3.11 def sparksee.Traversal.is_closed (*self*) [inherited]

Gets if Traversal has been closed or not.

See also

close()

Returns

TRUE if the Traversal instance has been closed, FALSE otherwise.

5.77.3.12 def sparksee.TraversalBFS.next (*self*)

Gets the next object of the traversal.

Returns

A node or edge identifier.

5.77.3.13 def sparksee.TraversalBFS.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

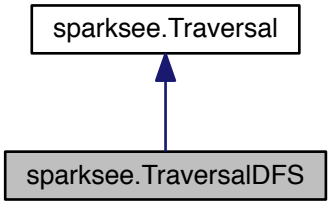
Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

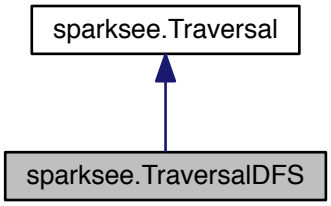
5.78 sparksee.TraversalDFS Class Reference

Depth-First Search (DFS) implementation of Traversal.

Inheritance diagram for sparksee.TraversalDFS:



Collaboration diagram for sparksee.TraversalDFS:



Public Member Functions

- def add_edge_type (self, type, dir)

  Allows for traversing edges of the given type.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self, dir)

  Allows for traversing all edge types of the graph.
- def add_node_type (self, type)

  Allows for traversing nodes of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def next (self)

  Gets the next object of the traversal.
- def has_next (self)

  Gets if there are more objects to be traversed.
- def get_current_depth (self)

  Returns the depth of the current node.
- def __init__ (self, session, node)

  Creates a new instance.

- def add_all_node_types (self)

  Allows for traversing all node types of the graph.
- def set_maximum_hops (self, maxhops)

  Sets the maximum hops restriction.
- def __iter__ (self)

  Gets a new TraversalIterator.
- def __next__ (self)

  Used in next()
- def close (self)

  Closes the Traversal instance.
- def is_closed (self)

  Gets if Traversal has been closed or not.

5.78.1 Detailed Description

Depth-First Search (DFS) implementation of Traversal.

Starting from a source or root node, it visits as far as possible along each branch before backtracking.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.78.2 Constructor & Destructor Documentation

5.78.2.1 def sparksee.TraversalDFS.__init__ (*self, session, node*)

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
node	[in] Node to start traversal from.

5.78.3 Member Function Documentation

5.78.3.1 def sparksee.Traversal.__iter__ (*self*) [inherited]

Gets a new TraversalIterator.

Returns

TraversalIterator instance

5.78.3.2 def sparksee.Traversal.__next__ (*self*) [inherited]

Used in next()

Returns

The next element

5.78.3.3 def sparksee.TraversalDFS.add_all_edge_types (*self, dir*)

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

5.78.3.4 def sparksee.TraversalDFS.add_edge_type (*self, type, dir*)

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

5.78.3.5 def sparksee.TraversalDFS.add_node_type (*self, type*)

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

5.78.3.6 def sparksee.Traversal.close (*self*) [inherited]

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

5.78.3.7 def sparksee.TraversalDFS.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.78.3.8 def sparksee.TraversalDFS.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.78.3.9 def sparksee.TraversalDFS.get_current_depth (*self*)

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

5.78.3.10 def sparksee.TraversalDFS.has_next (*self*)

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

5.78.3.11 def sparksee.Traversal.is_closed (*self*) [inherited]

Gets if Traversal has been closed or not.

See also

close()

Returns

TRUE if the Traversal instance has been closed, FALSE otherwise.

5.78.3.12 def sparksee.TraversalDFS.next (*self*)

Gets the next object of the traversal.

Returns

A node or edge identifier.

5.78.3.13 def sparksee.TraversalDFS.set_maximum_hops (*self, maxhops*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.79 sparksee.Type Class Reference

Type data class.

Public Member Functions

- def get_object_type (self)
  Gets the object type.
- def get_id (self)
  Gets the Sparksee type identifier.
- def get_restricted_from (self)
  Gets the tail or source type identifier for restricted edge types.
- def get_num_objects (self)
  Gets the number of objects belonging to the type.
- def get_are_neighbors_indexed (self)
  Gets if this is an edge type with neighbors index.
- def get_restricted_to (self)
  Gets the head or target type identifier for restricted edge types.
- def get_is_restricted (self)
  Gets if this is a restricted edge type.
- def get_name (self)
  Gets the unique type name.
- def get_is_directed (self)
  Gets if this is a directed edge type.

Static Public Attributes

- int INVALID_TYPE = 0
  Invalid type identifier.
- int NODES_TYPE = 2
  Identifier for all nodeType attributes.
- int GLOBAL_TYPE = 1
  Global type identifier.
- int EDGES_TYPE = 3
  Identifier for all edgeType attributes.

5.79.1 Detailed Description

Type data class.

It contains information about a node or edge type.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.79.2 Member Function Documentation

5.79.2.1 def sparksee.Type.get_are_neighbors_indexed (*self*)

Gets if this is an edge type with neighbors index.

Returns

TRUE for edges types with neighbors index, FALSE otherwise.

5.79.2.2 def sparksee.Type.get_id (*self*)

Gets the Sparksee type identifier.

Returns

The Sparksee type identifier.

5.79.2.3 def sparksee.Type.get_is_directed (*self*)

Gets if this is a directed edge type.

Returns

TRUE for directed edge types, FALSE otherwise.

5.79.2.4 def sparksee.Type.get_is_restricted (*self*)

Gets if this is a restricted edge type.

