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-Lluis Larriba-Pey - WWW 2014.

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

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

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

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.
Author:
Sparsity Technologies http://www.sparsity-technologies.com

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	excludeNodes(Objects nodes) Set which nodes can't be used.
DisjointCommunities	getCommunities() Returns the results generated by the execution of the algorithm.
void	includeEdges(Objects edges) Set additional edges that can be used.
void	includeNodes(Objects nodes) Set additional nodes that can be used.
void	run() Executes the algorithm.
void	setLookAhead(int lookahead) Sets the size of the lookahead iterations to look (5 by default).
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.DisjointCommunityDetection

addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, getCommunities, includeEdges, includeNodes, run, setMaterializedAttribute

Methods inherited from class com.sparsity.sparksee.algorithms.CommunityDetection

addAllNodeTypes, addNodeType, close, excludeEdges, excludeNodes, includeEdges, includeNodes, 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 inherited from interface java.lang.AutoCloseable

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.

(continued on next page)

(continued from last page)

Parameters:

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

Methods

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.

---

addAllEdgeTypes

`public void` **`addAllEdgeTypes`**`()`

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

---

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.

---

run

`public void` **`run`**`()`

Executes the algorithm.

---

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.

(continued on next page)

(continued from last page)

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.

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.

addNodeType

`public void addNodeType(int type)`

Allows connectivity through nodes of the given type.

Parameters:

`type` - null

includeNodes

`public void includeNodes(Objects nodes)`

Set additional nodes that can be used.

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

Parameters:

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

setLookAhead

`public void setLookAhead(int lookahead)`

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

(continued on next page)

(continued from last page)

Parameters:

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

addAllNodeTypes

`public void addAllNodeTypes()`

Allows connectivity through all node types of the graph.

includeEdges

`public void includeEdges(Objects edges)`

Set additional edges that can be used.

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

Parameters:

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

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.
void	includeEdges(Objects edges) Set additional edges that can be used.
void	includeNodes(Objects nodes) Set additional nodes that can 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 inherited from interface java.lang.AutoCloseable
close

Methods

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.

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()

addNodeType

public void **addNodeType**(int type)

Allows connectivity through nodes of the given type.

Parameters:

(continued from last page)

`type` - null

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.

close

`public void close()`

Closes the CommunityDetection instance.

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

includeNodes

`public void includeNodes(Objects nodes)`

Set additional nodes that can be used.

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

Parameters:

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

addAllNodeTypes

`public void addAllNodeTypes()`

Allows connectivity through all node types of the graph.

includeEdges

`public void includeEdges(Objects edges)`

Set additional edges that can be used.

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

Parameters:

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

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`

Methods inherited from interface `java.lang.AutoCloseable`
`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:

(continued from last page)

The number of connected components found in the graph.

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()

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.

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 inherited from interface java.lang.AutoCloseable
close

Methods

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.

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()`

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.

(continued from last page)

Returns:

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

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.

close

```
public void close()
```

Closes the Connectivity instance.

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

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

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`

Methods inherited from interface `java.lang.AutoCloseable`

`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

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)
```

(continued from last page)

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.

---

compute

```
public Objects compute()
```

Gets the resulting collection of nodes.

Returns:

The resulting collection of nodes.

---

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()`

---

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)
```

Helper method to easily compute a context from a node.

Parameters:

(continued from last page)

`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.

---

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.

---

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`t` - null

---

close

```
public void close()
```

Closes the Context instance.

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

---

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`d` - [in] Edge direction.

(continued from last page)

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

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

Methods inherited from interface java.lang.AutoCloseable
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

getNodes

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

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.

(continued from last page)

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

getCount

```
public long getCount()
```

Returns the number of communities found in the graph.

Returns:
The number of communities found in the graph.

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()`

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.

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
   |-com.sparsity.sparksee.algorithms.CommunityDetection
      |-com.sparsity.sparksee.algorithms.DisjointCommunityDetection
```

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	includeEdges(Objects edges) Set additional edges that can be used.
void	includeNodes(Objects nodes) Set additional nodes that can be used.

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, includeEdges, includeNodes, 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 inherited from interface java.lang.AutoCloseable
close

Methods

run

```
public void run()
```

Runs the algorithm in order to find the communities.

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.

addAllEdgeTypes

```
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

(continued from last page)

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.

---

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.

---

addNodeType

`public void addNodeType(int type)`

Allows connectivity through nodes of the given type.

Parameters:

`type` - null

---

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.

---

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.

(continued from last page)

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.

includeNodes

```
public void includeNodes(Objects nodes)
```

Set additional nodes that can be used.

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

Parameters:

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

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

includeEdges

```
public void includeEdges(Objects edges)
```

Set additional edges that can be used.

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

Parameters:

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

com.sparsity.sparksee.algorithms
Class KOpt

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

All Implemented Interfaces:
Closeable

---

public class **KOpt**
extends Object
implements Closeable

KOpt class.

Implements the 2-Opt and 3-Opt algorithms
Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

public	KOpt(Session session, OIDList tour) Creates a new instance.
public	KOpt(Session session) 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	close() Closes the KOpt instance.
double	getCurrentCost() Returns tour cost.
OIDList	getCurrentTour() Returns tour as a list of nodes.
boolean	isClosed() Gets if KOpt instance has been closed or not.

void	runThreeOpt() Runs 3-Opt local search.
void	runTwoOpt() Runs 2-Opt local search.
void	setCurrentTour(OIDList tour) Sets current tour as a list of nodes.
void	setEdgeWeightAttributeType(int attr) Sets the attribute to use as edge weight.
void	setMaxIterations(long maxIterations) Sets maximum number of iterations.
void	setTimeLimit(long maxTime) Limits execution time.

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.lang.AutoCloseable`

`close`

Constructors

KOpt

```
public KOpt(Session session,
            OIDList tour)
```

Creates a new instance.

Parameters:

`session` - [in] Session to get the graph from and perform algorithm.
`tour` - [in] Initial tour that needs to be improved.

KOpt

```
public KOpt(Session session)
```

Creates a new instance.

Parameters:

`session` - [in] Session to get the graph from and perform algorithm.

(continued from last page)

Methods

setCurrentTour

```
public void setCurrentTour(OIDList tour)
```

Sets current tour as a list of nodes.

Parameters:

`tour` - [in] Initial tour that needs to be improved.

---

getCurrentCost

```
public double getCurrentCost()
```

Returns tour cost.

---

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`dir` - [in] Edge direction.

---

getCurrentTour

```
public OIDList getCurrentTour()
```

Returns tour as a list of nodes.

---

setEdgeWeightAttributeType

```
public void setEdgeWeightAttributeType(int attr)
  throws RuntimeException
```

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double.

sparksee::gdb::Error

Parameters:

`attr` - [in] The attribute type to use as a weight. Default: InvalidAttribute

Throws:

`java.lang.RuntimeException` - null

(continued from last page)

runTwoOpt

```
public void runTwoOpt()
```

Runs 2-Opt local search.

addEdgeType

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

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters:

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

Throws:

`java.lang.RuntimeException` - null

runThreeOpt

```
public void runThreeOpt()
```

Runs 3-Opt local search.

isClosed

```
public boolean isClosed()
```

Gets if KOpt instance has been closed or not.

Returns:

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

See Also:

`close()`

setMaxIterations

```
public void setMaxIterations(long maxIterations)
```

Sets maximum number of iterations.

By default the algorithm will run until no improvement can be made in the current tour.

Parameters:

`maxIterations` - [in] Maximum number of iterations

(continued from last page)

setTimeLimit

```
public void setTimeLimit(long maxTime)
```

Limits execution time.

Parameters:

`maxTime` - [in] Time limit in miliseconds

---

addNodeType

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

Allows for traversing nodes of the given type.

sparksee::gdb::Error

Parameters:

`type` - [in] Node type.

Throws:

`java.lang.RuntimeException` - null

---

close

```
public void close()
```

Closes the KOpt 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.

com.sparsity.sparksee.algorithms
Class PageRank

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

All Implemented Interfaces:
Closeable

---

public class **PageRank**
extends Object
implements Closeable

PageRank class.

Implements the PageRank algorithm
Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

public	`PageRank(Session session)` Builds the PageRank.

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 PageRank instance.
boolean	`isClosed()` Gets if PageRank instance has been closed or not.
void	`run()` Runs the algorithm.
void	`setDamping(double damping)` Sets the damping value for the PageRank.
void	`setDefaultWeight(double weight)` Sets the default weight for those cases when a given edge does not have a weight attribute set.

void	setEdgeWeightAttributeType(int attr) Sets the attribute to use as edge weight.
void	setInitialPageRankValue(double startValue) Sets the initial PageRank value.
void	setNumIterations(int numIterations) Sets the number of iterations to run the PageRank for.
void	setOutputAttributeType(int attr) Sets the output attribute type.
void	setStartingNode(long startNode) Sets the starting node of the page rank to compute the Personalized PageRank variant.
void	setTolerance(double tolerance) Sets the tolerance threashold to continue computing the PageRank after each iteration.

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.lang.AutoCloseable
close

Constructors

PageRank

```
public PageRank(Session session)
```

Builds the PageRank.

Parameters:

`session` - [in] The session to use

Methods

setDamping

```
public void setDamping(double damping)
```

Sets the damping value for the PageRank.

Parameters:

`damping` - [in] The damping value. Default: 0.85

setDefaultWeight

```
public void setDefaultWeight(double weight)
```

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters:

`weight` - [in] The default weight

run

```
public void run()
  throws RuntimeException
```

Runs the algorithm.

sparksee::gdb::Error

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters:

`dir` - [in] Edge direction.

setOutputAttributeType

```
public void setOutputAttributeType(int attr)
  throws RuntimeException
```

Sets the output attribute type.

If the PageRank will run on more than one node type, then the output attribute must be of type GLOBAL_TYPE or NODES_TYPE. Otherwise, it must be a valid attribute for the used node type.

Parameters:

`attr` - [in] The attribute to store the result. Default: InvalidAttribute

Throws:

`java.lang.RuntimeException` - null

setStartingNode

```
public void setStartingNode(long startNode)
  throws RuntimeException
```

Sets the starting node of the page rank to compute the Personalized PageRank variant.

sparksee::gdb::Error

Parameters:

`startNode` - null

Throws:

(continued from last page)

`java.lang.RuntimeException` - null

setInitialPageRankValue

```
public void setInitialPageRankValue(double startValue)
```

Sets the initial PageRank value.

If a starting node is set, this initial value is only set for the starting node and the rest of nodes are set to 0.0

Parameters:

`startValue` - [in] The initial value to set. Default: 0.0

setEdgeWeightAttributeType

```
public void setEdgeWeightAttributeType(int attr)
  throws RuntimeException
```

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

sparksee::gdb::Error

Parameters:

`attr` - [in] The attribute type to use as a weight. Default: InvalidAttribute

Throws:

`java.lang.RuntimeException` - null

addEdgeType

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

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters:

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

Throws:

`java.lang.RuntimeException` - null

setNumIterations

```
public void setNumIterations(int numIterations)
```

Sets the number of iterations to run the PageRank for.

Parameters:

`numIterations` - [in] The number of interations to set. Default: 20

(continued from last page)

setTolerance

```
public void setTolerance(double tolerance)
```

Sets the tolerance threashold to continue computing the PageRank after each iteration.

If all the changes to any PPR value after an iteration are below that tolerance threshold, the algorithm finishes.

Parameters:

`tolerance` - [in] The tolerance to use normalized between 0 and 1. Default: 0.000001

isClosed

```
public boolean isClosed()
```

Gets if PageRank instance has been closed or not.

Returns:

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

See Also:

`close()`

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`type` - null

Throws:

`java.lang.RuntimeException` - null

close

```
public void close()
```

Closes the PageRank 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.

com.sparsity.sparksee.algorithms
Class RandomWalk

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

All Implemented Interfaces:
Closeable

---

public class **RandomWalk**
extends Traversal

RandomWalk class.

Implements the RandomWalk algorithm
Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

public	RandomWalk(Session session, long node) Builds the RandomWalk.

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	reset(long startNode) Sets the starting node of the RandomWalk.
void	setDefaultWeight(double weight) Sets the default weight for those cases when a given edge does not have a weight attribute set.
void	setEdgeWeightAttributeType(int attr) Sets the attribute to use as edge weight.
void	setInOutParameter(double val) Sets the In-Out parameter of the RandomWalk.
void	setMaximumHops(int maxhops) Sets the maximum hops restriction.
void	setReturnParameter(double val) Sets the return parameter of the RandomWalk.
void	setSeed(int seed) Sets the seed of the random walk.

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

Methods inherited from interface java.lang.AutoCloseable

close

Constructors

RandomWalk

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

Builds the RandomWalk.

Parameters:

session - [in] The session to use
node - [in] The starting node of the traversal

Methods

(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.

setDefaultWeight

```
public void setDefaultWeight(double weight)
```

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters:

`weight` - [in] The default weight

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`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.

setSeed

```
public void setSeed(int seed)
```

Sets the seed of the random walk.

Parameters:

`seed` - The seed to generate the random numbers that drive the random walk

(continued from last page)

setEdgeWeightAttributeType

```
public void setEdgeWeightAttributeType(int attr)
```

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

Parameters:

`attr` - [in] The attribute type to use as a weight. Default: InvalidAttribute

---

setInOutParameter

```
public void setInOutParameter(double val)
```

Sets the In-Out parameter of the RandomWalk.

Parameters:

`val` - The In-Out parameter to set. Default: 1.0

---

addEdgeType

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

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters:

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

---

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`type` - The node type to add

---

setReturnParameter

```
public void setReturnParameter(double val)
```

Sets the return parameter of the RandomWalk.

Parameters:

`val` - The return parameter to set. Default: 1.0

(continued from last page)

next

```
public long next()
```

Gets the next object of the traversal.

Returns:
A node or edge identifier.

hasNext

```
public boolean hasNext()
```

Gets if there are more objects to be traversed.

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

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.

reset

```
public void reset(long startNode)
```

Sets the starting node of the RandomWalk.

This method resets the RandomWalk.

sparksee::gdb::Error

Parameters:
`startNode` - null

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

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.

