

SPARKSEE

Sparsity Technologies

www.sparsity-technologies.com - 2011

Package

com.sparsity.sparksee.algorithms

com.sparsity.sparksee.algorithms

Class CommunitiesSCD

```

java.lang.Object
├── com.sparsity.sparksee.algorithms.CommunityDetection
│   ├── com.sparsity.sparksee.algorithms.DisjointCommunityDetection
│   │   └── com.sparsity.sparksee.algorithms.CommunitiesSCD

```

All Implemented Interfaces:
Closeable

```

public class CommunitiesSCD
extends DisjointCommunityDetection

```

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-Lluís 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>

Constructor Summary

public	CommunitiesSCD (Session session) Creates a new instance of CommunitiesSCD.
--------	--

Method Summary

void	addAllEdgeTypes () Allows connectivity through all edge types of the graph.
void	addAllNodeTypes () Allows connectivity through all node types of the graph.
void	addEdgeType (int type) Allows connectivity through edges of the given type.
void	addNodeType (int type) Allows connectivity through nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.

void	<code>excludeNodes(Objects nodes)</code> Set which nodes can't be used.
<code>DisjointCommunities</code>	<code>getCommunities()</code> Returns the results generated by the execution of the algorithm.
void	<code>run()</code> Executes the algorithm.
void	<code>setLookAhead(int lookahead)</code> Sets the size of the lookahead iterations to look.
void	<code>setMaterializedAttribute(String attributeName)</code> 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.DisjointCommunityDetection](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [excludeEdges](#), [excludeNodes](#), [getCommunities](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class [com.sparsity.sparksee.algorithms.CommunityDetection](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [isClosed](#), [run](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Constructors

CommunitiesSCD

```
public CommunitiesSCD(Session 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

Methods

addNodeType

```
public void addNodeType(int type)
```

(continued on next page)

(continued from last page)

Allows connectivity through nodes of the given type.

Parameters:

type - null

addEdgeType

```
public void addEdgeType(int type)
```

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters:

type - [in] Edge type.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

addAllEdgeTypes

```
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

excludeNodes

```
public void excludeNodes(Objects 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.

run

```
public void run()
```

Executes the algorithm.

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

(continued on next page)

(continued from last page)

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:

`attributeName` - [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.

getCommunities

```
public DisjointCommunities getCommunities()
```

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.

excludeEdges

```
public void excludeEdges(Objects 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.

setLookAhead

```
public void setLookAhead(int lookahead)
```

Sets the size of the lookahead iterations to look.

Parameters:

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

com.sparsity.sparksee.algorithms Class CommunityDetection

java.lang.Object

└─com.sparsity.sparksee.algorithms.CommunityDetection

All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[DisjointCommunityDetection](#)

public class **CommunityDetection**

extends Object

implements Closeable

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>

Method Summary

void	addAllNodeTypes() Allows connectivity through all node types of the graph.
void	addNodeType(int type) Allows connectivity through nodes of the given type.
void	close() Closes the CommunityDetection instance.
void	excludeEdges(Objects edges) Set which edges can't be used.
void	excludeNodes(Objects nodes) Set which nodes can't be used.
boolean	isClosed() Gets if CommunityDetection instance has been closed or not.
void	run() Runs the algorithm in order to find the connected components.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows connectivity through nodes of the given type.

Parameters:

type - null

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

run

```
public void run()
```

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

```
public void excludeNodes(Objects 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.

excludeEdges

```
public void excludeEdges(Objects 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.

(continued from last page)

isClosed

```
public boolean isClosed()
```

Gets if CommunityDetection instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the CommunityDetection instance.

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

com.sparsity.sparksee.algorithms

Class ConnectedComponents

java.lang.Object

└─com.sparsity.sparksee.algorithms.ConnectedComponents

All Implemented Interfaces:

Closeable

```
public class ConnectedComponents
    extends Object
    implements Closeable
```

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>

Constructor Summary

public	ConnectedComponents (Session s, String materializedattribute) Creates a new instance of ConnectedComponents.
--------	--

Method Summary

void	close () Closes the ConnectedComponents instance.
long	getConnectedComponent (long idNode) Returns the connected component where the given node belongs to.
long	getCount () Returns the number of connected components found in the graph.
Objects	getNodes (long idConnectedComponent) Returns the collection of nodes contained in the given connected component.
long	getSize (long idConnectedComponent) Returns the number of nodes contained in the given connected component.
boolean	isClosed () Gets if ConnectedComponents instance has been closed or not.

Methods inherited from class `java.lang.Object``clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`**Methods inherited from interface** `java.io.Closeable``close`

Constructors

ConnectedComponents

```
public ConnectedComponents(Session s,  
                           String 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.

Methods

getSize

```
public long getSize(long idConnectedComponent)
```

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

Parameters:

`idConnectedComponent` - 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.

getCount

```
public long getCount()
```

Returns the number of connected components found in the graph.

Returns:

The number of connected components found in the graph.

(continued from last page)

getConnectedComponent

```
public long getConnectedComponent(long idNode)
```

Returns the connected component where the given node belongs to.

Parameters:

`idNode` - [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.

getNodes

```
public Objects getNodes(long idConnectedComponent)
```

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

Parameters:

`idConnectedComponent` - 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.

isClosed

```
public boolean isClosed()
```

Gets if ConnectedComponents instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the ConnectedComponents instance.

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

com.sparsity.sparksee.algorithms

Class Connectivity

java.lang.Object

└─com.sparsity.sparksee.algorithms.Connectivity

All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[WeakConnectivity](#), [StrongConnectivity](#)

public class **Connectivity**

extends Object

implements Closeable

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>

Method Summary

void	addAllNodeTypes() Allows connectivity through all node types of the graph.
void	addNodeType(int t) Allows connectivity through nodes of the given type.
void	close() Closes the Connectivity instance.
void	excludeEdges(Objects edges) Set which edges can't be used.
void	excludeNodes(Objects nodes) Set which nodes can't be used.
ConnectedComponents	getConnectedComponents() Returns the results generated by the execution of the algorithm.
boolean	isClosed() Gets if Connectivity instance has been closed or not.
void	run() Runs the algorithm in order to find the connected components.
void	setMaterializedAttribute(String attributeName) 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.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.io.Closeable`

`close`

Methods

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

run

```
public void run()
```

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

```
public void excludeNodes(Objects 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.

addNodeType

```
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

Parameters:

`t` - null

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

(continued from last page)

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:

`attributeName` - [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.

excludeEdges

```
public void excludeEdges(Objects 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.

isClosed

```
public boolean isClosed()
```

Gets if Connectivity instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the Connectivity instance.

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

getConnectedComponents

```
public ConnectedComponents getConnectedComponents()
```

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.

com.sparsity.sparksee.algorithms

Class Context

```
java.lang.Object
|
|--com.sparsity.sparksee.algorithms.Context
```

All Implemented Interfaces:
Closeable

```
public class Context
extends Object
implements Closeable
```

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>

Constructor Summary

public	Context (Session session, long node) Creates a new instance.
--------	--

Method Summary

void	addAllEdgeTypes (EdgesDirection d) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int t, EdgesDirection d) Allows for traversing edges of the given type.
void	addNodeType (int t) Allows for traversing nodes of the given type.
void	close () Closes the Context instance.
Objects	compute () Gets the resulting collection of nodes.
static Objects	compute (Session session, long node, TypeList nodeTypes, TypeList edgeTypes, EdgesDirection dir, int maxhops, boolean include) Helper method to easily compute a context from a node.
void	excludeEdges (Objects edges) Set which edges can't be used.

void	excludeNodes (Objects nodes) Set which nodes can't be used.
boolean	isClosed () Gets if Context instance has been closed or not.
void	setMaximumHops (int maxhops, boolean include) Sets the maximum hops restriction.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Constructors

Context

```
public Context(Session session,
              Long node)
```

Creates a new instance.

Parameters:

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

Methods

addEdgeType

```
public void addEdgeType(int t,
                       EdgesDirection d)
```

Allows for traversing edges of the given type.

Parameters:

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

compute

```
public static Objects compute(Session session,
                             Long node,
                             TypeList nodeTypes,
                             TypeList edgeTypes,
                             EdgesDirection dir,
                             int maxhops,
                             boolean include)
```

(continued from last page)

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.
`nodeTypes` - [in] Allowed node type list. NULL means all node types are allowed.
`edgeTypes` - [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.

excludeNodes

```
public void excludeNodes(Objects 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.

compute

```
public Objects compute()
```

Gets the resulting collection of nodes.

Returns:

The resulting collection of nodes.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection d)
```

Allows for traversing all edge types of the graph.

Parameters:

`d` - [in] Edge direction.

addNodeType

```
public void addNodeType(int t)
```

Allows for traversing nodes of the given type.

(continued from last page)

Parameters:

t - null

close

```
public void close()
```

Closes the Context instance.

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

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

setMaximumHops

```
public void setMaximumHops(int maxhops,  
    boolean 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.

excludeEdges

```
public void excludeEdges(Objects 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.

isClosed

```
public boolean isClosed()
```

Gets if Context instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

com.sparsity.sparksee.algorithms

Class DisjointCommunities

java.lang.Object

└─com.sparsity.sparksee.algorithms.DisjointCommunities

All Implemented Interfaces:

Closeable

```
public class DisjointCommunities
    extends Object
    implements Closeable
```

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>

Constructor Summary

public	DisjointCommunities (Session session, String materializedattribute) Creates a new instance of DisjointCommunities.
--------	--

Method Summary

void	close () Closes the DisjointCommunities instance.
long	getCommunity (long idNode) Returns the disjoint community where the given node belongs to.
long	getCount () Returns the number of communities found in the graph.
Objects	getNodes (long idCommunity) Returns the collection of nodes contained in the given community.
long	getSize (long idCommunity) Returns the number of nodes contained in the given community.
boolean	isClosed () Gets if DisjointCommunities instance has been closed or not.

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Methods inherited from interface `java.io.Closeable`

```
close
```

Constructors

DisjointCommunities

```
public DisjointCommunities(Session session,  
                           String 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.

Methods

getSize

```
public long getSize(long idCommunity)
```

Returns the number of nodes contained in the given community.

Parameters:

`idCommunity` - The community for which the number of nodes contained in it will be returned.

Returns:

The number of nodes contained in the given community.

getCount

```
public long getCount()
```

Returns the number of communities found in the graph.

Returns:

The number of communities found in the graph.

getNodes

```
public Objects getNodes(long idCommunity)
```

(continued from last page)

Returns the collection of nodes contained in the given community.

Parameters:

`idCommunity` - 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.

getCommunity

```
public long getCommunity(long idNode)
```

Returns the disjoint community where the given node belongs to.

Parameters:

`idNode` - [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.

isClosed

```
public boolean isClosed()
```

Gets if DisjointCommunities instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

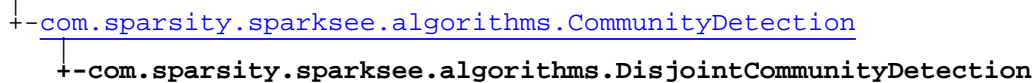
Closes the DisjointCommunities instance.

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

com.sparsity.sparksee.algorithms

Class DisjointCommunityDetection

java.lang.Object



All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[CommunitiesSCD](#)

public class **DisjointCommunityDetection**

extends [CommunityDetection](#)

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>

Method Summary

void	addAllEdgeTypes() Allows connectivity through all edge types of the graph.
void	addAllNodeTypes() Allows connectivity through all node types of the graph.
void	addEdgeType(int type) Allows connectivity through edges of the given type.
void	addNodeType(int type) Allows connectivity through nodes of the given type.
void	excludeEdges(Objects edges) Set which edges can't be used.
void	excludeNodes(Objects nodes) Set which nodes can't be used.
DisjointCommunities	getCommunities() Returns the results generated by the execution of the algorithm.
void	run() Runs the algorithm in order to find the communities.
void	setMaterializedAttribute(String attributeName) 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.CommunityDetection](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [isClosed](#), [run](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows connectivity through nodes of the given type.

Parameters:

`type` - null

addEdgeType

```
public void addEdgeType(int type)
```

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters:

`type` - [in] Edge type.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

addAllEdgeTypes

```
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

excludeNodes

```
public void excludeNodes(Objects nodes)
```


(continued from last page)

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.

run

```
public void run()
```

Runs the algorithm in order to find the communities.

This method can be called only once.

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

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:

attributeName - [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.

getCommunities

```
public DisjointCommunities getCommunities()
```

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.

excludeEdges

```
public void excludeEdges(Objects 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.

com.sparsity.sparksee.algorithms

Class ShortestPath

java.lang.Object

└─com.sparsity.sparksee.algorithms.ShortestPath

All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[SinglePairShortestPath](#)

public class **ShortestPath**

extends Object

implements Closeable

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>

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	close () Closes the ShortestPath instance.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
boolean	isClosed () Gets if ShortestPath instance has been closed or not.
void	run () Runs the algorithm.

void	setMaximumHops (int maxhops) Sets the maximum hops restriction.
------	--

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

`type` - null

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

run

```
public void run()
```

Runs the algorithm.

This method can only be called once.

excludeNodes

```
public void excludeNodes(Objects 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.

(continued from last page)

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.
dir - [in] Edge direction.

excludeEdges

```
public void excludeEdges(Objects 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.

setMaximumHops

```
public void setMaximumHops(int 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.

isClosed

```
public boolean isClosed()
```

Gets if ShortestPath instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

Parameters:

dir - [in] Edge direction.

close

```
public void close()
```

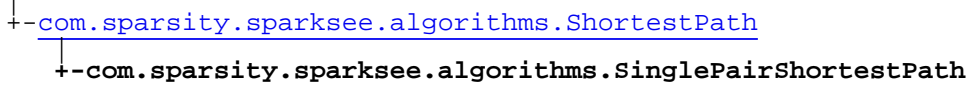
Closes the ShortestPath instance.

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

com.sparsity.sparksee.algorithms

Class SinglePairShortestPath

java.lang.Object



All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[SinglePairShortestPathDijkstra](#), [SinglePairShortestPathBFS](#)

public class **SinglePairShortestPath**

extends [ShortestPath](#)

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>

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
boolean	exists () Returns TRUE If a path exists or FALSE otherwise.
double	getCost () Gets the cost of the shortest path.
OIDList	getPathAsEdges () Gets the shortest path between the source node and the destination node as an ordered set of edges.

OIDList	getPathAsNodes() Gets the shortest path between the source node and the destination node as an ordered set of nodes.
void	run() Runs the algorithm.
void	setMaximumHops(int maxhops) Sets the maximum hops restriction.

Methods inherited from class [com.sparsity.sparksee.algorithms.ShortestPath](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [isClosed](#), [run](#), [setMaximumHops](#)

Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [java.io.Closeable](#)

[close](#)

Methods

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

type - null

excludeNodes

```
public void excludeNodes(Objects 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.

run

```
public void run()
```

Runs the algorithm.

This method can only be called once.

getPathAsEdges

```
public OIDList getPathAsEdges()
```

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

setMaximumHops

```
public void setMaximumHops(int 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

Parameters:

dir - [in] Edge direction.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

(continued from last page)

type - [in] Edge type.
dir - [in] Edge direction.

getCost

```
public double getCost()
```

Gets the cost of the shortest path.

The cost for unweighted algorithms is the number of hops of the shortest path. For weighted algorithms, the cost is the sum of the costs of the edges of the shortest path.

Returns:

The cost of the shortest path.

getPathAsNodes

```
public OIDList getPathAsNodes()
```

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

excludeEdges

```
public void excludeEdges(Objects 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.

com.sparsity.sparksee.algorithms

Class SinglePairShortestPathBFS

```

java.lang.Object
├── com.sparsity.sparksee.algorithms.ShortestPath
│   ├── com.sparsity.sparksee.algorithms.SinglePairShortestPath
│       └── com.sparsity.sparksee.algorithms.SinglePairShortestPathBFS

```

All Implemented Interfaces:
Closeable

public class **SinglePairShortestPathBFS**
extends [SinglePairShortestPath](#)

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>

Constructor Summary

public	SinglePairShortestPathBFS (Session session, long src, long dst) Creates a new instance.
--------	---

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	checkOnlyExistence () Set that only the path existence must be calculated and not the path itself.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
boolean	exists () Returns TRUE If a path exists or FALSE otherwise.

double	getCost() Gets the cost of the shortest path.
OIDList	getPathAsEdges() Gets the shortest path between the source node and the destination node as an ordered set of edges.
OIDList	getPathAsNodes() Gets the shortest path between the source node and the destination node as an ordered set of nodes.
void	run() Executes the algorithm.
void	setMaximumHops(int maxhops) Sets the maximum hops restriction.

Methods inherited from class [com.sparsity.sparksee.algorithms.SinglePairShortestPath](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [excludeEdges](#), [excludeNodes](#), [exists](#), [getCost](#), [getPathAsEdges](#), [getPathAsNodes](#), [run](#), [setMaximumHops](#)

Methods inherited from class [com.sparsity.sparksee.algorithms.ShortestPath](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [isClosed](#), [run](#), [setMaximumHops](#)

Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [java.io.Closeable](#)

[close](#)

Constructors

SinglePairShortestPathBFS

```
public SinglePairShortestPathBFS(Session session,
                                long src,
                                long 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.

Methods

(continued from last page)

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

type - null

excludeNodes

```
public void excludeNodes(Objects 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.

getPathAsEdges

```
public OIDList getPathAsEdges()
```

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

getPathAsNodes

```
public OIDList getPathAsNodes()
```

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

setMaximumHops

```
public void setMaximumHops(int maxhops)
```

(continued from last page)

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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

Parameters:

dir - [in] Edge direction.

getCost

```
public double getCost()
```

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.

dir - [in] Edge direction.

run

```
public void run()
```

Executes the algorithm.

(continued from last page)

checkOnlyExistence

```
public void checkOnlyExistence()
```

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.

excludeEdges

```
public void excludeEdges(Objects 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.

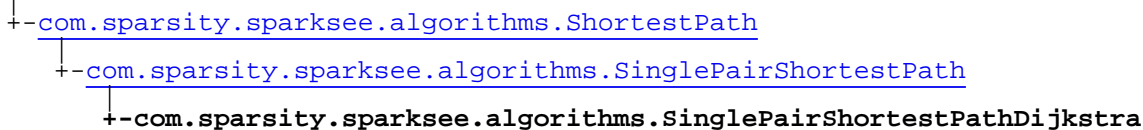
Parameters:

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

com.sparsity.sparksee.algorithms

Class SinglePairShortestPathDijkstra

```
java.lang.Object
```



All Implemented Interfaces:

Closeable

```
public class SinglePairShortestPathDijkstra
```

```
extends SinglePairShortestPath
```

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>

Constructor Summary

public	SinglePairShortestPathDijkstra (Session session, long src, long dst) Creates a new instance.
--------	--

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	addWeightedEdgeType (int type, EdgesDirection dir, int attr) Allows for traversing edges of the given type using the given attribute as the weight.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.

boolean	<u>exists()</u> Returns TRUE If a path exists or FALSE otherwise.
double	<u>getCost()</u> Gets the cost of the shortest path.
<u>OIDList</u>	<u>getPathAsEdges()</u> Gets the shortest path between the source node and the destination node as an ordered set of edges.
<u>OIDList</u>	<u>getPathAsNodes()</u> Gets the shortest path between the source node and the destination node as an ordered set of nodes.
void	<u>run()</u> Executes the algorithm.
void	<u>setMaximumHops(int maxhops)</u> Sets the maximum hops restriction.
void	<u>setUnweightedEdgeCost(double weight)</u> Sets the weight assigned to the unweighted edges.

Methods inherited from class [com.sparsity.sparksee.algorithms.SinglePairShortestPath](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [excludeEdges](#), [excludeNodes](#), [exists](#), [getCost](#), [getPathAsEdges](#), [getPathAsNodes](#), [run](#), [setMaximumHops](#)

Methods inherited from class [com.sparsity.sparksee.algorithms.ShortestPath](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [isClosed](#), [run](#), [setMaximumHops](#)

Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [java.io.Closeable](#)

[close](#)

Constructors

SinglePairShortestPathDijkstra

```
public SinglePairShortestPathDijkstra(Session session,
                                       long src,
                                       long dst)
```

Creates a new instance.

Parameters:

`session` - [in] Session to get the graph from and perform traversal.
`src` - [in] Source node.

(continued from last page)

dst - [dst] Destination node.

Methods

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

type - null

excludeNodes

```
public void excludeNodes(Objects 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.

getPathAsEdges

```
public OIDList getPathAsEdges()
```

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

getPathAsNodes

```
public OIDList getPathAsNodes()
```

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

(continued from last page)

setUnweightedEdgeCost

```
public void setUnweightedEdgeCost(double 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.

setMaximumHops

```
public void setMaximumHops(int 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

Parameters:

dir - [in] Edge direction.

getCost

```
public double getCost()
```

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.

addWeightedEdgeType

```
public void addWeightedEdgeType(int type,  
    EdgesDirection dir,  
    int 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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.

dir - [in] Edge direction.

run

```
public void run()
```

Executes the algorithm.

excludeEdges

```
public void excludeEdges(Objects 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.

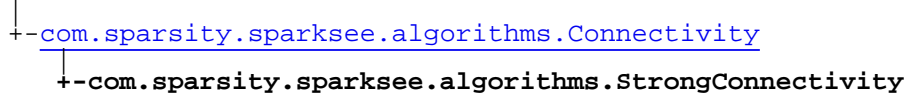
Parameters:

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

com.sparsity.sparksee.algorithms

Class StrongConnectivity

java.lang.Object



All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[StrongConnectivityGabow](#)

public class **StrongConnectivity**
extends [Connectivity](#)

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>

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows connectivity through all edge types of the graph.
void	addAllNodeTypes () Allows connectivity through all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows connectivity through edges of the given type.
void	addNodeType (int t) Allows connectivity through nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
ConnectedComponents	getConnectedComponents () Returns the results generated by the execution of the algorithm.

void	run() Runs the algorithm in order to find the connected components.
void	setMaterializedAttribute (String attributeName) 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.Connectivity](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [isClosed](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

run

```
public void run()
```

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

```
public void excludeNodes(Objects 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.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

(continued from last page)

Allows connectivity through edges of the given type.

Parameters:

type - [in] Edge type.
dir - [in] Edge direction.

addNodeType

```
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

Parameters:

t - null

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

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:

attributeName - [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.

excludeEdges

```
public void excludeEdges(Objects 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows connectivity through all edge types of the graph.

Parameters:

dir - [in] Edge direction.

getConnectedComponents

```
public ConnectedComponents getConnectedComponents ( )
```

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.

com.sparsity.sparksee.algorithms

Class StrongConnectivityGabow

```

java.lang.Object
├── com.sparsity.sparksee.algorithms.Connectivity
│   ├── com.sparsity.sparksee.algorithms.StrongConnectivity
│       └── com.sparsity.sparksee.algorithms.StrongConnectivityGabow

```

All Implemented Interfaces:

Closeable

```

public class StrongConnectivityGabow
extends StrongConnectivity

```

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>

Constructor Summary

public	StrongConnectivityGabow (Session session) Creates a new instance of StrongConnectivityGabow.
--------	--

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows connectivity through all edge types of the graph.
void	addAllNodeTypes () Allows connectivity through all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows connectivity through edges of the given type.
void	addNodeType (int t) Allows connectivity through nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.