Returns

TRUE for restricted edge types, FALSE otherwise.

5.79.2.5 def sparksee.Type.get_name (*self*)

Gets the unique type name.

Returns

The unique type name.

5.79.2.6 def sparksee.Type.get_num_objects (*self*)

Gets the number of objects belonging to the type.

Returns

The number of objects belonging to the type.

5.79.2.7 def sparksee.Type.get_object_type (*self*)

Gets the object type.

Returns

The object type.

5.79.2.8 def sparksee.Type.get_restricted_from (*self*)

Gets the tail or source type identifier for restricted edge types.

Returns

For restricted edge types, the tail or source type identifier, the Type InvalidType otherwise.

5.79.2.9 def sparksee.Type.get_restricted_to (*self*)

Gets the head or target type identifier for restricted edge types.

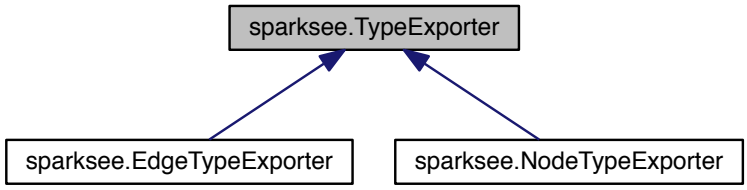
Returns

For restricted edge types, the head or target type identifier, the Type InvalidType otherwise.

5.80 sparksee.TypeExporter Class Reference

Base TypeExporter class.

Inheritance diagram for sparksee.TypeExporter:



Public Member Functions

- def run (self)

  Runs export process.
- def set_type (self, type)

  Sets the type to be exported.
- def set_header (self, header)

  Sets the presence of a header row.
- def set_row_writer (self, rw)

  Sets the output data destination.
- def set_graph (self, graph)

  Sets the graph that will be exported.
- def register (self, tel)

  Registers a new listener.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.

5.80.1 Detailed Description

Base TypeExporter class.

Base class to export a node or edge type from a graph using a RowWriter.

TypeExporterListener can be registered to receive information about the progress of the export process by means of TypeExporterEvent. The default frequency of notification to listeners is 100000.

By default no header row is created.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.80.2 Member Function Documentation

5.80.2.1 def sparksee.TypeExporter.register (*self, tel*)

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

5.80.2.2 def sparksee.TypeExporter.run (*self*)

Runs export process.

Exceptions

RuntimeError	null
IOError	If bad things happen writing to the RowWriter.

5.80.2.3 def sparksee.TypeExporter.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

5.80.2.4 def sparksee.TypeExporter.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.80.2.5 def sparksee.TypeExporter.set_graph (*self, graph*)

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

5.80.2.6 def sparksee.TypeExporter.set_header (*self, header*)

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

5.80.2.7 def sparksee.TypeExporter.set_row_writer (*self, rw*)

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

5.80.2.8 def sparksee.TypeExporter.set_type (*self, type*)

Sets the type to be exported.

Parameters

type	[in] Type identifier.

5.81 sparksee.TypeExporterEvent Class Reference

Provides information about the progress of an TypeExproter instance.

Public Member Functions

- def get_type_id (self)

  Gets the type identifier.
- def get_total (self)

  Gets the total number of objects exported.
- def get_count (self)

  Gets the current number of objects exported.
- def is_last (self)

  Gets if this is the last event or not.

5.81.1 Detailed Description

Provides information about the progress of an TypeExproter instance.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.81.2 Member Function Documentation

5.81.2.1 def sparksee.TypeExporterEvent.get_count (*self*)

Gets the current number of objects exported.

Returns

The current number of objects exported.

5.81.2.2 def sparksee.TypeExporterEvent.get_total (*self*)

Gets the total number of objects exported.

Returns

The total number of objects exported.

5.81.2.3 def sparksee.TypeExporterEvent.get_type_id (*self*)

Gets the type identifier.

Returns

The type identifier.

5.81.2.4 def sparksee.TypeExporterEvent.is_last (*self*)

Gets if this is the last event or not.

Returns

TRUE if this is the last event, FALSE otherwise.

5.82 sparksee.TypeExporterListener Class Reference

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Public Member Functions

- def notify_event (self, tee)
  Method to be notified from a TypeExporter.

5.82.1 Detailed Description

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.82.2 Member Function Documentation

5.82.2.1 def sparksee.TypeExporterListener.notify_event (*self, tee*)

Method to be notified from a TypeExporter.

Parameters

tee	[in] Notified event.

5.83 sparksee.TypeList Class Reference

Sparksee type identifier list.

Public Member Functions

- def clear (self)
  Clears the list.
- def __iter__ (self)
  Gets a new TypeListIterator.
- def iterator (self)
  Gets a new TypeListIterator.
- def __init__ (self)
  Constructor.
- def count (self)
  Number of elements in the list.
- def add (self, type)
  Adds a Sparksee type identifier at the end of the list.

5.83.1 Detailed Description

Sparksee type identifier list.

It stores a Sparksee node or edge type identifier list.

Use TypeListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.83.2 Constructor & Destructor Documentation

5.83.2.1 def sparksee.TypeList.__init__ (*self*)

Constructor.

This creates an empty list.

5.83.3 Member Function Documentation

5.83.3.1 def sparksee.TypeList.__iter__ (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance

5.83.3.2 def sparksee.TypeList.add (*self, type*)

Adds a Sparksee type identifier at the end of the list.

Parameters

type	[in] Sparksee type identifier.

5.83.3.3 def sparksee.TypeList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.83.3.4 def sparksee.TypeList.iterator (*self*)

Gets a new TypeListIterator.