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 inherited from interface java.lang.AutoCloseable

close

Methods

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()
```

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.

isClosed

```
public boolean isClosed()
```

(continued from last page)

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.

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:
`type` - null

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.

close

```
public void close()
```

Closes the ShortestPath instance.

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

setMaximumHops

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

Sets the maximum hops restriction.

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

(continued from last page)

Parameters:

`maxhops` - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

addAllNodeTypes

`public void` **`addAllNodeTypes`**`()`

Allows for traversing all node types of the graph.

com.sparsity.sparksee.algorithms
Class SinglePairShortestPath

```
java.lang.Object
   |-com.sparsity.sparksee.algorithms.ShortestPath
      |-com.sparsity.sparksee.algorithms.SinglePairShortestPath
```

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 inherited from interface java.lang.AutoCloseable

close

Methods

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.

addAllEdgeTypes

public void **addAllEdgeTypes**(EdgesDirection dir)

Allows for traversing all edge types of the graph.

(continued from last page)

Parameters:

`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.

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.

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.

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.

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

(continued from last page)

`type` - null

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

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 algoritms, the cost is the sum of the costs of the edges of the shortest path.

Returns:

The cost of the shortest path.

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

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

Methods inherited from interface java.lang.AutoCloseable

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.

(continued from last page)

Methods

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.

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.

run

```
public void run()
```

Executes the algorithm.

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`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.

(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.

---

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.

---

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`type` - null

---

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

---

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.

---

getPathAsEdges

```
public OIDList getPathAsEdges()
```

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

(continued from last page)

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

com.sparsity.sparksee.algorithms
Class SinglePairShortestPathDijkstra

```
java.lang.Object
   |-com.sparsity.sparksee.algorithms.ShortestPath
      |-com.sparsity.sparksee.algorithms.SinglePairShortestPath
         |-com.sparsity.sparksee.algorithms.SinglePairShortestPathDijkstra
```

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	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	setDynamicEdgeCostCallback(SinglePairShortestPathDijkstraDynamicCost dynCostCalculator) Set a class callback to dynamically calculate the cost of the edges.
void	setMaximumHops(int maxhops) Sets the maximum hops restriction.
void	setUnweightedEdgeCost(double weight) 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

Methods inherited from interface java.lang.AutoCloseable

close

Constructors

SinglePairShortestPathDijkstra

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

(continued from last page)

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

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.

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.

run

`public void run()`

Executes the algorithm.

addAllEdgeTypes

`public void addAllEdgeTypes(EdgesDirection dir)`

Allows for traversing all edge types of the graph.

Parameters:

`dir` - [in] Edge direction.

excludeEdges

`public void excludeEdges(Objects edges)`

(continued from last page)

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.

---

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.

---

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.

---

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.

---

setDynamicEdgeCostCallback

```
public void setDynamicEdgeCostCallback(SinglePairShortestPathDijkstraDynamicCost
dynCostCalculator)
```

Set a class callback to dynamically calculate the cost of the edges.

The callback can be set to NULL (the default) to use the normal attribute based cost weights. The given class must be kept alive by the user for as long as the algorithm is running.

Parameters:

`dynCostCalculator` - [in] Class callback to calculate the edge costs

---

addNodeType

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

(continued from last page)

Allows for traversing nodes of the given type.

Parameters:

`type` - null

exists

```
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

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.

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.

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

com.sparsity.sparksee.algorithms
Class SinglePairShortestPathDijkstraDynamicCost

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

public class **SinglePairShortestPathDijkstraDynamicCost**
extends Object

Defines how to calculate an edge weight.

This is an interface which must be implemented by the user. While the algorithm is running, one or more calls for each edge that can be in the shortest path will be issued to get the weight of the edge that could change based on the predecessor node and the current minimum path from the source.

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

Method Summary

double	`calculateEdgeCost(long sourceNode, double sourceCost, int sourceLevel, long targetNode, long edge, int edgeWeightAttr)` Dynamic calculation of an edge weight.

Methods inherited from class `java.lang.Object`

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

Methods

calculateEdgeCost

```
public double calculateEdgeCost(long sourceNode,
         double sourceCost,
         int sourceLevel,
         long targetNode,
         long edge,
         int edgeWeightAttr)
```

Dynamic calculation of an edge weight.

This method will be called several times during the algorithm execution. It can be called for the same edge multiple times because the path to the source node (and as a consequence the source weight and levels) might change while the algorithm is running. This method will have the normal default implementation that only uses the specified edge attribute to get the weight or the unweighted edge cost if the edge doesn't have the specified cost attribute. But it can be overriden in a derived user class to implement a dinamic edge weight, that may need the sourceCost and sourceLevels to calculate the weight. Returns the current weight of the edge (>=0) or <0 if the edge can not be used.

Parameters:

`sourceNode` - [in] The current node
`sourceCost` - [in] Current total cost up to the source node
`sourceLevel` - [in] Current path size up to the source node
`targetNode` - [in] The next node
`edge` - [in] The edge to the next node that this methos must weight
`edgeWeightAttr` - [in] The attribute that stores the edge weight or InvalidAttribute

(continued from last page)

Returns:

Returns the current weight of the edge (>=0) or <0 if the edge can not be used.

com.sparsity.sparksee.algorithms
Class StrongConnectivity

```
java.lang.Object
   |-com.sparsity.sparksee.algorithms.Connectivity
      |-com.sparsity.sparksee.algorithms.StrongConnectivity
```

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 inherited from interface java.lang.AutoCloseable

close

Methods

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()
```

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.

(continued from last page)

Parameters:

`nodes` - [in] A set of node 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.

addAllEdgeTypes

`public void addAllEdgeTypes(EdgesDirection dir)`

Allows connectivity through all edge types of the graph.

Parameters:

`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.

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.

---

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

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

Methods inherited from interface java.lang.AutoCloseable

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

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.

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.

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.

run

```
public void run()
```

Executes the algorithm.

addAllEdgeTypes

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

Allows connectivity through all edge types of the graph.

Parameters:

`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.

(continued from last page)

Parameters:

`edges` - [in] A set of edge 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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

com.sparsity.sparksee.algorithms
Class Traversal

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

All Implemented Interfaces:
Closeable

Direct Known Subclasses:
TraversalDFS, TraversalBFS, RandomWalk

---

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 inherited from interface `java.lang.AutoCloseable`

`close`

Methods

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.

hasNext

`public boolean hasNext()`

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

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.

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`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.

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.

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()`

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

(continued from last page)

`type` - The node type to add

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

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.

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.

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

Methods inherited from interface java.lang.AutoCloseable

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

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.

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.

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`dir` - [in] Edge direction.

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`type` - The node type to add

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.

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

hasNext

```
public boolean hasNext()
```

Gets if there are more objects to be traversed.

(continued from last page)

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

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.

addAllNodeTypes

`public void addAllNodeTypes()`

Allows for traversing all node types of the graph.

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.

com.sparsity.sparksee.algorithms
Class TraversalDFS

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

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

Methods inherited from interface java.lang.AutoCloseable

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

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.

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.

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:

`dir` - [in] Edge direction.

addNodeType

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

Allows for traversing nodes of the given type.

Parameters:

`type` - The node type to add

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.

next

```
public long next()
```

Gets the next object of the traversal.

Returns:

A node or edge identifier.

hasNext

```
public boolean hasNext()
```

Gets if there are more objects to be traversed.

(continued from last page)

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

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

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.

com.sparsity.sparksee.algorithms
Class WeakConnectivity

```
java.lang.Object
   |-com.sparsity.sparksee.algorithms.Connectivity
      |-com.sparsity.sparksee.algorithms.WeakConnectivity
```

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	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 inherited from interface java.lang.AutoCloseable
close

Methods

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.

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.

(continued from last page)

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.

---

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.

---

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.

---

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.

(continued from last page)

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

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

Methods inherited from interface java.lang.AutoCloseable

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

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.

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.

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.

run

```
public void run()
```

Executes the algorithm.

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.

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.

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.

addAllNodeTypes

```
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

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

int	getArraySize() Gets the number of elements in the array.
long	getCount() Gets the number of non-NULL values.
DataType	getDataType() Gets the data type.
Value	getDefaultValue()
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	isArrayAttribute() Check if it's an array attribute.
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

InvalidAttribute

`public static int` **`InvalidAttribute`**

Invalid attribute identifier constant.

Methods

getSize

`public long` **`getSize`**`()`

Gets the number of different values.

Returns:
The number of different values.

getId

`public int` **`getId`**`()`

Gets the Sparksee attribute identifier.

Returns:
The Sparksee attribute identifier.

getTypeId

`public int` **`getTypeId`**`()`

Gets the Sparksee type identifier.

Returns:
The Sparksee type identifier.

isArrayAttribute

`public boolean` **`isArrayAttribute`**`()`

(continued from last page)

Check if it's an array attribute.

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

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.

getDataType

`public DataType getDataType()`

Gets the data type.

Returns:
The DataType.

getKind

`public AttributeKind getKind()`

Gets the attribute kind.

Returns:
The AttributeKind.

getDefaultValue

`public Value getDefaultValue()`

getArraySize

```
public int getArraySize()
```

Gets the number of elements in the array.

Returns:
The size of the array

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).

Basic attribute (non indexed attribute).

Indexed

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

Indexed attribute.

Indexed attribute.

Unique

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

Unique attribute (indexed + unique restriction).

Unique attribute (indexed + unique restriction).

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

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`

`forEach, iterator, spliterator`

Constructors

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.

AttributeList

`public AttributeList(Collection col)`

Creates a new instance from an integer collection.

Parameters:

`col` - Collection to initialize the instance.

Methods

add

`public void add(int attr)`

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

Parameters:

`attr` - [in] Sparksee attribute identifier.

clear

`public void clear()`

Clears the list.

iterator

`public AttributeListIterator iterator()`

(continued from last page)

Gets a new AttributeListIterator.

Returns:
AttributeListIterator instance.

count

```
public int count()
```

Number of elements in the list.

Returns:
Number of elements in the list.

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`

`forEachRemaining, hasNext, next, remove`

Methods

next

`public Integer next()`

See nextAttribute().

hasNext

```
public boolean hasNext()
```

Gets if there are more elements.

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

nextAttribute

```
public int nextAttribute()
```

Gets the next element.

remove

```
public void remove()
```

Operation not supported.

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

getMinLengthString

public int **getMinLengthString**()

Gets the minimum length.

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

Returns:
The minimum 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.

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.

getMin

`public Value getMin()`

Gets the minimum existing value (BASIC statistics).

Returns:
The minimum 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:
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 an string attribute, it just returns 0.

Returns:
The maximum length.

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:

(continued from last page)

The mean.

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.

getAvgLengthString

`public double getAvgLengthString()`

Gets the average length.

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

Returns:
The average length.

getMax

`public Value getMax()`

Gets the maximum existing value (BASIC statistics).

Returns:
The maximum existing value.

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

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`

`forEach, iterator, spliterator`

Constructors

BooleanList

`public BooleanList()`

Constructor.

This creates an empty list.

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.

Methods

clear

`public void clear()`

Clears the list.

iterator

`public BooleanListIterator iterator()`

Gets a new BooleanListIterator.

Returns:
BooleanListIterator instance.

add

`public void add(boolean value)`

(continued from last page)

Adds a Boolean at the end of the list.

Parameters:

`value` - [in] Boolean.

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`

`forEachRemaining, hasNext, next, remove`

Methods

next

`public Boolean next()`

See nextBoolean().

nextBoolean

```
public boolean nextBoolean()
```

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.

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 (==).

Equal condition (==).

GreaterEqual

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

Greater or equal condition (>=).

Greater or equal condition (>=).

GreaterThan

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

Greater than condition (>).

Greater than condition (>).

LessEqual

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

Less or equal condition (<=).

Less or equal condition (<=).

LessThan

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

Less than condition (<).

Less than condition (<).

NotEqual

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

Not equal condition (!=).

Not equal condition (!=).

Like

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

Substring condition.

Substring condition.

LikeNoCase

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

Substring (no case sensitive) condition.

Substring (no case sensitive) condition.

Between

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

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

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

RegExp

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

Regular expression condition.

Regular expression condition.

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 SPARKSEEConfig.

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

Method Summary

void	close() Closes the Database instance.
void	disableRollback() Disables the rollback mechanism.
void	dumpSchema(String filePath) Dump the database schema to the given file using the Sparksee scripting format.
void	enableRollback() Enables the rollback mechanism.
void	fixCurrentCacheMaxSize() Sets the cache maximum size to the current cache size in use.
String	getAlias() Gets the alias of the Database.
int	getCacheMaxSize() Gets the cache maximum size (in MB).
String	getPath() Gets the path of the Database.
void	getStatistics(DatabaseStatistics stats) Gets Database statistics.
boolean	isClosed() Gets if Database instance has been closed or not.

Session	newSession() Creates a new Session.
void	redoPrecommitted(long txId) Redo a pending precommitted transaction recovered.
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 inherited from interface java.lang.AutoCloseable
close

Methods

dumpSchema

public void **dumpSchema**(String filePath)

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

Parameters:
filePath - null

newSession

public Session **newSession**()

Creates a new Session.

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.

fixCurrentCacheMaxSize

```
public void fixCurrentCacheMaxSize()
```

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

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()

redoPrecommitted

```
public void redoPrecommitted(long txId)
```

Redo a pending precommitted transaction recovered.

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

Parameters:
`txId` - null

getStatistics

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

Gets Database statistics.

Parameters:
`stats` - [out] The DatabaseStatistics instance.

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.

(continued from last page)

Parameters:

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

disableRollback

`public void disableRollback()`

Disables the rollback mechanism.

close

`public void close()`

Closes the Database instance.

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

getAlias

`public String getAlias()`

Gets the alias of the Database.

Returns:

The alias of the Database.

getCacheMaxSize

`public int getCacheMaxSize()`

Gets the cache maximum size (in MB).

Returns:

Returns the current cache max size.

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

getRead

`public long` **`getRead`**`()`

Gets total read data in KBytes.

Returns:
Total read data in KBytes.

(continued from last page)

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.

getWrite

`public long getWrite()`

Gets total written data in KBytes.

Returns:
Total read written in KBytes.

getCache

`public long getCache()`

Gets cache size in KBytes.

Returns:
Cache size in KBytes.

getSessions

`public long getSessions()`

Gets the number of sessions.

Returns:
The number of sessions.

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.

Boolean data type.

Integer

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

32-bit signed integer data type.

32-bit signed integer data type.

Long

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

64-bit signed integer data type.

64-bit signed integer data type.