ConnectedComponents	getConnectedComponents() Returns the results generated by the execution of the algorithm.
void	run() Executes the algorithm.
void	setMaterializedAttribute (String attributeName) 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.StrongConnectivity](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class [com.sparsity.sparksee.algorithms.Connectivity](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [isClosed](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Constructors

StrongConnectivityGabow

```
public StrongConnectivityGabow(Session 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

Methods

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

(continued from last page)

excludeNodes

```
public void excludeNodes(Objects 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.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows connectivity through edges of the given type.

Parameters:

type - [in] Edge type.

dir - [in] Edge direction.

run

```
public void run()
```

Executes the algorithm.

addNodeType

```
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

Parameters:

t - null

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

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:

attributeName - [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.

excludeEdges

```
public void excludeEdges(Objects 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows connectivity through all edge types of the graph.

Parameters:

dir - [in] Edge direction.

getConnectedComponents

```
public ConnectedComponents getConnectedComponents()
```

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.

com.sparsity.sparksee.algorithms

Class Traversal

```
java.lang.Object
|
+--com.sparsity.sparksee.algorithms.Traversal
```

All Implemented Interfaces:
Closeable

Direct Known Subclasses:
[TraversalDFS](#), [TraversalBFS](#)

```
public class Traversal
extends Object
implements Closeable
```

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>

Method Summary		
void	addAllEdgeTypes (EdgesDirection dir)	Allows for traversing all edge types of the graph.
void	addAllNodeTypes ()	Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir)	Allows for traversing edges of the given type.
void	addNodeType (int type)	Allows for traversing nodes of the given type.
void	close ()	Closes the Traversal instance.
void	excludeEdges (Objects edges)	Set which edges can't be used.
void	excludeNodes (Objects nodes)	Set which nodes can't be used.
int	getCurrentDepth ()	Returns the depth of the current node.
boolean	hasNext ()	Gets if there are more objects to be traversed.

boolean	isClosed() Gets if Traversal instance has been closed or not.
long	next() Gets the next object of the traversal.
void	setMaximumHops(int maxhops) Sets the maximum hops restriction.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

type - null

hasNext

```
public boolean hasNext()
```

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

excludeNodes

```
public void excludeNodes(Objects 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.

(continued from last page)

setMaximumHops

```
public void setMaximumHops(int 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

Parameters:

dir - [in] Edge direction.

getCurrentDepth

```
public int getCurrentDepth()
```

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.

close

```
public void close()
```

Closes the Traversal instance.

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

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.

(continued from last page)

`dir` - [in] Edge direction.

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

excludeEdges

```
public void excludeEdges(Objects 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.

isClosed

```
public boolean isClosed()
```

Gets if Traversal instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

com.sparsity.sparksee.algorithms

Class TraversalBFS

```

java.lang.Object
|
+-com.sparsity.sparksee.algorithms.Traversal
   |
   +-com.sparsity.sparksee.algorithms.TraversalBFS

```

All Implemented Interfaces:
Closeable

public class **TraversalBFS**
extends [Traversal](#)

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>

Constructor Summary

public	TraversalBFS (Session session, long node) Creates a new instance.
--------	---

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
int	getCurrentDepth () Returns the depth of the current node.
boolean	hasNext () Gets if there are more objects to be traversed.
long	next () Gets the next object of the traversal.

void	setMaximumHops (int maxhops) Sets the maximum hops restriction.
------	--

Methods inherited from class [com.sparsity.sparksee.algorithms.Traversal](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getCurrentDepth](#), [hasNext](#), [isClosed](#), [next](#), [setMaximumHops](#)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Constructors

TraversalBFS

```
public TraversalBFS(Session session,
                    long node)
```

Creates a new instance.

Parameters:

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

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

`type` - null

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

hasNext

```
public boolean hasNext()
```

(continued from last page)

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

excludeNodes

```
public void excludeNodes(Objects 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.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.

dir - [in] Edge direction.

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

getCurrentDepth

```
public int getCurrentDepth()
```

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.

setMaximumHops

```
public void setMaximumHops(int maxhops)
```

(continued from last page)

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.

excludeEdges

```
public void excludeEdges(Objects 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

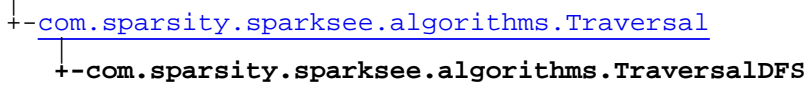
Parameters:

dir - [in] Edge direction.

com.sparsity.sparksee.algorithms

Class TraversalDFS

java.lang.Object



All Implemented Interfaces:
Closeable

public class **TraversalDFS**
extends [Traversal](#)

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>

Constructor Summary

public	TraversalDFS (Session session, long node) Creates a new instance.
--------	---

Method Summary

void	addAllEdgeTypes (EdgesDirection dir) Allows for traversing all edge types of the graph.
void	addAllNodeTypes () Allows for traversing all node types of the graph.
void	addEdgeType (int type, EdgesDirection dir) Allows for traversing edges of the given type.
void	addNodeType (int type) Allows for traversing nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.
void	excludeNodes (Objects nodes) Set which nodes can't be used.
int	getCurrentDepth () Returns the depth of the current node.
boolean	hasNext () Gets if there are more objects to be traversed.
long	next () Gets the next object of the traversal.

void	setMaximumHops (int maxhops) Sets the maximum hops restriction.
------	--

Methods inherited from class [com.sparsity.sparksee.algorithms.Traversal](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getCurrentDepth](#), [hasNext](#), [isClosed](#), [next](#), [setMaximumHops](#)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Constructors

TraversalDFS

```
public TraversalDFS(Session session,
                    long node)
```

Creates a new instance.

Parameters:

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

Methods

addNodeType

```
public void addNodeType(int type)
```

Allows for traversing nodes of the given type.

Parameters:

`type` - null

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

hasNext

```
public boolean hasNext()
```

(continued from last page)

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

excludeNodes

```
public void excludeNodes(Objects 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.

addEdgeType

```
public void addEdgeType(int type,  
    EdgesDirection dir)
```

Allows for traversing edges of the given type.

Parameters:

type - [in] Edge type.

dir - [in] Edge direction.

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

getCurrentDepth

```
public int getCurrentDepth()
```

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.

setMaximumHops

```
public void setMaximumHops(int maxhops)
```

(continued from last page)

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.

excludeEdges

```
public void excludeEdges(Objects 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.

addAllEdgeTypes

```
public void addAllEdgeTypes(EdgesDirection dir)
```

Allows for traversing all edge types of the graph.

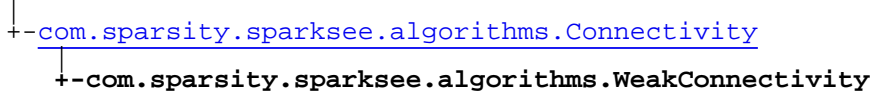
Parameters:

dir - [in] Edge direction.

com.sparsity.sparksee.algorithms

Class WeakConnectivity

java.lang.Object



All Implemented Interfaces:

Closeable

Direct Known Subclasses:

[WeakConnectivityDFS](#)

```
public class WeakConnectivity
extends Connectivity
```

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>

Method Summary

void	addAllEdgeTypes() Allows connectivity through all edge types of the graph.
void	addAllNodeTypes() Allows connectivity through all node types of the graph.
void	addEdgeType(int type) Allows connectivity through edges of the given type.
void	addNodeType(int t) Allows connectivity through nodes of the given type.
void	excludeEdges(Objects edges) Set which edges can't be used.
void	excludeNodes(Objects nodes) Set which nodes can't be used.
ConnectedComponents	getConnectedComponents() Returns the results generated by the execution of the algorithm.

void	<u>run()</u> Runs the algorithm in order to find the connected components.
void	<u>setMaterializedAttribute</u> (String attributeName) 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.Connectivity](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [isClosed](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods

addEdgeType

```
public void addEdgeType(int 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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

addAllEdgeTypes

```
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

run

```
public void run()
```

(continued from last page)

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

```
public void excludeNodes(Objects 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.

addNodeType

```
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

Parameters:

t - null

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

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:

attributeName - [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.

excludeEdges

```
public void excludeEdges(Objects 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.

getConnectedComponents

```
public ConnectedComponents getConnectedComponents( )
```

(continued from last page)

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.

com.sparsity.sparksee.algorithms

Class WeakConnectivityDFS

```

java.lang.Object
├── com.sparsity.sparksee.algorithms.Connectivity
│   ├── com.sparsity.sparksee.algorithms.WeakConnectivity
│       └── com.sparsity.sparksee.algorithms.WeakConnectivityDFS

```

All Implemented Interfaces:

Closeable

```

public class WeakConnectivityDFS
extends WeakConnectivity

```

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>

Constructor Summary

public	WeakConnectivityDFS (Session session) Creates a new instance of WeakConnectivityDFS.
--------	--

Method Summary

void	addAllEdgeTypes () Allows connectivity through all edge types of the graph.
void	addAllNodeTypes () Allows connectivity through all node types of the graph.
void	addEdgeType (int type) Allows connectivity through edges of the given type.
void	addNodeType (int t) Allows connectivity through nodes of the given type.
void	excludeEdges (Objects edges) Set which edges can't be used.

void	excludeNodes (Objects nodes) Set which nodes can't be used.
ConnectedComponents	getConnectedComponents () Returns the results generated by the execution of the algorithm.
void	run () Executes the algorithm.
void	setMaterializedAttribute (String attributeName) 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.

Methods inherited from class [com.sparsity.sparksee.algorithms.WeakConnectivity](#)

[addAllEdgeTypes](#), [addAllNodeTypes](#), [addEdgeType](#), [addNodeType](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class [com.sparsity.sparksee.algorithms.Connectivity](#)

[addAllNodeTypes](#), [addNodeType](#), [close](#), [excludeEdges](#), [excludeNodes](#), [getConnectedComponents](#), [isClosed](#), [run](#), [setMaterializedAttribute](#)

Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [java.io.Closeable](#)

[close](#)

Constructors

WeakConnectivityDFS

```
public WeakConnectivityDFS(Session 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

Methods

addEdgeType

```
public void addEdgeType(int type)
```

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

(continued from last page)

Parameters:`type` - [in] Edge type.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

addAllEdgeTypes

```
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

excludeNodes

```
public void excludeNodes(Objects 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.

run

```
public void run()
```

Executes the algorithm.

addNodeType

```
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

Parameters:

`t` - null

setMaterializedAttribute

```
public void setMaterializedAttribute(String attributeName)
```

(continued from last page)

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:

`attributeName` - [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.

excludeEdges

```
public void excludeEdges(Objects 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.

getConnectedComponents

```
public ConnectedComponents getConnectedComponents( )
```

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.

Package

com.sparsity.sparksee.gdb

com.sparsity.sparksee.gdb Class Attribute

java.lang.Object

└─com.sparsity.sparksee.gdb.Attribute

public class **Attribute**
extends Object

Attribute data class.

It contains information about an attribute.

Author:

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

Field Summary

public static	InvalidAttribute Invalid attribute identifier constant.
---------------	--

Method Summary

long	getCount() Gets the number of non-NULL values.
DataType	getDataType() Gets the data type.
int	getId() Gets the Sparksee attribute identifier.
AttributeKind	getKind() Gets the attribute kind.
String	getName() Gets the unique attribute name.
long	getSize() Gets the number of different values.
int	getTypeId() Gets the Sparksee type identifier.
boolean	isSessionAttribute() Check if it's a session attribute or a persistent one.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

(continued from last page)

InvalidAttribute

```
public static int InvalidAttribute
```

Invalid attribute identifier constant.

Methods

getKind

```
public AttributeKind getKind()
```

Gets the attribute kind.

Returns:

The AttributeKind.

getCount

```
public long getCount()
```

Gets the number of non-NULL values.

Returns:

The number of non-NULL values.

isSessionAttribute

```
public boolean isSessionAttribute()
```

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

Returns:

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

getSize

```
public long getSize()
```

Gets the number of different values.

Returns:

The number of different values.

getTypeId

```
public int getTypeId()
```

(continued from last page)

Gets the Sparksee type identifier.

Returns:

The Sparksee type identifier.

getDataType

```
public DataType getDataType()
```

Gets the data type.

Returns:

The DataType.

getId

```
public int getId()
```

Gets the Sparksee attribute identifier.

Returns:

The Sparksee attribute identifier.

getName

```
public String getName()
```

Gets the unique attribute name.

Returns:

The unique attribute name.

com.sparsity.sparksee.gdb Class AttributeKind

```

java.lang.Object
  |
  +- java.lang.Enum
        |
        +- com.sparsity.sparksee.gdb.AttributeKind
  
```

All Implemented Interfaces:
Serializable, Comparable

public final class **AttributeKind**
extends Enum

Attribute kind enumeration.

It determines the indexing-capabilities of an attribute.

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Basic Basic attribute (non indexed attribute).
public static final	Indexed Indexed attribute.
public static final	Unique Unique attribute (indexed + unique restriction).

Method Summary

static AttributeKind	valueOf (String name)
static AttributeKind[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

(continued from last page)

Fields

Basic

```
public static final com.sparsity.sparksee.gdb.AttributeKind Basic
```

Basic attribute (non indexed attribute).

Indexed

```
public static final com.sparsity.sparksee.gdb.AttributeKind Indexed
```

Indexed attribute.

Unique

```
public static final com.sparsity.sparksee.gdb.AttributeKind Unique
```

Unique attribute (indexed + unique restriction).

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

Methods

values

```
public static AttributeKind\[\] values()
```

valueOf

```
public static AttributeKind valueOf(String name)
```

com.sparsity.sparksee.gdb Class AttributeList

java.lang.Object

└─com.sparsity.sparksee.gdb.AttributeList

All Implemented Interfaces:

Iterable

public class **AttributeList**
extends Object
implements Iterable

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>

Constructor Summary

public	AttributeList (Collection col) Creates a new instance from an integer collection.
public	AttributeList () Constructor.
public	AttributeList (int[] list) Creates a new instance from an integer array.

Method Summary

void	add (int attr) Adds a Sparksee attribute identifier at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
AttributeListIterator	iterator () Gets a new AttributeListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Constructors

AttributeList

```
public AttributeList(Collection col)
```

Creates a new instance from an integer collection.

Parameters:

col - Collection to initialize the instance.

AttributeList

```
public AttributeList()
```

Constructor.

This creates an empty list.

AttributeList

```
public AttributeList(int[] list)
```

Creates a new instance from an integer array.

Parameters:

list - Integer array to initialize the instance.

Methods

clear

```
public void clear()
```

Clears the list.

iterator

```
public AttributeListIterator iterator()
```

Gets a new AttributeListIterator.

Returns:

AttributeListIterator instance.

count

```
public int count()
```

(continued from last page)

Number of elements in the list.

Returns:

Number of elements in the list.

add

```
public void add(int attr)
```

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

Parameters:

`attr` - [in] Sparksee attribute identifier.

com.sparsity.sparksee.gdb Class AttributeListIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.AttributeListIterator

All Implemented Interfaces:
Iterator

public class **AttributeListIterator**
extends Object
implements Iterator

AttributeList iterator class.

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

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext() Gets if there are more elements.
Integer	next() See nextAttribute().
int	nextAttribute() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean **hasNext()**

Gets if there are more elements.

(continued from last page)

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Integer next()
```

See nextAttribute().

nextAttribute

```
public int nextAttribute()
```

Gets the next element.

com.sparsity.sparksee.gdb Class AttributeStatistics

java.lang.Object

└─com.sparsity.sparksee.gdb.AttributeStatistics

public class **AttributeStatistics**
extends Object

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 `getAttributeStatistics` or check out the SPARKSEE User Manual for more details on this.

Author:

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

Method Summary

double	getAvgLengthString() Gets the average length.
long	getDistinct() Gets the number of distinct values (BASIC statistics).
Value	getMax() Gets the maximum existing value (BASIC statistics).
int	getMaxLengthString() Gets the maximum length.
double	getMean() Gets the mean or average.
double	getMedian() Gets the median.
Value	getMin() Gets the minimum existing value (BASIC statistics).
int	getMinLengthString() Gets the minimum length.
Value	getMode() Gets the mode.
long	getModeCount() Gets the number of objects with a Value equal to the mode.
long	getNull() Gets the number of objects NULL a Value (BASIC statistics).

long	getTotal() Gets the number of objects with a non-NULL Value (BASIC statistic).
double	getVariance() Gets the variance.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods

getMin

```
public Value getMin()
```

Gets the minimum existing value (BASIC statistics).

Returns:

The minimum existing value.

getMinLengthString

```
public int getMinLengthString()
```

Gets the minimum length.

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

Returns:

The minimum length.

getVariance

```
public double getVariance()
```

Gets the variance.

It is computed just for numerical attributes.

Returns:

The variance.

getMode

```
public Value getMode()
```

Gets the mode.

Mode: Most frequent Value.

Returns:

The mode.

getNull

```
public long getNull()
```

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

Returns:

The number of objects NULL a Value.

getDistinct

```
public long getDistinct()
```

Gets the number of distinct values (BASIC statistics).

Returns:

The number of distinct values.

getMean

```
public double getMean()
```

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.

getMax

```
public Value getMax()
```

Gets the maximum existing value (BASIC statistics).

Returns:

The maximum existing value.

getMedian

```
public double getMedian()
```

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:

(continued from last page)

The median.

getTotal

```
public long getTotal()
```

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

Returns:

The number of objects with a non-NULL Value.

getMaxLengthString

```
public int getMaxLengthString()
```

Gets the maximum length.

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

Returns:

The maximum length.

getAvgLengthString

```
public double getAvgLengthString()
```

Gets the average length.

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

Returns:

The average length.

getModeCount

```
public long getModeCount()
```

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

Returns:

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

com.sparsity.sparksee.gdb

Class BooleanList

java.lang.Object

└─com.sparsity.sparksee.gdb.BooleanList

All Implemented Interfaces:

Iterable

public class **BooleanList**
 extends Object
 implements Iterable

Boolean list.

It stores a Boolean list.

Use BooleanListIterator to access all elements into this collection.

Author:

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

Constructor Summary

public	BooleanList (Collection col) Creates a new instance from a boolean collection.
public	BooleanList (boolean[] list) Creates a new instance from a boolean array.
public	BooleanList () Constructor.

Method Summary

void	add (boolean value) Adds a Boolean at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
BooleanListIterator	iterator () Gets a new BooleanListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Constructors

BooleanList

```
public BooleanList(Collection col)
```

Creates a new instance from a boolean collection.

Parameters:

col - Collection to initialize the instance.

BooleanList

```
public BooleanList(boolean[] list)
```

Creates a new instance from a boolean array.

Parameters:

list - Boolean array to initialize the instance.

BooleanList

```
public BooleanList()
```

Constructor.

This creates an empty list.

Methods

add

```
public void add(boolean value)
```

Adds a Boolean at the end of the list.

Parameters:

value - [in] Boolean.

clear

```
public void clear()
```

Clears the list.

iterator

```
public BooleanListIterator iterator()
```

(continued from last page)

Gets a new BooleanListIterator.

Returns:

BooleanListIterator instance.

count

public int **count**()

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb Class BooleanListIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.BooleanListIterator

All Implemented Interfaces:
Iterator

public class **BooleanListIterator**
extends Object
implements Iterator

BooleanList iterator class.

Iterator to traverse all the strings into a BooleanList instance.

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext() Gets if there are more elements.
Boolean	next() See nextBoolean().
boolean	nextBoolean() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean **hasNext()**

Gets if there are more elements.

(continued from last page)

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Boolean next()
```

See nextBoolean().

nextBoolean

```
public boolean nextBoolean()
```

Gets the next element.

com.sparsity.sparksee.gdb Class Condition

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.Condition

```

All Implemented Interfaces:
Serializable, Comparable

public final class **Condition**
extends Enum

Condition operators enumeration.

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

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Between In range operator condition ([x,y]).
public static final	Equal Equal condition (==).
public static final	GreaterEqual Greater or equal condition (>=).
public static final	GreaterThan Greater than condition (>).
public static final	LessEqual Less or equal condition (<=).
public static final	LessThan Less than condition (<).
public static final	Like Substring condition.
public static final	LikeNoCase Substring (no case sensitive) condition.
public static final	NotEqual Not equal condition (!=).
public static final	RegExp Regular expression condition.

Method Summary

static Condition	valueOf (String name)
----------------------------------	---------------------------------------

static Condition[]	values()
------------------------------------	--------------------------

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

Equal

```
public static final com.sparsity.sparksee.gdb.Condition Equal
```

Equal condition (==).

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

GreaterEqual

```
public static final com.sparsity.sparksee.gdb.Condition GreaterEqual
```

Greater or equal condition (>=).

Null values cannot be used together with this condition.

GreaterThan

```
public static final com.sparsity.sparksee.gdb.Condition GreaterThan
```

Greater than condition (>).

Null values cannot be used together with this condition.

LessEqual

```
public static final com.sparsity.sparksee.gdb.Condition LessEqual
```

Less or equal condition (<=).

Null values cannot be used together with this condition.

LessThan

```
public static final com.sparsity.sparksee.gdb.Condition LessThan
```

Less than condition (<).

Null values cannot be used together with this condition.

NotEqual

```
public static final com.sparsity.sparksee.gdb.Condition NotEqual
```

Not equal condition (!=).

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

Like

```
public static final com.sparsity.sparksee.gdb.Condition Like
```

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

LikeNoCase

```
public static final com.sparsity.sparksee.gdb.Condition LikeNoCase
```

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

Between

```
public static final com.sparsity.sparksee.gdb.Condition Between
```

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

Null values cannot be used together with this condition.

RegExp

```
public static final com.sparsity.sparksee.gdb.Condition RegExp
```

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.

(continued from last page)

Methods

values

```
public static Condition\[\] values()
```

valueOf

```
public static Condition valueOf(String name)
```

com.sparsity.sparksee.gdb Class Database

java.lang.Object

└─com.sparsity.sparksee.gdb.Database

All Implemented Interfaces:

Closeable

```
public class Database
extends Object
implements Closeable
```

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 SPARKSEConfig.

Author:

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

Method Summary

void	<code>close()</code> Closes the Database instance.
void	<code>disableRollback()</code> Disables the rollback mechanism.
void	<code>enableRollback()</code> Enables the rollback mechanism.
void	<code>fixCurrentCacheMaxSize()</code> Sets the cache maximum size to the current cache size in use.
String	<code>getAlias()</code> Gets the alias of the Database.
int	<code>getCacheMaxSize()</code> Gets the cache maximum size (in MB).
String	<code>getPath()</code> Gets the path of the Database.
void	<code>getStatistics(DatabaseStatistics stats)</code> Gets Database statistics.
boolean	<code>isClosed()</code> Gets if Database instance has been closed or not.
<code>Session</code>	<code>newSession()</code> Creates a new Session.

void

[`setCacheMaxSize`](#)(int megaBytes)
Sets the cache maximum size (in MB).

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

Methods

disableRollback

```
public void disableRollback()
```

Disables the rollback mechanism.

fixCurrentCacheMaxSize

```
public void fixCurrentCacheMaxSize()
```

Sets the cache maximum size to the current cache size in use.

Returns:

Returns true if successful or false otherwise.

getAlias

```
public String getAlias()
```

Gets the alias of the Database.

Returns:

The alias of the Database.

getPath

```
public String getPath()
```

Gets the path of the Database.

Returns:

The path of the Database.