Returns

TypeListIterator instance.

5.84 sparksee.TypeListIterator Class Reference

TypeList iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.84.1 Detailed Description

TypeList iterator class.

Iterator to traverse all the Sparksee node or edge type identifiers into a TypeList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.84.2 Member Function Documentation

5.84.2.1 def sparksee.TypeListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.84.2.2 def sparksee.TypeListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.84.2.3 def sparksee.TypeListIterator.next (*self*)

Moves to the next element.

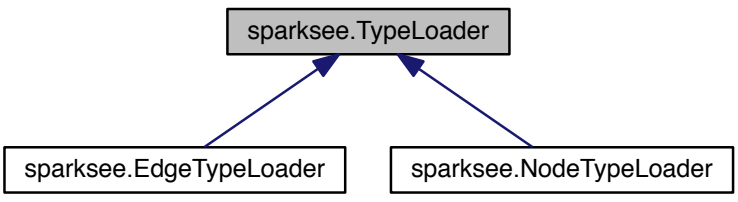
Returns

The next element.

5.85 sparksee.TypeLoader Class Reference

Base TypeLoader class.

Inheritance diagram for sparksee.TypeLoader:



Public Member Functions

- def run (self)

  Run the loader.
- def run_two_phases (self)

  Run the loader for two phases loading.
- def set_log_error (self, path)

  Sets a log error file.
- def set_type (self, type)

  Sets the type to be loaded.
- def set_locale (self, locale_str)

  Sets the locale that will be used to read the data.
- def set_timestamp_format (self, timestamp_format)

  Sets a specific timestamp format.
- def set_attributes (self, attrs)

  Sets the list of Attributes.
- def register (self, tel)

  Registers a new listener.
- def set_frequency (self, freq)

  Sets the frequency of listener notification.
- def set_row_reader (self, rr)

  Sets the input data source.
- def set_attribute_positions (self, attrs_pos)

  Sets the list of attribute positions.
- def run_n_phases (self, partitions)

  Run the loader for N phases loading.
- def set_graph (self, graph)

  Sets the graph where the data will be loaded.
- def set_log_off (self)

  Truns off all the error reporting.

5.85.1 Detailed Description

Base TypeLoader class.

Base class to load a node or edge type from a graph using a RowReader.

TypeLoaderListener can be registered to receive information about the progress of the load process by means of TypeLoaderEvent. The default frequency of notification to listeners is 100000.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.85.2 Member Function Documentation

5.85.2.1 def sparksee.TypeLoader.register (*self, tel*)

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

5.85.2.2 def sparksee.TypeLoader.run (*self*)

Run the loader.

Exceptions

RuntimeError	null
IOError	null

5.85.2.3 def sparksee.TypeLoader.run_n_phases (*self, partitions*)

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

Parameters

partitions	[in] Number of horizontal partitions to perform the load.

Exceptions

RuntimeError	null
IOError	null

5.85.2.4 def sparksee.TypeLoader.run_two_phases (*self*)

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

Exceptions

RuntimeError	null
IOError	null

5.85.2.5 def sparksee.TypeLoader.set_attribute_positions (*self, attrs_pos*)

Sets the list of attribute positions.

Parameters

attrs_pos	[in] Attribute positions (column index >=0).

5.85.2.6 def sparksee.TypeLoader.set_attributes (*self, attrs*)

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

5.85.2.7 def sparksee.TypeLoader.set_frequency (*self, freq*)

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

5.85.2.8 def sparksee.TypeLoader.set_graph (*self, graph*)

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

5.85.2.9 def sparksee.TypeLoader.set_locale (*self, locale_str*)

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

locale_str	[in] The locale string for the read data. See CSVReader.

5.85.2.10 def sparksee.TypeLoader.set_log_error (*self, path*)

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

IOError	If bad things happen opening the file.

5.85.2.11 def sparksee.TypeLoader.set_log_off (*self*)

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.85.2.12 def sparksee.TypeLoader.set_row_reader (*self, rr*)

Sets the input data source.

Parameters

rr	[in] Input RowReader.

5.85.2.13 def sparksee.TypeLoader.set_timestamp_format (*self, timestamp_format*)

Sets a specific timestamp format.

Parameters

timestamp_format	[in] A string with the timestamp format definition.

5.85.2.14 def sparksee.TypeLoader.set_type (*self, type*)

Sets the type to be loaded.

Parameters

type	[in] Type identifier.

5.86 sparksee.TypeLoaderEvent Class Reference

Provides information about the progress of a TypeLoader instance.

Public Member Functions

- def get_type_id (self)
  Gets the type identifier.
- def get_phase (self)
  Gets the current phase.

- def get_count (self)
  Gets the current number of objects created.
- def get_total_partition_steps (self)
  Gets the total number of steps in the current partition.
- def get_partition (self)
  Gets the current partition.
- def get_total_partitions (self)
  Gets the total number of partitions.
- def is_last (self)
  Gets if this is the last event or not.
- def get_total_phases (self)
  Gets the total number of phases.

5.86.1 Detailed Description

Provides information about the progress of a TypeLoader instance.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.86.2 Member Function Documentation

5.86.2.1 def sparksee.TypeLoaderEvent.get_count (*self*)

Gets the current number of objects created.

Returns

The current number of objects created.

5.86.2.2 def sparksee.TypeLoaderEvent.get_partition (*self*)

Gets the current partition.

Returns

The current partition.

5.86.2.3 def sparksee.TypeLoaderEvent.get_phase (*self*)

Gets the current phase.