Double

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

64-bit signed double data type.

64-bit signed double data type.

Timestamp

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

Distance from Epoch (UTC) time in milliseconds precision.

Distance from Epoch (UTC) time in milliseconds precision.

(continued from last page)

String

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

Unicode string data type.

Unicode string data type.

Text

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

Large unicode character object (CLOB) data type.

Large unicode character object (CLOB) data type.

OID

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

Object identifier (OID) data type.

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

prepare

public void **prepare**(Graph graph)

Default implementation of the ExportManager class method Prepare.

Parameters:
graph - null

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.

Returns:
TRUE.

release

public void **release**()

Default implementation of the ExportManager class method Release.

getNodeType

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

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

(continued from last page)

Parameters:

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

Returns:

TRUE.

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.

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.

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.

getEdgeType

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

(continued from last page)

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.

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 eges, 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.

getEdge

`public long` **`getEdge`**`()`

Gets the edge identifier.

(continued from last page)

Returns:
The Sparksee edge identifier.

getTail

```
public long getTail()
```

Gets the tail of the edge.

Returns:
The Sparksee edge identifier of the tail of the edge.

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

setAsDirected

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

Sets if the edge should be managed as directed.

(continued from last page)

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

getLabelColorRGB

`public int getLabelColorRGB()`

Gets the edge label color.

Returns:
The edge label color.

setLabel

`public void setLabel(String label)`

Sets the edge label.

Parameters:
`label` - [in] The edge label.

setWidth

`public void setWidth(int width)`

Sets the edge width.

Parameters:
`width` - [in] The edge width.

setColorRGB

`public void setColorRGB(int color)`

Sets the edge color.

Parameters:
`color` - [in] The edge color.

setFontSize

`public void setFontSize(int size)`

Sets the edge label font size.

Parameters:
`size` - [in] The edge label font size.

getLabelColor

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

Gets the color of the label.

getColor

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

Gets the color of the edge.

setLabelColorRGB

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

Sets the edge label color.

Parameters:

`color` - [in] The edge label color.

getColorRGB

```
public int getColorRGB()
```

Gets the edge color.

Returns:

The edge color.

getFontSize

```
public int getFontSize()
```

Gets the edge label font size.

Returns:

The edge label font size.

setDefaults

```
public void setDefaults()
```

Sets to default values.

(continued from last page)

setColor

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

Sets the color of the edge.

Parameters:
c - New value.

getLabel

```
public String getLabel()
```

Gets the edge label.

Returns:
The edge label.

setLabelColor

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

Sets the color of the label.

Parameters:
c - New value.

getWidth

```
public int getWidth()
```

Gets the edge width.

Returns:
The edge width.

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.

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.

In-going edges.

Outgoing

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

Out-going edges.

Out-going edges.

Any

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

In-going or out-going edges.

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 fortmats. 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)

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.

---

release

```
public void release()
```

Ends the export process.

It is called once after the export process.

---

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.

---

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.

(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.

---

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.

---

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.

---

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

Returns:

TRUE.

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.

Export to Graphviz format.

GraphML

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

Export to GraphML format.

Export to GraphML format.

YGraphML

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

Export to YGRAPHML format.

Export to YGRAPHML format.

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).
void	encryptedBackup(String file, String keyInHex, String ivInHex) Dumps all the data to a backup file.
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.
ValueArray	getArrayAttribute(long oid, int attr) Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.
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	getObjectType(long oid) Gets the Sparksee node or edge type identifier for the given OID.
Session	getSession()
Type	getType(int type) Gets information about the given type.
Values	getValues(int attr) Gets the Value collection for the given attribute.
Objects	heads(Objects edges) Gets all the heads from the given edges collection.
void	indexAttribute(int attr, AttributeKind kind) Updates the index of the given attribute.
void	indexNeighbors(int edgeType, boolean neighbors) Creates or destroys the neighbors index of an edge type.
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.
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.
int	newArrayAttribute(int type, String name, DataType dt, int size) Creates a new array attribute.
int	newAttribute(int type, String name, DataType dt, AttributeKind kind) Creates a new attribute.
int	newAttribute(int type, String name, DataType dt, AttributeKind kind, Value defaultValue) Creates a new attribute with a default value.
long	newEdge(int type, int tailAttr, Value tailV, int headAttr, Value headV) Creates a new edge instance.
long	newEdge(int type, long tail, long head) Creates a new edge instance.
int	newEdgeType(String name, boolean directed, boolean neighbors) Creates a new edge type.
long	newNode(int type) Creates a new node instance.

int	newNodeType(String name) Creates a new node type.
int	newRestrictedEdgeType(String name, int tail, int head, boolean neighbors) Creates a new restricted edge type.
int	newSessionArrayAttribute(int type, DataType dt, int size) Creates a new Session array attribute.
int	newSessionAttribute(int type, DataType dt, AttributeKind kind) Creates a new Session attribute.
int	newSessionAttribute(int type, DataType dt, AttributeKind kind, Value defaultValue) Creates a new Session attribute with a default value.
void	removeAttribute(int attr) Removes the given attribute.
void	removeType(int type) Removes the given type.
void	renameAttribute(int attr, String newName) Renames an attribute.
void	renameType(int type, String newName) Renames a type.
void	renameType(String oldName, String newName) Renames a type.
Objects	select(int type) Selects all OIDs belonging to the given type.
Objects	select(int attr, Condition cond, Value value) Selects all OIDs satisfying the given condition for the given attribute.
Objects	select(int attr, Condition cond, Value value, Objects restriction) Selects all OIDs satisfying the given condition for the given attribute.
Objects	select(int attr, Condition cond, Value lower, Value higher) Selects all OIDs satisfying the given condition for the given attribute.
Objects	select(int attr, Condition cond, Value lower, Value higher, Objects restriction) Selects all OIDs satisfying the given condition for the given attribute.
void	setArrayAttribute(int attr, Value value) Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.
ValueArray	setArrayAttribute(long oid, int attr, Value value) Sets all the elements of the ValueArray for the given array attribute and OID and returns it.
void	setArrayAttributeVoid(long oid, int attr, Value value) Sets all the elements of the ValueArray for the given array attribute and OID.
void	setAttribute(int attr, Value value) Sets the value of the given attribute for all the objects of the types the attribute applies to.

void	setAttribute(long oid, int attr, Value value) Sets the Value for the given attribute and OID.
void	setAttributeDefaultValue(int attr, Value value) Sets a default value for an attribute.
void	setAttributeText(long oid, int attr, TextStream tstream) Sets the writable TextStream for the given text attribute and OID.
Objects	tails(Objects edges) Gets all the tails from the given edges collection.
void	tailsAndHeads(Objects edges, Objects tails, Objects heads) Gets all the tails and heads from the given edges collection.
KeyValues	topK(int attribute, Condition operation, Value lower, Order order, int k) Gets a KeyValues iterator as a result of the TopK operation.
KeyValues	topK(int attribute, Condition operation, Value lower, Order order, int k, Objects restriction) Gets a KeyValues iterator as a result of the TopK operation.
KeyValues	topK(int attribute, Condition operation, Value lower, Value higher, Order order, int k) Gets a KeyValues iterator as a result of the TopK operation.
KeyValues	topK(int attribute, Condition operation, Value lower, Value higher, Order order, int k, Objects restriction) Gets a KeyValues iterator as a result of the TopK operation.
KeyValues	topK(int attribute, Order order, int k) Gets a KeyValues iterator as a result of the TopK operation.
KeyValues	topK(int attribute, Order order, int k, Objects restriction) Gets a KeyValues iterator as a result of the TopK operation.

Methods inherited from class `java.lang.Object`

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

Methods

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.

(continued from last page)

Returns:
Any of the edges or the Objects InvalidOID.

findEdgeTypes

```
public TypeList findEdgeTypes()
```

Gets all existing Sparksee edge type identifiers.

Returns:
Sparksee edge type identifier list.

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.

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.

(continued from last page)

backup

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

Dumps all the data to a backup file.

See the Sparksee class Restore methods.

Parameters:

`file` - [in] Output backup file path.

Throws:

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

---

removeAttribute

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

Removes the given attribute.

Parameters:

`attr` - [in] Sparksee attribute identifier.

---

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.

---

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.

---

findTypes

```
public TypeList findTypes()
```

(continued from last page)

Gets all existing Sparksee node and edge type identifiers.

Returns:

Sparksee node and edge type identifier list.

newArrayAttribute

```
public int newArrayAttribute(int type,
          String name,
          DataType dt,
          int size)
```

Creates a new array attribute.

Parameters:

`type` - [in] Sparksee node or edge type identifier.
`name` - [in] Unique name for the new attribute.
`dt` - [in] Base Data type for the new array attribute elements.
`size` - [in] The array size.

Returns:

Unique Sparksee attribute identifier.

findType

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

Gets the Sparksee type identifier for the given type name.

Parameters:

`name` - [in] Unique type name.

Returns:

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

topK

```
public KeyValues topK(int attribute,
          Condition operation,
          Value lower,
          Order order,
          int k)
```

Gets a KeyValues iterator as a result of the TopK operation.

Parameters:

`attribute` - The attribute to perform the TopK on
`operation` - The operation to perform in the top k
`lower` - The lower bound Value to be satisfied
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK

(continued from last page)

Returns:
A KeyValues instance

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.

setAttributeText

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

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

Parameters:
`oid` - [in] Sparksee OID.
`attr` - [in] Sparksee attribute identifier.
`tstream` - [in] New Text value. This corresponds to a TextStream to write.

setAttribute

```
public void setAttribute(int attr,
         Value value)
```

Sets the value of the given attribute for all the objects of the types the attribute applies to.

Does not work for TEXT attribute types

txThe Transaction this operation belongs to attributeThe attribute to initialize

Parameters:
`attr` - null
`value` - The value to initialize the attribute to

indexNeighbors

```
public void indexNeighbors(int edgeType,
         boolean neighbors)
```

Creates or destroys the neighbors index of an edge type.

Parameters:
`edgeType` - [in] Sparksee Edge type identifier.
`neighbors` - [in] If TRUE, it indexes the neighbor nodes, otherwise it removes the index.

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.

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.

setArrayAttribute

```
public void setArrayAttribute(int attr,
        Value value)
```

Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.

Parameters:

`attr` - [in] Sparksee array attribute identifier.
`value` - [in] Value for all the array elements of the arrays.

topK

```
public KeyValues topK(int attribute,
        Order order,
        int k)
```

Gets a KeyValues iterator as a result of the TopK operation.

Parameters:

`attribute` - The attribute to perform the TopK on
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK

Returns:

A KeyValues instance

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.

getAttributeStatistics

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

Gets statistics from the given attribute.

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

Parameters:

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

Returns:

An AttributeStatistics instace.

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.

countNodes

```
public long countNodes()
```

(continued from last page)

Gets the number of nodes.

Returns:

The number of nodes.

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.

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.

Returns:

Sparksee OID or the Objects InvalidOID.

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.

(continued from last page)

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.

---

encryptedBackup

```
public void encryptedBackup(String file,
          String keyInHex,
          String ivInHex)
  throws RuntimeException,
          FileNotFoundException
```

Dumps all the data to a backup file.

See the Sparksee class RestoreEncryptedBackup methods.

Parameters:

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

Throws:

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

---

drop

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

Drops the given OID.

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

Parameters:

`oid` - [in] Sparksee OID to be removed.

---

setArrayAttributeVoid

```
public void setArrayAttributeVoid(long oid,
          int attr,
          Value value)
```

(continued from last page)

Sets all the elements of the ValueArray for the given array attribute and OID.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray which can be obtained using the GetArrayAttribute operation.

Parameters:

`oid` - [in] Sparksee OID.
`attr` - [in] Sparksee array attribute identifier.
`value` - [in] Value for all the array elements of the given attribute and OID.

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.

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.

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.

tails

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

Gets all the tails from the given edges collection.

Parameters:

`edges` - [in] Sparksee edge identifier collection.

Returns:

The tails collection.

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.

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.

export

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

Exports the Graph.

(continued from last page)

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

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.

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.

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:

(continued from last page)

Objects instance.

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.

topK

```
public KeyValues topK(int attribute,
         Condition operation,
         Value lower,
         Value higher,
         Order order,
         int k)
```

Gets a KeyValues iterator as a result of the TopK operation.

Parameters:

`attribute` - The attribute to perform the TopK on
`operation` - The operation to perform in the top k
`lower` - The lower bound Value to be satisfied
`higher` - The upper bound Value to be satisfied
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK

Returns:

A KeyValues instance

dumpData

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

Dumps logical data to a file.

Parameters:

`file` - [in] Output file path.

Throws:

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

setArrayAttribute

```
public ValueArray setArrayAttribute(long oid,
         int attr,
         Value value)
```

Sets all the elements of the ValueArray for the given array attribute and OID and returns it.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray. And you can get it again with the GetArrayAttribute operation.

Parameters:

`oid` - [in] Sparksee OID.
`attr` - [in] Sparksee array attribute identifier.
`value` - [in] Value for all the array elements of the given attribute and OID.

Returns:

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

topK

```
public KeyValues topK(int attribute,
         Condition operation,
         Value lower,
         Value higher,
         Order order,
         int k,
         Objects restriction)
```

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters:

`attribute` - The attribute to perform the TopK on
`operation` - The operation to perform in the top k
`lower` - The lower bound Value to be satisfied
`higher` - The upper bound Value to be satisfied
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK
`restriction` - null

Returns:

A KeyValues instance

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:

(continued from last page)

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.

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.

findNodeTypes

```
public TypeList findNodeTypes()
```

Gets all existing Sparksee node type identifiers.

Returns:

Sparksee node type identifier list.

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.

topK

```
public KeyValues topK(int attribute,
          Condition operation,
          Value lower,
          Order order,
          int k,
          Objects restriction)
```

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

(continued from last page)

Parameters:

`attribute` - The attribute to perform the TopK on
`operation` - The operation to perform in the top k
`lower` - The lower bound Value to be satisfied
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK
`restriction` - null

Returns:

A KeyValues instance

---

newSessionArrayAttribute

```
public int newSessionArrayAttribute(int type,
          DataType dt,
          int size)
```

Creates a new Session array attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters:

`type` - [in] Sparksee node or edge type identifier.
`dt` - [in] Base Data type for the new array attribute elements.
`size` - [in] The array size.

Returns:

Unique Sparksee attribute identifier.