(continued from last page)

enableRollback

```
public void enableRollback()
```

Enables the rollback mechanism.

newSession

```
public Session newSession()
```

Creates a new Session.

getCacheMaxSize

```
public int getCacheMaxSize()
```

Gets the cache maximum size (in MB).

Returns:

Returns the current cache max size.

isClosed

```
public boolean isClosed()
```

Gets if Database instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

setCacheMaxSize

```
public void setCacheMaxSize(int megaBytes)
```

Sets the cache maximum size (in MB).

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

Parameters:

`megaBytes` - [in] The new cache max size.

close

```
public void close()
```

Closes the Database instance.

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

getStatistics

```
public void getStatistics(DatabaseStatistics stats)
```

Gets Database statistics.

Parameters:

stats - [out] The DatabaseStatistics instance.

com.sparsity.sparksee.gdb Class DatabaseStatistics

java.lang.Object

└─com.sparsity.sparksee.gdb.DatabaseStatistics

public class **DatabaseStatistics**
extends Object

Database statistics.

Author:

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

Method Summary

long	getCache() Gets cache size in KBytes.
long	getData() Gets database size in KBytes.
long	getRead() Gets total read data in KBytes.
long	getSessions() Gets the number of sessions.
long	getTemp() Gets temporary storage file size in KBytes.
long	getWrite() Gets total written data in KBytes.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getWrite

public long **getWrite()**

Gets total written data in KBytes.

Returns:

Total read written in KBytes.

(continued from last page)

getSessions

```
public long getSessions()
```

Gets the number of sessions.

Returns:

The number of sessions.

getData

```
public long getData()
```

Gets database size in KBytes.

Returns:

Database size in KBytes.

getTemp

```
public long getTemp()
```

Gets temporary storage file size in KBytes.

Returns:

Temporary storage file size in KBytes.

getRead

```
public long getRead()
```

Gets total read data in KBytes.

Returns:

Total read data in KBytes.

getCache

```
public long getCache()
```

Gets cache size in KBytes.

Returns:

Cache size in KBytes.

com.sparsity.sparksee.gdb

Class DataType

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.DataType

```

All Implemented Interfaces:
 Serializable, Comparable

public final class **DataType**
 extends Enum

Data type (domain) enumeration.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Boolean Boolean data type.
public static final	Double 64-bit signed double data type.
public static final	Integer 32-bit signed integer data type.
public static final	Long 64-bit signed integer data type.
public static final	OID Object identifier (OID) data type.
public static final	String Unicode string data type.
public static final	Text Large unicode character object (CLOB) data type.
public static final	Timestamp Distance from Epoch (UTC) time in milliseconds precision.

Method Summary

static DataType	valueOf (String name)
static DataType[]	values ()

Methods inherited from class `java.lang.Enum`

```
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal,
toString, valueOf
```

Methods inherited from class `java.lang.Object`

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait
```

Methods inherited from interface `java.lang.Comparable`

```
compareTo
```

Fields

Boolean

```
public static final com.sparsity.sparksee.gdb.DataType Boolean
```

Boolean data type.

Integer

```
public static final com.sparsity.sparksee.gdb.DataType Integer
```

32-bit signed integer data type.

Long

```
public static final com.sparsity.sparksee.gdb.DataType Long
```

64-bit signed integer data type.

Double

```
public static final com.sparsity.sparksee.gdb.DataType Double
```

64-bit signed double data type.

Timestamp

```
public static final com.sparsity.sparksee.gdb.DataType Timestamp
```

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].

(continued from last page)

String

```
public static final com.sparsity.sparksee.gdb.DataType String
```

Unicode string data type.

2048 characters maximum length.

Text

```
public static final com.sparsity.sparksee.gdb.DataType Text
```

Large unicode character object (CLOB) data type.

TextStream

OID

```
public static final com.sparsity.sparksee.gdb.DataType OID
```

Object identifier (OID) data type.

Methods

values

```
public static DataType\[\] values()
```

valueOf

```
public static DataType valueOf(String name)
```


com.sparsity.sparksee.gdb Class DefaultExport

```

java.lang.Object
|
+-com.sparsity.sparksee.gdb.ExportManager
   |
   +-com.sparsity.sparksee.gdb.DefaultExport

```

public class **DefaultExport**
extends [ExportManager](#)

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>

Constructor Summary

public	DefaultExport() Creates a new instance.
--------	--

Method Summary

boolean	enableType (int type) Default implementation of the ExportManager class method EnableType.
boolean	getEdge (long edge, EdgeExport edgeExport) Default implementation of the ExportManager class method GetEdge.
boolean	getEdgeType (int type, EdgeExport edgeExport) Default implementation of the ExportManager class method GetEdgeType.
boolean	getGraph (GraphExport graphExport) Default implementation of the ExportManager class method GetGraph.
boolean	getNode (long node, NodeExport nodeExport) Default implementation of the ExportManager class method GetNode.
boolean	getNodeType (int type, NodeExport nodeExport) Default implementation of the ExportManager class method GetNodeType.
void	prepare (Graph graph) Default implementation of the ExportManager class method Prepare.
void	release () Default implementation of the ExportManager class method Release.

Methods inherited from class [com.sparsity.sparksee.gdb.ExportManager](#)

[enableType](#), [getEdge](#), [getEdgeType](#), [getGraph](#), [getNode](#), [getNodeType](#), [prepare](#), [release](#)

Methods inherited from class [java.lang.Object](#)

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Constructors

DefaultExport

```
public DefaultExport()
```

Creates a new instance.

Methods

enableType

```
public boolean enableType(int 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.

getEdge

```
public boolean getEdge(long edge,  
    EdgeExport edgeExport)
```

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.

edgeExport - [out] The EdgeExport that will store the information.

Returns:

TRUE.

getGraph

```
public boolean getGraph(GraphExport graphExport)
```

Default implementation of the ExportManager class method GetGraph.

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

Parameters:

graphExport - [out] The GraphExport that will store the information.

(continued from last page)

Returns:
TRUE.

getEdgeType

```
public boolean getEdgeType(int type,  
    EdgeExport edgeExport)
```

Default implementation of the ExportManager class method GetEdgeType.

This sets de default EdgeExport values.

Parameters:
type - [in] An edge type.
edgeExport - [out] The EdgeExport that will store the information.

Returns:
TRUE.

getNodeType

```
public boolean getNodeType(int type,  
    NodeExport nodeExport)
```

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

Parameters:
type - [in] A node type.
nodeExport - [out] The NodeExport that will store the information.

Returns:
TRUE.

release

```
public void release()
```

Default implementation of the ExportManager class method Release.

getNode

```
public boolean getNode(long node,  
    NodeExport nodeExport)
```

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.
nodeExport - [out] The NodeExport that will store the information.

Returns:
TRUE.

(continued from last page)

prepare

```
public void prepare(Graph graph)
```

Default implementation of the ExportManager class method Prepare.

Parameters:

graph - null

com.sparsity.sparksee.gdb Class EdgeData

```
java.lang.Object
└--com.sparsity.sparksee.gdb.EdgeData
```

```
public class EdgeData
    extends Object
```

Edge data class.

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

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

Author:

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

Method Summary

long	getEdge() Gets the edge identifier.
long	getHead() Gets the head of the edge.
long	getTail() Gets the tail of the edge.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getHead

```
public long getHead()
```

Gets the head of the edge.

Returns:

The Sparksee edge identifier of the head of the edge.

getTail

```
public long getTail()
```

Gets the tail of the edge.

(continued from last page)

Returns:

The Sparksee edge identifier of the tail of the edge.

getEdge

```
public long getEdge()
```

Gets the edge identifier.

Returns:

The Sparksee edge identifier.

com.sparsity.sparksee.gdb Class EdgeExport

java.lang.Object

└─com.sparsity.sparksee.gdb.EdgeExport

public class **EdgeExport**
extends Object

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>

Constructor Summary

public	EdgeExport() Creates a new instance.
--------	---

Method Summary

boolean	asDirected() Gets if the edge should be managed as directed.
java.awt.Color	getColor() Gets the color of the edge.
int	getColorRGB() Gets the edge color.
int	getFontSize() Gets the edge label font size.
String	getLabel() Gets the edge label.
java.awt.Color	getLabelColor() Gets the color of the label.

int	<code>getLabelColorRGB()</code> Gets the edge label color.
int	<code>getWidth()</code> Gets the edge width.
void	<code>setAsDirected(boolean directed)</code> Sets if the edge should be managed as directed.
void	<code>setColor(java.awt.Color c)</code> Sets the color of the edge.
void	<code>setColorRGB(int color)</code> Sets the edge color.
void	<code>setDefaults()</code> Sets to default values.
void	<code>setFontSize(int size)</code> Sets the edge label font size.
void	<code>setLabel(String label)</code> Sets the edge label.
void	<code>setLabelColor(java.awt.Color c)</code> Sets the color of the label.
void	<code>setLabelColorRGB(int color)</code> Sets the edge label color.
void	<code>setWidth(int width)</code> Sets the edge width.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

EdgeExport

```
public EdgeExport()
```

Creates a new instance.

Methods

getColor

```
public java.awt.Color getColor()
```

Gets the color of the edge.

setColorRGB

```
public void setColorRGB(int color)
```

Sets the edge color.

Parameters:

color - [in] The edge color.

setAsDirected

```
public void setAsDirected(boolean directed)
```

Sets if the edge should be managed as directed.

Parameters:

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

getFontSize

```
public int getFontSize()
```

Gets the edge label font size.

Returns:

The edge label font size.

setDefaults

```
public void setDefaults()
```

Sets to default values.

getColorRGB

```
public int getColorRGB()
```

Gets the edge color.

Returns:

The edge color.

getLabelColorRGB

```
public int getLabelColorRGB()
```

(continued from last page)

Gets the edge label color.

Returns:

The edge label color.

getWidth

```
public int getWidth()
```

Gets the edge width.

Returns:

The edge width.

setLabel

```
public void setLabel(String label)
```

Sets the edge label.

Parameters:

label - [in] The edge label.

getLabelColor

```
public java.awt.Color getLabelColor()
```

Gets the color of the label.

setColor

```
public void setColor(java.awt.Color c)
```

Sets the color of the edge.

Parameters:

c - New value.

asDirected

```
public boolean asDirected()
```

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.

(continued from last page)

getLabel

```
public String getLabel()
```

Gets the edge label.

Returns:

The edge label.

setLabelColorRGB

```
public void setLabelColorRGB(int color)
```

Sets the edge label color.

Parameters:

color - [in] The edge label color.

setWidth

```
public void setWidth(int width)
```

Sets the edge width.

Parameters:

width - [in] The edge width.

setFontSize

```
public void setFontSize(int size)
```

Sets the edge label font size.

Parameters:

size - [in] The edge label font size.

setLabelColor

```
public void setLabelColor(java.awt.Color c)
```

Sets the color of the label.

Parameters:

c - New value.

com.sparsity.sparksee.gdb

Class EdgesDirection

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.EdgesDirection
  
```

All Implemented Interfaces:
 Serializable, Comparable

public final class **EdgesDirection**
 extends Enum

Edges direction enumeration.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Any In-going or out-going edges.
public static final	Ingoing In-going edges.
public static final	Outgoing Out-going edges.

Method Summary

static EdgesDirection	valueOf (String name)
static EdgesDirection[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

Ingoing

```
public static final com.sparsity.sparksee.gdb.EdgesDirection Ingoing
```

In-going edges.

Outgoing

```
public static final com.sparsity.sparksee.gdb.EdgesDirection Outgoing
```

Out-going edges.

Any

```
public static final com.sparsity.sparksee.gdb.EdgesDirection Any
```

In-going or out-going edges.

Methods

values

```
public static EdgesDirection\[\] values()
```

valueOf

```
public static EdgesDirection valueOf(String name)
```

com.sparsity.sparksee.gdb Class ExportManager

java.lang.Object

└─com.sparsity.sparksee.gdb.ExportManager

Direct Known Subclasses:

[DefaultExport](#)

```
public class ExportManager
extends Object
```

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 formats. Nowadays, available formats are defined in the ExportType enum.

Author:

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

Method Summary

boolean	enableType (int type) Gets whether a node or edge type must be exported or not.
boolean	getEdge (long edge, EdgeExport edgeExport) Gets the edge export definition for the given edge.
boolean	getEdgeType (int type, EdgeExport edgeExport) Gets the default node export definition for the given edge type.
boolean	getGraph (GraphExport graphExport) Gets the graph export definition.
boolean	getNode (long node, NodeExport nodeExport) Gets the node export definition for the given node.
boolean	getNodeType (int type, NodeExport nodeExport) Gets the default node export definition for the given node type.
void	prepare (Graph graph) Prepares the graph for the export process.
void	release () Ends the export process.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

(continued from last page)

getNodeType

```
public boolean getNodeType(int type,  
    NodeExport nodeExport)
```

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.

nodeExport - [out] The NodeExport which defines how to export the nodes of the given type.

Returns:

TRUE.

getEdge

```
public boolean getEdge(long edge,  
    EdgeExport edgeExport)
```

Gets the edge export definition for the given edge.

Parameters:

edge - Edge identifier.

edgeExport - [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.

getGraph

```
public boolean getGraph(GraphExport graphExport)
```

Gets the graph export definition.

Parameters:

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

Returns:

TRUE.

getEdgeType

```
public boolean getEdgeType(int type,  
    EdgeExport edgeExport)
```

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.

(continued from last page)

edgeExport - [out] The EdgeExport which defines how to export the edges of the given type.

Returns:

TRUE.

prepare

```
public void prepare(Graph graph)
```

Prepares the graph for the export process.

It is called once before the export process.

Parameters:

graph - Graph to be exported.

getNode

```
public boolean getNode(long node,  
    NodeExport nodeExport)
```

Gets the node export definition for the given node.

Parameters:

node - Node identifier.

nodeExport - [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.

release

```
public void release()
```

Ends the export process.

It is called once after the export process.

enableType

```
public boolean enableType(int 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.

com.sparsity.sparksee.gdb

Class ExportType

```

java.lang.Object
  |
  +- java.lang.Enum
        |
        +- com.sparsity.sparksee.gdb.ExportType
  
```

All Implemented Interfaces:
 Serializable, Comparable

public final class **ExportType**
 extends Enum

Export type.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	GraphML Export to GraphML format.
public static final	Graphviz Export to Graphviz format.
public static final	YGraphML Export to YGRAPHML format.

Method Summary

static ExportType	valueOf (String name)
static ExportType[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

Graphviz

```
public static final com.sparsity.sparksee.gdb.ExportType Graphviz
```

Export to Graphviz format.

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

GraphML

```
public static final com.sparsity.sparksee.gdb.ExportType GraphML
```

Export to GraphML format.

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

YGraphML

```
public static final com.sparsity.sparksee.gdb.ExportType YGraphML
```

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>").

Methods

values

```
public static ExportType\[\] values()
```

valueOf

```
public static ExportType valueOf(String name)
```

com.sparsity.sparksee.gdb Class Graph

java.lang.Object

└--com.sparsity.sparksee.gdb.Graph

public class **Graph**
extends Object

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>

Method Summary

void	backup (String file) Dumps all the data to a backup file.
long	countEdges () Gets the number of edges.
long	countNodes () Gets the number of nodes.
long	degree (long oid, int etype, EdgesDirection dir) Gets the number of edges from or to the given node OID and for the given edge type.
void	drop (long oid) Drops the given OID.
void	drop (Objects objs) Drops all the OIDs from the given collection.
void	dumpData (String file) Dumps logical data to a file.
void	dumpStorage (String file) Dumps internal storage data to a file.
Objects	edges (int etype, long tail, long head) Gets all the edges of the given type between two given nodes (tail and head).
Objects	explode (long oid, int etype, EdgesDirection dir) Selects all edges from or to the given node OID and for the given edge type.
Objects	explode (Objects objs, int etype, EdgesDirection dir) Selects all edges from or to each of the node OID in the given collection and for the given edge type.

void	export (String file, ExportType type, ExportManager em) Exports the Graph.
int	findAttribute (int type, String name) Gets the Sparksee attribute identifier for the given type identifier and attribute name.
AttributeList	findAttributes (int type) Gets all existing Sparksee attribute identifiers for the given type identifier.
long	findEdge (int etype, long tail, long head) Gets any of the edges of the given type between two given nodes (tail and head).
TypeList	findEdgeTypes () Gets all existing Sparksee edge type identifiers.
TypeList	findNodeTypes () Gets all existing Sparksee node type identifiers.
long	findObject (int attr, Value value) Finds one object having the given Value for the given attribute.
long	findOrCreateEdge (int etype, long tail, long head) Gets any of the edges of the specified type between two given nodes (tail and head).
long	findOrCreateObject (int attr, Value value) Finds one object having the given Value for the attribute or it creates one does not exist any.
int	findType (String name) Gets the Sparksee type identifier for the given type name.
TypeList	findTypes () Gets all existing Sparksee node and edge type identifiers.
Attribute	getAttribute (int attr) Gets information about the given attribute.
Value	getAttribute (long oid, int attr) Gets the Value for the given attribute and OID.
void	getAttribute (long oid, int attr, Value value) Gets the Value for the given attribute and OID.
long	getAttributeIntervalCount (int attr, Value lower, boolean includeLower, Value higher, boolean includeHigher) Gets how many objects have a value into the given range for the given attribute.
AttributeList	getAttributes (long oid) Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.
AttributeStatistics	getAttributeStatistics (int attr, boolean basic) Gets statistics from the given attribute.
TextStream	getAttributeText (long oid, int attr) Gets the read-only TextStream for the given text attribute and OID.
EdgeData	getEdgeData (long edge) Gets information about an edge.
long	getEdgePeer (long edge, long node) Gets the other end for the given edge.

int	<u>getObjectType</u> (long oid) Gets the Sparksee node or edge type identifier for the given OID.
<u>Type</u>	<u>getType</u> (int type) Gets information about the given type.
<u>Values</u>	<u>getValues</u> (int attr) Gets the Value collection for the given attribute.
<u>Objects</u>	<u>heads</u> (<u>Objects</u> edges) Gets all the heads from the given edges collection.
void	<u>indexAttribute</u> (int attr, <u>AttributeKind</u> kind) Updates the index of the given attribute.
<u>Objects</u>	<u>neighbors</u> (long oid, int etype, <u>EdgesDirection</u> dir) Selects all neighbor nodes from or to the given node OID and for the given edge type.
<u>Objects</u>	<u>neighbors</u> (<u>Objects</u> objs, int etype, <u>EdgesDirection</u> dir) Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.
int	<u>newAttribute</u> (int type, String name, <u>DataType</u> dt, <u>AttributeKind</u> kind) Creates a new attribute.
int	<u>newAttribute</u> (int type, String name, <u>DataType</u> dt, <u>AttributeKind</u> kind, <u>Value</u> defaultValue) Creates a new attribute with a default value.
long	<u>newEdge</u> (int type, int tailAttr, <u>Value</u> tailV, int headAttr, <u>Value</u> headV) Creates a new edge instance.
long	<u>newEdge</u> (int type, long tail, long head) Creates a new edge instance.
int	<u>newEdgeType</u> (String name, boolean directed, boolean neighbors) Creates a new edge type.
long	<u>newNode</u> (int type) Creates a new node instance.
int	<u>newNodeType</u> (String name) Creates a new node type.
int	<u>newRestrictedEdgeType</u> (String name, int tail, int head, boolean neighbors) Creates a new restricted edge type.
int	<u>newSessionAttribute</u> (int type, <u>DataType</u> dt, <u>AttributeKind</u> kind) Creates a new Session attribute.
int	<u>newSessionAttribute</u> (int type, <u>DataType</u> dt, <u>AttributeKind</u> kind, <u>Value</u> defaultValue) Creates a new Session attribute with a default value.
void	<u>removeAttribute</u> (int attr) Removes the given attribute.
void	<u>removeType</u> (int type) Removes the given type.

void	<code>renameAttribute</code> (int attr, String newName) Renames an attribute.
void	<code>renameType</code> (int type, String newName) Renames a type.
void	<code>renameType</code> (String oldName, String newName) Renames a type.
Objects	<code>select</code> (int type) Selects all OIDs belonging to the given type.
Objects	<code>select</code> (int attr, Condition cond, Value value) Selects all OIDs satisfying the given condition for the given attribute.
Objects	<code>select</code> (int attr, Condition cond, Value value, Objects restriction) Selects all OIDs satisfying the given condition for the given attribute.
Objects	<code>select</code> (int attr, Condition cond, Value lower, Value higher) Selects all OIDs satisfying the given condition for the given attribute.
Objects	<code>select</code> (int attr, Condition cond, Value lower, Value higher, Objects restriction) Selects all OIDs satisfying the given condition for the given attribute.
void	<code>setAttribute</code> (long oid, int attr, Value value) Sets the Value for the given attribute and OID.
void	<code>setAttributeDefaultValue</code> (int attr, Value value) Sets a default value for an attribute.
void	<code>setAttributeText</code> (long oid, int attr, TextStream tstream) Sets the writable TextStream for the given text attribute and OID.
Objects	<code>tails</code> (Objects edges) Gets all the tails from the given edges collection.
void	<code>tailsAndHeads</code> (Objects edges, Objects tails, Objects heads) Gets all the tails and heads from the given edges collection.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods

`setAttributeText`

```
public void setAttributeText(long oid,
    int attr,
    TextStream tstream)
```

Sets the writable TextStream for the given text attribute and OID.

Parameters:

(continued from last page)

oid - [in] Sparksee OID.

attr - [in] Sparksee attribute identifier.

tstream - [in] New Text value. This corresponds to a TextStream to write.

dumpData

```
public void dumpData(String file)
    throws FileNotFoundException,
        RuntimeException
```

Dumps logical data to a file.

Parameters:

file - [in] Output file path.

Throws:

java.io.FileNotFoundException - If the given file cannot be created.

java.lang.RuntimeException - null

findTypes

```
public TypeList findTypes()
```

Gets all existing Sparksee node and edge type identifiers.

Returns:

Sparksee node and edge type identifier list.

renameAttribute

```
public void renameAttribute(int attr,
    String newName)
```

Renames an attribute.

The new name must be available.

Parameters:

attr - [in] Sparksee attribute identifier.

newName - [in] The new name for the attribute.

select

```
public Objects select(int attr,
    Condition cond,
    Value value,
    Objects 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.

(continued from last page)

restriction - [in] Objects to limit the select in this set of objects.

Returns:

Objects instance.

getAttributeIntervalCount

```
public long getAttributeIntervalCount(int attr,  
    Value lower,  
    boolean includeLower,  
    Value higher,  
    boolean includeHigher)
```

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.

includeLower - [in] If TRUE, include lower bound Value of the range.

higher - [in] Higher bound Value of the range.

includeHigher - [in] If TRUE, include higher bound Value of the range.

Returns:

Number of objects having a value into the given range.

tailsAndHeads

```
public void tailsAndHeads(Objects edges,  
    Objects tails,  
    Objects 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.

degree

```
public long degree(long oid,  
    int etype,  
    EdgesDirection 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.

renameType

```
public void renameType(String oldName,  
                        String newName)
```

Renames a type.

The new name must be available.

Parameters:

oldName - [in] The current name of the type to be renamed.

newName - [in] The new name for the type.

dumpStorage

```
public void dumpStorage(String file)  
    throws FileNotFoundException,  
           RuntimeException
```

Dumps internal storage data to a file.

Parameters:

file - [in] Output file path.