Returns

The current phase.

5.86.2.4 def sparksee.TypeLoaderEvent.get_total_partition_steps (*self*)

Gets the total number of steps in the current partition.

Returns

The total number steps in the current partition.

5.86.2.5 def sparksee.TypeLoaderEvent.get_total_partitions (*self*)

Gets the total number of partitions.

Returns

The total number of partitions.

5.86.2.6 def sparksee.TypeLoaderEvent.get_total_phases (*self*)

Gets the total number of phases.

Returns

The total number of phases.

5.86.2.7 def sparksee.TypeLoaderEvent.get_type_id (*self*)

Gets the type identifier.

Returns

The type identifier.

5.86.2.8 def sparksee.TypeLoaderEvent.is_last (*self*)

Gets if this is the last event or not.

Returns

TRUE if this is the last event, FALSE otherwise.

5.87 sparksee.TypeLoaderListener Class Reference

Public Member Functions

- def notify_event (self, ev)
  Method to receive events from a Loader.

5.87.1 Member Function Documentation

5.87.1.1 def sparksee.TypeLoaderListener.notify_event (*self, ev*)

Method to receive events from a Loader.

Parameters

ev	Loader.LoaderEvent with information from a running Loader.

5.88 sparksee.Value Class Reference

Value class.

Public Member Functions

- def set_null (self)

  Sets the Value to NULL.
- def get_timestamp (self)

  Gets Timestamp Value.
- def get_long (self)

  Gets Long Value.
- def operator (self, value)

  Assignment operator.
- def set_oid (self, value)

  Sets the Value.
- def set_double_void (self, value)

  Sets the Value.
- def set_boolean (self, value)

  Sets the Value.
- def __init__ (self)

  Creates a new instance.
- def get_integer (self)

  Gets Integer Value.
- def set_timestamp_void (self, year, month, day, hour, minutes, seconds, millisecs)

  Sets the Value.
- def equals (self, value)

  Compares with the given Value.
- def set_boolean_void (self, value)

  Sets the Value.
- def set_oid_void (self, value)

  Sets the OID Value.
- def set_long_void (self, value)

  Sets the Value.
- def set_string (self, value)

  Sets the Value.
- def compare (self, value)

  Compares with the given Value.
- def get_oid (self)

  Gets OID Value.
- def get_boolean (self)

  Gets Boolean Value.
- def set_timestamp (self, value)

  Sets the Value.

- def get_data_type (self)

  Gets the DataType.
- def set_long (self, value)

  Sets the Value.
- def set_integer (self, value)

  Sets the Value.
- def set_double (self, value)

  Sets the Value.
- def get_double (self)

  Gets Double Value.
- def __init__ (self, value)

  Copy constructor.
- def set_timestamp (self, year, month, day, hour, minutes, seconds, millisecs)

  Sets the Value.
- def set_string_void (self, value)

  Sets the Value.
- def set_void (self, value)

  Sets the Value.
- def set_timestamp_void (self, value)

  Sets the Value.
- def set_integer_void (self, value)

  Sets the Value.
- def is_null (self)

  Gets if this is a NULL Value.
- def get_string (self)

  Gets String Value.
- def set_null_void (self)

  Sets the Value to NULL.
- def to_string (self)

  *Returns a String representation of the Value, used in **unicode** and **str***

Static Public Attributes

- int MAX_LENGTH_STRING = 2047

  Maximum number of characters allowed for a String.

5.88.1 Detailed Description

Value class.

It is a container which stores a value and its data type (domain). A Value can be NULL.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.88.2 Constructor & Destructor Documentation

5.88.2.1 def sparksee.Value.__init__ (*self*)

Creates a new instance.

It creates a NULL Value.

Referenced by sparksee.Value.__init__().

5.88.2.2 def sparksee.Value.__init__ (*self, value*)

Copy constructor.

Parameters

value	[in] Value to be copied.

References sparksee.Value.__init__().

5.88.3 Member Function Documentation

5.88.3.1 def sparksee.Value.compare (*self, value*)

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

Parameters

value	Given value to compare to.

Returns

0 if this Value is equal to the given one; a value less than 0 if this Value is less than the given one; and a value greater than 0 if this Value is greater than the given one.

5.88.3.2 def sparksee.Value.equals (*self, value*)

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

Parameters

value	Given value to compare to.

Returns

TRUE if this Value is equal to the given one; FALSE otherwise.

5.88.3.3 def sparksee.Value.get_boolean (*self*)

Gets Boolean Value.

This must be a non-NULL Boolean Value.

Returns

The Boolean Value.

5.88.3.4 def sparksee.Value.get_data_type (*self*)

Gets the DataType.

Value cannot be NULL.

Returns

The DataType.

5.88.3.5 def sparksee.Value.get_double (*self*)

Gets Double Value.

This must be a non-NULL Double Value.

Returns

The Double Value.

5.88.3.6 def sparksee.Value.get_integer (*self*)

Gets Integer Value.

This must be a non-NULL Integer Value.

Returns

The Integer Value.

5.88.3.7 def sparksee.Value.get_long (*self*)

Gets Long Value.

This must be a non-NULL Long Value.

Returns

The Long Value.

5.88.3.8 def sparksee.Value.get_oid (*self*)

Gets OID Value.

This must be an non-NULL OID Value.

Returns

The OID Value.

5.88.3.9 def sparksee.Value.get_string (*self*)

Gets String Value.

This must be a non-NULL String Value.

Returns

The String Value.