---

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.

---

getAttribute

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

(continued from last page)

Gets information about the given attribute.

Parameters:
`attr` - [in] Sparksee attribute identifier.

Returns:
The Attribute for the given Sparksee attribute identifier.

getSession

```
public Session getSession()
```

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.

dumpStorage

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

Dumps internal storage data to a file.

Parameters:
`file` - [in] Output file path.

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

select

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

Selects all OIDs belonging to the given type.

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

(continued from last page)

Returns:
Objects instance.

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

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.

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.

topK

```
public KeyValues topK(int attribute,
        Order order,
        int k,
        Objects restriction)
```

(continued from last page)

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters:

`attribute` - The attribute to perform the TopK on
`order` - The ordering semantincs of the top-k
`k` - The size of the TopK
`restriction` - null

Returns:

A KeyValues instance

---

countEdges

```
public long countEdges()
```

Gets the number of edges.

Returns:

The number of edges.

---

renameType

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

Renames a type.

The new name must be available.

Parameters:

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

---

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.

---

newSessionAttribute

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

(continued from last page)

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.

---

heads

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

Gets all the heads from the given edges collection.

Parameters:

`edges` - [in] Sparksee edge identifier collection.

Returns:

The heads collection.

---

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.

---

renameType

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

Renames a type.

The new name must be available.

(continued from last page)

Parameters:

`oldName` - [in] The current name of the type to be renamed.
`newName` - [in] The new name for the type.

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.

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.

Returns:

Sparksee attribute identifier list.

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.

getArrayAttribute

```
public ValueArray getArrayAttribute(long oid,
                                    int attr)
```

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

Parameters:

(continued from last page)

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

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

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.

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.

getEdgeData

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

Gets information about an edge.

Parameters:
`edge` - [in] Sparksee edge identifier.

Returns:
An EdgeData instance.

(continued from last page)

drop

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

Drops all the OIDs from the given collection.

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

Parameters:

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

---

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.

---

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.

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

setDefaults

public void **setDefaults**()

(continued from last page)

Sets to default values.

setLabel

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

Sets the graph label.

Parameters:

`label` - [in] The graph label.

getLabel

```
public String getLabel()
```

Gets the graph label.

Returns:

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

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`

`forEach, iterator, spliterator`

Constructors

Int32List

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

Creates a new instance from an integer array.

Parameters:
`list` - Integer array to initialize the instance.

Int32List

```
public Int32List()
```

Constructor.

This creates an empty list.

Int32List

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

Creates a new instance from an integer collection.

Parameters:
`col` - Collection to initialize the instance.

Methods

clear

```
public void clear()
```

Clears the list.

iterator

```
public Int32ListIterator iterator()
```

Gets a new Int32ListIterator.

Returns:
Int32ListIterator instance.

add

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

(continued from last page)

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

Parameters:

`value` - [in] The integer.

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`

`forEachRemaining, hasNext, next, remove`

Methods

next

`public Integer` **`next`**`()`

See nextInt32().

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.

nextInt32

```
public Integer nextInt32()
```

Gets the next element.

com.sparsity.sparksee.gdb
Class KeyValue

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

public class **KeyValue**
extends Object

Constructor Summary

public	KeyValue()

Method Summary

int	compareTo(KeyValue kvalue) See compare().
int	compareTo(Object value) See compare().
boolean	equals(Object other)
long	getKey()
Value	getValue()
int	hashCode()
String	toString() 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`

Constructors

KeyValue

`public` **`KeyValue`**`()`

(continued from last page)

Methods

equals

```
public boolean equals(Object other)
```

Parameters:
`other` - null

compareTo

```
public int compareTo(KeyValue kvalue)
```

See compare().

Parameters:
`kvalue` - null

hashCode

```
public int hashCode()
```

getKey

```
public long getKey()
```

compareTo

```
public int compareTo(Object value)
```

See compare().

This just works if the given object is a Value instance.

Parameters:
`value` - null

toString

```
public String toString()
```

(continued from last page)

Gets a String representation of the Value.

getValue

```
public Value getValue()
```

com.sparsity.sparksee.gdb
Class KeyValues

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

All Implemented Interfaces:
Iterator, Closeable

---

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

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 KeyValues instance.
boolean	hasNext() Checks if the KeyValues has more KeyValue pairs.
boolean	isClosed() Gets if KeyValues instance has been closed or not.
KeyValue	next() Gets the next KeyValue pair.
void	next(KeyValue kv) Gets the next KeyValue pair.

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.lang.AutoCloseable

close

Methods inherited from interface java.util.Iterator

forEachRemaining, hasNext, next, remove

Methods

isClosed

public boolean **isClosed**()

Gets if KeyValues instance has been closed or not.

Returns:
TRUE if the KeyValues instance has been closed, FALSE otherwise.

See Also:
close()

next

public KeyValue **next**()

Gets the next KeyValue pair.

Returns:
Returns the next KeyValue pair

next

public void **next**(KeyValue kv)

Gets the next KeyValue pair.

Parameters:
kv - Returns the next KeyValue pair

hasNext

public boolean **hasNext**()

Checks if the KeyValues has more KeyValue pairs.

Returns:
Returns ture if there are more KeyValue pairs

close

public void **close**()

(continued from last page)

Closes the KeyValues instance.

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

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.

Disable log.

Severe

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

Severe log level.

Severe log level.

Warning

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

Warning log level.

Warning log level.

Info

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

Info log level.

Info log level.

Config

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

Config log level.

Config log level.

Fine

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

(continued from last page)

Fine log level.

Fine log level.

Debug

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

Debug log level.

Debug log level.

Methods

values

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

valueOf

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

com.sparsity.sparksee.gdb
Class MissingEndpoint

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

All Implemented Interfaces:
Serializable, Comparable

---

public final class **MissingEndpoint**
extends Enum

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

Field Summary

public static final	Create Create the endpoint.
public static final	Ignore Ignore the edge.
public static final	IsError Throw an error.

Method Summary

static MissingEndpoint	valueOf(String name)
static MissingEndpoint[]	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)

IsError

`public static final com.sparsity.sparksee.gdb.MissingEndpoint IsError`

Throw an error.

Throw an error.

Create

`public static final com.sparsity.sparksee.gdb.MissingEndpoint Create`

Create the endpoint.

Create the endpoint.

Ignore

`public static final com.sparsity.sparksee.gdb.MissingEndpoint Ignore`

Ignore the edge.

Ignore the edge.

Methods

values

`public static MissingEndpoint[] values()`

---

valueOf

`public static MissingEndpoint 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 (Ox000000, 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

getLabelColorRGB

```
public int getLabelColorRGB()
```

Gets the node label color.

Returns:
The node label color.

---

setLabel

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

Sets the node label.

Parameters:
`label` - [in] The node label.

---

setWidth

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

Gets the node width.

Parameters:
`width` - The node width in pixels.

---

setColorRGB

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

Sets the node color.

Parameters:
`color` - The node color.

---

setFontSize

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

(continued from last page)

Sets the node label font size.

Parameters:

`size` - [in] The node label font size.

setHeight

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

Sets the node height.

Parameters:

`height` - [in] The node height in pixels.

getLabelColor

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

Gets the color of the label.

getColor

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

Gets the color of the node.

getShape

```
public NodeShape getShape()
```

Gets the node shape.

Returns:

The node shape.

setLabelColorRGB

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

Sets the node label color.

Parameters:

`color` - [in] The node label color.

isFit

```
public boolean isFit()
```

(continued from last page)

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'.

getColorRGB

```
public int getColorRGB()
```

Gets the node color.

Returns:
The node color.

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'.

getFontSize

```
public int getFontSize()
```

Gets the node label font size.

Returns:
The node label font size.

setDefaults

```
public void setDefaults()
```

Sets to default values.

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.

setLabelColor

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

Sets the color of the label.

Parameters:
`c` - New value.

getWidth

```
public int getWidth()
```

Gets the node width.

Returns:
The node width in pixels.

setShape

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

Sets the node shape.

Parameters:
`shape` - [in] The node shape.

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.

Box shape.

Round

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

Round shape.

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, spliterator, toArray, toArray

Methods inherited from interface java.util.Collection

add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, parallelStream, remove, removeAll, removeIf, retainAll, size, spliterator, stream, toArray, toArray

Methods inherited from interface java.lang.Iterable

forEach, iterator, spliterator

Methods inherited from interface java.io.Closeable

close

Methods inherited from interface java.lang.AutoCloseable

close

Methods inherited from interface java.lang.Iterable

forEach, iterator, spliterator

Fields

InvalidOID

```
public static int InvalidOID
```

Invalid object identifier constant.

Methods

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.

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.

any

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

Gets an element from the collection.

Returns:

Any element from the collection.

Throws:

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

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.

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.

iterator

```
public ObjectsIterator iterator()
```

Gets an ObjectsIterator.

Returns:
ObjectsIterator instance.

union

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

Performs the union operation.

This adds all existing elements of the given Objects instance to the Objects calling instance

Parameters:
`objs` - [in] Objects instance.

Returns:
Number of elements into the collection once the operation has been executed.

intersection

```
public long intersection(Objects objs)
```

(continued from last page)

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.

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.

clear

```
public void clear()
```

Clears the collection removing all its elements.

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.

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.

(continued from last page)

Returns:

true if this set changed as a result of the call

copy

```
public Objects copy()
```

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

Returns:

The new Objects instance.

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.
`objs2` - [in] Objects instance.

Returns:

New Objects instance.

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.

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:

(continued from last page)

Sample collection.

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.

close

```
public void close()
```

Closes the Objects instance.

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

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.

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.

(continued from last page)

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.

Returns:
an array containing all of the elements in this set.

---

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.

---

count

```
public long count()
```

Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

---

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.

---

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.

(continued from last page)

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

isEmpty

```
public boolean isEmpty()
```

Returns true if this Objects contains no elements.

Returns:
true if the collection contains no elements.

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.

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()`

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.

(continued from last page)

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.

---

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.

---

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.

---

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.

(continued from last page)

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.

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.lang.AutoCloseable`

`close`

Methods inherited from interface `java.util.Iterator`

forEachRemaining, hasNext, next, remove

Methods

next

public Long **next**()

See nextObject().

nextObject

public long **nextObject**()

Gets the next element.

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()

hasNext

public boolean **hasNext**()

Gets if there are more elements to traverse.

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

close

public void **close**()

Closes the ObjectsIterator instance.

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

remove

public void **remove**()

(continued from last page)

Operation not supported.

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.

Node object type.

Edge

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

Edge object type.

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

Constructors

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**(long[] list)

Creates a new instance from a long array.

Parameters:
list - Long array to initialize the instance.

OIDList

public **OIDList**()

Constructor.

This creates an empty list.

Methods

clear

public void **clear**()

(continued from last page)

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.

add

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

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

Parameters:

`attr` - [in] Sparksee object identifier.

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`

`forEachRemaining, hasNext, next, remove`

Methods

nextOID

`public long` **`nextOID`**`()`

Gets the next element.

next

```
public Long next()
```

See nextOID().

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.

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.

From lower to higher.

Descendent

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

From higher to lower.

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

getTotalMem

`public long getTotalMem()`

Gets physical memory size in Bytes.

Returns:
Physical memory size in Bytes.

getAvailableMem

`public long getAvailableMem()`

Gets avialable (free) memory size in Bytes.

Returns:
Avialable (free) memory size in Bytes.

getRealTime

`public long getRealTime()`

Gets time in microseconds (since epoch).

Returns:
Time in microseconds (since epoch).

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, boolean reiterable) 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

setStream

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

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.

Parameters:
`stream` - [in] The stream name
`handler` - [in] Query stream handler

Returns:
The previous handler, or NULL if it does not exists

setDynamic

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

(continued from last page)

Sets the value for a dynamic paramater.

Parameters:

`name` - [in] Parameter name
`value` - [in] Parameter value

---

execute

```
public ResultSet execute(String stmt,
        boolean reiterable)
```

Executes the given statement.

Parameters:

`stmt` - [in] Query statement.
`reiterable` - [in] Whether we want the resultset to be reiterable or not

Returns:

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

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 QueryLanguage

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

All Implemented Interfaces:
Serializable, Comparable

---

public final class **QueryLanguage**
extends Enum

The supported query languages.

Field Summary

public static final	SparkseeAlgebra The internal Sparksee Algebra.
public static final	SparkseeCypher The Sparksee Cypher Language inspired by the OpenCypher Query Language.

Method Summary

static QueryLanguage	valueOf(String name)
static QueryLanguage[]	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

SparkseeAlgebra

public static final com.sparsity.sparksee.gdb.QueryLanguage **SparkseeAlgebra**

(continued from last page)

The internal Sparksee Algebra.

The internal Sparksee Algebra.

SparkseeCypher

```
public static final com.sparsity.sparksee.gdb.QueryLanguage SparkseeCypher
```

The Sparksee Cypher Language inspired by the OpenCypher Query Language.

The Sparksee Cypher Language inspired by the OpenCypher Query Language.

Methods

values

```
public static QueryLanguage[] values()
```

valueOf

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

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

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

(continued from last page)

prepare

`public boolean prepare(ValueList list)`

Prepares the stream before it is started.

Parameters:
`list` - [in] Optional list of arguments

Returns:
FALSE on error

---

start

`public boolean start(ResultSetList list)`

Starts the stream.

Parameters:
`list` - [in] Optional list of input ResultSets

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.

getColumnDataType

`public DataType getColumnDataType(int index)`

Gets the datatype for the given column.

Parameters:
`index` - [in] Column index.

Returns:
DataType for the given column.

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.

getNumColumns

`public int getNumColumns()`

Gets the number of columns.

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

Returns:
The number of columns.

getColumnName

`public String getColumnName(int index)`

Gets the name for the given column.

Parameters:
`index` - [in] Column index.

Returns:

(continued from last page)

Column name.

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.

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.

JSON representation of the next <rows> rows in the resultset

Parameters:

`rows` - [in] Maximum number of rows

Returns:

JSON representation of the next <rows> rows in the resultset

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.

getColumnIndex

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

Gets the column index for the given column name.

Parameters:

`name` - [in] Column name.

Returns:

Column index.

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)

clear

```
public void clear()
```

Clears the list.

---

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.

---

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

next

`public ResultSet next()`

Moves to the next element.

Returns:
The next element.

hasNext

`public boolean hasNext()`

Gets if there are more elements.