Throws:

java.io.FileNotFoundException - If the given file cannot be created.

java.lang.RuntimeException - null

neighbors

```
public Objects neighbors(Objects objs,  
                        int etype,  
                        EdgesDirection 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.

etype - [in] Sparksee edge type identifier.

dir - [in] Direction.

Returns:

Objects instance.

getAttributes

```
public AttributeList getAttributes(long oid)
```

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

Parameters:

oid - [in] Sparksee OID.

(continued from last page)

Returns:

Sparksee attribute identifier list.

getAttributeStatistics

```
public AttributeStatistics getAttributeStatistics(int attr,  
                                                boolean basic)
```

Gets statistics from the given attribute.

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.

newNode

```
public long newNode(int type)
```

Creates a new node instance.

Parameters:`type` - [in] Sparksee type identifier.**Returns:**

Unique OID of the new node instance.

getAttributeText

```
public TextStream getAttributeText(long oid,  
                                    int 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.

countEdges

```
public long countEdges()
```

Gets the number of edges.

Returns:

(continued from last page)

The number of edges.

findEdgeTypes

```
public TypeList findEdgeTypes()
```

Gets all existing Sparksee edge type identifiers.

Returns:

Sparksee edge type identifier list.

select

```
public Objects select(int attr,  
    Condition cond,  
    Value lower,  
    Value 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.

indexAttribute

```
public void indexAttribute(int attr,  
    AttributeKind 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.

getType

```
public Type getType(int type)
```

Gets information about the given type.

Parameters:

type - [in] Sparksee type identifier.

Returns:

The Type for the given Sparksee type identifier.

findAttribute

```
public int findAttribute(int type,  
    String 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.

newAttribute

```
public int newAttribute(int type,  
    String name,  
    DataType dt,  
    AttributeKind 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.

edges

```
public Objects edges(int etype,  
    long tail,  
    long 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.

select

```
public Objects select(int type)
```

(continued from last page)

Selects all OIDs belonging to the given type.

Parameters:

type - [in] Sparksee type identifier.

Returns:

Objects instance.

select

```
public Objects select(int attr,  
    Condition cond,  
    Value 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.

findOrCreateObject

```
public long findOrCreateObject(int attr,  
    Value 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.

findNodeTypes

```
public TypeList findNodeTypes( )
```

Gets all existing Sparksee node type identifiers.

Returns:

Sparksee node type identifier list.

(continued from last page)

getAttribute

```
public Value getAttribute(long oid,  
                           int 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.

removeAttribute

```
public void removeAttribute(int attr)
```

Removes the given attribute.

Parameters:

attr - [in] Sparksee attribute identifier.

setAttributeDefaultValue

```
public void setAttributeDefaultValue(int attr,  
                                       Value 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.

backup

```
public void backup(String file)  
    throws FileNotFoundException,  
           RuntimeException
```

Dumps all the data to a backup file.

See the Sparksee class Restore method.

Parameters:

file - [in] Output backup file path.

Throws:

java.io.FileNotFoundException - If the given file cannot be created.

java.lang.RuntimeException - null

newSessionAttribute

```
public int newSessionAttribute(int type,
    DataType dt,
    AttributeKind kind,
    Value defaultValue)
```

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.
defaultValue - [in] The default value to use in each new node/edge.

Returns:

Unique Sparksee attribute identifier.

findAttributes

```
public AttributeList findAttributes(int type)
```

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

Parameters:

type - [in] Sparksee type identifier.

Returns:

Sparksee attribute identifier list.

getAttribute

```
public void getAttribute(long oid,
    int attr,
    Value 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.

countNodes

```
public long countNodes()
```

Gets the number of nodes.

(continued from last page)

Returns:

The number of nodes.

setAttribute

```
public void setAttribute(long oid,  
    int attr,  
    Value 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.

getEdgeData

```
public EdgeData getEdgeData(long edge)
```

Gets information about an edge.

Parameters:

edge - [in] Sparksee edge identifier.

Returns:

An EdgeData instance.

neighbors

```
public Objects neighbors(long oid,  
    int etype,  
    EdgesDirection 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.

renameType

```
public void renameType(int type,  
    String newName)
```

Renames a type.

The new name must be available.

Parameters:

(continued from last page)

type - [in] The type to be renamed.
newName - [in] The new name for the type.

explode

```
public Objects explode(Objects objs,  
                       int etype,  
                       EdgesDirection 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.

newNodeType

```
public int newNodeType(String name)
```

Creates a new node type.

Parameters:

name - [in] Unique name for the new node type.

Returns:

Unique Sparksee type identifier.

newSessionAttribute

```
public int newSessionAttribute(int type,  
                                DataType dt,  
                                AttributeKind 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.

tails

```
public Objects tails(Objects edges)
```

(continued from last page)

Gets all the tails from the given edges collection.

Parameters:

edges - [in] Sparksee edge identifier collection.

Returns:

The tails collection.

findOrCreateEdge

```
public long findOrCreateEdge(int etype,  
    long tail,  
    long 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.

drop

```
public void drop(long oid)
```

Drops the given OID.

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

Parameters:

oid - [in] Sparksee OID to be removed.

newEdgeType

```
public int newEdgeType(String name,  
    boolean directed,  
    boolean 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.

heads

```
public Objects heads(Objects edges)
```

(continued from last page)

Gets all the heads from the given edges collection.

Parameters:

edges - [in] Sparksee edge identifier collection.

Returns:

The heads collection.

findEdge

```
public long findEdge(int etype,  
    long tail,  
    long 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.

explode

```
public Objects explode(long oid,  
    int etype,  
    EdgesDirection 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.

findObject

```
public long findObject(int attr,  
    Value 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.

(continued from last page)

Returns:

Sparksee OID or the Objects InvalidOID.

drop

```
public void drop(Objects objs)
```

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs that call for all the elements into the collection.

Parameters:

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

newAttribute

```
public int newAttribute(int type,  
    String name,  
    DataType dt,  
    AttributeKind kind,  
    Value defaultValue)
```

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.

defaultValue - [in] The default value to use in each new node/edge.

Returns:

Unique Sparksee attribute identifier.

getObjectType

```
public int getObjectType(long 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.

getAttribute

```
public Attribute getAttribute(int attr)
```

Gets information about the given attribute.

Parameters:

(continued from last page)

`attr` - [in] Sparksee attribute identifier.

Returns:

The Attribute for the given Sparksee attribute identifier.

export

```
public void export(String file,  
    ExportType type,  
    ExportManager em)  
throws IOException
```

Exports the Graph.

Parameters:

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

Throws:

`java.io.IOException` - null

newEdge

```
public long newEdge(int type,  
    int tailAttr,  
    Value tailV,  
    int headAttr,  
    Value headV)
```

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.
`tailAttr` - [in] Sparksee attribute identifier.
`tailV` - [in] Value.
`headAttr` - [in] Sparksee attribute identifier.
`headV` - [in] Value.

Returns:

Unique OID of the new edge instance.

getValues

```
public Values getValues(int attr)
```

Gets the Value collection for the given attribute.

Parameters:

`attr` - [in] Sparksee attribute identifier.

Returns:

Returns a Values object.

getEdgePeer

```
public long getEdgePeer(long edge,  
                        long 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.

newEdge

```
public long newEdge(int type,  
                   long tail,  
                   long 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.

select

```
public Objects select(int attr,  
                    Condition cond,  
                    Value lower,  
                    Value higher,  
                    Objects 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.

findType

```
public int findType(String name)
```

(continued from last page)

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.

removeType

```
public void removeType(int 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.

newRestrictedEdgeType

```
public int newRestrictedEdgeType(String name,  
    int tail,  
    int head,  
    boolean 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.

com.sparsity.sparksee.gdb

Class GraphExport

java.lang.Object

└─com.sparsity.sparksee.gdb.GraphExport

public class **GraphExport**
extends Object

Stores the graph exporting values.

Author:

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

Constructor Summary

public	GraphExport() Creates a new GraphExport instance.
--------	--

Method Summary

String	getLabel() Gets the graph label.
void	setDefaults() Sets to default values.
void	setLabel(String label) Sets the graph label.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

GraphExport

public **GraphExport()**

Creates a new GraphExport instance.

Methods

getLabel

public String **getLabel()**

(continued from last page)

Gets the graph label.

Returns:

The graph label.

setDefault

```
public void setDefault()
```

Sets to default values.

setLabel

```
public void setLabel(String label)
```

Sets the graph label.

Parameters:

label - [in] The graph label.

com.sparsity.sparksee.gdb

Class Int32List

java.lang.Object

└─com.sparsity.sparksee.gdb.Int32List

All Implemented Interfaces:

Iterable

public class **Int32List**
 extends Object
 implements Iterable

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>

Constructor Summary

public	Int32List (Collection col) Creates a new instance from an integer collection.
public	Int32List () Constructor.
public	Int32List (int[] list) Creates a new instance from an integer array.

Method Summary

void	add (int value) Adds an 32-bit signed integer at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
Int32ListIterator	iterator () Gets a new Int32ListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Constructors

Int32List

```
public Int32List(Collection col)
```

Creates a new instance from an integer collection.

Parameters:

col - Collection to initialize the instance.

Int32List

```
public Int32List()
```

Constructor.

This creates an empty list.

Int32List

```
public Int32List(int[] list)
```

Creates a new instance from an integer array.

Parameters:

list - Integer array to initialize the instance.

Methods

add

```
public void add(int value)
```

Adds an 32-bit signed integer at the end of the list.

Parameters:

value - [in] The integer.

clear

```
public void clear()
```

Clears the list.

iterator

```
public Int32ListIterator iterator()
```

(continued from last page)

Gets a new Int32ListIterator.

Returns:

Int32ListIterator instance.

count

public int **count**()

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb Class Int32ListIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.Int32ListIterator

All Implemented Interfaces:
Iterator

public class **Int32ListIterator**
extends Object
implements Iterator

Int32List iterator class.

Iterator to traverse all the integer into a Int32List instance.

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext() Gets if there are more elements.
Integer	next() See nextInt32().
Integer	nextInt32() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

nextInt32

public Integer **nextInt32()**

Gets the next element.

hasNext

```
public boolean hasNext()
```

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Integer next()
```

See nextInt32().

com.sparsity.sparksee.gdb Class LogLevel

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.LogLevel

```

All Implemented Interfaces:
Serializable, Comparable

public final class **LogLevel**
extends Enum

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>

Field Summary

public static final	Config Config log level.
public static final	Debug Debug log level.
public static final	Fine Fine log level.
public static final	Info Info log level.
public static final	Off Disable log.
public static final	Severe Severe log level.
public static final	Warning Warning log level.

Method Summary

static LogLevel	valueOf (String name)
static LogLevel[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Fields

Off

```
public static final com.sparsity.sparksee.gdb.LogLevel Off
```

Disable log.

Severe

```
public static final com.sparsity.sparksee.gdb.LogLevel Severe
```

Severe log level.

This is the lower log level, just errors are shown.

Warning

```
public static final com.sparsity.sparksee.gdb.LogLevel Warning
```

Warning log level.

Errors and warnings are shown.

Info

```
public static final com.sparsity.sparksee.gdb.LogLevel Info
```

Info log level.

Errors, warnings and information messages are shown.

Config

```
public static final com.sparsity.sparksee.gdb.LogLevel Config
```

Config log level.

Errors, warnings, information messages and configuration details of the different components are shown.

Fine

```
public static final com.sparsity.sparksee.gdb.LogLevel Fine
```


(continued from last page)

Fine log level.

This is the higher and finest public log level, everything is dumped to the log.

Debug

```
public static final com.sparsity.sparksee.gdb.LogLevel Debug
```

Debug log level.

This is for Sparksee development purposes and just works with debug versions of the library.

Methods

values

```
public static LogLevel\[\] values()
```

valueOf

```
public static LogLevel valueOf(String name)
```

com.sparsity.sparksee.gdb Class NodeExport

java.lang.Object

└─com.sparsity.sparksee.gdb.NodeExport

public class **NodeExport**
extends Object

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 (0x000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

Author:

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

Constructor Summary

public	NodeExport() Creates a new instance.
--------	---

Method Summary

java.awt.Color	getColor() Gets the color of the node.
int	getColorRGB() Gets the node color.
int	getFontSize() Gets the node label font size.
int	getHeight() Gets the node height.
String	getLabel() Gets the node label.

java.awt.Color	getLabelColor() Gets the color of the label.
int	getLabelColorRGB() Gets the node label color.
NodeShape	getShape() Gets the node shape.
int	getWidth() Gets the node width.
boolean	isFit() Gets whether the node size is fitted to the label or not.
void	setColor(java.awt.Color c) Sets the color of the node.
void	setColorRGB(int color) Sets the node color.
void	setDefaults() Sets to default values.
void	setFit(boolean fit) Sets the node fit property.
void	setFontSize(int size) Sets the node label font size.
void	setHeight(int height) Sets the node height.
void	setLabel(String label) Sets the node label.
void	setLabelColor(java.awt.Color c) Sets the color of the label.
void	setLabelColorRGB(int color) Sets the node label color.
void	setShape(NodeShape shape) Sets the node shape.
void	setWidth(int width) Gets the node width.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

NodeExport

```
public NodeExport()
```

Creates a new instance.

Methods

getShape

```
public NodeShape getShape()
```

Gets the node shape.

Returns:

The node shape.

getColor

```
public java.awt.Color getColor()
```

Gets the color of the node.

setColorRGB

```
public void setColorRGB(int color)
```

Sets the node color.

Parameters:

color - The node color.

setHeight

```
public void setHeight(int height)
```

Sets the node height.

Parameters:

height - [in] The node height in pixels.

getFontSize

```
public int getFontSize()
```

Gets the node label font size.

(continued from last page)

Returns:

The node label font size.

setDefault

```
public void setDefault()
```

Sets to default values.

getColorRGB

```
public int getColorRGB()
```

Gets the node color.

Returns:

The node color.

isFit

```
public boolean isFit()
```

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'.

getLabelColorRGB

```
public int getLabelColorRGB()
```

Gets the node label color.

Returns:

The node label color.

getWidth

```
public int getWidth()
```

Gets the node width.

Returns:

The node width in pixels.

setLabel

```
public void setLabel(String label)
```

(continued from last page)

Sets the node label.

Parameters:

label - [in] The node label.

getLabelColor

```
public java.awt.Color getLabelColor()
```

Gets the color of the label.

setColor

```
public void setColor(java.awt.Color c)
```

Sets the color of the node.

Parameters:

c - New value.

getLabel

```
public String getLabel()
```

Gets the node label.

Returns:

The node label.

getHeight

```
public int getHeight()
```

Gets the node height.

Returns:

The node height in pixels.

setLabelColorRGB

```
public void setLabelColorRGB(int color)
```

Sets the node label color.

Parameters:

color - [in] The node label color.

setWidth

```
public void setWidth(int width)
```

Gets the node width.

Parameters:

width - The node width in pixels.

setShape

```
public void setShape(NodeShape shape)
```

Sets the node shape.

Parameters:

shape - [in] The node shape.

setFit

```
public void setFit(boolean 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'.

setFontSize

```
public void setFontSize(int size)
```

Sets the node label font size.

Parameters:

size - [in] The node label font size.

setLabelColor

```
public void setLabelColor(java.awt.Color c)
```

Sets the color of the label.

Parameters:

c - New value.

com.sparsity.sparksee.gdb

Class NodeShape

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.NodeShape
  
```

All Implemented Interfaces:
 Serializable, Comparable

public final class **NodeShape**
 extends Enum

Node shape.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Box Box shape.
public static final	Round Round shape.

Method Summary

static NodeShape	valueOf (String name)
static NodeShape[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

Box

```
public static final com.sparsity.sparksee.gdb.NodeShape Box
```

Box shape.

Round

```
public static final com.sparsity.sparksee.gdb.NodeShape Round
```

Round shape.

Methods

values

```
public static NodeShape\[\] values()
```

valueOf

```
public static NodeShape valueOf(String name)
```

com.sparsity.sparksee.gdb Class Objects

java.lang.Object

└─com.sparsity.sparksee.gdb.Objects

All Implemented Interfaces:

Iterable, Closeable, Set

```
public class Objects
extends Object
implements Set, Closeable, Iterable
```

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>

Field Summary

public static	InvalidOID Invalid object identifier constant.
---------------	---

Method Summary

boolean	add (long e) Adds an element into the collection.
boolean	add (Long e) Adds the specified element to this set if it is not already present (optional operation).
boolean	addAll (Collection clctn) Adds all of the elements in the specified collection to this set if they're not already present (optional operation).
long	any () Gets an element from the collection.
void	clear () Clears the collection removing all its elements.
void	close () Closes the Objects instance.

static Objects	combineDifference (Objects objs1, Objects objs2) Creates a new Objects instance which is the difference of the two given.
static Objects	combineIntersection (Objects objs1, Objects objs2) Creates a new Objects instance which is the intersection of the two given.
static Objects	combineUnion (Objects objs1, Objects objs2) Creates a new Objects instance which is the union of the two given.
boolean	contains (Object o) Returns true if this collections contains the specified element or Objects.
boolean	contains (Objects objs) Check if this objects contains the other one.
boolean	containsAll (Collection clctn) Returns true if this set contains all of the elements of the specified collection.
Objects	copy () Creates a new Objects instance as a copy of the given one.
long	copy (Objects objs) Performs the copy operation.
long	count () Gets the number of elements into the collection.
long	difference (Objects objs) Performs the difference operation.
boolean	equals (Object o) Returns true if the collection is equal to the object.
boolean	equals (Objects objs) Checks if the given Objects contains the same information.
boolean	exists (long e) Gets if the given element exists into the collection.
long	intersection (Objects objs) Performs the intersection operation.
boolean	isClosed () Gets if Objects instance has been closed or not.
boolean	isEmpty () Returns true if this Objects contains no elements.
ObjectsIterator	iterator () Gets an ObjectsIterator.
ObjectsIterator	iteratorFromElement (long e) Gets an ObjectsIterator starting from the given element.
ObjectsIterator	iteratorFromIndex (long index) Gets an ObjectsIterator skipping index elements.
boolean	remove (long e) Removes an element from the collection.

boolean	remove (Object o) Removes the specified element from this set if it is present (optional operation).
boolean	removeAll (Collection clctn) Removes from this set all of its elements that are contained in the specified collection (optional operation).
boolean	retainAll (Collection clctn) Retains only the elements in this set that are contained in the specified collection (optional operation).
Objects	sample (Objects exclude, long samples) Creates a new Objects instance which is a sample of the calling one.
int	size () Gets the size of the collection.
Object[]	toArray () Returns an array containing all of the object identifiers in this set.
Object[]	toArray (Object[] ts) Returns an array containing all of the object identifiers in this set; the runtime type of the returned array is that of the specified array.
long	union (Objects objs) Performs the union operation.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Set

add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toArray

Methods inherited from interface java.util.Collection

add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toArray

Methods inherited from interface java.lang.Iterable

iterator

Methods inherited from interface java.io.Closeable

close

Methods inherited from interface java.lang.Iterable

iterator

Fields

(continued from last page)

InvalidOID

```
public static int InvalidOID
```

Invalid object identifier constant.

Methods

any

```
public long any()  
    throws NoSuchElementException,  
        RuntimeException
```

Gets an element from the collection.

Returns:

Any element from the collection.

Throws:

java.util.NoSuchElementException - whether the collection is empty.
java.lang.RuntimeException - null

contains

```
public boolean contains(Object o)
```

Returns true if this collections contains the specified element or Objects.

Parameters:

o - element or Objects whose presence in this set is to be tested.

Returns:

true if this set contains the specified element or Objects.

remove

```
public boolean remove(Object o)
```

Removes the specified element from this set if it is present (optional operation).

More formally, removes an element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)), if the set contains such an element. Returns true if the set contained the specified element (or equivalently, if the set changed as a result of the call). (The set will not contain the specified element once the call returns.)

Parameters:

o - object to be removed from this set, if present.

Returns:

true if the set contained the specified element.

(continued from last page)

equals

```
public boolean equals(Object o)
```

Returns true if the collection is equal to the object.

Parameters:

o - object to compare with the collection.

Returns:

true if the objects are equal or false otherwise.

difference

```
public long difference(Objects 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.

combineUnion

```
public static Objects combineUnion(Objects objs1,  
    Objects 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.

containsAll

```
public boolean containsAll(Collection clctn)
```

Returns true if this set contains all of the elements of the specified collection.

If the specified collection is also a set, this method returns true if it is a subset of this set.

Parameters:

clctn - collection to be checked for containment in this set.

Returns:

true if this set contains all of the elements of the specified collection.

(continued from last page)

iteratorFromElement

```
public ObjectsIterator iteratorFromElement(long 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.

equals

```
public boolean equals(Objects 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.

add

```
public boolean add(Long e)
```

Adds the specified element to this set if it is not already present (optional operation).

More formally, adds the specified element, o, to this set if this set contains no element e such that (o==null ? e==null : o.equals(e)). If this set already contains the specified element, the call leaves this set unchanged and returns false. In combination with the restriction on constructors, this ensures that sets never contain duplicate elements. The stipulation above does not imply that sets must accept all elements; sets may refuse to add any particular element, including null, and throwing an exception, as described in the specification for Collection.add. Individual set implementations should clearly document any restrictions on the the elements that they may contain.

Parameters:

e - element to be added to this set.

Returns:

true if this set did not already contain the specified element.

copy

```
public long copy(Objects 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.

combineIntersection

```
public static Objects combineIntersection(Objects objs1,  
                                         Objects 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.

close

```
public void close()
```

Closes the Objects instance.

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

isEmpty

```
public boolean isEmpty()
```

Returns true if this Objects contains no elements.

Returns:

true if the collection contains no elements.

contains

```
public boolean contains(Objects objs)
```

Check if this objects contains the other one.

Parameters:

objs - Objects collection.

Returns:

True if it contains the given object.

clear

```
public void clear()
```

Clears the collection removing all its elements.

(continued from last page)

count

```
public long count()
```

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

retainAll

```
public boolean retainAll(Collection clctn)
```

Retains only the elements in this set that are contained in the specified collection (optional operation).

In other words, removes from this set all of its elements that are not contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the intersection of the two sets.

Parameters:

`clctn` - collection that defines which elements this set will retain.

Returns:

true if this collection changed as a result of the call.

remove

```
public boolean remove(long 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.

iteratorFromIndex

```
public ObjectsIterator iteratorFromIndex(long 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.

union

```
public long union(Objects objs)
```

(continued from last page)

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.

isClosed

```
public boolean isClosed()
```

Gets if Objects instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

add

```
public boolean add(long 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.

removeAll

```
public boolean removeAll(Collection clctn)
```

Removes from this set all of its elements that are contained in the specified collection (optional operation).

If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

Parameters:

clctn - collection that defines which elements will be removed from this set.

Returns:

true if this set changed as a result of the call

toArray

```
public Object[] toArray()
```

Returns an array containing all of the object identifiers in this set.

Obeys the general contract of the Collection.toArray method.

(continued from last page)

Returns:

an array containing all of the elements in this set.

toArray

```
public Object[] toArray(Object[] ts)
```

Returns an array containing all of the object identifiers in this set; the runtime type of the returned array is that of the specified array.

Obeys the general contract of the `Collection.toArray(Object[])` method.

Parameters:

`ts` - the array into which the elements of this set are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose.

Returns:

an array containing the elements of this set.

addAll

```
public boolean addAll(Collection clctn)
```

Adds all of the elements in the specified collection to this set if they're not already present (optional operation).