5.88.3.10 def sparksee.Value.get_timestamp (*self*)

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

Returns

The Timestamp Value.

5.88.3.11 def sparksee.Value.is_null (*self*)

Gets if this is a NULL Value.

Returns

TRUE if this is a NULL Value, FALSE otherwise.

5.88.3.12 def sparksee.Value.operator (*self, value*)

Assignment operator.

Parameters

value	[in] Value to be copied.

Returns

Returns the Value reference.

5.88.3.13 def sparksee.Value.set_boolean (*self, value*)

Sets the Value.

Parameters

value	[in] Nex Boolean value.

Returns

The calling instance.

5.88.3.14 def sparksee.Value.set_boolean_void (*self, value*)

Sets the Value.

Parameters

value	[in] New Boolean value.

5.88.3.15 def sparksee.Value.set_double (*self, value*)

Sets the Value.

Parameters

value	[in] New Double value.

Returns

The calling instance.

5.88.3.16 def sparksee.Value.set_double_void (*self, value*)

Sets the Value.

Parameters

value	[in] New Double value.

5.88.3.17 def sparksee.Value.set_integer (*self, value*)

Sets the Value.

Parameters

value	[in] New Integer value.

Returns

The calling instance.

5.88.3.18 def sparksee.Value.set_integer_void (*self, value*)

Sets the Value.

Parameters

value	[in] New Integer value.

5.88.3.19 def sparksee.Value.set_long (*self, value*)

Sets the Value.

Parameters

value	[in] New Long value.

Returns

The calling instance.

5.88.3.20 def sparksee.Value.set_long_void (*self, value*)

Sets the Value.

Parameters

value	[in] New Long value.

5.88.3.21 def sparksee.Value.set_null (*self*)

Sets the Value to NULL.

Returns

The calling instance.

5.88.3.22 def sparksee.Value.set_oid (*self, value*)

Sets the Value.

Parameters

value	[in] New OID Value.

Returns

The calling instance.

5.88.3.23 def sparksee.Value.set_oid_void (*self, value*)

Sets the OID Value.

Parameters

value	[in] New OID value.

5.88.3.24 def sparksee.Value.set_string (*self, value*)

Sets the Value.

Parameters

value	[in] New String value.

Returns

The calling instance.

5.88.3.25 def sparksee.Value.set_string_void (*self, value*)

Sets the Value.

Parameters

value	[in] New String value.

5.88.3.26 def sparksee.Value.set_timestamp (*self, value*)

Sets the Value.

Parameters

value	[in] New Timestamp value.

Returns

The calling instance.

Referenced by sparksee.Value.set_timestamp().

5.88.3.27 def sparksee.Value.set_timestamp (*self, year, month, day, hour, minutes, seconds, millisecs*)

Sets the Value.

Parameters

year	[in] The year (>=1970).
month	[in] The month ([1..12]).
day	[in] The of the month ([1..31]).
hour	[in] The hour ([0..23]).
minutes	[in] The minutes ([0..59]).
seconds	[in] The seconds ([0..59]).
millisecs	[in] The milliseconds ([0..999]).

Returns

The calling instance.

References sparksee.Value.set_timestamp().

5.88.3.28 def sparksee.Value.set_timestamp_void (*self, year, month, day, hour, minutes, seconds, millisecs*)

Sets the Value.

Parameters

year	[in] The year (>=1970).
month	[in] The month ([1..12]).
day	[in] The of the month ([1..31]).
hour	[in] The hour ([0..23]).
minutes	[in] The minutes ([0..59]).
seconds	[in] The seconds ([0..59]).
millisecs	[in] The milliseconds ([0..999]).

Referenced by sparksee.Value.set_timestamp_void().

5.88.3.29 def sparksee.Value.set_timestamp_void (*self, value*)

Sets the Value.

Parameters

value	[in] New Timestamp value.

References sparksee.Value.set_timestamp_void().

5.88.3.30 def sparksee.Value.set_void (*self, value*)

Sets the Value.

Parameters

value	[in] New value.

5.88.3.31 def sparksee.Value.to_string (*self*)

Returns a String representation of the Value, used in **unicode** and **str**

5.89 sparksee.ValueArray Class Reference

ValueArray class.

Public Member Functions

- def set (self, index, value)

  Set a Value to a specific array position.
- def get_integer (self)

  Get all the values from this int array.
- def get_long (self)

  Get all the values from this long array.
- def set_timestamp (self, values)

  Set all the values of this timestamp array.
- def set_oid (self, values)

  Set all the values of this oid array.
- def get_oid (self)

  Get all the values from this oid array.
- def get (self, index, value)

  Get a Value from the array.
- def get_timestamp (self)

  Get all the values from this timestamp array.
- def get_double_range (self, index, size)

  Get a subset of the values from this double array.
- def get_integer_range (self, index, size)

  Get a subset of the values from this integer array.
- def get_boolean (self)

  Get all the values from this bool array.
- def set_boolean (self, values)

  Set all the values of this bool array.
- def set_double_range (self, index, values)

  Set a subset of the values from this double array.
- def size (self)

  Returns the array size.
- def get_long_range (self, index, size)

  Get a subset of the values from this long array.
- def set_timestamp_range (self, index, values)

  Set a subset of the values from this timestamp array.
- def make_null (self)

  Sets the attribute array to Null.
- def set_integer_range (self, index, values)

  Set a subset of the values from this integer array.
- def get_oid_range (self, index, size)

  Get a subset of the values from this oid array.
- def set_boolean_range (self, index, values)

Set a subset of the values from this boolean array.

- def set_oid_range (self, index, values)

  Set a subset of the values from this oid array.