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

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	begin() Begins a transaction.
void	beginUpdate() Begins an update transaction.
void	close() Closes the Session instance.
void	commit() Commits a transaction.
Graph	getGraph() Gets the Graph instance.
long	getInMemoryPoolCapacity() Gets the capacity of the in-memory pool.
boolean	isClosed() Gets if Session instance has been closed or not.
Objects	newObjects() Creates a new Objects instance.

Query	newQuery(QueryLanguage lang) Creates a new Query.
long	preCommit() PreCommits a transaction.
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 inherited from interface `java.lang.AutoCloseable`

`close`

Methods

rollback

`public void rollback()`

Rollbacks a transaction.

preCommit

`public long preCommit()`

PreCommits a transaction.

YOU SHOULD NOT USE THIS METHOD. This method exists for a specific service and it is used to force that the transaction is written in the recovery log even though it had not been committed yet (and may never be).

Returns:

The transaction id. This has to be used in RedoPrecommited in case of a recovery process.

begin

`public void begin()`

Begins a transaction.

beginUpdate

`public void beginUpdate()`

(continued from last page)

Begins an update transaction.

newQuery

`public Query newQuery(QueryLanguage lang)`

Creates a new Query.

Parameters:

`lang` - The query language to create the query for

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()`

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.

newObjects

`public Objects newObjects()`

(continued from last page)

Creates a new Objects instance.

Returns:

The new Objects instance.

getInMemoryPoolCapacity

```
public long getInMemoryPoolCapacity()
```

Gets the capacity of the in-memory pool.

Returns:

Returns the capacity of the in-memory pool

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

boolean	addChecksums(String path) Converts a database WITHOUT checksums into a database WITH checksums.
void	close() Closes the Sparksee instance.
Database	create(String path, String alias) Creates a new Database instance.
Database	create(String path, String alias, SparkseeConfig config) Creates a new Database instance.
boolean	decrypt(String path) Converts a database WITH encryption into a database WITHOUT encryption.
boolean	encrypt(String path) Converts a database WITHOUT encryption into an encrypted database.
long	getInMemoryPoolCapacity() Gets the capacity of the in-memory pool (in Megabytes)

boolean	isClosed() Gets if Sparksee instance has been closed or not.
Database	open(String path, boolean readOnly) Opens an existing Database instance.
Database	open(String path, boolean readOnly, SparkseeConfig config) Opens an existing Database instance.
boolean	removeChecksums(String path) Converts a database WITH checksums into a database WITHOUT checksums.
void	resizeInMemoryPool(long newCapacity) Resizes the in memory pool allocator.
Database	restore(String path, String backupFile) Restores a Database from a backup file.
Database	restore(String path, String backupFile, SparkseeConfig config) Restores a Database from a backup file.
Database	restoreEncryptedBackup(String path, String backupFile, SparkseeConfig config, String keyInHex, String ivInHex) Restores a Database from an encrypted backup file.
Database	restoreEncryptedBackup(String path, String backupFile, String keyInHex, String ivInHex) Restores a Database from an encrypted backup file.
boolean	verifyChecksums(String path) Verifies the Checksum integrity of the given image file and it's recovery log 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`

Methods inherited from interface `java.lang.AutoCloseable`

`close`

Fields

Version

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

Sparksee version.

Sparksee version.

Constructors

(continued from last page)

Sparksee

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

Creates a new instance.

Parameters:

`config` - [in/out] Sparksee configuration (may be updated).

Methods

restoreEncryptedBackup

```
public Database restoreEncryptedBackup(String path,
        String backupFile,
        SparkseeConfig config,
        String keyInHex,
        String ivInHex)
  throws RuntimeException,
        FileNotFoundException
```

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters:

`path` - [in] Database storage file.
`backupFile` - [in] The Backup file to be restored.
`config` - [in] Use a specific configuration instead of the one in this Sparksee
`keyInHex` - [In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
`ivInHex` - [In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns:

A Database instance.

Throws:

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

---

resizeInMemoryPool

```
public void resizeInMemoryPool(long newCapacity)
  throws RuntimeException
```

Resizes the in memory pool allocator.

Parameters:

`newCapacity` - The new capacity of the in memory pool allocator

Returns:

Returns true if was successful

Throws:

`java.lang.RuntimeException` - null

create

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

Creates a new Database instance.

Parameters:

`path` - [in] Database storage file.
`alias` - [in] Database alias name.
`config` - [in] Use a specific configuration instead of the one in this Sparksee

Returns:

A Database instance.

Throws:

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

verifyChecksums

```
public boolean verifyChecksums(String path)
  throws RuntimeException,
         FileNotFoundException
```

Verifies the Checksum integrity of the given image file and it's recovery log file.

The main database file checksums are always checked.

When the recovery log file exists, the recovery log file is also checked for checksum and format errors.

When the checksums of the main file or the recovery file are incorrect, the funcion returns false, in any other error it throws an exception.

Any error found is logged as a WARNING.

sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters:

`path` - The path to the image file

Returns:

Returns true if the checksum verification succeeded. Returns false if an invalid checksum was detected.

Throws:

`java.lang.RuntimeException` - null
`java.io.FileNotFoundException` - null

removeChecksums

```
public boolean removeChecksums(String path)
  throws RuntimeException,
         FileNotFoundException
```

(continued from last page)

Converts a database WITH checksums into a database WITHOUT checksums.

Any error found is logged. The database (and it's recovery log if present) have checksums.
sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters:

`path` - The path to the database image file

Returns:

Returns true if the checksum could be removed. The errors are returned with an exception.

Throws:

`java.lang.RuntimeException` - null
`java.io.FileNotFoundException` - null

---

restore

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

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

---

addChecksums

```
public boolean addChecksums(String path)
  throws RuntimeException,
         FileNotFoundException
```

Converts a database WITHOUT checksums into a database WITH checksums.

Any error found is logged. The database (and it's recovery log if present) does NOT have checksums.
sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters:

`path` - The path to the database image file

Returns:

Returns true if the checksum could be added or false otherwise.

Throws:

`java.lang.RuntimeException` - null
`java.io.FileNotFoundException` - null

(continued from last page)

restoreEncryptedBackup

```
public Database restoreEncryptedBackup(String path,
         String backupFile,
         String keyInHex,
         String ivInHex)
  throws RuntimeException,
         FileNotFoundException
```

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters:

`path` - [in] Database storage file.
`backupFile` - [in] The Backup file to be restored.
`keyInHex` - [In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
`ivInHex` - [In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns:

A Database instance.

Throws:

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

decrypt

```
public boolean decrypt(String path)
  throws RuntimeException,
         FileNotFoundException
```

Converts a database WITH encryption into a database WITHOUT encryption.

Any error found is logged. The database (and it's recovery log if present) is encrypted. The current encryption had been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters:

`path` - The path to the database image file

Returns:

Returns true if the database could be unencrypted. The errors are returned with an exception.

Throws:

`java.lang.RuntimeException` - null
`java.io.FileNotFoundException` - null

restore

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

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.

(continued from last page)

`config` - [in] Use a specific configuration instead of the one in this Sparksee

Returns:

A Database instance.

Throws:

`java.lang.RuntimeException` - null

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

create

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

Creates a new Database instance.

Parameters:

`path` - [in] Database storage file.

`alias` - [in] Database alias name.

Returns:

A Database instance.

Throws:

`java.lang.RuntimeException` - null

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

open

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

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.lang.RuntimeException` - null

`java.io.FileNotFoundException` - If the given file does not exist.

isClosed

```
public boolean isClosed()
```

Gets if Sparksee instance has been closed or not.

(continued from last page)

Returns:

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

See Also:

close()

---

open

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

Opens an existing Database instance.

The provided configuration will be used for all the database settings but not for the license, which must already be properly setup in this class.

Parameters:

`path` - [in] Database storage file.

`readOnly` - [in] If TRUE, open Database in read-only mode.

`config` - [in] Use a specific configuration instead of the one in this Sparksee

Returns:

A Database instance.

Throws:

`java.lang.RuntimeException` - null

`java.io.FileNotFoundException` - If the given file does not exist.

---

encrypt

```
public boolean encrypt(String path)
  throws RuntimeException,
         FileNotFoundException
```

Converts a database WITHOUT encryption into an encrypted database.

Any error found is logged. The database (and it's recovery log if present) does is NOT encrypted. The encryption had already been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters:

`path` - The path to the database image file

Returns:

Returns true if the database could be encrypted. The errors are returned with an exception.

Throws:

`java.lang.RuntimeException` - null

`java.io.FileNotFoundException` - null

---

close

```
public void close()
```

Closes the Sparksee instance.

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

getInMemoryPoolCapacity

`public long` **`getInMemoryPoolCapacity`**`()`

Gets the capacity of the in-memory pool (in Megabytes)

Returns:

Returns the capacity of the in memory pool

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 overrided with values from a properties file (see SparkseeProperties class). Also, this settings can be overrided calling a specific setter.

For each field, it is shown its default value and the property to override this value:

Extent size: 4KB ('sparksee.storage.extentsize' at SparkseeProperties).

Pages per extent: 1 page ('sparksee.storage.extentpages' at SparkseeProperties).

Checksums enabled: true ('sparksee.storage.checksum' at SparkseeProperties).

Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at SparkseeProperties).

Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at SparkseeProperties).

Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at SparkseeProperties).

Minimum size for the temporary pool: 16 frames ('sparksee.io.pool.temporal.minsize' at SparkseeProperties).

Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at SparkseeProperties).

Number of pools in the pool cluster: 0 pools ('sparksee.io.pool.clustersize' at SparkseeProperties). 0 or 1 means the clustering is disabled.

Maximum size for the cache (all pools): 0 MB ('sparksee.io.cache.maxsize' at SparkseeProperties).

License code: "" ('sparksee.license' at SparkseeProperties). No license code means evaluation license.

Log level: Info ('sparksee.log.level' at SparkseeProperties).

Log file: "sparksee.log" ('sparksee.log.file' at SparkseeProperties).

Cache statistics: false (disabled) ('sparksee.cache.statistics' at SparkseeProperties).

Cache statistics log file: "statistics.log" ('sparksee.cache.statisticsFile' at SparkseeProperties).

Cache statistics snapshot time: 1000 msecs [TimeUnit] ('sparksee.cache.statisticsSnapshotTime' at 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:

<X>[D|H|M|S|s|m|u]

where <X> is a number followed by an optional character which represents the unit: D for days, H for hours, M for minutes, S or s for seconds, m for milliseconds and u for microseconds. If no unit character is given, seconds are assumed.

Capture abort signals to dump the call stack ('sparksee.callstackdump' at SparkseeProperties) is enabled by default on most platforms.

Author:

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

Constructor Summary

public	`SparkseeConfig(String path)` Creates a new instance with a specific config file.
public	`SparkseeConfig(String path, String clientId, String licenseId)` Creates a new instance with a specific config file and IDs.
public	`SparkseeConfig()` Creates a new instance.

Method Summary

boolean	`downloadExpected()` Check if a new license will be automatically downloaded with the current settings.
int	`downloadLicense()` Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).
String	`getAESIV()` Get the AES initialization vector in a hexadecimal encoded string.
String	`getAESKey()` Get the AES encryption key in a hexadecimal encoded string.
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.
boolean	getCallStackDump() Gets whether the signals will be captured to dump the call stack or not.
boolean	getChecksumEnabled() Gets whether the storage checksum usage is enabled or disabled.
String	getClientId() Gets the client identifier.
String	getDownloadStatus() Gets a message with the license download result.
boolean	getEncryptionEnabled() Gets whether the storage encryption is enabled or disabled.
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.
long	getInMemAllocSize() Gets the in-memory allocator size.
String	getLicense() Gets the license key.
String	getLicenseId() Gets the license identifier.
int	getLicensePreDownloadDays() Get the number of days before expiration when a new license will be downloaded.
String	getLicenseRequest() Get information useful to manually request a license.
String	getLogFile() Gets the log file.
LogLevel	getLogLevel() Gets the log level.

int	getPoolFrameSize() Gets the size of a pool frame in number of extents.
int	getPoolPartitions() Gets the number of partitions in each PartitionedPool.
int	getPoolPersistentMaxSize() Gets the maximum size for the persistent pool in number of frames.
int	getPoolPersistentMinSize() Gets the minimum size for the persistent pool in number of frames.
int	getPoolTemporaryMaxSize() Gets the maximum size for the temporary pool in number of frames.
int	getPoolTemporaryMinSize() Gets the minimum size for the temporary pool in number of frames.
int	getRecoveryCacheMaxSize() Gets the maximum size for the recovery log cache in extents.
long	getRecoveryCheckpointTime() Gets the delay time (in microseconds) between automatic checkpoints.
boolean	getRecoveryEnabled() Gets whether the recovery is enabled or disabled.
String	getRecoveryLogFile() Gets the recovery log file.
boolean	getRollbackEnabled() Gets whether the rollback is enabled or disabled.
String	getSparkseeConfigFile() Gets the config file path.
boolean	getTmpEnabled() Gets whether using temporary storage for computations is enabled.
String	getTmpFolder() Gets the temporary folder used for temporary staging.
boolean	save() Save the current configuration in the specified config file.
boolean	saveAll() Save all the current configuration settings in the specified config file.
boolean	setAESEncryptionEnabled(int keySize) Enables storage encryption using AES and GENARATES a key and an iv.
boolean	setAESEncryptionEnabled(String keyInHex, String ivInHex) Enables storage encryption using AES with the given key and iv.
void	setCacheMaxSize(int megaBytes) Sets the maximum size for the cache (all pools) in MB.
void	setCacheStatisticsEnabled(boolean status) Enables or disables cache statistics.

void	setCacheStatisticsFile(String filePath) Sets the cache statistics log file.
void	setCacheStatisticsSnapshotTime(long microSeconds) Sets the cache statistics snapshot time.
void	setCallStackDump(boolean status) Sets whether the signals will be captured to dump the call stack or not.
void	setChecksumEnabled(boolean status) Enables or disables the storage checksum usage.
void	setClientId(String clientId) Set the client identifier.
void	setEncryptionDisabled() Disables storage encryption.
void	setExtentPages(int pages) Sets the number of pages per extent.
void	setExtentSize(int kBytes) Sets the size of the extents in KB.
void	setHighAvailabilityCoordinators(String ip) Sets the coordinators address and port list.
void	setHighAvailabilityEnabled(boolean status) Enables or disables high availability mode.
void	setHighAvailabilityIP(String ip) Sets the IP address and port of the instance.
void	setHighAvailabilityMasterHistory(long filePath) Sets the master's history log.
void	setHighAvailabilitySynchronization(long microSeconds) Sets the synchronization polling time.
void	setInMemAllocSize(long size) Sets the in-memory allocator size.
int	setLicense(String key) Sets the license key.
void	setLicenseId(String licenseId) Set the license identifier.
void	setLicensePreDownloadDays(int days) Start trying to automatically download a new license at the specified number of days before expiration.
void	setLogFile(String filePath) Sets the log file.
void	setLogLevel(LogLevel level) Sets the log level.
void	setPoolFrameSize(int extents) Sets the size of a pool frame in number of extents.

void	setPoolPartitions(int pools) Sets the number of pools in each PartitionedPool.
void	setPoolPersistentMaxSize(int frames) Sets the maximum size for the persistent pool in number of frames.
void	setPoolPersistentMinSize(int frames) Sets the minimum size for the persistent pool in number of frames.
void	setPoolTemporaryMaxSize(int frames) Sets the maximum size for the temporary pool in number of frames.
void	setPoolTemporaryMinSize(int frames) Sets the minimum size for the temporary pool in number of frames.
void	setRecoveryCacheMaxSize(int extents) Sets the maximum size for the recovery log cache in extents.
void	setRecoveryCheckpointTime(long microSeconds) Sets the delay time (in microseconds) between automatic checkpoints.
void	setRecoveryEnabled(boolean status) Enables or disables the recovery.
void	setRecoveryLogFile(String filePath) Sets the recovery log file.
void	setRollbackEnabled(boolean status) Enables or disables the rollback.
void	setSparkseeConfigFile(String path) Sets the config file path.
void	setTmpEnabled(boolean enabled) Sets whether to use temporary storage for computations or not.
void	setTmpFolder(String tmpFolder) Sets the temporary folder used for temporary staging.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

SparkseeConfig

public **SparkseeConfig**(String path)

Creates a new instance with a specific config file.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters:

path - [in] File path to the config file.