If the specified collection is also a set, the `addAll` operation effectively modifies this set so that its value is the union of the two sets. The behavior of this operation is unspecified if the specified collection is modified while the operation is in progress.

Parameters:

`clctn` - collection whose elements are to be added to this set.

Returns:

true if this set changed as a result of the call.

exists

```
public boolean exists(long 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.

combineDifference

```
public static Objects combineDifference(Objects objs1,  
                                     Objects 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.

(continued from last page)

objs2 - [in] Objects instance.

Returns:

New Objects instance.

sample

```
public Objects sample(Objects exclude,  
                       long 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.

size

```
public int size()
```

Gets the size of the collection.

It is the same as count() if the number of elements is <= java.lang.Integer.MAX_VALUE, otherwise java.lang.Integer.MAX_VALUE is returned.

Returns:

It returns the same as count() or java.lang.Integer.MAX_VALUE.

intersection

```
public long intersection(Objects 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.

iterator

```
public ObjectsIterator iterator()
```

Gets an ObjectsIterator.

Returns:

ObjectsIterator instance.

(continued from last page)

copy

```
public Objects copy( )
```

Creates a new Objects instance as a copy of the given one.

Returns:

The new Objects instance.

com.sparsity.sparksee.gdb Class ObjectsIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.ObjectsIterator

All Implemented Interfaces:

Iterator, Closeable

public class **ObjectsIterator**
extends Object
implements Closeable, Iterator

ObjectsIterator class.

Iterator to traverse all the object identifiers from an Objects instance.

Author:

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

Method Summary

void	close() Closes the ObjectsIterator instance.
boolean	hasNext() Gets if there are more elements to traverse.
boolean	isClosed() Gets if ObjectsIterator instance has been closed or not.
Long	next() See nextObject().
long	nextObject() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods inherited from interface java.util.Iterator

hasNext, next, remove

(continued from last page)

Methods

nextObject

```
public long nextObject()
```

Gets the next element.

hasNext

```
public boolean hasNext()
```

Gets if there are more elements to traverse.

Returns:

TRUE if there are more elements to traverse, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Long next()
```

See nextObject().

isClosed

```
public boolean isClosed()
```

Gets if ObjectsIterator instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the ObjectsIterator instance.

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

com.sparsity.sparksee.gdb

Class ObjectType

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.ObjectType

```

All Implemented Interfaces:
 Serializable, Comparable

public final class **ObjectType**
 extends Enum

Object type enumeration.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Edge Edge object type.
public static final	Node Node object type.

Method Summary

static ObjectType	valueOf (String name)
static ObjectType[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

Node

```
public static final com.sparsity.sparksee.gdb.ObjectType Node
```

Node object type.

Edge

```
public static final com.sparsity.sparksee.gdb.ObjectType Edge
```

Edge object type.

Methods

values

```
public static ObjectType\[\] values()
```

valueOf

```
public static ObjectType valueOf(String name)
```

com.sparsity.sparksee.gdb

Class OIDList

java.lang.Object

└─com.sparsity.sparksee.gdb.OIDList

All Implemented Interfaces:

Iterable

```
public class OIDList
extends Object
implements Iterable
```

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>

Constructor Summary

public	OIDList (long[] list) Creates a new instance from a long array.
public	OIDList (Collection col) Creates a new instance from a long collection.
public	OIDList (int numInvalidOIDs) Constructor.
public	OIDList () Constructor.

Method Summary

void	add (long attr) Adds a Sparksee object identifier at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
OIDListIterator	iterator () Gets a new OIDListIterator.
void	set (int pos, long oid) Sets a Sparksee object identifier at the specified position of the list.

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Methods inherited from interface `java.lang.Iterable`

```
iterator
```

Constructors

`OIDList`

```
public OIDList(long[] list)
```

Creates a new instance from a long array.

Parameters:

`list` - Long array to initialize the instance.

`OIDList`

```
public OIDList(Collection col)
```

Creates a new instance from a long collection.

Parameters:

`col` - Collection to initialize the instance.

`OIDList`

```
public OIDList(int numInvalidOIDs)
```

Constructor.

This creates a list with N invalid oids.

Parameters:

`numInvalidOIDs` - [in] The number of invalid oids added to the list.

`OIDList`

```
public OIDList()
```

Constructor.

This creates an empty list.

Methods

`add`

```
public void add(long attr)
```

(continued from last page)

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

Parameters:

`attr` - [in] Sparksee object identifier.

clear

```
public void clear()
```

Clears the list.

set

```
public void set(int pos,  
                long 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.

iterator

```
public OIDListIterator iterator()
```

Gets a new OIDListIterator.

Returns:

OIDListIterator instance.

count

```
public int count()
```

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb Class OIDListIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.OIDListIterator

All Implemented Interfaces:
Iterator

public class **OIDListIterator**
extends Object
implements Iterator

OIDList iterator class.

Iterator to traverse all the Sparksee object identifier into a OIDList instance.

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext() Gets if there are more elements.
Long	next() See nextOID().
long	nextOID() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean **hasNext()**

Gets if there are more elements.

(continued from last page)

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Long next()
```

See nextOID().

nextOID

```
public long nextOID()
```

Gets the next element.

com.sparsity.sparksee.gdb

Class Order

```

java.lang.Object
  |
  +- java.lang.Enum
        +- com.sparsity.sparksee.gdb.Order
  
```

All Implemented Interfaces:
 Serializable, Comparable

public final class **Order**
 extends Enum

Order enumeration.

Author:
 Sparsity Technologies <http://www.sparsity-technologies.com>

Field Summary

public static final	Ascendent From lower to higher.
public static final	Descendent From higher to lower.

Method Summary

static Order	valueOf (String name)
static Order[]	values ()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

Ascendent

```
public static final com.sparsity.sparksee.gdb.Order Ascendent
```

From lower to higher.

Descendent

```
public static final com.sparsity.sparksee.gdb.Order Descendent
```

From higher to lower.

Methods

values

```
public static Order\[\] values()
```

valueOf

```
public static Order valueOf(String name)
```


com.sparsity.sparksee.gdb Class Platform

java.lang.Object

└─com.sparsity.sparksee.gdb.Platform

public class **Platform**
extends Object

Platform class.

Author:

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

Method Summary

static void	getStatistics (PlatformStatistics stats)
	Gets platform data and statistics.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getStatistics

public static void **getStatistics**([PlatformStatistics](#) stats)

Gets platform data and statistics.

Parameters:

stats - [in/out] This updates the given PlatformStatistics.

com.sparsity.sparksee.gdb Class PlatformStatistics

java.lang.Object

└─com.sparsity.sparksee.gdb.PlatformStatistics

public class **PlatformStatistics**
extends Object

Platform data and statistics.

Author:

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

Constructor Summary

public	PlatformStatistics() Creates a new instance setting all values to 0.
--------	---

Method Summary

long	getAvailableMem() Gets avialable (free) memory size in Bytes.
int	getNumCPUs() Gets the number of CPUs.
long	getRealTime() Gets time in microseconds (since epoch).
long	getSystemTime() Gets CPU system time.
long	getTotalMem() Gets physical memory size in Bytes.
long	getUserTime() Gets CPU user time.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

PlatformStatistics

public **PlatformStatistics()**

(continued from last page)

Creates a new instance setting all values to 0.

Methods

getRealTime

```
public long getRealTime()
```

Gets time in microseconds (since epoch).

Returns:

Time in microseconds (since epoch).

getAvailableMem

```
public long getAvailableMem()
```

Gets available (free) memory size in Bytes.

Returns:

Available (free) memory size in Bytes.

getTotalMem

```
public long getTotalMem()
```

Gets physical memory size in Bytes.

Returns:

Physical memory size in Bytes.

getSystemTime

```
public long getSystemTime()
```

Gets CPU system time.

Returns:

CPU system time.

getUserTime

```
public long getUserTime()
```

Gets CPU user time.

Returns:

(continued from last page)

CPU user time.

getNumCPUs

```
public int getNumCPUs()
```

Gets the number of CPUs.

Returns:

The number of CPUs.

com.sparsity.sparksee.gdb

Class Query

java.lang.Object

└─com.sparsity.sparksee.gdb.Query

public class **Query**
extends Object

Query class.

Author:

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

Method Summary

ResultSet	execute (String stmt) Executes the given statement.
void	setDynamic (String name, Value value) Sets the value for a dynamic paramater.
QueryStream	setStream (String stream, QueryStream handler) Sets a query stream handler.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

setDynamic

```
public void setDynamic(String name,
    Value value)
```

Sets the value for a dynamic paramater.

Parameters:

name - [in] Parameter name

value - [in] Parameter value

setStream

```
public QueryStream setStream(String stream,
    QueryStream handler)
```

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.

(continued from last page)

Parameters:

stream - [in] The stream name
handler - [in] Query stream handler

Returns:

The previous handler, or NULL if it does not exists

execute

```
public ResultSet execute(String stmt)
```

Executes the given statement.

Parameters:

stmt - [in] Query statement.

Returns:

A ResultSet instance with the contents of the result of the query.

com.sparsity.sparksee.gdb

Class QueryContext

java.lang.Object

└─com.sparsity.sparksee.gdb.QueryContext

public class **QueryContext**
extends Object

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>

Constructor Summary

public	QueryContext() Default constructor.
--------	--

Method Summary

Query	newQuery() Creates a new Query.
-----------------------	--

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

QueryContext

public **QueryContext()**

Default constructor.

Methods

newQuery

public [Query](#) **newQuery()**

Creates a new Query.

com.sparsity.sparksee.gdb Class QueryStream

java.lang.Object

└--com.sparsity.sparksee.gdb.QueryStream

public class **QueryStream**
extends Object

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>

Method Summary

boolean	fetch (ValueList list) Gets the next row and moves the iterator forward.
boolean	prepare (ValueList list) Prepares the stream before it is started.
boolean	start (ResultSetList list) Starts the stream.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

start

public boolean **start**([ResultSetList](#) list)

Starts the stream.

Parameters:

list - [in] Optional list of input ResultSets

Returns:

FALSE on error

(continued from last page)

fetch

```
public boolean fetch(ValueList 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

prepare

```
public boolean prepare(ValueList list)
```

Prepares the stream before it is started.

Parameters:

list - [in] Optional list of arguments

Returns:

FALSE on error

com.sparsity.sparksee.gdb

Class ResultSet

java.lang.Object

└─com.sparsity.sparksee.gdb.ResultSet

public class **ResultSet**
extends Object

ResultSet class.

Author:

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

Method Summary

Value	getColumn (int index) Gets the value for the given column.
void	getColumn (int index, Value value) Gets the value for the given column.
DataType	getColumnDataType (int index) Gets the datatype for the given column.
int	getColumnIndex (String name) Gets the column index for the given column name.
String	getColumnName (int index) Gets the name for the given column.
String	getJSON (int rows) Returns rows in JSON format.
int	getNumColumns () Gets the number of columns.
boolean	next () Fetches the next row.
void	rewind () Positions the cursor before the first row.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

(continued from last page)

rewind

```
public void rewind()
```

Positions the cursor before the first row.

getColumn

```
public void getColumn(int index,  
    Value value)
```

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

Parameters:

index - [in] Column index.

value - [in|out] Value.

getColumn

```
public Value getColumn(int 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.

getColumnIndex

```
public int getColumnIndex(String name)
```

Gets the column index for the given column name.

Parameters:

name - [in] Column name.

Returns:

Column index.

next

```
public boolean next()
```

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.

(continued from last page)

Returns:

TRUE if the next row has been successfully fetched, FALSE otherwise.

getJSON

```
public String getJSON(int 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 in the resultset

getColumnName

```
public String getColumnName(int index)
```

Gets the name for the given column.

Parameters:

index - [in] Column index.

Returns:

Column name.

getColumnDataType

```
public DataType getColumnDataType(int index)
```

Gets the datatype for the given column.

Parameters:

index - [in] Column index.

Returns:

DataType for the given column.

getNumColumns

```
public int getNumColumns()
```

Gets the number of columns.

Columns are in the range [0...COLUMNS).

Returns:

The number of columns.

com.sparsity.sparksee.gdb

Class ResultSetList

java.lang.Object

└─com.sparsity.sparksee.gdb.ResultSetList

public class **ResultSetList**
extends Object

ResultSet list.

It stores a ResultSet list.

Author:

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

Constructor Summary

public	ResultSetList() Constructor.
--------	---

Method Summary

void	clear() Clears the list.
int	count() Number of elements in the list.
ResultSet	get(int index) Returns the ResultSet at the specified position in the list.
ResultSetListIterator	iterator() Gets a new ResultSetListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

ResultSetList

public **ResultSetList()**

Constructor.

This creates an empty list.

Methods

(continued from last page)

get

```
public ResultSet get(int index)
```

Returns the ResultSet at the specified position in the list.

Parameters:

index - [in] Index of the element to return, starting at 0.

clear

```
public void clear()
```

Clears the list.

iterator

```
public ResultSetListIterator iterator()
```

Gets a new ResultSetListIterator.

Returns:

ResultSetListIterator instance.

count

```
public int count()
```

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb Class ResultSetListIterator

java.lang.Object
└─com.sparsity.sparksee.gdb.ResultSetListIterator

public class **ResultSetListIterator**
extends Object

ResultSetList iterator class.

Iterator to traverse all the values into a ResultSetList instance.
Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext () Gets if there are more elements.
ResultSet	next () Moves to the next element.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

hasNext

public boolean **hasNext**()

Gets if there are more elements.

Returns:
TRUE if there are more elements, FALSE otherwise.

next

public [ResultSet](#) **next**()

Moves to the next element.

Returns:
The next element.

com.sparsity.sparksee.gdb Class Session

java.lang.Object

└─com.sparsity.sparksee.gdb.Session

All Implemented Interfaces:

Closeable

```
public class Session
extends Object
implements Closeable
```

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>

Method Summary

void	<code>begin()</code> Begins a transaction.
void	<code>beginUpdate()</code> Begins an update transaction.
void	<code>close()</code> Closes the Session instance.
void	<code>commit()</code> Commits a transaction.
<code>Graph</code>	<code>getGraph()</code> Gets the Graph instance.
boolean	<code>isClosed()</code> Gets if Session instance has been closed or not.
<code>Objects</code>	<code>newObjects()</code> Creates a new Objects instance.
<code>Query</code>	<code>newQuery()</code> Creates a new Query.

void

[rollback\(\)](#)

Rollbacks a transaction.

Methods inherited from class `java.lang.Object``clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`**Methods inherited from interface** `java.io.Closeable``close`

Methods

rollback

```
public void rollback()
```

Rollbacks a transaction.

beginUpdate

```
public void beginUpdate()
```

Begins an update transaction.

newQuery

```
public Query newQuery()
```

Creates a new Query.

isClosed

```
public boolean isClosed()
```

Gets if Session instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

commit

```
public void commit()
```

(continued from last page)

Commits a transaction.

getGraph

```
public Graph getGraph()
```

Gets the Graph instance.

Returns:

The Graph instance.

close

```
public void close()
```

Closes the Session instance.

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

begin

```
public void begin()
```

Begins a transaction.

newObjects

```
public Objects newObjects()
```

Creates a new Objects instance.

Returns:

The new Objects instance.

com.sparsity.sparksee.gdb

Class Sparksee

java.lang.Object

└─com.sparsity.sparksee.gdb.Sparksee

All Implemented Interfaces:

Closeable

```
public class Sparksee
  extends Object
  implements Closeable
```

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>

Field Summary

public static	Version Sparksee version.
---------------	--

Constructor Summary

public	Sparksee (SparkseeConfig config) Creates a new instance.
--------	--

Method Summary

void	close () Closes the Sparksee instance.
Database	create (String path, String alias) Creates a new Database instance.
boolean	isClosed () Gets if Sparksee instance has been closed or not.
Database	open (String path, boolean readOnly) Opens an existing Database instance.
Database	restore (String path, String backupFile) Restores a Database from a backup file.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Fields

Version

```
public static java.lang.String Version
```

Sparksee version.

Constructors

Sparksee

```
public Sparksee(SparkseeConfig config)
```

Creates a new instance.

Parameters:

config - [in] Sparksee configuration.

Methods

create

```
public Database create(String path,  
                        String alias)  
    throws FileNotFoundException,  
           RuntimeException
```

Creates a new Database instance.

Parameters:

path - [in] Database storage file.
alias - [in] Database alias name.

Returns:

A Database instance.

Throws:

java.io.FileNotFoundException - If the given file cannot be created.
java.lang.RuntimeException - null

restore

```
public Database restore(String path,  
                        String backupFile)  
    throws FileNotFoundException,  
           RuntimeException
```

(continued from last page)

Restores a Database from a backup file.

See the Graph class Backup method.

Parameters:

path - [in] Database storage file.
backupFile - [in] The Backup file to be restored.

Returns:

A Database instance.

Throws:

java.io.FileNotFoundException - If the given file cannot be created, or the exported data file does not exists.
java.lang.RuntimeException - null

isClosed

```
public boolean isClosed()
```

Gets if Sparksee instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

open

```
public Database open(String path,  
                    boolean readOnly)  
throws FileNotFoundException,  
    RuntimeException
```

Opens an existing Database instance.

Parameters:

path - [in] Database storage file.
readOnly - [in] If TRUE, open Database in read-only mode.

Returns:

A Database instance.

Throws:

java.io.FileNotFoundException - If the given file does not exist.
java.lang.RuntimeException - null

close

```
public void close()
```

Closes the Sparksee instance.

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

com.sparsity.sparksee.gdb Class SparkseeConfig

```
java.lang.Object
  |
  +--com.sparsity.sparksee.gdb.SparkseeConfig
```

```
public class SparkseeConfig
  extends Object
```

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 overridden with values from a properties file (see SparkseeProperties class). Also, this settings can be overridden 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).

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 msec [TimeUnit] ('sparksee.cache.statisticsSnapshotTime' at SparkseeProperties).

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:

[D|H|M|S|s|m|u]

where 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.

Author:

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

Constructor Summary

public	SparkseeConfig() Creates a new instance.
--------	---

Method Summary

int	getCacheMaxSize() Gets the maximum size for the cache (all pools) in MB.
boolean	getCacheStatisticsEnabled() Gets whether cache statistics are enabled or disabled.
String	getCacheStatisticsFile() Gets the cache statistics log file.
long	getCacheStatisticsSnapshotTime() Gets the cache statistics snapshot time in microseconds.
int	getExtentPages() Gets the number of pages per extent.
int	getExtentSize() Gets the size of a extent.
String	getHighAvailabilityCoordinators() Gets the coordinators address and port list.
boolean	getHighAvailabilityEnabled() Gets whether high availability mode is enabled or disabled.
String	getHighAvailabilityIP() Gets the IP address and port of the instance.
long	getHighAvailabilityMasterHistory() Gets the master's history log.
long	getHighAvailabilitySynchronization() Gets the synchronization polling time.

String	<u>getLicense()</u> Gets the license code.
String	<u>getLogFile()</u> Gets the log file.
<u>LogLevel</u>	<u>getLogLevel()</u> Gets the log level.
int	<u>getPoolClusterSize()</u> Gets the number of pools in each PoolCluster.
int	<u>getPoolFrameSize()</u> Gets the size of a pool frame in number of extents.
int	<u>getPoolPersistentMaxSize()</u> Gets the maximum size for the persistent pool in number of frames.
int	<u>getPoolPersistentMinSize()</u> Gets the minimum size for the persistent pool in number of frames.
int	<u>getPoolTemporaryMaxSize()</u> Gets the maximum size for the temporary pool in number of frames.
int	<u>getPoolTemporaryMinSize()</u> Gets the minimum size for the temporary pool in number of frames.
int	<u>getRecoveryCacheMaxSize()</u> Gets the maximum size for the recovery log cache in extents.
long	<u>getRecoveryCheckpointTime()</u> Gets the delay time (in microseconds) between automatic checkpoints.
boolean	<u>getRecoveryEnabled()</u> Gets whether the recovery is enabled or disabled.
String	<u>getRecoveryLogFile()</u> Gets the recovery log file.
boolean	<u>getRollbackEnabled()</u> Gets whether the rollback is enabled or disabled.
void	<u>setCacheMaxSize(int megaBytes)</u> Sets the maximum size for the cache (all pools) in MB.
void	<u>setCacheStatisticsEnabled(boolean status)</u> Enables or disables cache statistics.
void	<u>setCacheStatisticsFile(String filePath)</u> Sets the cache statistics log file.
void	<u>setCacheStatisticsSnapshotTime(long microSeconds)</u> Sets the cache statistics snapshot time.
void	<u>setExtentPages(int pages)</u> Sets the number of pages per extent.
void	<u>setExtentSize(int kBytes)</u> Sets the size of the extents in KB.

void	<u>setHighAvailabilityCoordinators</u> (String ip) Sets the coordinators address and port list.
void	<u>setHighAvailabilityEnabled</u> (boolean status) Enables or disables high availability mode.
void	<u>setHighAvailabilityIP</u> (String ip) Sets the IP address and port of the instance.
void	<u>setHighAvailabilityMasterHistory</u> (long filePath) Sets the master's history log.
void	<u>setHighAvailabilitySynchronization</u> (long microSeconds) Sets the synchronization polling time.
void	<u>setLicense</u> (String key) Sets the license code.
void	<u>setLogFile</u> (String filePath) Sets the log file.
void	<u>setLogLevel</u> (<u>LogLevel</u> level) Sets the log level.
void	<u>setPoolClusterSize</u> (int pools) Sets the number of pools in each PoolCluster.
void	<u>setPoolFrameSize</u> (int extents) Sets the size of a pool frame in number of extents.
void	<u>setPoolPersistentMaxSize</u> (int frames) Sets the maximum size for the persistent pool in number of frames.
void	<u>setPoolPersistentMinSize</u> (int frames) Sets the minimum size for the persistent pool in number of frames.
void	<u>setPoolTemporaryMaxSize</u> (int frames) Sets the maximum size for the temporary pool in number of frames.
void	<u>setPoolTemporaryMinSize</u> (int frames) Sets the minimum size for the temporary pool in number of frames.
void	<u>setRecoveryCacheMaxSize</u> (int extents) Sets the maximum size for the recovery log cache in extents.
void	<u>setRecoveryCheckpointTime</u> (long microSeconds) Sets the delay time (in microseconds) between automatic checkpoints.
void	<u>setRecoveryEnabled</u> (boolean status) Enables or disables the recovery.
void	<u>setRecoveryLogFile</u> (String filePath) Sets the recovery log file.
void	<u>setRollbackEnabled</u> (boolean status) Enables or disables the rollback.

Methods inherited from class `java.lang.Object`

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Constructors

SparkseeConfig

```
public SparkseeConfig()
```

Creates a new instance.

Values are set with default values.

Methods

setLicense

```
public void setLicense(String key)
```

Sets the license code.

Parameters:

key - [in] The license code.

getHighAvailabilitySynchronization

```
public long getHighAvailabilitySynchronization()
```

Gets the synchronization polling time.

Returns:

The Synchronization polling time.

setExtentPages

```
public void setExtentPages(int 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.

setHighAvailabilityCoordinators

```
public void setHighAvailabilityCoordinators(String ip)
```

Sets the coordinators address and port list.

(continued from last page)

Parameters:

ip - [in] The coordinators address and port list.

getExtentSize

```
public int getExtentSize()
```

Gets the size of a extent.