- def get_double (self)

  Get all the values from this double array.

- def set_long (self, values)

  Set all the values of this long array.

- def set_double (self, values)

  Set all the values of this double array.

- def set_integer (self, values)

  Set all the values of this int array.

- def set_long_range (self, index, values)

  Set a subset of the values from this long array.

- def set (self, value)

  Set a Value to the whole array.

- def get_boolean_range (self, index, size)

  Get a subset of the values from this boolean array.

- def get_timestamp_range (self, index, size)

  Get a subset of the values from this timestamp array.

5.89.1 Detailed Description

ValueArray class.

It allows for getting and setting ValueArray attribute values.

It is very important to close the ValueArray once no more get or set operations will be performed.

Creation of a new ValueArray: (i) Set all the ValueArray elements using Graph::SetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Use of an existing ValueArray: (i) Get a ValueArray instance using Graph::GetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.89.2 Member Function Documentation

5.89.2.1 def sparksee.ValueArray.get (*self, index, value*)

Get a Value from the array.

AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the element to get [0..N-1]
value	[out] Value to get the array element

Exceptions

ValueError	If the index is out of range or the Value not valid
RuntimeError	null

5.89.2.2 def sparksee.ValueArray.get_boolean (*self*)

Get all the values from this bool array.

UnsupportedOperationErrorIf array DataType is not bool

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.3 def sparksee.ValueArray.get_boolean_range (*self, index, size*)

Get a subset of the values from this boolean array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.4 def sparksee.ValueArray.get_double (*self*)

Get all the values from this double array.

UnsupportedOperationErrorIf array DataType is not Double

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.5 def sparksee.ValueArray.get_double_range (*self, index, size*)

Get a subset of the values from this double array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.6 def sparksee.ValueArray.get_integer (*self*)

Get all the values from this int array.

UnsupportedOperationErrorIf array DataType is not int

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.7 def sparksee.ValueArray.get_integer_range (*self, index, size*)

Get a subset of the values from this integer array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.8 def sparksee.ValueArray.get_long (*self*)

Get all the values from this long array.

UnsupportedOperationErrorIf array DataType is not long

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.9 def sparksee.ValueArray.get_long_range (*self, index, size*)

Get a subset of the values from this long array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.10 def sparksee.ValueArray.get_oid (*self*)

Get all the values from this oid array.

UnsupportedOperationErrorIf array DataType is not oid

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.11 def sparksee.ValueArray.get_oid_range (*self, index, size*)

Get a subset of the values from this oid array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.12 def sparksee.ValueArray.get_timestamp (*self*)

Get all the values from this timestamp array.

UnsupportedOperationErrorIf array DataType is not timestamp

Returns

All the array values

Exceptions

RuntimeError	If the ValueArray is not available
RuntimeError	null

5.89.2.13 def sparksee.ValueArray.get_timestamp_range (*self, index, size*)

Get a subset of the values from this timestamp array.

Returns all the requested values

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

5.89.2.14 def sparksee.ValueArray.make_null (*self*)

Sets the attribute array to Null.

The ValueArray can not be used after this call.

Exceptions

RuntimeError	null

5.89.2.15 def sparksee.ValueArray.set (*self, index, value*)

Set a Value to a specific array position.

AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the element to set [0..N-1]
value	[in] Value to set in the array element

Exceptions

ValueError	If the index is out of range or the Value not valid
RuntimeError	null

Referenced by sparksee.ValueArray.set().

5.89.2.16 def sparksee.ValueArray.set (*self, value*)

Set a Value to the whole array.

AppErrorIf the ValueArray is not available

Parameters

value	[in] Value to set in all the array elements

Exceptions

ValueError	If the Value is not valid
RuntimeError	null

References sparksee.ValueArray.set().

5.89.2.17 def sparksee.ValueArray.set_boolean (*self, values*)

Set all the values of this bool array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not bool

5.89.2.18 def sparksee.ValueArray.set_boolean_range (*self, index, values*)

Set a subset of the values from this boolean array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.89.2.19 def sparksee.ValueArray.set_double (*self, values*)

Set all the values of this double array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not Double

5.89.2.20 def sparksee.ValueArray.set_double_range (*self, index, values*)

Set a subset of the values from this double array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.89.2.21 def sparksee.ValueArray.set_integer (*self, values*)

Set all the values of this int array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not int

5.89.2.22 def sparksee.ValueArray.set_integer_range (*self, index, values*)

Set a subset of the values from this integer array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.89.2.23 def sparksee.ValueArray.set_long (*self, values*)

Set all the values of this long array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not long

5.89.2.24 def sparksee.ValueArray.set_long_range (*self, index, values*)

Set a subset of the values from this long array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.89.2.25 def sparksee.ValueArray.set_oid (*self, values*)

Set all the values of this oid array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not oid

5.89.2.26 def sparksee.ValueArray.set_oid_range (*self, index, values*)

Set a subset of the values from this oid array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.89.2.27 def sparksee.ValueArray.set_timestamp (*self, values*)

Set all the values of this timestamp array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

RuntimeError	null
RuntimeError	If array DataType is not timestamp

5.89.2.28 def sparksee.ValueArray.set_timestamp_range (*self, index, values*)

Set a subset of the values from this timestamp array.

Parameters

index	[in] Position of the first element to get [0..N-1]
values	[in] All the values to set

5.90 sparksee.ValueList Class Reference

Value list.