SparkseeConfig

```
public SparkseeConfig(String path,
                      String clientId,
                      String licenseId)
```

Creates a new instance with a specific config file and IDs.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the config file already exists, the client and license ids should have the same values as the given arguments. If the file doesn't exists, it will be created.

Parameters:

`path` - [in] File path to the config file.
`clientId` - [in] The client identifier.
`licenseId` - [in] The license identifier.

SparkseeConfig

```
public SparkseeConfig()
```

Creates a new instance.

Values are set with default values.

Methods

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.

setLicensePreDownloadDays

```
public void setLicensePreDownloadDays(int days)
```

Start trying to automatically download a new license at the specified number of days before expiration.

Parameters:

`days` - [in] Number of days before expiration or -1 if the license should never be downloaded.

getExtentPages

```
public int getExtentPages()
```

Gets the number of pages per extent.

Returns:

The number of pages per extent.

saveAll

```
public boolean saveAll()
```

Save all the current configuration settings in the specified config file.

It will try to save the current configuration only if a config file was specified. The saved config file WILL CONATAIN all the modified settings including the secret ENCRYPTION KEYS. You should usually just use the Save method instead.

Returns:
Returns true if the config file is successfully written or false otherwise.

getAESKey

```
public String getAESKey()
```

Get the AES encryption key in a hexadecimal encoded string.

Returns:
The AES encryption Key as an hexadecimal string.

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.

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.

getCallStackDump

```
public boolean getCallStackDump()
```

Gets whether the signals will be captured to dump the call stack or not.

Returns:
TRUE if the signals must be captured, FALSE otherwise.

(continued from last page)

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).

---

setAESEncryptionEnabled

```
public boolean setAESEncryptionEnabled(String keyInHex,
         String ivInHex)
```

Enables storage encryption using AES with the given key and iv.

The key and initialization vector will not be saved in the config file.

Parameters:

`keyInHex` - [In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes).
`ivInHex` - [In] The AES Initialization Vector as an hexadecimal string (16 bytes).

Returns:

Returns true if the encryption had been enabled with the given arguments or false otherwise.

---

getHighAvailabilityEnabled

```
public boolean getHighAvailabilityEnabled()
```

Gets whether high availability mode is enabled or disabled.

Returns:

TRUE if high availability mode is enabled, FALSE otherwise.

---

getPoolPersistentMinSize

```
public int getPoolPersistentMinSize()
```

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

Returns:

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

---

setChecksumEnabled

```
public void setChecksumEnabled(boolean status)
```

Enables or disables the storage checksum usage.

Parameters:

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

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.

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.

setRecoveryCheckpointTime

`public void setRecoveryCheckpointTime(long microSeconds)`

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 commited transaction.

setEncryptionDisabled

`public void setEncryptionDisabled()`

Disables storage encryption.

setRecoveryLogFile

`public void setRecoveryLogFile(String filePath)`

Sets the recovery log file.

filePath[in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log)

Parameters:

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

setPoolFrameSize

`public void setPoolFrameSize(int extents)`

(continued from last page)

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

Parameters:

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

getRecoveryLogFile

`public String getRecoveryLogFile()`

Gets the recovery log file.

Returns:

The recovery log file.

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.

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.

getExtentSize

`public int getExtentSize()`

Gets the size of a extent.

Returns:

The size of a extent in KB.

getSparkseeConfigFile

`public String getSparkseeConfigFile()`

Gets the config file path.

Returns:

(continued from last page)

Returns the configured path or an empty string

setExtentSize

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

Sets the size of the extents in KB.

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.

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.

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.

setInMemAllocSize

```
public void setInMemAllocSize(long size)
```

Sets the in-memory allocator size.

Parameters:

`size` - The size to set the in-memory allocator to

setHighAvailabilityMasterHistory

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

Sets the master's history log.

Parameters:

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

(continued from last page)

setTmpEnabled

`public void setTmpEnabled(boolean enabled)`

Sets whether to use temporary storage for computations or not.

Parameters:
`enabled` - True to use temporary storage for computation

getLogFile

`public String getLogFile()`

Gets the log file.

Returns:
The log file.

getTmpEnabled

`public boolean getTmpEnabled()`

Gets whether using temporary storage for computations is enabled.

Returns:
True if enabled

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.

getCacheStatisticsEnabled

`public boolean getCacheStatisticsEnabled()`

Gets whether cache statistics are enabled or disabled.

Returns:
TRUE if cache statistics are enabled, FALSE otherwise.

setHighAvailabilityCoordinators

`public void setHighAvailabilityCoordinators(String ip)`

(continued from last page)

Sets the coordinators address and port list.

Parameters:

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

setHighAvailabilitySynchronization

```
public void setHighAvailabilitySynchronization(long microSeconds)
```

Sets the synchronization polling time.

Parameters:

`microSeconds` - [in] The synchronization polling time.

getClientId

```
public String getClientId()
```

Gets the client identifier.

Returns:

The client identifier.

getCacheStatisticsFile

```
public String getCacheStatisticsFile()
```

Gets the cache statistics log file.

Useless if cache statistics are disabled.

Returns:

The cache statistics log file.

setSparkseeConfigFile

```
public void setSparkseeConfigFile(String path)
```

Sets the config file path.

The file is not loaded in this operation, see the constructor options if you need to load the file. The config file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters:

`path` - [in] File path to the config file.

save

```
public boolean save()
```

Save the current configuration in the specified config file.

It will try to save the current configuration only if a config file was specified. This method will be automatically used when a Sparksee class downloads a new License.

(continued from last page)

Returns:
Returns true if the config file is successfully written or false otherwise.

setPoolPersistentMaxSize

`public void setPoolPersistentMaxSize(int frames)`

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

Parameters:
`frames` - [in] The maximum size for the persistent pool in number of frames. It must be non-negative.

getLogLevel

`public LogLevel getLogLevel()`

Gets the log level.

Returns:
The LogLevel.

setLicense

`public int setLicense(String key)`

Sets the license key.

Parameters:
`key` - [in] The license key. Returns -1 if the new key can't be used, 0 if it's the same license or 1 if the new license is applied.

getDownloadStatus

`public String getDownloadStatus()`

Gets a message with the license download result.

It can be used when the DownloadLicense call failed.

Returns:
The log download result.

setLicenseId

`public void setLicenseId(String licenseId)`

Set the license identifier.

If you don't have a license identifier, you may want to register at http://sparsity-technologies.com/#sparksee

Parameters:
`licenseId` - [in] The license identifier.

getLicense

`public String getLicense()`

Gets the license key.

Returns:
The license key.

getHighAvailabilityIP

`public String getHighAvailabilityIP()`

Gets the IP address and port of the instance.

Returns:
The IP address and port of the instance.

setAESEncryptionEnabled

`public boolean setAESEncryptionEnabled(int keySize)`

Enables storage encryption using AES and GENARATES a key and an iv.

YOU MUST GET AND KEEP SAFE THE GENERATED KEY AND IV! The key and initialization vector will not be saved in the config file.

Parameters:
`keySize` - [In] The key size in bytes (8, 16 or 32).

Returns:
Returns true if the encryption had been enabled.

getInMemAllocSize

`public long getInMemAllocSize()`

Gets the in-memory allocator size.

Returns:
Returns the in-memory allocator size

getLicensePreDownloadDays

`public int getLicensePreDownloadDays()`

Get the number of days before expiration when a new license will be downloaded.

Returns:
Returns the number of days or -1 if it will never be automatically downloaded.

setCallStackDump

```
public void setCallStackDump(boolean status)
```

Sets whether the signals will be captured to dump the call stack or not.

Parameters:

`status` - [in] If TRUE signals must be captured.

downloadLicense

```
public int downloadLicense()
```

Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).

Returns:

Returns -1 if a valid license key couldn't be downloaded, 0 if the same license was downloaded or 1 if a different license is downloaded.

getRecoveryEnabled

```
public boolean getRecoveryEnabled()
```

Gets whether the recovery is enabled or disabled.

Returns:

TRUE if the recovery is enabled, FALSE otherwise.

getLicenseRequest

```
public String getLicenseRequest()
```

Get information useful to manually request a license.

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.

getTmpFolder

```
public String getTmpFolder()
```

(continued from last page)

Gets the temporary folder used for temporary staging.

Returns:
The temporary folder path

getPoolPartitions

`public int getPoolPartitions()`

Gets the number of partitions in each PartitionedPool.

Returns:
The number of partitions in each PartitionedPool.

getHighAvailabilityCoordinators

`public String getHighAvailabilityCoordinators()`

Gets the coordinators address and port list.

Returns:
The coordinators address and port list.

getRecoveryCheckpointTime

`public long getRecoveryCheckpointTime()`

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

Returns:
The delay time (in microseconds) between automatic checkpoints.

setLogFile

`public void setLogFile(String filePath)`

Sets the log file.

Parameters:
`filePath` - [in] The log file.

setExtentPages

`public void setExtentPages(int pages)`

Sets the number of pages per extent.

Parameters:

(continued from last page)

`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.

setClientId

`public void` **`setClientId`**`(String clientId)`

Set the client identifier.

If you don't have a client identifier, you may want to register at http://sparsity-technologies.com/#sparksee

Parameters:

`clientId` - [in] The client identifier.

setLogLevel

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

Sets the log level.

Parameters:

`level` - [in] The LogLevel.

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.

getChecksumEnabled

`public boolean` **`getChecksumEnabled`**`()`

Gets whether the storage checksum usage is enabled or disabled.

Returns:

TRUE if the checksum is enabled, FALSE otherwise.

getHighAvailabilitySynchronization

`public long` **`getHighAvailabilitySynchronization`**`()`

Gets the synchronization polling time.

Returns:

The Synchronization polling time.

(continued from last page)

getHighAvailabilityMasterHistory

`public long getHighAvailabilityMasterHistory()`

Gets the master's history log.

Returns:
The master's history log.

getLicenseId

`public String getLicenseId()`

Gets the license identifier.

Returns:
The license identifier.

getAESIV

`public String getAESIV()`

Get the AES initialization vector in a hexadecimal encoded string.

Returns:
The AES initialization vector as an hexadecimal string.

getEncryptionEnabled

`public boolean getEncryptionEnabled()`

Gets whether the storage encryption is enabled or disabled.

Returns:
TRUE if the encryption is enabled, FALSE otherwise.

setPoolPartitions

`public void setPoolPartitions(int pools)`

Sets the number of pools in each PartitionedPool.

Parameters:
`pools` - [in] The number of pools in each PartitionedPool. It must be non-negative.

getRollbackEnabled

`public boolean getRollbackEnabled()`

(continued from last page)

Gets whether the rollback is enabled or disabled.

Returns:

TRUE if the rollback is enabled, FALSE otherwise.

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.

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.

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.

setTmpFolder

```
public void setTmpFolder(String tmpFolder)
```

Sets the temporary folder used for temporary staging.

Parameters:

`tmpFolder` - The temporary folder to set

setCacheStatisticsSnapshotTime

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

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters:

(continued from last page)

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

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.

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.

downloadExpected

`public boolean` **`downloadExpected`**`()`

Check if a new license will be automatically downloaded with the current settings.

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 void	clear() Remove all the properties.
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.
static void	setHI(String value) YOU SHOULD NOT USE THIS METHOD.

Methods inherited from class `java.lang.Object`

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

Methods

(continued from last page)

clear

```
public static void clear()
```

Remove all the properties.

---

get

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

Gets a property.

Parameters:

`key` - [in] The name of the property to lookup.
`def` - [in] Default value to be returned in case there is no property with the name key.

Returns:

The value of the property, or def if the key is not found.

---

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.

---

setHI

```
public static void setHI(String value)
```

YOU SHOULD NOT USE THIS METHOD.

This method exists only for a specific service.

Parameters:

`value` - null

---

load

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

Loads properties from the given file path.

(continued from last page)

Parameters:

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

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 begining or at the end are ignored. No blanks are allowed between the time duration and the unit time.

Allowed time units: 'D' for days, 'H' for hours, 'M' for minutes, 'S' o 's' for seconds, 'm' for milliseconds and 'u' for microseconds.

There is a special case: If no time unit is given, seconds is the default. So, '10' means 10 seconds.

Parameters:

`key` - [in] The name of the property to lookup.