Returns:

The size of a extent in KB.

setLogFile

```
public void setLogFile(String filePath)
```

Sets the log file.

Parameters:

filePath - [in] The log file.

setLogLevel

```
public void setLogLevel(LogLevel level)
```

Sets the log level.

Parameters:

level - [in] The LogLevel.

setCacheStatisticsEnabled

```
public void setCacheStatisticsEnabled(boolean status)
```

Enables or disables cache statistics.

Parameters:

status - [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

getLogFile

```
public String getLogFile()
```

Gets the log file.

Returns:

The log file.

(continued from last page)

setPoolClusterSize

```
public void setPoolClusterSize(int pools)
```

Sets the number of pools in each PoolCluster.

Parameters:

pools - [in] The number of pools in each PoolCluster. It must be non-negative.

setCacheStatisticsSnapshotTime

```
public void setCacheStatisticsSnapshotTime(long microSeconds)
```

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters:

microSeconds - [in] The cache statistics snapshot time in microseconds.

getRecoveryCheckpointTime

```
public long getRecoveryCheckpointTime()
```

Gets the delay time (in microseconds) between automatic checkpoints.

Returns:

The delay time (in microseconds) between automatic checkpoints.

getCacheStatisticsEnabled

```
public boolean getCacheStatisticsEnabled()
```

Gets whether cache statistics are enabled or disabled.

Returns:

TRUE if cache statistics are enabled, FALSE otherwise.

getPoolPersistentMaxSize

```
public int getPoolPersistentMaxSize()
```

Gets the maximum size for the persistent pool in number of frames.

Returns:

The maximum size for the persistent pool in number of frames.

setPoolPersistentMaxSize

```
public void setPoolPersistentMaxSize(int frames)
```

(continued from last page)

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.

getRecoveryLogFile

```
public String getRecoveryLogFile()
```

Gets the recovery log file.

Returns:

The recovery log file.

setHighAvailabilityMasterHistory

```
public void setHighAvailabilityMasterHistory(long filePath)
```

Sets the master's history log.

Parameters:

filePath - [in] The master's history log.

setCacheStatisticsFile

```
public void setCacheStatisticsFile(String filePath)
```

Sets the cache statistics log file.

Useless if cache statistics are disabled.

Parameters:

filePath - [in] The cache statistics log file.

getHighAvailabilityCoordinators

```
public String getHighAvailabilityCoordinators()
```

Gets the coordinators address and port list.

Returns:

The coordinators address and port list.

setPoolFrameSize

```
public void setPoolFrameSize(int extents)
```

Sets the size of a pool frame in number of extents.

Parameters:

(continued from last page)

extents - [in] The size of a pool frame in number of extents. It must be non-negative.

getCacheStatisticsFile

```
public String getCacheStatisticsFile()
```

Gets the cache statistics log file.

Useless if cache statistics are disabled.

Returns:

The cache statistics log file.

getCacheStatisticsSnapshotTime

```
public long getCacheStatisticsSnapshotTime()
```

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

Returns:

The cache statistics snapshot time in microseconds.

getPoolTemporaryMaxSize

```
public int getPoolTemporaryMaxSize()
```

Gets the maximum size for the temporary pool in number of frames.

Returns:

The maximum size for the temporary pool in number of frames.

setRecoveryEnabled

```
public void setRecoveryEnabled(boolean status)
```

Enables or disables the recovery.

Parameters:

status - [in] If TRUE this enables the recovery, if FALSE then disables it.

getLicense

```
public String getLicense()
```

Gets the license code.

Returns:

The license code.

(continued from last page)

setPoolTemporaryMinSize

```
public void setPoolTemporaryMinSize(int 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.

getHighAvailabilityIP

```
public String getHighAvailabilityIP()
```

Gets the IP address and port of the instance.

Returns:

The IP address and port of the instance.

getLogLevel

```
public LogLevel getLogLevel()
```

Gets the log level.

Returns:

The LogLevel.

setHighAvailabilitySynchronization

```
public void setHighAvailabilitySynchronization(long microseconds)
```

Sets the synchronization polling time.

Parameters:

microseconds - [in] The synchronization polling time.

setCacheMaxSize

```
public void setCacheMaxSize(int megaBytes)
```

Sets the maximum size for the cache (all pools) in MB.

Parameters:

megaBytes - [in] The maximum size for the cache (all pools) in MB. It must be non-negative.

getPoolPersistentMinSize

```
public int getPoolPersistentMinSize()
```

(continued from last page)

Gets the minimum size for the persistent pool in number of frames.

Returns:

The minimum size for the persistent pool in number of frames.

setHighAvailabilityEnabled

```
public void setHighAvailabilityEnabled(boolean status)
```

Enables or disables high availability mode.

Parameters:

status - [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.

getPoolClusterSize

```
public int getPoolClusterSize()
```

Gets the number of pools in each PoolCluster.

Returns:

The number of pools in each PoolCluster.

setRecoveryCacheMaxSize

```
public void setRecoveryCacheMaxSize(int 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).

setHighAvailabilityIP

```
public void setHighAvailabilityIP(String ip)
```

Sets the IP address and port of the instance.

Parameters:

ip - [in] The IP address and port of the instance.

setExtentSize

```
public void setExtentSize(int kBytes)
```

Sets the size of the extents in KB.

(continued from last page)

Parameters:

kBytes - [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.

setRollbackEnabled

```
public void setRollbackEnabled(boolean status)
```

Enables or disables the rollback.

Parameters:

status - [in] If TRUE this enables the rollback, if FALSE then disables it.

getExtentPages

```
public int getExtentPages()
```

Gets the number of pages per extent.

Returns:

The number of pages per extent.

setPoolTemporaryMaxSize

```
public void setPoolTemporaryMaxSize(int 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.

getHighAvailabilityEnabled

```
public boolean getHighAvailabilityEnabled()
```

Gets whether high availability mode is enabled or disabled.

Returns:

TRUE if high availability mode is enabled, FALSE otherwise.

getRecoveryEnabled

```
public boolean getRecoveryEnabled()
```

Gets whether the recovery is enabled or disabled.

Returns:

TRUE if the recovery is enabled, FALSE otherwise.

setPoolPersistentMinSize

```
public void setPoolPersistentMinSize(int 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.

getCacheMaxSize

```
public int getCacheMaxSize()
```

Gets the maximum size for the cache (all pools) in MB.

Returns:

The maximum size for the cache (all pools) in MB.

getPoolFrameSize

```
public int getPoolFrameSize()
```

Gets the size of a pool frame in number of extents.

Returns:

The size of a pool frame in number of extents.

getRollbackEnabled

```
public boolean getRollbackEnabled()
```

Gets whether the rollback is enabled or disabled.

Returns:

TRUE if the rollback is enabled, FALSE otherwise.

getPoolTemporaryMinSize

```
public int getPoolTemporaryMinSize()
```

Gets the minimum size for the temporary pool in number of frames.

Returns:

The minimum size for the temporary pool in number of frames.

setRecoveryCheckpointTime

```
public void setRecoveryCheckpointTime(long microseconds)
```

(continued from last page)

Sets the delay time (in microseconds) between automatic checkpoints.

Parameters:

`microSeconds` - [in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each committed transaction.

getRecoveryCacheMaxSize

```
public int getRecoveryCacheMaxSize()
```

Gets the maximum size for the recovery log cache in extents.

Returns:

The maximum size for the recovery log cache in extents.

getHighAvailabilityMasterHistory

```
public long getHighAvailabilityMasterHistory()
```

Gets the master's history log.

Returns:

The master's history log.

setRecoveryLogFile

```
public void setRecoveryLogFile(String filePath)
```

Sets the recovery log file.

Parameters:

`filePath` - [in] The recovery log file. Left it empty for the default log file (same as .log)

com.sparsity.sparksee.gdb

Class SparkseeProperties

```
java.lang.Object
└--com.sparsity.sparksee.gdb.SparkseeProperties
```

```
public class SparkseeProperties
extends Object
```

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.

Method Summary		
static String	get (String key, String def)	Gets a property.
static boolean	getBoolean (String key, boolean def)	Gets a property as a boolean.
static int	getInteger (String key, int def)	Gets a property as an integer.
static long	getTimeUnit (String key, long def)	Gets a property as a time unit.
static void	load (String path)	Loads properties from the given file path.
Methods inherited from class java.lang.Object		
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait		

Methods

get

```
public static String get(String key,
                        String def)

    Gets a property.
```

(continued from last page)

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.

getTimeUnit

```
public static long getTimeUnit(String key,  
                                long 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 beginning 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' or '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.

getBoolean

```
public static boolean getBoolean(String key,  
                                  boolean 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.

load

```
public static void load(String path)
```

Loads properties from the given file path.

Parameters:

path - [in] File path to load properties from.

(continued from last page)

getInteger

```
public static int getInteger(String key,  
                             int 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.

com.sparsity.sparksee.gdb

Class StringList

java.lang.Object

└─com.sparsity.sparksee.gdb.StringList

All Implemented Interfaces:

Iterable

public class **StringList**
 extends Object
 implements Iterable

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>

Constructor Summary

public	StringList (Collection col) Creates a new instance from an string collection.
public	StringList () Constructor.
public	StringList (String[] list) Creates a new instance from an string array.

Method Summary

void	add (String str) Adds a String at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
StringListIterator	iterator () Gets a new StringListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Constructors

StringList

```
public StringList(Collection col)
```

Creates a new instance from an string collection.

Parameters:

col - Collection to initialize the instance.

StringList

```
public StringList()
```

Constructor.

This creates an empty list.

StringList

```
public StringList(String[] list)
```

Creates a new instance from an string array.

Parameters:

list - String array to initialize the instance.

Methods

clear

```
public void clear()
```

Clears the list.

iterator

```
public StringListIterator iterator()
```

Gets a new StringListIterator.

Returns:

StringListIterator instance.

count

```
public int count()
```

(continued from last page)

Number of elements in the list.

Returns:

Number of elements in the list.

add

```
public void add(String str)
```

Adds a String at the end of the list.

Parameters:

`str` - [in] String.

com.sparsity.sparksee.gdb Class StringListIterator

```
java.lang.Object
  |
  +--com.sparsity.sparksee.gdb.StringListIterator
```

All Implemented Interfaces:
Iterator

```
public class StringListIterator
  extends Object
  implements Iterator
```

StringList iterator class.

Iterator to traverse all the strings into a StringList instance.

Author:
Sparsity Technologies <http://www.sparsity-technologies.com>

Method Summary

boolean	hasNext() Gets if there are more elements.
String	next() See nextString().
String	nextString() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

```
public boolean hasNext()
```

Gets if there are more elements.

(continued from last page)

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public String next()
```

See nextString().

nextString

```
public String nextString()
```

Gets the next element.

com.sparsity.sparksee.gdb Class TextStream

java.lang.Object

└--com.sparsity.sparksee.gdb.TextStream

All Implemented Interfaces:

Closeable

```
public class TextStream
extends Object
implements Closeable
```

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 GetAttributeText 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>

Constructor Summary

public	TextStream (boolean append) Creates a new instance.
--------	--

Method Summary

void	close () Closes the stream.
boolean	isNull () Returns TRUE if the stream is not available.
int	read (char[] dataOUT, int length) Read data from the stream.
void	write (char[] dataIN, int length) Write data to the stream.

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Methods inherited from interface `java.io.Closeable`

```
close
```

Constructors

TextStream

```
public TextStream(boolean 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.

Methods

read

```
public int read(char[] dataOUT,  
               int length)
```

Read data from the stream.

Parameters:

dataOUT - [out] Buffer to read data to. It must be allocated by the user.
length - [in] Length of the given data buffer. It must be > 0.

Returns:

Amount of read data (<= length). If 0, there is no more data to be read from the stream.

isNull

```
public boolean isNull()
```

Returns TRUE if the stream is not available.

It returns for reading or writing data.

Returns:

FALSE if the stream is ready

write

```
public void write(char[] dataIN,  
                 int length)
```

(continued from last page)

Write data to the stream.

Parameters:

`dataIN` - [in] Buffer to write data from.

`length` - [in] Length of the data buffer. It must be > 0 .

close

```
public void close()
```

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.

com.sparsity.sparksee.gdb

Class Type

java.lang.Object

└─com.sparsity.sparksee.gdb.Type

public class **Type**
extends Object

Type data class.

It contains information about a node or edge type.

Author:

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

Field Summary

public static	EdgesType Identifier for all edgeType attributes.
public static	GlobalType Global type identifier constant.
public static	InvalidType Invalid type identifier constant.
public static	NodesType Identifier for all nodeType attributes.

Method Summary

boolean	getAreNeighborsIndexed() Gets if this is an edge type with neighbors index.
int	getId() Gets the Sparksee type identifier.
boolean	getIsDirected() Gets if this is a directed edge type.
boolean	getIsRestricted() Gets if this is a restricted edge type.
String	getName() Gets the unique type name.
long	getNumObjects() Gets the number of objects belonging to the type.
ObjectType	getObjectType() Gets the object type.
int	getRestrictedFrom() Gets the tail or source type identifier for restricted edge types.

int	getRestrictedTo() Gets the head or target type identifier for restricted edge types.
-----	---

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

EdgesType

```
public static int EdgesType
```

Identifier for all edgeType attributes.

GlobalType

```
public static int GlobalType
```

Global type identifier constant.

InvalidType

```
public static int InvalidType
```

Invalid type identifier constant.

NodesType

```
public static int NodesType
```

Identifier for all nodeType attributes.

Methods

getRestrictedFrom

```
public int getRestrictedFrom()
```

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.

(continued from last page)

getAreNeighborsIndexed

```
public boolean getAreNeighborsIndexed()
```

Gets if this is an edge type with neighbors index.

Returns:

TRUE for edges types with neighbors index, FALSE otherwise.

getObjectType

```
public ObjectType getObjectType()
```

Gets the object type.

Returns:

The object type.

getRestrictedTo

```
public int getRestrictedTo()
```

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.

getIsRestricted

```
public boolean getIsRestricted()
```

Gets if this is a restricted edge type.

Returns:

TRUE for restricted edge types, FALSE otherwise.

getNumObjects

```
public long getNumObjects()
```

Gets the number of objects belonging to the type.

Returns:

The number of objects belonging to the type.

getId

```
public int getId()
```

(continued from last page)

Gets the Sparksee type identifier.

Returns:

The Sparksee type identifier.

getIsDirected

```
public boolean getIsDirected()
```

Gets if this is a directed edge type.

Returns:

TRUE for directed edge types, FALSE otherwise.

getName

```
public String getName()
```

Gets the unique type name.

Returns:

The unique type name.

com.sparsity.sparksee.gdb

Class TypeList

java.lang.Object

└─com.sparsity.sparksee.gdb.TypeList

All Implemented Interfaces:

Iterable

public class **TypeList**
 extends Object
 implements Iterable

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>

Constructor Summary

public	TypeList (int[] list) Creates a new instance from an integer array.
public	TypeList (Collection col) Creates a new instance from an integer collection.
public	TypeList () Constructor.

Method Summary

void	add (int type) Adds a Sparksee type identifier at the end of the list.
void	clear () Clears the list.
int	count () Number of elements in the list.
TypeListIterator	iterator () Gets a new TypeListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Constructors

TypeList

```
public TypeList(int[] list)
```

Creates a new instance from an integer array.

Parameters:

`list` - Integer array to initialize the instance.

TypeList

```
public TypeList(Collection col)
```

Creates a new instance from an integer collection.

Parameters:

`col` - Collection to initialize the instance.

TypeList

```
public TypeList()
```

Constructor.

This creates an empty list.

Methods

add

```
public void add(int type)
```

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

Parameters:

`type` - [in] Sparksee type identifier.

clear

```
public void clear()
```

Clears the list.

iterator

```
public TypeListIterator iterator()
```

(continued from last page)

Gets a new TypeListIterator.

Returns:

TypeListIterator instance.

count

```
public int count()
```

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb

Class TypeListIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.TypeListIterator

All Implemented Interfaces:
Iterator

public class **TypeListIterator**
extends Object
implements Iterator

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>

Method Summary

boolean	hasNext() Gets if there are more elements.
Integer	next() See nextType().
int	nextType() Gets the next element.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean **hasNext()**

Gets if there are more elements.

(continued from last page)

Returns:

TRUE if there are more elements, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Integer next()
```

See nextType().

nextType

```
public int nextType()
```

Gets the next element.

com.sparsity.sparksee.gdb

Class Value

java.lang.Object

└─com.sparsity.sparksee.gdb.Value

public class **Value**
extends Object

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>

Field Summary

public static	MaxLengthString Maximum number of characters allowed for a String.
---------------	---

Constructor Summary

public	Value (Value value) Copy constructor.
public	Value () Creates a new instance.

Method Summary

int	compare (Value value) Compares with the given Value.
int	compareTo (Object value) See compare().
int	compareTo (Value value) See compare().
boolean	equals (Object other)
boolean	equals (Value value) Compares with the given Value.
boolean	getBoolean () Gets Boolean Value.
DataType	getDataType () Gets the DataType.
double	getDouble () Gets Double Value.

int	<code>getInteger()</code> Gets Integer Value.
long	<code>getLong()</code> Gets Long Value.
long	<code>getOID()</code> Gets OID Value.
String	<code>getString()</code> Gets String Value.
long	<code>getTimestamp()</code> Gets Timestamp Value.
Calendar	<code>getTimestampAsCalendar()</code> Gets the Value as a Calendar instance.
Date	<code>getTimestampAsDate()</code> Gets the Value as a Date instance.
int	<code>hashCode()</code>
boolean	<code>isNull()</code> Gets if this is a NULL Value.
<code>Value</code>	<code>set(Value value)</code> Sets the Value.
<code>Value</code>	<code>setBoolean(boolean value)</code> Sets the Value.
void	<code>setBooleanVoid(boolean value)</code> Sets the Value.
<code>Value</code>	<code>setDouble(double value)</code> Sets the Value.
void	<code>setDoubleVoid(double value)</code> Sets the Value.
<code>Value</code>	<code>setInteger(int value)</code> Sets the Value.
void	<code>setIntegerVoid(int value)</code> Sets the Value.
<code>Value</code>	<code>setLong(long value)</code> Sets the Value.
void	<code>setLongVoid(long value)</code> Sets the Value.
<code>Value</code>	<code>setNull()</code> Sets the Value to NULL.
void	<code>setNullVoid()</code> Sets the Value to NULL.

Value	setOID (long value) Sets the Value.
void	setOIDVoid (long value) Sets the OID Value.
Value	setString (String value) Sets the Value.
void	setStringVoid (String value) Sets the Value.
Value	setTimestamp (Calendar value) Sets the Value.
Value	setTimestamp (Date value) Sets the Value.
Value	setTimestamp (int year, int month, int day, int hour, int minutes, int seconds, int millisec) Sets the Value.
void	setTimestampVoid (int year, int month, int day, int hour, int minutes, int seconds, int millisecs) Sets the Value.
void	setTimestampVoid (long value) Sets the Value.
void	setVoid (Value value) Sets the Value.
String	toString () Gets a String representation of the Value.
String	toString (String str) Gets a string representation of the Value.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Fields

MaxLengthString

```
public static int MaxLengthString
```

Maximum number of characters allowed for a String.

Constructors

(continued from last page)

Value

```
public Value(Value value)
```

Copy constructor.

Parameters:

value - [in] Value to be copied.

Value

```
public Value()
```

Creates a new instance.

It creates a NULL Value.

Methods

setLongVoid

```
public void setLongVoid(long value)
```

Sets the Value.

Parameters:

value - [in] New Long value.

setTimestamp

```
public Value setTimestamp(Date value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

getTimestampAsCalendar

```
public Calendar getTimestampAsCalendar()
```

Gets the Value as a Calendar instance.

Returns:

The returning Calendar instance.

(continued from last page)

equals

```
public boolean equals(Object other)
```

Parameters:

other - null

getOID

```
public long getOID()
```

Gets OID Value.

This must be an non-NULL OID Value.

Returns:

The OID Value.

set

```
public Value set(Value value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

setString

```
public Value setString(String value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

hashCode

```
public int hashCode()
```

(continued from last page)

setBoolean

```
public Value setBoolean(boolean value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

getBoolean

```
public boolean getBoolean()
```

Gets Boolean Value.

This must be a non-NULL Boolean Value.

Returns:

The Boolean Value.

setDouble

```
public Value setDouble(double value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

setNullVoid

```
public void setNullVoid()
```

Sets the Value to NULL.

setTimestampVoid

```
public void setTimestampVoid(long value)
```

Sets the Value.

Parameters:

value - [in] New Timestamp value.

(continued from last page)

setStringVoid

```
public void setStringVoid(String value)
```

Sets the Value.

Parameters:

value - [in] New String value.

setIntegerVoid

```
public void setIntegerVoid(int value)
```

Sets the Value.

Parameters:

value - [in] New Integer value.

getDataType

```
public DataType getDataType()
```

Gets the DataType.

Value cannot be NULL.

Returns:

The DataType.

setLong

```
public Value setLong(long value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

getTimestampAsDate

```
public Date getTimestampAsDate()
```

Gets the Value as a Date instance.

Returns:

The returning Date instance.

(continued from last page)

getString

```
public String getString()
```

Gets String Value.

This must be a non-NULL String Value.

Returns:

The String Value.

setDoubleVoid

```
public void setDoubleVoid(double value)
```

Sets the Value.

Parameters:

value - [in] New Double value.

equals

```
public boolean equals(Value 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.

isNull

```
public boolean isNull()
```

Gets if this is a NULL Value.

Returns:

TRUE if this is a NULL Value, FALSE otherwise.

setVoid

```
public void setVoid(Value value)
```

Sets the Value.

Parameters:

value - [in] New value.

(continued from last page)

setTimestamp

```
public Value setTimestamp(int year,  
    int month,  
    int day,  
    int hour,  
    int minutes,  
    int seconds,  
    int millisec)
```

Sets the Value.

Parameters:

year - The year (≥ 1970).
month - The month ([1..12]).
day - The day of the month ([1..31]).
hour - The hour ([0..23]).
minutes - The minutes ([0..59]).
seconds - The seconds ([0..59]).
millisec - The milliseconds ([0..999]).

Returns:

The calling instance.

setTimestampVoid

```
public void setTimestampVoid(int year,  
    int month,  
    int day,  
    int hour,  
    int minutes,  
    int seconds,  
    int 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]).

setOIDVoid

```
public void setOIDVoid(long value)
```

Sets the OID Value.

Parameters:

value - [in] New OID value.

(continued from last page)

getDouble

```
public double getDouble()
```

Gets Double Value.

This must be a non-NULL Double Value.

Returns:

The Double Value.

toString

```
public String toString(String str)
```

Gets a string representation of the Value.

Parameters:

`str` - String to be used. It is cleared and set with the string representation of the Value.

Returns:

The given string which has been updated.

setNull

```
public Value setNull()
```

Sets the Value to NULL.

Returns:

The calling instance.

setOID

```
public Value setOID(long value)
```

Sets the Value.

Parameters:

`value` - New value.