Public Member Functions

- def clear (self)
  Clears the list.
- def get (self, index)
  Returns the Value at the specified position in the list.
- def __iter__ (self)
  Gets a new ValueListIterator.
- def iterator (self)
  Gets a new ValueListIterator.
- def __init__ (self)
  Constructor.
- def add (self, value)
  Adds a value to the end of the list.
- def count (self)
  Number of elements in the list.

5.90.1 Detailed Description

Value list.

It stores a Value list.

Use ValueListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.90.2 Constructor & Destructor Documentation

5.90.2.1 def sparksee.ValueList.__init__ (*self*)

Constructor.

This creates an empty list.

5.90.3 Member Function Documentation

5.90.3.1 def sparksee.ValueList.__iter__ (*self*)

Gets a new ValueListIterator.

Returns

ValueListIterator instance

5.90.3.2 def sparksee.ValueList.add (*self, value*)

Adds a value to the end of the list.

Parameters

value	[in] The value to add

5.90.3.3 def sparksee.ValueList.count (*self*)

Number of elements in the list.

Returns

Number of elements in the list.

5.90.3.4 def sparksee.ValueList.get (*self, index*)

Returns the Value at the specified position in the list.

Parameters

index	[in] Index of the element to return, starting at 0.

5.90.3.5 def sparksee.ValueList.iterator (*self*)

Gets a new ValueListIterator.

Returns

ValueListIterator instance.

5.91 sparksee.ValueListIterator Class Reference

ValueList iterator class.

Public Member Functions

- def next (self)

  Moves to the next element.
- def has_next (self)

  Gets if there are more elements.
- def __next__ (self)

  Used in next()

5.91.1 Detailed Description

ValueList iterator class.

Iterator to traverse all the values into a ValueList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.91.2 Member Function Documentation

5.91.2.1 def sparksee.ValueListIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.91.2.2 def sparksee.ValueListIterator.has_next (*self*)

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

5.91.2.3 def sparksee.ValueListIterator.next (*self*)

Moves to the next element.

Returns

The next element.

5.92 sparksee.Values Class Reference

Value set class.

Public Member Functions

- def count (self)

  Gets the number of elements into the collection.
- def __iter__ (self)

  Gets a new ValuesIterator.
- def iterator (self, order)

  Gets a ValuesIterator.
- def close (self)

  Closes the Values instance.
- def is_closed (self)

  Gets if Values instance has been closed or not.

5.92.1 Detailed Description

Value set class.

This is a set of Value instances, that is there is no duplicated elements.

Use a ValuesIterator to traverse all the elements into the set.

When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.92.2 Member Function Documentation

5.92.2.1 def sparksee.Values.__iter__ (*self*)

Gets a new ValuesIterator.

Returns

ValuesIterator instance

5.92.2.2 def sparksee.Values.close (*self*)

Closes the Values instance.

It must be called to ensure the integrity of all data.

5.92.2.3 def sparksee.Values.count (*self*)

Gets the number of elements into the collection.

Returns

The number of elements into the collection.

5.92.2.4 def sparksee.Values.is_closed (*self*)

Gets if Values instance has been closed or not.

See also

close()

Returns

TRUE if the Values instance has been closed, FALSE otherwise.

5.92.2.5 def sparksee.Values.iterator (*self, order*)

Gets a ValuesIterator.

Parameters

order	[in] Ascending or descending order.

Returns

ValuesIterator instance.

5.93 sparksee.ValuesIterator Class Reference

Values iterator class.

Public Member Functions

- def next (self)

  Gets the next element to traverse.
- def has_next (self)

Gets if there are more elements to traverse.

- def close (self)

  Closes the ValuesIterator instance.

- def __next__ (self)

  Used in next()

- def is_closed (self)

  Gets if ValuesIterator instance has been closed or not.

5.93.1 Detailed Description

Values iterator class.

It allows for traversing all the elements into a Values instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.93.2 Member Function Documentation

5.93.2.1 def sparksee.ValuesIterator.__next__ (*self*)

Used in next()

Returns

The next element

5.93.2.2 def sparksee.ValuesIterator.close (*self*)

Closes the ValuesIterator instance.

It must be called to ensure the integrity of all data.

5.93.2.3 def sparksee.ValuesIterator.has_next (*self*)

Gets if there are more elements to traverse.

Returns

TRUE if there are more elements to traverse, FALSE otherwise.

5.93.2.4 def sparksee.ValuesIterator.is_closed (*self*)

Gets if ValuesIterator instance has been closed or not.

See also

close()

Returns

TRUE if the ValuesIterator instance has been closed, FALSE otherwise.

5.93.2.5 def sparksee.ValuesIterator.next (*self*)

Gets the next element to traverse.

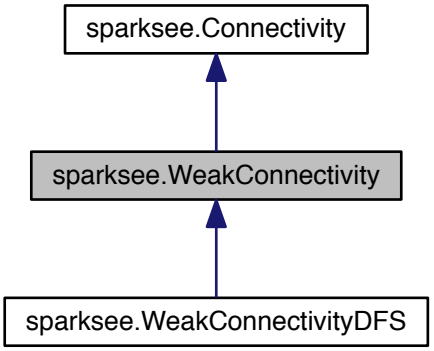
Returns

The next element.

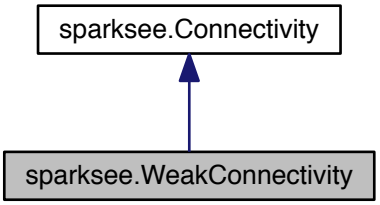
5.94 sparksee.WeakConnectivity Class Reference

WeakConnectivity class.