`def` - [in] The default value (in microseconds) to be returned in case there is no property with the name key.

Returns:

The time duration in microseconds, or def if the key is not found or in case of error.

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.

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

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`

`forEach, iterator, spliterator`

Constructors

StringList

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

Creates a new instance from an string array.

Parameters:
`list` - String array to initialize the instance.

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.

Methods

clear

```
public void clear()
```

Clears the list.

iterator

```
public StringListIterator iterator()
```

Gets a new StringListIterator.

Returns:
StringListIterator instance.

add

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

(continued from last page)

Adds a String at the end of the list.

Parameters:

`str` - [in] String.

---

count

```
public int count()
```

Number of elements in the list.

Returns:

Number of elements in the list.

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`

`forEachRemaining, hasNext, next, remove`

Methods

next

`public String next()`

See nextString().

nextString

```
public String nextString()
```

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.

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

Methods inherited from interface java.lang.AutoCloseable
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

write

```
public void write(char[] dataIN,
         int length)
```

Write data to the stream.

length[in] Length of the data buffer. It must be > 0.

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.

read

```
public int read(char[] dataOUT,
         int length)
```

(continued from last page)

Read data from the stream.

length[in] Length of the given data buffer. It must be > 0. Amount of read data (<= length). If 0, there is no more data to be read 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

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

InvalidType

`public static int InvalidType`

Invalid type identifier constant.

NodesType

`public static int NodesType`

Identifier for all nodeType attributes.

Identifier for all nodeType attributes.

GlobalType

`public static int GlobalType`

Global type identifier constant.

EdgesType

`public static int EdgesType`

Identifier for all edgeType attributes.

Identifier for all edgeType attributes.

Methods

getObjectType

`public ObjectType getObjectType()`

Gets the object type.

Returns:

The object type.

(continued from last page)

getId

`public int getId()`

Gets the Sparksee type identifier.

Returns:
The Sparksee type identifier.

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.

getNumObjects

`public long getNumObjects()`

Gets the number of objects belonging to the type.

Returns:
The number of objects belonging to the type.

getAreNeighborsIndexed

`public boolean getAreNeighborsIndexed()`

Gets if this is an edge type with neighbors index.

Returns:
TRUE for edges types with neighbors index, FALSE otherwise.

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()`

(continued from last page)

Gets if this is a restricted edge type.

Returns:
TRUE for restricted edge types, FALSE otherwise.

getName

`public String getName()`

Gets the unique type name.

Returns:
The unique type name.

getIsDirected

`public boolean getIsDirected()`

Gets if this is a directed edge type.

Returns:
TRUE for directed edge types, FALSE otherwise.

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

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`

`forEach, iterator, spliterator`

Constructors

TypeList

```
public TypeList()
```

Constructor.

This creates an empty list.

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.

Methods

clear

```
public void clear()
```

Clears the list.

iterator

```
public TypeListIterator iterator()
```

Gets a new TypeListIterator.

Returns:

TypeListIterator 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 type)
```

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

Parameters:

`type` - [in] Sparksee type identifier.

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`

`forEachRemaining, hasNext, next, remove`

Methods

next

`public Integer next()`

See nextType().

nextType

```
public int nextType()
```

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.

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() Creates a new instance.
public	Value(Value value) Copy constructor.

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	getInteger() Gets Integer Value.
long	getLong() Gets Long Value.
long	getOID() Gets OID Value.
String	getString() Gets String Value.
long	getTimestamp() Gets Timestamp Value.
Calendar	getTimestampAsCalendar() Gets the Value as a Calendar instance.
Date	getTimestampAsDate() Gets the Value as a Date instance.
int	hashCode()
boolean	isNull() Gets if this is a NULL Value.
Value	set(Value value) Sets the Value.
Value	setBoolean(boolean value) Sets the Value.
void	setBooleanVoid(boolean value) Sets the Value.
Value	setDouble(double value) Sets the Value.
void	setDoubleVoid(double value) Sets the Value.
Value	setInteger(int value) Sets the Value.
void	setIntegerVoid(int value) Sets the Value.
Value	setLong(long value) Sets the Value.
void	setLongVoid(long value) Sets the Value.
Value	setNull() Sets the Value to NULL.
void	setNullVoid() 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()`

Creates a new instance.

It creates a NULL Value.

Value

`public Value(Value value)`

Copy constructor.

Parameters:
`value` - [in] Value to be copied.

Methods

equals

`public boolean equals(Object other)`

Parameters:
`other` - null

getTimestamp

`public long getTimestamp()`

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

Returns:
The Timestamp Value.

getLong

`public long getLong()`

Gets Long Value.

This must be a non-NULL Long Value.

Returns:
The Long Value.

setNull

`public Value setNull()`

(continued from last page)

Sets the Value to NULL.

Returns:
The calling instance.

setInteger

`public Value setInteger(int value)`

Sets the Value.

Parameters:
`value` - New value.

Returns:
The calling instance.

setDoubleVoid

`public void setDoubleVoid(double value)`

Sets the Value.

Parameters:
`value` - [in] New 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.

setOID

`public Value setOID(long 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.

getTimestampAsCalendar

```
public Calendar getTimestampAsCalendar()
```

Gets the Value as a Calendar instance.

Returns:
The returning Calendar instance.

setDouble

```
public Value setDouble(double 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.

setTimestampVoid

```
public void setTimestampVoid(int year,
         int month,
         int day,
         int hour,
         int minutes,
         int seconds,
         int millisecs)
```

Sets the Value.

year[in] The year (>=1970).

(continued from last page)

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]).

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.

setBooleanVoid

```
public void setBooleanVoid(boolean value)
```

Sets the Value.

Parameters:

`value` - [in] New Boolean value.

hashCode

```
public int hashCode()
```

setOIDVoid

```
public void setOIDVoid(long value)
```

Sets the OID Value.

Parameters:

`value` - [in] New OID value.

compareTo

```
public int compareTo(Value value)
```

(continued from last page)

See compare().

Parameters:

`value` - null

---

setLongVoid

```
public void setLongVoid(long value)
```

Sets the Value.

Parameters:

`value` - [in] New Long value.

---

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.

---

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.

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.

(continued from last page)

getOID

`public long getOID()`

Gets OID Value.

This must be an non-NULL OID Value.

Returns:
The OID Value.

getBoolean

`public boolean getBoolean()`

Gets Boolean Value.

This must be a non-NULL Boolean Value.

Returns:
The Boolean Value.

set

`public Value set(Value value)`

Sets the Value.

Parameters:
`value` - New value.

Returns:
The calling instance.

setTimestamp

`public Value setTimestamp(Calendar value)`

Sets the Value.

Parameters:
`value` - New value.

Returns:
The calling instance.

getDataType

`public DataType getDataType()`

Gets the DataType.

Value cannot be NULL.

Returns:
The DataType.

setTimestamp

`public Value setTimestamp(Date value)`

Sets the Value.

Parameters:
`value` - New value.

Returns:
The calling instance.

compareTo

`public int compareTo(Object value)`

See compare().

This just works if the given object is a Value instance.

Parameters:
`value` - null

setLong

`public Value setLong(long 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.

getDouble

`public double getDouble()`

(continued from last page)

Gets Double Value.

This must be a non-NULL Double Value.

Returns:
The Double Value.

setStringVoid

```
public void setStringVoid(String value)
```

Sets the Value.

Parameters:
`value` - [in] New String value.

setVoid

```
public void setVoid(Value value)
```

Sets the Value.

Parameters:
`value` - [in] New value.

setTimestampVoid

```
public void setTimestampVoid(long value)
```

Sets the Value.

Parameters:
`value` - [in] New Timestamp value.

toString

```
public String toString()
```

Gets a String representation of the Value.

setIntegerVoid

```
public void setIntegerVoid(int value)
```

Sets the Value.

Parameters:
`value` - [in] New Integer value.

(continued from last page)

isNull

```
public boolean isNull()
```

Gets if this is a NULL Value.

Returns:
TRUE if this is a NULL Value, FALSE otherwise.

---

getString

```
public String getString()
```

Gets String Value.

This must be a non-NULL String Value.

Returns:
The String Value.

---

setNullVoid

```
public void setNullVoid()
```

Sets the Value to NULL.

---

setBoolean

```
public Value setBoolean(boolean value)
```

Sets the Value.

Parameters:
`value` - New value.

Returns:
The calling instance.

com.sparsity.sparksee.gdb
Class ValueArray

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

All Implemented Interfaces:
Closeable

---

public class **ValueArray**
extends Object
implements Closeable

ValueArray class.

It allows for getting and setting ValueArray attribute values.

It is very important to close the ValueArray once no more get or set operations will be performed.

Creation of a new ValueArray: (i) Set all the ValueArray elements using Graph::SetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Use of an existing ValueArray: (i) Get a ValueArray instance using Graph::GetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.
Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

void	close() Closes the ValueArray instance.
void	get(int index, Value value) Get a Value from the array.
boolean[]	getBoolean() Get all the Values from this boolean array
boolean[]	getBoolean(int index, int size) Get a subset of the Values from this boolean array
double[]	getDouble() Get all the Values from this double array
double[]	getDouble(int index, int size) Get a subset of the Values from this double array
int[]	getInteger() Get all the Values from this int array
int[]	getInteger(int index, int size) Get a subset of the Values from this int array

long[]	getLong() Get all the Values from this long array
long[]	getLong(int index, int size) Get a subset of the Values from this long array
long[]	getOID() Get all the Values from this OID array
long[]	getOID(int index, int size) Get a subset of the Values from this long array
long[]	getTimestamp() Get all the Values from this timestamp array
long[]	getTimestamp(int index, int size) Get a subset of the Values from this timestamp array
boolean	isClosed() Gets if ValueArray instance has been closed or not.
void	makeNull() Sets the attribute array to Null.
void	set(int index, Value value) Set a Value to a specific array position.
void	set(Value value) Set a Value to the whole array.
void	setBoolean(boolean[] values) Set all the values of this bool array.
void	setBoolean(int index, boolean[] values) Set a subset of the values of this bool array.
void	setDouble(double[] values) Set all the values of this double array.
void	setDouble(int index, double[] values) Set a subset of the values of this double array.
void	setInteger(int[] values) Set all the values of this int array.
void	setInteger(int index, int[] values) Set a subset of the values of this int array.
void	setLong(int index, long[] values) Set a subset of the values of this long array.
void	setLong(long[] values) Set all the values of this long array.
void	setOID(int index, long[] values) Set a subset of the values of this oid array.
void	setOID(long[] values) Set all the values of this oid array.

void	setTimestamp(int index, long[] values) Set a subset of the values of this timestamp array.
void	setTimestamp(long[] values) Set all the values of this timestamp array.
int	size() Returns the array size.

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.lang.AutoCloseable

close

Methods

getInteger

```
public int[] getInteger(int index,
         int size)
```

Get a subset of the Values from this int array

Parameters:

index - Position of the first element to get [0..N-1]
size - Number of elements to get [1..N]

Returns:

Returns all the requested values

getTimestamp

```
public long[] getTimestamp(int index,
         int size)
```

Get a subset of the Values from this timestamp array

Parameters:

index - Position of the first element to get [0..N-1]
size - Number of elements to get [1..N]

Returns:

Returns all the requested values

set

```
public void set(int index,
         Value value)
  throws RuntimeException,
         RuntimeException
```

Set a Value to a specific array position.

AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the element to set [0..N-1]
`value` - [in] Value to set in the array element

Throws:

`java.lang.RuntimeException` - If the index is out of range or the Value not valid
`java.lang.RuntimeException` - null

setBoolean

```
public void setBoolean(int index,
         boolean[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this bool array.

UnsupportedOperationErrorIf array DataType is not bool AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

setTimestamp

```
public void setTimestamp(long[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this timestamp array.

AppErrorIf the ValueArray is not available

Parameters:

`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If array DataType is not timestamp

(continued from last page)

setOID

```
public void setOID(long[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this oid array.

AppErrorIf the ValueArray is not available

Parameters:

`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If array DataType is not oid

---

get

```
public void get(int index,
         Value value)
  throws RuntimeException,
         RuntimeException
```

Get a Value from the array.

AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the element to get [0..N-1]
`value` - [out] Value to get the array element

Throws:

`java.lang.RuntimeException` - If the index is out of range or the Value not valid
`java.lang.RuntimeException` - null

---

setTimestamp

```
public void setTimestamp(int index,
         long[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this timestamp array.

UnsupportedOperationErrorIf array DataType is not timestamp AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

---

getOID

```
public long[] getOID()
```

(continued from last page)

Get all the Values from this OID array

Returns:

Returns all the values

---

setInteger

```
public void setInteger(int index,
         int[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this int array.

UnsupportedOperationErrorIf array DataType is not int AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

---

getLong

```
public long[] getLong(int index,
         int size)
```

Get a subset of the Values from this long array

Parameters:

`index` - Position of the first element to get [0..N-1]
`size` - Number of elements to get [1..N]

Returns:

Returns all the requested values

---

getBoolean

```
public boolean[] getBoolean(int index,
         int size)
```

Get a subset of the Values from this boolean array

Parameters:

`index` - Position of the first element to get [0..N-1]
`size` - Number of elements to get [1..N]

Returns:

Returns all the requested values

close

```
public void close()
```

Closes the ValueArray instance.

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

getTimestamp

```
public long[] getTimestamp()
```

Get all the Values from this timestamp array

Returns:
Returns all the values

getBoolean

```
public boolean[] getBoolean()
```

Get all the Values from this boolean array

Returns:
Returns all the values

setBoolean

```
public void setBoolean(boolean[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this bool array.

AppErrorIf the ValueArray is not available

Parameters:
`values` - [in] All the values to set

Throws:
`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If array DataType is not bool

getInteger

```
public int[] getInteger()
```

Get all the Values from this int array

Returns:
Returns all the values

(continued from last page)

getDouble

```
public double[] getDouble(int index,
         int size)
```

Get a subset of the Values from this double array

Parameters:

`index` - Position of the first element to get [0..N-1]
`size` - Number of elements to get [1..N]

Returns:

Returns all the requested values

---

size

```
public int size()
```

Returns the array size.