Returns:

The calling instance.

compare

```
public int compare(Value 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.

(continued from last page)

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.

getLong

```
public long getLong()
```

Gets Long Value.

This must be a non-NULL Long Value.

Returns:

The Long Value.

compareTo

```
public int compareTo(Object value)
```

See compare().

This just works if the given object is a Value instance.

Parameters:

value - null

setBooleanVoid

```
public void setBooleanVoid(boolean value)
```

Sets the Value.

Parameters:

value - [in] New Boolean value.

toString

```
public String toString()
```

Gets a String representation of the Value.

getTimestamp

```
public long getTimestamp()
```

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

Returns:

The Timestamp Value.

setTimestamp

```
public Value setTimestamp(Calendar value)
```

(continued from last page)

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

setInteger

```
public Value setInteger(int value)
```

Sets the Value.

Parameters:

value - New value.

Returns:

The calling instance.

getInteger

```
public int getInteger()
```

Gets Integer Value.

This must be a non-NULL Integer Value.

Returns:

The Integer Value.

compareTo

```
public int compareTo(Value value)
```

See compare().

Parameters:

value - null

com.sparsity.sparksee.gdb

Class ValueList

java.lang.Object

└--com.sparsity.sparksee.gdb.ValueList

public class **ValueList**
extends Object

Value list.

It stores a Value list.

Use ValueListIterator to access all elements into this collection.

Author:

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

Constructor Summary

public	ValueList() Constructor.
--------	---

Method Summary

void	add(Value value) Adds a value to the end of the list.
void	clear() Clears the list.
int	count() Number of elements in the list.
Value	get(int index) Returns the Value at the specified position in the list.
ValueListIterator	iterator() Gets a new ValueListIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

ValueList

public **ValueList()**

Constructor.

This creates an empty list.

Methods

get

```
public Value get(int index)
```

Returns the Value at the specified position in the list.

Parameters:

index - [in] Index of the element to return, starting at 0.

clear

```
public void clear()
```

Clears the list.

add

```
public void add(Value value)
```

Adds a value to the end of the list.

Parameters:

value - [in] The value to add

iterator

```
public ValueListIterator iterator()
```

Gets a new ValueListIterator.

Returns:

ValueListIterator instance.

count

```
public int count()
```

Number of elements in the list.

Returns:

Number of elements in the list.

com.sparsity.sparksee.gdb

Class ValueListIterator

java.lang.Object

└--com.sparsity.sparksee.gdb.ValueListIterator

public class **ValueListIterator**
extends Object

ValueList iterator class.

Iterator to traverse all the values into a ValueList instance.

Author:

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

Method Summary

boolean	hasNext () Gets if there are more elements.
Value	next () Moves to the next element.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

hasNext

public boolean **hasNext**()

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

next

public [Value](#) **next**()

Moves to the next element.

Returns:

The next element.

com.sparsity.sparksee.gdb Class Values

java.lang.Object

└─com.sparsity.sparksee.gdb.Values

All Implemented Interfaces:

Closeable, Iterable

public class **Values**
extends Object
implements Iterable, Closeable

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>

Method Summary

void	close() Closes the Values instance.
long	count() Gets the number of elements into the collection.
boolean	isClosed() Gets if Values instance has been closed or not.
ValuesIterator	iterator() See iterator().
ValuesIterator	iterator(Order order) Gets a ValuesIterator.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator

Methods inherited from interface java.io.Closeable

close

(continued from last page)

Methods

iterator

```
public ValuesIterator iterator()
```

See iterator().

Creates an Ascendent iterator.

count

```
public long count()
```

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

iterator

```
public ValuesIterator iterator(Order order)
```

Gets a ValuesIterator.

Parameters:

order - [in] Ascending or descending order.

Returns:

ValuesIterator instance.

isClosed

```
public boolean isClosed()
```

Gets if Values instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the Values instance.

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

com.sparsity.sparksee.gdb Class ValuesIterator

java.lang.Object

└─com.sparsity.sparksee.gdb.ValuesIterator

All Implemented Interfaces:

Iterator, Closeable

public class **ValuesIterator**
extends Object
implements Closeable, Iterator

Values iterator class.

It allows for traversing all the elements into a Values instance.

Author:

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

Method Summary

void	close() Closes the ValuesIterator instance.
boolean	hasNext() Gets if there are more elements to traverse.
boolean	isClosed() Gets if ValuesIterator instance has been closed or not.
Value	next() Gets the next element to traverse.
void	remove() Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

(continued from last page)

hasNext

```
public boolean hasNext()
```

Gets if there are more elements to traverse.

Returns:

TRUE if there are more elements to traverse, FALSE otherwise.

remove

```
public void remove()
```

Operation not supported.

next

```
public Value next()
```

Gets the next element to traverse.

Returns:

The next element.

isClosed

```
public boolean isClosed()
```

Gets if ValuesIterator instance has been closed or not.

Returns:

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

See Also:

[close\(\)](#)

close

```
public void close()
```

Closes the ValuesIterator instance.

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

Package

com.sparsity.sparksee.io

com.sparsity.sparksee.io Class CSVReader

```
java.lang.Object
|
+-com.sparsity.sparksee.io.RowReader
|
+-com.sparsity.sparksee.io.CSVReader
```

public class **CSVReader**
extends [RowReader](#)

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>

Constructor Summary

public	CSVReader() Constructs CSVReader.
--------	--

Method Summary

void	close() Closes the reader.
int	getRow() The row number for the current row.

void	<code>open(String filePath)</code> Opens the source file path.
boolean	<code>read(StringList row)</code> Reads the next row as a string array.
boolean	<code>reset()</code> Moves the reader to the beginning.
void	<code>setLocale(String localeStr)</code> Sets the locale that will be used to read the file.
void	<code>setMultilines(int numExtralines)</code> Allows the use of fields with more than one line.
void	<code>setNumLines(int numLines)</code> Used to limit the number of lines that will be read.
void	<code>setQuotes(String quotes)</code> Sets the character used to quote fields.
void	<code>setSeparator(String sep)</code> Sets the character used to separate fields in the file.
void	<code>setSingleLine()</code> Only allows single line fields.
void	<code>setStartLine(int startLine)</code> Sets the number of lines to be skipped from the beginning.

Methods inherited from class [com.sparsity.sparksee.io.RowReader](#)

[`close`](#), [`getRow`](#), [`read`](#), [`reset`](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

CSVReader

```
public CSVReader()
```

Constructs CSVReader.

Methods

reset

```
public boolean reset()
    throws IOException
```

(continued from last page)

Moves the reader to the beginning.

Restarts the reader.

Returns:

true if the reader can be restarted, false otherwise.

Throws:

java.io.IOException - If bad things happen during the restart.

close

```
public void close()  
    throws IOException
```

Closes the reader.

setNumLines

```
public void setNumLines(int numLines)
```

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

Parameters:

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

setMultilines

```
public void setMultilines(int numExtralines)
```

Allows the use of fields with more than one line.

Parameters:

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

setSeparator

```
public void setSeparator(String sep)  
    throws RuntimeException
```

Sets the character used to separate fields in the file.

Parameters:

sep - [in] Separator character.

Throws:

java.lang.RuntimeException - null

setQuotes

```
public void setQuotes(String quotes)  
    throws RuntimeException
```

(continued from last page)

Sets the character used to quote fields.

Parameters:

quotes - [in] Quote character.

Throws:

java.lang.RuntimeException - null

open

```
public void open(String filePath)
    throws IOException
```

Opens the source file path.

File can be optionally compressed in GZIP format.

Parameters:

filePath - [in] CSV file path.

Throws:

java.io.IOException - If bad things happen opening the file.

setSingleLine

```
public void setSingleLine()
```

Only allows single line fields.

read

```
public boolean read(StringList row)
    throws IOException
```

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.

Throws:

java.io.IOException - If bad things happen during the read.

setStartLine

```
public void setStartLine(int startLine)
```

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

(continued from last page)

Parameters:

`startLine` - [in] The line number to skip for start reading

setLocale

```
public void setLocale(String localeStr)
```

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

Parameters:

`localeStr` - [in] The locale string for the file encoding.

getRow

```
public int getRow()  
    throws IOException
```

The row number for the current row.

Returns:

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

Throws:

`java.io.IOException` - If it fails.

com.sparsity.sparksee.io Class CSVWriter

```
java.lang.Object
|
+-com.sparsity.sparksee.io.RowWriter
|
+-com.sparsity.sparksee.io.CSVWriter
```

public class **CSVWriter**
extends [RowWriter](#)

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>

Constructor Summary

public	CSVWriter() Creates a new instance.
--------	--

Method Summary

void	close() Closes the writer.
void	open(String f) Opens the output file path.
void	setAutoQuotes(boolean autoquotes) Sets on/off the automatic quote mode.
void	setForcedQuotes(BooleanList forcequotes) Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.
void	setLocale(String localeStr) Sets the locale that will be used to write the file.
void	setQuotes(String quotes) Sets the character used to quote fields.
void	setSeparator(String sep) Sets the character used to separate fields in the file.

void	<code>write(StringList row)</code> Writes the next row.
------	--

Methods inherited from class [com.sparsity.sparksee.io.RowWriter](#)

[close](#), [write](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

CSVWriter

```
public CSVWriter()
```

Creates a new instance.

Methods

setAutoQuotes

```
public void setAutoQuotes(boolean 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.

setSeparator

```
public void setSeparator(String sep)
    throws RuntimeException
```

Sets the character used to separate fields in the file.

Parameters:

`sep` - [in] Separator character.

Throws:

`java.lang.RuntimeException` - null

setQuotes

```
public void setQuotes(String quotes)
    throws RuntimeException
```

(continued from last page)

Sets the character used to quote fields.

Parameters:

quotes - [in] Quote character.

Throws:

java.lang.RuntimeException - null

setLocale

```
public void setLocale(String localeStr)
```

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

Parameters:

localeStr - [in] The locale string for the file encoding.

write

```
public void write(StringList row)  
    throws IOException,  
           RuntimeException
```

Writes the next row.

Parameters:

row - [in] Row of data.

Throws:

java.io.IOException - If bad things happen during the write.
java.lang.RuntimeException - null

setForcedQuotes

```
public void setForcedQuotes(BooleanList 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.

close

```
public void close()  
    throws IOException,  
           RuntimeException
```

Closes the writer.

(continued from last page)

open

```
public void open(String f)  
    throws IOException
```

Opens the output file path.

Parameters:

f - [in] Output file path.

Throws:

java.io.IOException - If bad things happen opening the file.

com.sparsity.sparksee.io Class EdgeTypeExporter

java.lang.Object

```

  |
+-com.sparsity.sparksee.io.TypeExporter
  |
+-com.sparsity.sparksee.io.EdgeTypeExporter

```

public class **EdgeTypeExporter**
extends [TypeExporter](#)

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>

Constructor Summary

public	EdgeTypeExporter (RowWriter rowWriter, Graph graph, int type, AttributeList attrs, int hPos, int tPos, int hAttr, int tAttr) Creates a new instance.
public	EdgeTypeExporter () Creates a new instance.

Method Summary

void	register (TypeExporterListener tel) Registers a new listener.
void	run () See the TypeExporter class Run method.
void	setAttributes (AttributeList attrs) Sets the list of Attributes.
void	setFrequency (int freq) Sets the frequency of listener notification.
void	setGraph (Graph graph) Sets the graph that will be exported.
void	setHeadAttribute (int attr) Sets the attribute that will be used to get the value to be dumped for the head of the edge.
void	setHeader (boolean header) Sets the presence of a header row.
void	setHeadPosition (int pos) Sets the position (index column) of the head attribute in the exported data.

void	setRowWriter (RowWriter rw) Sets the output data destination.
void	setTailAttribute (int attr) Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
void	setTailPosition (int pos) Sets the position (index column) of the tail attribute in the exported data.
void	setType (int type) Sets the type to be exported.

Methods inherited from class [com.sparsity.sparksee.io.TypeExporter](#)

[register](#), [run](#), [setAttributes](#), [setFrequency](#), [setGraph](#), [setHeader](#), [setRowWriter](#), [setType](#)

Methods inherited from class [java.lang.Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Constructors

EdgeTypeExporter

```
public EdgeTypeExporter(RowWriter rowWriter,
                        Graph graph,
                        int type,
                        AttributeList attrs,
                        int hPos,
                        int tPos,
                        int hAttr,
                        int tAttr)
```

Creates a new instance.

Parameters:

[rowWriter](#) - [in] Output RowWriter.
[graph](#) - [in] Graph.
[type](#) - [in] Type identifier.
[attrs](#) - [in] Attribute identifiers to be exported.
[hPos](#) - [in] The position (index column) for the head value.
[tPos](#) - [in] The position (index column) for the tail value.
[hAttr](#) - [in] The attribute identifier to get the value to be dumped for the head.
[tAttr](#) - [in] The attribute identifier to get the value to be dumped for the tail.

EdgeTypeExporter

```
public EdgeTypeExporter()
```

Creates a new instance.

(continued from last page)

Methods

setTailAttribute

```
public void setTailAttribute(int 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

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

Parameters:

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

setHeadAttribute

```
public void setHeadAttribute(int 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

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

type - [in] Type identifier.

setTailPosition

```
public void setTailPosition(int pos)
```

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

Parameters:

pos - [in] Tail position

(continued from last page)

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

rw - [in] Input RowWriter.

register

```
public void register(TypeExporterListener tel)
```

Registers a new listener.

Parameters:

tel - [in] TypeExporterListener to be registered.

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

See the TypeExporter class Run method.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph that will be exported.

Parameters:

graph - [in] Graph.

setHeader

```
public void setHeader(boolean header)
```

Sets the presence of a header row.

Parameters:

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

setHeadPosition

```
public void setHeadPosition(int pos)
```


(continued from last page)

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

Parameters:

pos - [in] Head position

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be exported

com.sparsity.sparksee.io Class EdgeTypeLoader

```
java.lang.Object
├── com.sparsity.sparksee.io.TypeLoader
│   └── com.sparsity.sparksee.io.EdgeTypeLoader
```

public class **EdgeTypeLoader**
extends [TypeLoader](#)

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>

Constructor Summary

public	EdgeTypeLoader (RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr) Creates a new instance.
public	EdgeTypeLoader () Creates a new instance.

Method Summary

void	register (TypeLoaderListener tel) Registers a new listener.
void	run () See the TypeLoader class Run method.
void	runNPhases (int partitions) See the TypeLoader class RunNPhases method.
void	runTwoPhases () See the TypeLoader class RunTwoPhases method.
void	setAttributePositions (Int32List attrsPos) Sets the list of attribute positions.
void	setAttributes (AttributeList attrs) Sets the list of Attributes.
void	setFrequency (int freq) Sets the frequency of listener notification.
void	setGraph (Graph graph) Sets the graph where the data will be loaded.

void	<code>setHeadAttribute</code> (int attr) Sets the attribute that will be used to find the head of the edge.
void	<code>setHeadPosition</code> (int pos) Sets the position of the head attribute in the source data.
void	<code>setLocale</code> (String localeStr) Sets the locale that will be used to read the data.
void	<code>setLogError</code> (String path) Sets a log error file.
void	<code>setLogOff</code> () Turns off all the error reporting.
void	<code>setRowReader</code> (<code>RowReader</code> rr) Sets the input data source.
void	<code>setTailAttribute</code> (int attr) Sets the attribute that will be used to find the tail of the edge.
void	<code>setTailPosition</code> (int pos) Sets the position of the tail attribute in the source data.
void	<code>setTimestampFormat</code> (String timestampFormat) Sets a specific timestamp format.
void	<code>setType</code> (int type) Sets the type to be loaded.

Methods inherited from class [com.sparsity.sparksee.io.TypeLoader](#)

[register](#), [run](#), [runNPhases](#), [runTwoPhases](#), [setAttributePositions](#), [setAttributes](#), [setFrequency](#), [setGraph](#), [setLocale](#), [setLogError](#), [setLogOff](#), [setRowReader](#), [setTimestampFormat](#), [setType](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

EdgeTypeLoader

```
public EdgeTypeLoader(RowReader rowReader,
                     Graph graph,
                     int type,
                     AttributeList attrs,
                     Int32List attrsPos,
                     int hPos,
                     int tPos,
                     int hAttr,
                     int tAttr)
```

Creates a new instance.

(continued from last page)

Parameters:

rowReader - [in] Input RowReader.
graph - [in] Graph.
type - [in] Type identifier.
attrs - [in] Attribute identifiers to be loaded.
attrsPos - [in] Attribute positions (column index ≥ 0). to all listeners.
hPos - [in] The position (index column) for the head value.
tPos - [in] The position (index column) for the tail value.
hAttr - [in] The attribute identifier for the head.
tAttr - [in] The attribute identifier for the tail.

EdgeTypeLoader

```
public EdgeTypeLoader( )
```

Creates a new instance.

Methods

setTailAttribute

```
public void setTailAttribute(int 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

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

Parameters:

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

setLogOff

```
public void setLogOff( )
```

Turns 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.

setLogError

```
public void setLogError(String path)  
    throws IOException
```

(continued from last page)

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.

Throws:

java.io.IOException - If bad things happen opening the file.

setHeadAttribute

```
public void setHeadAttribute(int 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

setType

```
public void setType(int type)
```

Sets the type to be loaded.

Parameters:

type - [in] Type identifier.

runTwoPhases

```
public void runTwoPhases()  
    throws IOException,  
           RuntimeException
```

See the TypeLoader class RunTwoPhases method.

setTailPosition

```
public void setTailPosition(int 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

setRowReader

```
public void setRowReader(RowReader rr)
```

Sets the input data source.

(continued from last page)

Parameters:`rr` - [in] Input RowReader.

setAttributePositions

```
public void setAttributePositions(Int32List attrsPos)
```

Sets the list of attribute positions.

Parameters:`attrsPos` - [in] Attribute positions (column index >=0).

register

```
public void register(TypeLoaderListener tel)
```

Registers a new listener.

Parameters:`tel` - TypeLoaderListener to be registered.

setLocale

```
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

Parameters:`localeStr` - [in] The locale string for the read data. See CSVReader.

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

See the TypeLoader class Run method.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph where the data will be loaded.

Parameters:`graph` - [in] Graph.

(continued from last page)

runNPhases

```
public void runNPhases(int partitions)
    throws IOException,
        RuntimeException
```

See the TypeLoader class RunNPhases method.

Parameters:

partitions - null

Throws:

java.io.IOException - null

java.lang.RuntimeException - null

setTimestampFormat

```
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

Parameters:

timestampFormat - [in] A string with the timestamp format definition.

setHeadPosition

```
public void setHeadPosition(int 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

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be loaded

com.sparsity.sparksee.io Class NodeTypeExporter

```

java.lang.Object
|
+-com.sparsity.sparksee.io.TypeExporter
   |
   +-com.sparsity.sparksee.io.NodeTypeExporter

```

public class **NodeTypeExporter**
extends [TypeExporter](#)

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>

Constructor Summary

public	NodeTypeExporter (RowWriter rowWriter, Graph graph, int type, AttributeList attrs) Creates a new instance.
public	NodeTypeExporter () Creates a new instance.

Method Summary

void	register (TypeExporterListener tel) Registers a new listener.
void	run () See the TypeExporter class Run method.
void	setAttributes (AttributeList attrs) Sets the list of Attributes.
void	setFrequency (int freq) Sets the frequency of listener notification.
void	setGraph (Graph graph) Sets the graph that will be exported.
void	setHeader (boolean header) Sets the presence of a header row.
void	setRowWriter (RowWriter rw) Sets the output data destination.
void	setType (int type) Sets the type to be exported.

Methods inherited from class [com.sparsity.sparksee.io.TypeExporter](#)

[register](#), [run](#), [setAttributes](#), [setFrequency](#), [setGraph](#), [setHeader](#), [setRowWriter](#), [setType](#)

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

NodeTypeExporter

```
public NodeTypeExporter(RowWriter rowWriter,  
                        Graph graph,  
                        int type,  
                        AttributeList attrs)
```

Creates a new instance.

Parameters:

`rowWriter` - [in] Output RowWriter.
`graph` - [in] Graph.
`type` - [in] Type identifier.
`attrs` - [in] Attribute identifiers to be exported.

NodeTypeExporter

```
public NodeTypeExporter()
```

Creates a new instance.

Methods

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

`rw` - [in] Input RowWriter.

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

(continued from last page)

Parameters:

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

register

```
public void register(TypeExporterListener tel)
```

Registers a new listener.

Parameters:

tel - [in] TypeExporterListener to be registered.

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

See the TypeExporter class Run method.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph that will be exported.

Parameters:

graph - [in] Graph.

setHeader

```
public void setHeader(boolean header)
```

Sets the presence of a header row.

Parameters:

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

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

type - [in] Type identifier.

(continued from last page)

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

`attrs` - [in] Attribute identifiers to be exported

com.sparsity.sparksee.io Class NodeTypeLoader

```

java.lang.Object
  |
  +--com.sparsity.sparksee.io.TypeLoader
        |
        +--com.sparsity.sparksee.io.NodeTypeLoader
  
```

public class **NodeTypeLoader**
extends [TypeLoader](#)

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>

Constructor Summary

public	NodeTypeLoader (RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos) Creates a new instance.
public	NodeTypeLoader () Creates a new instance.

Method Summary

void	register (TypeLoaderListener tel) Registers a new listener.
void	run () See the TypeLoader class Run method.
void	runNPhases (int partitions) See the TypeLoader class RunNPhases method.
void	runTwoPhases () See the TypeLoader class RunTwoPhases method.
void	setAttributePositions (Int32List attrsPos) Sets the list of attribute positions.
void	setAttributes (AttributeList attrs) Sets the list of Attributes.
void	setFrequency (int freq) Sets the frequency of listener notification.
void	setGraph (Graph graph) Sets the graph where the data will be loaded.

void	setLocale (String localeStr) Sets the locale that will be used to read the data.
void	setLogError (String path) Sets a log error file.
void	setLogOff () Turns off all the error reporting.
void	setRowReader (RowReader rr) Sets the input data source.
void	setTimestampFormat (String timestampFormat) Sets a specific timestamp format.
void	setType (int type) Sets the type to be loaded.

Methods inherited from class [com.sparsity.sparksee.io.TypeLoader](#)

[register](#), [run](#), [runNPhases](#), [runTwoPhases](#), [setAttributePositions](#), [setAttributes](#), [setFrequency](#), [setGraph](#), [setLocale](#), [setLogError](#), [setLogOff](#), [setRowReader](#), [setTimestampFormat](#), [setType](#)

Methods inherited from class java.lang.Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Constructors

NodeTypeLoader

```
public NodeTypeLoader(RowReader rowReader,
                     Graph graph,
                     int type,
                     AttributeList attrs,
                     Int32List attrsPos)
```

Creates a new instance.

Parameters:

[rowReader](#) - [in] Input RowReader.
[graph](#) - [in] Graph.
[type](#) - [in] Type identifier.
[attrs](#) - [in] Attribute identifiers to be loaded.
[attrsPos](#) - [in] Attribute positions (column index >=0).

NodeTypeLoader

```
public NodeTypeLoader()
```

Creates a new instance.

Methods

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

Parameters:

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

setLogOff

```
public void setLogOff()
```

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.

setLogError

```
public void setLogError(String path)  
    throws IOException
```

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.

Throws:

java.io.IOException - If bad things happen opening the file.

setType

```
public void setType(int type)
```

Sets the type to be loaded.

Parameters:

type - [in] Type identifier.

runTwoPhases

```
public void runTwoPhases()  
    throws IOException,  
        RuntimeException
```

See the TypeLoader class RunTwoPhases method.

setRowReader

```
public void setRowReader(RowReader rr)
```

Sets the input data source.

Parameters:

rr - [in] Input RowReader.

setAttributePositions

```
public void setAttributePositions(Int32List attrsPos)
```

Sets the list of attribute positions.

Parameters:

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

register

```
public void register(TypeLoaderListener tel)
```

Registers a new listener.

Parameters:

tel - TypeLoaderListener to be registered.

setLocale

```
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

Parameters:

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

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

See the TypeLoader class Run method.

setGraph

```
public void setGraph(Graph graph)
```

(continued from last page)

Sets the graph where the data will be loaded.