Inheritance diagram for sparksee.WeakConnectivity:



Collaboration diagram for sparksee.WeakConnectivity:



Public Member Functions

- def run (self)

  Runs the algorithm in order to find the connected components.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self)

  Allows connectivity through all edge types of the graph.
- def get_connected_components (self)

  Returns the results generated by the execution of the algorithm.
- def add_edge_type (self, type)

  Allows connectivity through edges of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_node_type (self, t)

  Allows connectivity through nodes of the given type.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def close (self)

  Closes the Connectivity instance.
- def is_closed (self)

  Gets if Connectivity has been closed or not.

5.94.1 Detailed Description

WeakConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.94.2 Member Function Documentation

5.94.2.1 def sparksee.WeakConnectivity.add_all_edge_types (*self*)

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

5.94.2.2 def sparksee.WeakConnectivity.add_edge_type (*self, type*)

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

Parameters

type	[in] Edge type.

5.94.2.3 def sparksee.WeakConnectivity.add_node_type (*self, t*)

Allows connectivity through nodes of the given type.

Parameters

t	null

5.94.2.4 def sparksee.Connectivity.close (*self*) [inherited]

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

5.94.2.5 def sparksee.WeakConnectivity.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.94.2.6 def sparksee.WeakConnectivity.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.94.2.7 def sparksee.WeakConnectivity.get_connected_components (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

5.94.2.8 def sparksee.Connectivity.is_closed (*self*) [inherited]

Gets if Connectivity has been closed or not.

See also

close()

Returns

TRUE if the Connectivity instance has been closed, FALSE otherwise.

5.94.2.9 def sparksee.WeakConnectivity.run (*self*)

Runs the algorithm in order to find the connected components.

This method can be called only once.

5.94.2.10 def sparksee.WeakConnectivity.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

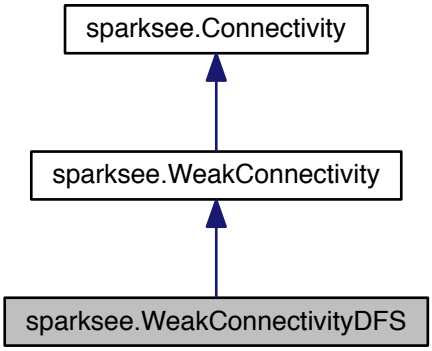
Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

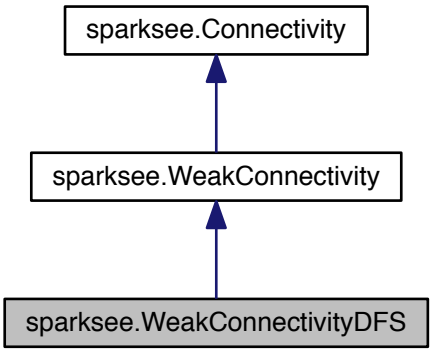
5.95 sparksee.WeakConnectivityDFS Class Reference

WeakConnectivityDFS class.

Inheritance diagram for sparksee.WeakConnectivityDFS:



Collaboration diagram for sparksee.WeakConnectivityDFS:



Public Member Functions

- def __init__ (self, session)

  Creates a new instance of WeakConnectivityDFS.
- def exclude_nodes (self, nodes)

  Set which nodes can't be used.
- def add_all_edge_types (self)

  Allows connectivity through all edge types of the graph.
- def get_connected_components (self)

  Returns the results generated by the execution of the algorithm.
- def run (self)

  Executes the algorithm.

- def add_edge_type (self, type)

  Allows connectivity through edges of the given type.
- def exclude_edges (self, edges)

  Set which edges can't be used.
- def add_node_type (self, t)

  Allows connectivity through nodes of the given type.
- def set_materialized_attribute (self, attribute_name)

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- def add_all_node_types (self)

  Allows connectivity through all node types of the graph.
- def close (self)

  Closes the Connectivity instance.
- def is_closed (self)

  Gets if Connectivity has been closed or not.

5.95.1 Detailed Description

WeakConnectivityDFS class.

This class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u. This implementation is based on the Depth-First Search (DFS) algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

5.95.2 Constructor & Destructor Documentation

5.95.2.1 def sparksee.WeakConnectivityDFS.__init__ (*self, session*)

Creates a new instance of WeakConnectivityDFS.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the weak connected components.

Parameters

session	[in] Session to get the graph from and calculate the connectivity

5.95.3 Member Function Documentation

5.95.3.1 def sparksee.WeakConnectivityDFS.add_all_edge_types (*self*)

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

5.95.3.2 def sparksee.WeakConnectivityDFS.add_edge_type (*self, type*)

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

Parameters

type	[in] Edge type.

5.95.3.3 def sparksee.WeakConnectivityDFS.add_node_type (*self, t*)

Allows connectivity through nodes of the given type.

Parameters

t	null

5.95.3.4 def sparksee.Connectivity.close (*self*) [inherited]

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

5.95.3.5 def sparksee.WeakConnectivityDFS.exclude_edges (*self, edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.95.3.6 def sparksee.WeakConnectivityDFS.exclude_nodes (*self, nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

5.95.3.7 def sparksee.WeakConnectivityDFS.get_connected_components (*self*)

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

5.95.3.8 def sparksee.Connectivity.is_closed (*self*) [inherited]

Gets if Connectivity has been closed or not.

See also

close()

Returns

TRUE if the Connectivity instance has been closed, FALSE otherwise.

5.95.3.9 def sparksee.WeakConnectivityDFS.set_materialized_attribute (*self, attribute_name*)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attribute_name	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

Index