Parameters:

graph - [in] Graph.

runNPhases

```
public void runNPhases(int partitions)
    throws IOException,
        RuntimeException
```

See the TypeLoader class RunNPhases method.

Parameters:

partitions - null

Throws:

java.io.IOException - null

java.lang.RuntimeException - null

setTimestampFormat

```
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

Parameters:

timestampFormat - [in] A string with the timestamp format definition.

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be loaded

com.sparsity.sparksee.io

Class RowReader

java.lang.Object

└─com.sparsity.sparksee.io.RowReader

Direct Known Subclasses:

[CSVReader](#)

```
public class RowReader
extends Object
```

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>

Method Summary

void	close() Closes the reader.
int	getRow() The row number for the current row.
boolean	read(StringList row) Reads the next row as a string array.
boolean	reset() Moves the reader to the beginning.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

close

```
public void close()
throws IOException
```

Closes the reader.

getRow

```
public int getRow()  
    throws IOException
```

The row number for the current row.

Returns:

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

Throws:

`java.io.IOException` - If it fails.

reset

```
public boolean reset()  
    throws IOException
```

Moves the reader to the beginning.

Restarts the reader.

Returns:

true if the reader can be restarted, false otherwise.

Throws:

`java.io.IOException` - If bad things happen during the restart.

read

```
public boolean read(StringList row)  
    throws IOException
```

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.

Throws:

`java.io.IOException` - If bad things happen during the read.

com.sparsity.sparksee.io

Class RowWriter

java.lang.Object

└─com.sparsity.sparksee.io.RowWriter

Direct Known Subclasses:

[CSVWriter](#)

public class **RowWriter**
extends Object

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>

Method Summary

void	close() Closes the writer.
void	write(StringList row) Writes the next row.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

write

public void **write**([StringList](#) row)
throws IOException,
RuntimeException

Writes the next row.

Parameters:

row - [in] Row of data.

Throws:

java.io.IOException - If bad things happen during the write.

(continued from last page)

`java.lang.RuntimeException - null`

close

```
public void close()  
    throws IOException,  
           RuntimeException
```

Closes the writer.

com.sparsity.sparksee.io Class TypeExporter

java.lang.Object

└─com.sparsity.sparksee.io.TypeExporter

Direct Known Subclasses:

[NodeTypeExporter](#), [EdgeTypeExporter](#)

```
public class TypeExporter
extends Object
```

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>

Method Summary

void	register (TypeExporterListener tel)	Registers a new listener.
void	run ()	Runs export process.
void	setAttributes (AttributeList attrs)	Sets the list of Attributes.
void	setFrequency (int freq)	Sets the frequency of listener notification.
void	setGraph (Graph graph)	Sets the graph that will be exported.
void	setHeader (boolean header)	Sets the presence of a header row.
void	setRowWriter (RowWriter rw)	Sets the output data destination.
void	setType (int type)	Sets the type to be exported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

rw - [in] Input RowWriter.

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

Parameters:

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

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

Runs export process.

register

```
public void register(TypeExporterListener tel)
```

Registers a new listener.

Parameters:

tel - [in] TypeExporterListener to be registered.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph that will be exported.

Parameters:

graph - [in] Graph.

(continued from last page)

setHeader

```
public void setHeader(boolean header)
```

Sets the presence of a header row.

Parameters:

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

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

type - [in] Type identifier.

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be exported

com.sparsity.sparksee.io Class TypeExporterEvent

java.lang.Object

└─com.sparsity.sparksee.io.TypeExporterEvent

public class **TypeExporterEvent**
extends Object

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>

Method Summary

long	getCount() Gets the current number of objects exported.
long	getTotal() Gets the total number of objects exported.
int	getTypeId() Gets the type identifier.
boolean	isLast() Gets if this is the last event or not.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getCount

public long **getCount()**

Gets the current number of objects exported.

Returns:

The current number of objects exported.

isLast

public boolean **isLast()**

(continued from last page)

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

getTypeId

```
public int getTypeId()
```

Gets the type identifier.

Returns:

The type identifier.

getTotal

```
public long getTotal()
```

Gets the total number of objects exported.

Returns:

The total number of objects exported.

com.sparsity.sparksee.io Class TypeExporterListener

java.lang.Object

└─com.sparsity.sparksee.io.TypeExporterListener

public class **TypeExporterListener**
extends Object

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>

Method Summary

void	notifyEvent (TypeExporterEvent tee) Method to be notified from a TypeExporter.
------	--

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

notifyEvent

public void **notifyEvent**([TypeExporterEvent](#) tee)

Method to be notified from a TypeExporter.

Parameters:

tee - [in] Notified event.

com.sparsity.sparksee.io Class TypeLoader

java.lang.Object

└─com.sparsity.sparksee.io.TypeLoader

Direct Known Subclasses:

[NodeTypeLoader](#), [EdgeTypeLoader](#)

public class **TypeLoader**
extends Object

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>

Method Summary

void	register (TypeLoaderListener tel) Registers a new listener.
void	run () Run the loader.
void	runNPhases (int partitions) Run the loader for N phases loading.
void	runTwoPhases () Run the loader for two phases loading.
void	setAttributePositions (Int32List attrsPos) Sets the list of attribute positions.
void	setAttributes (AttributeList attrs) Sets the list of Attributes.
void	setFrequency (int freq) Sets the frequency of listener notification.
void	setGraph (Graph graph) Sets the graph where the data will be loaded.
void	setLocale (String localeStr) Sets the locale that will be used to read the data.
void	setLogError (String path) Sets a log error file.

void	setLogOff() Truns off all the error reporting.
void	setRowReader (RowReader rr) Sets the input data source.
void	setTimestampFormat (String timestampFormat) Sets a specific timestamp format.
void	setType (int type) Sets the type to be loaded.

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods

runTwoPhases

```
public void runTwoPhases()
    throws IOException,
           RuntimeException
```

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.

runNPhases

```
public void runNPhases(int partitions)
    throws IOException,
           RuntimeException
```

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.

Throws:

`java.io.IOException` - null

`java.lang.RuntimeException` - null

setFrequency

```
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

(continued from last page)

Parameters:

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

setLogOff

```
public void setLogOff()
```

Turns 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.

run

```
public void run()  
    throws IOException,  
           RuntimeException
```

Run the loader.

setLogError

```
public void setLogError(String path)  
    throws IOException
```

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.

Throws:

java.io.IOException - If bad things happen opening the file.

setType

```
public void setType(int type)
```

Sets the type to be loaded.

Parameters:

type - [in] Type identifier.

setRowReader

```
public void setRowReader(RowReader rr)
```

Sets the input data source.

Parameters:

rr - [in] Input RowReader.

register

```
public void register(TypeLoaderListener tel)
```

Registers a new listener.

Parameters:

tel - TypeLoaderListener to be registered.

setAttributePositions

```
public void setAttributePositions(Int32List attrsPos)
```

Sets the list of attribute positions.

Parameters:

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

setLocale

```
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

Parameters:

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

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph where the data will be loaded.

Parameters:

graph - [in] Graph.

setTimestampFormat

```
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

Parameters:

timestampFormat - [in] A string with the timestamp format definition.

setAttributes

```
public void setAttributes(AttributeList attrs)
```

(continued from last page)

Sets the list of Attributes.

Parameters:

`attrs` - [in] Attribute identifiers to be loaded

com.sparsity.sparksee.io Class TypeLoaderEvent

java.lang.Object

└─com.sparsity.sparksee.io.TypeLoaderEvent

public class **TypeLoaderEvent**
extends Object

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>

Method Summary

long	getCount() Gets the current number of objects created.
int	getPartition() Gets the current partition.
int	getPhase() Gets the current phase.
int	getTotalPartitions() Gets the total number of partitions.
int	getTotalPartitionSteps() Gets the total number of steps in the current partition.
int	getTotalPhases() Gets the total number of phases.
int	getTypeId() Gets the type identifier.
boolean	isLast() Gets if this is the last event or not.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getTotalPhases

public int **getTotalPhases()**

(continued from last page)

Gets the total number of phases.

Returns:

The total number of phases.

getCount

```
public long getCount()
```

Gets the current number of objects created.

Returns:

The current number of objects created.

getTotalPartitionSteps

```
public int getTotalPartitionSteps()
```

Gets the total number of steps in the current partition.

Returns:

The total number steps in the current partition.

isLast

```
public boolean isLast()
```

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

getPartition

```
public int getPartition()
```

Gets the current partition.

Returns:

The current partition.

getTypeId

```
public int getTypeId()
```

Gets the type identifier.

Returns:

(continued from last page)

The type identifier.

getTotalPartitions

```
public int getTotalPartitions()
```

Gets the total number of partitions.

Returns:

The total number of partitions.

getPhase

```
public int getPhase()
```

Gets the current phase.

Returns:

The current phase.

com.sparsity.sparksee.io Class TypeLoaderListener

```
java.lang.Object
  |
  +--com.sparsity.sparksee.io.TypeLoaderListener
```

```
public class TypeLoaderListener
    extends Object
```

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>

Method Summary

void	notifyEvent (TypeLoaderEvent ev) Method to receive events from a Loader.
------	--

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

notifyEvent

```
public void notifyEvent(TypeLoaderEvent ev)
```

Method to receive events from a Loader.

Parameters:

ev - Loader.LoaderEvent with information from a running Loader.

Package

com.sparsity.sparksee.script

com.sparsity.sparksee.script Class ScriptParser

java.lang.Object

└─com.sparsity.sparksee.script.ScriptParser

public class **ScriptParser**
extends Object

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>

Constructor Summary

public	ScriptParser() Constructor.
--------	--

Method Summary

static void	generateSchemaScript (String path, Database db) Writes an script with the schema definition for the given database.
static void	main () Executes ScriptParser for the given file path.
boolean	parse (String path, boolean execute, String localeStr) Parses the given input file.
void	setErrorLog (String path) Sets the error log.
void	setOutputLog (String path) Sets the output log.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,

Constructors

(continued from last page)

ScriptParser

```
public ScriptParser()
```

Constructor.

Methods

parse

```
public boolean parse(String path,  
                    boolean execute,  
                    String localeStr)  
    throws IOException
```

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.
localeStr - [in] The locale string for reading the input file. See CSVReader.

Returns:

TRUE if ok, FALSE in case of error.

Throws:

java.io.IOException - If bad things happen opening the file.

setErrorLog

```
public void setErrorLog(String path)  
    throws IOException
```

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters:

path - [in] Path of the error log.

Throws:

java.io.IOException - If bad things happen opening the file.

generateSchemaScript

```
public static void generateSchemaScript(String path,  
                                         Database db)  
    throws IOException
```

Writes an script with the schema definition for the given database.

Parameters:

path - [in] Filename of the script to be written.

(continued from last page)

db - [in] Database.

Throws:

java.io.IOException - If bad things happen opening or writing the file.

setOutputLog

```
public void setOutputLog(String path)
    throws IOException
```

Sets the output log.

If not set, output log corresponds to standard output.

Parameters:

path - [in] Path of the output log.

Throws:

java.io.IOException - If bad things happen opening the file.

main

```
public static void main()
```

Executes ScriptParser for the given file path.

One argument is required, a file path which contains the script to be parsed.

A second argument may be given, a boolean to set if the script must be executed or just parsed. If not given, the script will be executed.

Index

A

add 80, 88, 147, 167, 170, 179, 225, 236, 253
addAll 171
addAllEdgeTypes 5, 18, 24, 28, 32, 37, 42, 46, 51, 54, 59, 63, 65, 70
addAllNodeTypes 5, 8, 14, 19, 24, 27, 32, 37, 43, 45, 49, 54, 57, 61, 65, 70
addEdgeType 5, 17, 24, 27, 32, 37, 43, 45, 50, 54, 58, 62, 65, 69
addNodeType 4, 8, 14, 18, 24, 27, 31, 36, 41, 46, 50, 53, 57, 61, 66, 70
addWeightedEdgeType 42
Any 117
any 165
Ascendent 183
asDirected 114
AttributeList 79

B

backup 134
Basic 76
begin 202
beginUpdate 201
Between 94
Boolean 103
BooleanList 88
Box 160

C

checkOnlyExistence 37
clear 79, 88, 147, 168, 180, 198, 224, 236, 253
close 9, 12, 15, 19, 22, 29, 54, 98, 168, 175, 202, 205, 230, 256, 258, 262, 267, 289, 292
combineDifference 171
combineIntersection 168
combineUnion 166
commit 201
CommunitiesSCD 4
compare 249
compareTo 250, 251

compute 17, 18
Config 152
ConnectedComponents 11
contains 165, 168
containsAll 166
Context 17
copy 167, 172
count 79, 89, 148, 168, 180, 198, 224, 237, 253, 256
countEdges 130
countNodes 135
create 204
CSVReader 261
CSVWriter 266

D

Debug 153
DefaultExport 106
degree 128
Descendent 184
difference 166
disableRollback 97
DisjointCommunities 21
Double 103
drop 138, 140
dumpData 127
dumpStorage 129

E

Edge 177
EdgeExport 112
edges 132
EdgesType 232
EdgeTypeExporter 270
EdgeTypeLoader 275, 276
enableRollback 97
enableType 106, 120
Equal 93
equals 165, 167, 243, 247
excludeEdges 6, 8, 15, 19, 25, 28, 33, 38, 43, 46, 51, 55, 59, 63, 66, 71
excludeNodes 5, 8, 14, 18, 24, 27, 31, 36, 41, 45, 49, 53, 58, 62, 66, 70
execute 190

exists 31, 35, 41, 171

explode 137, 139

export 141

F

fetch 192

findAttribute 132

findAttributes 135

findEdge 139

findEdgeTypes 131

findNodeTypes 133

findObject 139

findOrCreateEdge 138

findOrCreateObject 133

findType 142

findTypes 127

Fine 152

fixCurrentCacheMaxSize 97

G

generateSchemaScript 310

get 197, 220, 252

getAlias 97

getAreNeighborsIndexed 232

getAttribute 133, 135, 140

getAttributeIntervalCount 128

getAttributes 129

getAttributeStatistics 130

getAttributeText 130

getAvailableMem 187

getAvgLengthString 86

getBoolean 221, 245

getCache 101

getCacheMaxSize 98, 218

getCacheStatisticsEnabled 212

getCacheStatisticsFile 214

getCacheStatisticsSnapshotTime 214

getColor 112, 156

getColorRGB 113, 157

getColumn 195

getColumnDataType 196

getColumnIndex 195

getColumnName 196

getCommunities 6, 25

getCommunity 22

getConnectedComponent 11

getConnectedComponents 15, 47, 51, 66, 71

getCost 33, 37, 42

getCount 11, 21, 74, 296, 305

getCurrentDepth 54, 58, 62

getData 101

getDataType 75, 246

getDistinct 85

getDouble 248

getEdge 106, 110, 119

getEdgeData 136

getEdgePeer 142

getEdgeType 107, 119

getExtentPages 217

getExtentSize 211

getFontSize 113, 156

getGraph 106, 119, 202

getHead 109

getHeight 158

getHighAvailabilityCoordinators 213

getHighAvailabilityEnabled 217

getHighAvailabilityIP 215

getHighAvailabilityMasterHistory 219

getHighAvailabilitySynchronization 210

getId 75, 233

getInteger 221, 251

getIsDirected 234

getIsRestricted 233

getJSON 196

getKind 74

getLabel 114, 144, 158

getLabelColor 114, 158

getLabelColorRGB 113, 157

getLicense 214

getLogFile 211

getLogLevel 215

getLong 250

getMax 85

getMaxLengthString 86

getMean 85

getMedian 85

getMin 84

getMinLengthString 84

getMode 84
getModeCount 86
getName 75, 234
getNode 107, 120
getNodes 12, 21
getNodeType 107, 118
getNull 85
getNumColumns 196
getNumCPUs 188
getNumObjects 233
getObjectType 140, 233
getOID 244
getPartition 305
getPath 97
getPathAsEdges 32, 36, 41
getPathAsNodes 33, 36, 41
getPhase 306
getPoolClusterSize 216
getPoolFrameSize 218
getPoolPersistentMaxSize 212
getPoolPersistentMinSize 215
getPoolTemporaryMaxSize 214
getPoolTemporaryMinSize 218
getRead 101
getRealTime 187
getRecoveryCacheMaxSize 219
getRecoveryCheckpointTime 212
getRecoveryEnabled 217
getRecoveryLogFile 213
getRestrictedFrom 232
getRestrictedTo 233
getRollbackEnabled 218
getRow 264, 290
getSessions 100
getShape 156
getSize 11, 21, 74
getStatistics 99, 185
getString 246
getSystemTime 187
getTail 109
getTemp 101
getTimestamp 250
getTimestampAsCalendar 243
getTimestampAsDate 246
getTimeUnit 221

getTotal 86, 297
getTotalMem 187
getTotalPartitions 306
getTotalPartitionSteps 305
getTotalPhases 304
getType 131
getTypeId 74, 297, 305
getUserTime 187
getValues 141
getVariance 84
getWidth 114, 157
getWrite 100
GlobalType 232
GraphExport 144
GraphML 122
Graphviz 121
GreaterEqual 93
GreaterThan 93

H

hashCode 244
hasNext 53, 57, 61, 81, 90, 150, 175, 181, 199, 226, 238, 254, 257
heads 138

I

indexAttribute 131
Indexed 77
Info 152
Ingoing 116
Int32List 147
Integer 103
intersection 172
InvalidAttribute 73
InvalidOID 164
InvalidType 232
isClosed 8, 12, 15, 19, 22, 28, 55, 98, 170, 175, 201, 205, 256, 258
isEmpty 168
isFit 157
isLast 296, 305
isNull 229, 247
isSessionAttribute 74

iterator 79, 88, 147, 172, 180, 198, 224, 236, 253, 255, 256
iteratorFromElement 166
iteratorFromIndex 169

L

LessEqual 93
LessThan 93
Like 94
LikeNoCase 94
load 221
Long 103

M

main 311
MaxLengthString 242

N

neighbors 129, 136
newAttribute 132, 140
newEdge 141, 142
newEdgeType 138
newNode 130
newNodeType 137
newObjects 202
newQuery 191, 201
newRestrictedEdgeType 143
newSession 98
newSessionAttribute 135, 137
next 55, 58, 62, 82, 91, 150, 175, 182, 195, 199, 227, 239, 254, 258
nextAttribute 82
nextBoolean 91
nextInt32 149
nextObject 174
nextOID 182
nextString 227
nextType 239
Node 176
NodeExport 155
NodesType 232
NodeTypeExporter 281
NodeTypeLoader 285

NotEqual 94
notifyEvent 298, 307

O

Off 152
OID 104
OIDList 179
open 205, 263, 267
Outgoing 117

P

parse 310
PlatformStatistics 186
prepare 107, 120, 193

Q

QueryContext 191

R

read 229, 263, 290
RegExp 94
register 272, 278, 282, 287, 294, 302
release 107, 120
remove 82, 91, 150, 165, 169, 175, 182, 227, 239, 258
removeAll 170
removeAttribute 134
removeType 143
renameAttribute 127
renameType 129, 136
reset 261, 290
restore 204
ResultSetList 197
retainAll 169
rewind 194
rollback 201
Round 161
run 5, 8, 14, 25, 27, 32, 37, 43, 45, 50, 65, 70, 272, 278, 282, 287, 294, 301
runNPhases 278, 288, 300
runTwoPhases 277, 286, 300

S

- sample 172
- ScriptParser 309
- select 127, 131, 132, 133, 142
- set 180, 244
- setAsDirected 113
- setAttribute 136
- setAttributeDefaultValue 134
- setAttributePositions 278, 287, 302
- setAttributes 273, 279, 282, 288, 295, 302
- setAttributeText 126
- setAutoQuotes 266
- setBoolean 244
- setBooleanVoid 250
- setCacheMaxSize 98, 215
- setCacheStatisticsEnabled 211
- setCacheStatisticsFile 213
- setCacheStatisticsSnapshotTime 212
- setColor 114, 158
- setColorRGB 113, 156
- setDefaults 113, 145, 157
- setDouble 245
- setDoubleVoid 247
- setDynamic 189
- setErrorLog 310
- setExtentPages 210
- setExtentSize 216
- setFit 159
- setFontSize 115, 159
- setForcedQuotes 267
- setFrequency 271, 276, 281, 285, 294, 300
- setGraph 272, 278, 282, 287, 294, 302
- setHeadAttribute 271, 277
- setHeader 272, 282, 294
- setHeadPosition 272, 279
- setHeight 156
- setHighAvailabilityCoordinators 210
- setHighAvailabilityEnabled 216
- setHighAvailabilityIP 216
- setHighAvailabilityMasterHistory 213
- setHighAvailabilitySynchronization 215
- setInteger 251
- setIntegerVoid 246
- setLabel 114, 145, 157
- setLabelColor 115, 159
- setLabelColorRGB 115, 158
- setLicense 210
- setLocale 264, 267, 278, 287, 302
- setLogError 276, 286, 301
- setLogFile 211
- setLogLevel 211
- setLogOff 276, 286, 301
- setLong 246
- setLongVoid 243
- setLookAhead 6
- setMaterializedAttribute 5, 14, 25, 46, 50, 66, 70
- setMaximumHops 19, 28, 32, 36, 42, 53, 58, 62
- setMultilines 262
- setNull 249
- setNullVoid 245
- setNumLines 262
- setOID 249
- setOIDVoid 248
- setOutputLog 311
- setPoolClusterSize 211
- setPoolFrameSize 213
- setPoolPersistentMaxSize 212
- setPoolPersistentMinSize 218
- setPoolTemporaryMaxSize 217
- setPoolTemporaryMinSize 214
- setQuotes 262, 266
- setRecoveryCacheMaxSize 216
- setRecoveryCheckpointTime 218
- setRecoveryEnabled 214
- setRecoveryLogFile 219
- setRollbackEnabled 217
- setRowReader 277, 287, 301
- setRowWriter 271, 281, 294
- setSeparator 262, 266
- setShape 159
- setSingleLine 263
- setStartLine 263
- setStream 189
- setString 244
- setStringVoid 245
- setTailAttribute 270, 276
- setTailPosition 271, 277
- setTimestamp 243, 247, 250
- setTimestampFormat 279, 288, 302

setTimestampVoid 245, 248
setType 271, 277, 282, 286, 295, 301
setUnweightedEdgeCost 41
setVoid 247
setWidth 115, 159
Severe 152
SinglePairShortestPathBFS 35
SinglePairShortestPathDijkstra 40
size 172
Sparksee 204
SparkseeConfig 210
start 192
String 103
StringList 224
StrongConnectivityGabow 49

T

tails 137
tailsAndHeads 128
Text 104
TextStream 229
Timestamp 103
toArray 170, 171
toString 249, 250
TraversalBFS 57
TraversalDFS 61
TypeList 236

U

union 169
Unique 77

V

Value 242, 243
ValueList 252
valueOf 77, 95, 104, 117, 122, 153, 161, 177, 184
values 77, 94, 104, 117, 122, 153, 161, 177, 184
Version 204

W

Warning 152

WeakConnectivityDFS 69
write 229, 267, 291

Y

YGraphML 122