---

setDouble

```
public void setDouble(int index,
         double[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this double array.

UnsupportedOperationErrorIf array DataType is not Double AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

---

makeNull

```
public void makeNull()
  throws RuntimeException
```

Sets the attribute array to Null.

The ValueArray can not be used after this call.

---

getOID

```
public long[] getOID(int index,
         int size)
```

(continued from last page)

Get a subset of the Values from this long array

Parameters:

`index` - Position of the first element to get [0..N-1]
`size` - Number of elements to get [1..N]

Returns:

Returns all the requested values

---

setLong

```
public void setLong(int index,
         long[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this long array.

UnsupportedOperationErrorIf array DataType is not long AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

---

getDouble

```
public double[] getDouble()
```

Get all the Values from this double array

Returns:

Returns all the values

---

setOID

```
public void setOID(int index,
         long[] values)
  throws RuntimeException,
         RuntimeException,
         RuntimeException
```

Set a subset of the values of this oid array.

UnsupportedOperationErrorIf array DataType is not oid AppErrorIf the ValueArray is not available

Parameters:

`index` - [in] Position of the first element to set [0..N-1]
`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null

(continued from last page)

`java.lang.RuntimeException` - If the index or size is out of range
`java.lang.RuntimeException` - null

isClosed

```
public boolean isClosed()
```

Gets if ValueArray instance has been closed or not.

Returns:
TRUE if the ValueArray instance has been closed, FALSE otherwise.

See Also:
`close()`

setLong

```
public void setLong(long[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this long array.

AppErrorIf the ValueArray is not available

Parameters:
`values` - [in] All the values to set

Throws:
`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If array DataType is not long

setDouble

```
public void setDouble(double[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this double array.

AppErrorIf the ValueArray is not available

Parameters:
`values` - [in] All the values to set

Throws:
`java.lang.RuntimeException` - null
`java.lang.RuntimeException` - If array DataType is not Double

setInteger

```
public void setInteger(int[] values)
  throws RuntimeException,
         RuntimeException
```

Set all the values of this int array.

AppErrorIf the ValueArray is not available

(continued from last page)

Parameters:

`values` - [in] All the values to set

Throws:

`java.lang.RuntimeException` - null

`java.lang.RuntimeException` - If array DataType is not int

---

set

```
public void set(Value value)
  throws RuntimeException,
         RuntimeException
```

Set a Value to the whole array.

AppErrorIf the ValueArray is not available

Parameters:

`value` - [in] Value to set in all the array elements

Throws:

`java.lang.RuntimeException` - If the Value is not valid

`java.lang.RuntimeException` - null

---

getLong

```
public long[] getLong()
```

Get all the Values from this long array

Returns:

Returns all the values

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

clear

`public void clear()`

Clears the list.

---

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.

---

iterator

`public ValueListIterator iterator()`

Gets a new ValueListIterator.

Returns:

ValueListIterator instance.

---

add

`public void add(Value value)`

Adds a value to the end of the list.

Parameters:

`value` - [in] The value to add

---

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

next

`public Value next()`

Moves to the next element.

Returns:
The next element.

hasNext

`public boolean hasNext()`

Gets if there are more elements.

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

com.sparsity.sparksee.gdb
Class Values

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

All Implemented Interfaces:
Iterable, Closeable

---

public class **Values**
extends Object
implements Closeable, Iterable

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.io.Closeable

close

Methods inherited from interface java.lang.AutoCloseable

close

Methods inherited from interface java.lang.Iterable

forEach, iterator, spliterator

Methods

count

`public long count()`

Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

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()`

iterator

`public ValuesIterator iterator(Order order)`

Gets a ValuesIterator.

Parameters:
`order` - [in] Ascending or descending order.

Returns:
ValuesIterator instance.

iterator

`public ValuesIterator iterator()`

See iterator().

Creates an Ascendent iterator.

close

`public void close()`

(continued from last page)

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.lang.AutoCloseable

close

Methods inherited from interface java.util.Iterator

forEachRemaining, hasNext, next, remove

Methods

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()`

---

next

`public Value next()`

Gets the next element to traverse.

Returns:
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.

---

close

`public void close()`

Closes the ValuesIterator instance.

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

---

remove

`public void remove()`

Operation not supported.

Package
com.sparsity.sparksee.io

com.sparsity.sparksee.io
Class CSVLoader

```
java.lang.Object
   |
   +-com.sparsity.sparksee.io.CSVLoader
```

public class **CSVLoader**
extends Object

Method Summary

static void	loadEdges(Graph graph, String fileName, String edgeType, String tailNodeType, String headNodeType, int tail, int head, String separator, boolean directed, boolean header, MissingEndpoint onMissingTail, MissingEndpoint onMissingHead, int[] columns, String[] attrNames, DataType[] dataTypes, AttributeKind[] attrKinds) Loads edges from a CSV file.
static void	loadNodes(Graph graph, String fileName, String nodeType, String separator, boolean header, int[] columns, String[] attrNames, DataType[] dataTypes, AttributeKind[] attrKinds) Loads nodes from a CSV file.

Methods inherited from class `java.lang.Object`

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

Methods

loadEdges

```
public static void loadEdges(Graph graph,
         String fileName,
         String edgeType,
         String tailNodeType,
         String headNodeType,
         int tail,
         int head,
         String separator,
         boolean directed,
         boolean header,
         MissingEndpoint onMissingTail,
         MissingEndpoint onMissingHead,
         int[] columns,
         String[] attrNames,
         DataType[] dataTypes,
         AttributeKind[] attrKinds)
```

Loads edges from a CSV file.

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

Parameters:

(continued from last page)

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

loadNodes

```
public static void loadNodes(Graph graph,
          String fileName,
          String nodeType,
          String separator,
          boolean header,
          int[] columns,
          String[] attrNames,
          DataType[] dataTypes,
          AttributeKind[] attrKinds)
```

Loads nodes from a CSV file.

Parameters:

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

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	`open(String filePath)` Opens the source file path.
boolean	`read(StringList row)` Reads the next row as a string array.
boolean	`reset()` Moves the reader to the beginning.
void	`setLocale(String localeStr)` Sets the locale that will be used to read the file.
void	`setMultilines(int numExtralines)` Allows the use of fields with more than one line.
void	`setNumLines(int numLines)` Used to limit the number of lines that will be read.
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	`setSingleLine()` Only allows single line fields.
void	`setStartLine(int startLine)` Sets the number of lines to be skiped 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

setSeparator

```
public void setSeparator(String sep)
  throws RuntimeException
```

(continued from last page)

Sets the character used to separate fields in the file.

Parameters:

`sep` - [in] Separator character.

Throws:

`java.lang.RuntimeException` - null

---

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)

---

close

```
public void close()
  throws IOException
```

Closes the reader.

---

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.

---

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.

(continued from last page)

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).

---

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.

---

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.

---

setStartLine

```
public void setStartLine(int startLine)
```

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

Parameters:

`startLine` - [in] The line number to skip for start reading

---

setSingleLine

```
public void setSingleLine()
```

Only allows single line fields.

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.

setQuotes

```
public void setQuotes(String quotes)
  throws RuntimeException
```

Sets the character used to quote fields.

Parameters:
`quotes` - [in] Quote character.

Throws:
`java.lang.RuntimeException` - null

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	write(StringList row) 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

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

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.

open

```
public void open(String f)
  throws IOException
```

(continued from last page)

Opens the output file path.

Parameters:

`f` - [in] Output file path.

Throws:

`java.io.IOException` - If bad things happen opening the file.

write

```
public void write(StringList row)
  throws RuntimeException,
         IOException
```

Writes the next row.

Parameters:

`row` - [in] Row of data.

Throws:

`java.lang.RuntimeException` - null
`java.io.IOException` - If bad things happen during the write.

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 RuntimeException,
         IOException
```

Closes the writer.

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.

(continued from last page)

setQuotes

```
public void setQuotes(String quotes)
  throws RuntimeException
```

Sets the character used to quote fields.

Parameters:

`quotes` - [in] Quote character.

Throws:

`java.lang.RuntimeException` - null

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()` Creates a new instance.
public	`EdgeTypeExporter(RowWriter rowWriter, Graph graph, int type, AttributeList attrs, int hPos, int tPos, int hAttr, int tAttr)` 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()
```

Creates a new instance.

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.

(continued from last page)

Methods

run

```
public void run()
  throws RuntimeException,
         IOException
```

See the TypeExporter class Run method.

---

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

`type` - [in] Type identifier.

---

setHeadPosition

```
public void setHeadPosition(int pos)
```

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

---

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

---

(continued from last page)

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

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

setTailPosition

```
public void setTailPosition(int pos)
```

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

Parameters:

`pos` - [in] Tail position

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.

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

`rw` - [in] Input RowWriter.

setGraph

```
public void setGraph(Graph graph)
```

(continued from last page)

Sets the graph that will be exported.

Parameters:

`graph` - [in] Graph.

register

```
public void register(TypeExporterListener tel)
```

Registers a new listener.

Parameters:

`tel` - [in] TypeExporterListener to be registered.

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()` Creates a new instance.
public	`EdgeTypeLoader(RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr, MissingEndpoint headMEP, MissingEndpoint tailMEP)` 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	`setHeadAttribute(int attr)` Sets the attribute that will be used to find the head of the edge.
void	`setHeadMEP(MissingEndpoint mep)` Sets the flag to create missing head nodes when loading nodes or edges.
void	`setHeadPosition(int pos)` Sets the position of the head attribute in the source data.
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	`setTailAttribute(int attr)` Sets the attribute that will be used to find the tail of the edge.
void	`setTailMEP(MissingEndpoint mep)` Sets the flag to create missing tail nodes when loading nodes or edges.
void	`setTailPosition(int pos)` Sets the position of the tail attribute in the source data.
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

EdgeTypeLoader

`public` **`EdgeTypeLoader`**`()`

Creates a new instance.

EdgeTypeLoader

```
public EdgeTypeLoader(RowReader rowReader,
                      Graph graph,
                      int type,
                      AttributeList attrs,
                      Int32List attrsPos,
                      int hPos,
                      int tPos,
                      int hAttr,
                      int tAttr,
                      MissingEndpoint headMEP,
                      MissingEndpoint tailMEP)
```

Creates a new instance.

attrsPos[in] Attribute positions (column index >=0). to all listeners.

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.
`headMEP` - [in] The MissingEndpoint policy for the head nodes
`tailMEP` - [in] The MissingEndpoint policy for the tail nodes

Methods

run

```
public void run()
  throws RuntimeException,
         IOException
```

See the TypeLoader class Run method.

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.

(continued from last page)

setType

`public void setType(int type)`

Sets the type to be loaded.

Parameters:
`type` - [in] Type identifier.

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

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.

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

register

`public void register(TypeLoaderListener tel)`

(continued from last page)

Registers a new listener.

Parameters:

`tel` - TypeLoaderListener to be registered.

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

runTwoPhases

```
public void runTwoPhases()
  throws RuntimeException,
         IOException
```

See the TypeLoader class RunTwoPhases method.

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

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

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.

Parameters:
`rr` - [in] Input RowReader.

setTailMEP

```
public void setTailMEP(MissingEndpoint mep)
```

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

flagTrue if the nodes need to be created. False otherwise

Parameters:
`mep` - null

setAttributePositions

```
public void setAttributePositions(Int32List attrsPos)
```

Sets the list of attribute positions.

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

Parameters:
`attrsPos` - [in] Attribute positions (column index >=0).

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph where the data will be loaded.

Parameters:
`graph` - [in] Graph.

runNPhases

```
public void runNPhases(int partitions)
  throws RuntimeException,
         IOException
```

See the TypeLoader class RunNPhases method.

Parameters:
`partitions` - null

Throws:
`java.lang.RuntimeException` - null

(continued from last page)

`java.io.IOException` - null

setHeadMEP

`public void` **`setHeadMEP`**`(MissingEndpoint mep)`

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

flagTrue if the nodes need to be created. False otherwise

Parameters:

`mep` - null

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.

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()` Creates a new instance.
public	`NodeTypeExporter(RowWriter rowWriter, Graph graph, int type, AttributeList attrs)` 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()
```

Creates a new instance.

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.

Methods

run

```
public void run()
  throws RuntimeException,
         IOException
```

See the TypeExporter class Run method.

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

- `type` - [in] Type identifier.

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.

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

`rw` - [in] Input RowWriter.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph that will be exported.

Parameters:

`graph` - [in] Graph.

register

```
public void register(TypeExporterListener tel)
```

Registers a new listener.

Parameters:

`tel` - [in] TypeExporterListener to be registered.

setAttributes

```
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

`attrs` - [in] Attribute identifiers to be exported

setFrequency

```
public void setFrequency(int freq)
```

(continued from last page)

Sets the frequency of listener notification.

Parameters:

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

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()` Creates a new instance.
public	`NodeTypeLoader(RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos)` 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() 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 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()
```

Creates a new instance.

NodeTypeLoader

```
public NodeTypeLoader(RowReader rowReader,
                      Graph graph,
                      int type,
                      AttributeList attrs,
                      Int32List attrsPos)
```

Creates a new instance.

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

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).

Methods

run

```
public void run()
  throws RuntimeException,
         IOException
```

See the TypeLoader class Run method.

---

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.

---

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.

---

setTimestampFormat

```
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

Parameters:

(continued from last page)

`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

register

```
public void register(TypeLoaderListener tel)
```

Registers a new listener.

Parameters:

`tel` - TypeLoaderListener to be registered.

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

runTwoPhases

```
public void runTwoPhases()
  throws RuntimeException,
         IOException
```

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)
```

(continued from last page)

Sets the list of attribute positions.

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

Parameters:

`attrsPos` - [in] Attribute positions (column index >=0).

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph where the data will be loaded.

Parameters:

`graph` - [in] Graph.

runNPhases

```
public void runNPhases(int partitions)
  throws RuntimeException,
         IOException
```

See the TypeLoader class RunNPhases method.

Parameters:

`partitions` - null

Throws:

`java.lang.RuntimeException` - null

`java.io.IOException` - null

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.

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.

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.

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 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

close

```
public void close()
  throws RuntimeException,
         IOException
```

Closes the writer.

---

write

```
public void write(StringList row)
  throws RuntimeException,
         IOException
```

(continued from last page)

Writes the next row.

Parameters:

`row` - [in] Row of data.

Throws:

`java.lang.RuntimeException` - null

`java.io.IOException` - If bad things happen during the write.

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

run

```
public void run()
  throws RuntimeException,
         IOException
```

Runs export process.

setType

```
public void setType(int type)
```

Sets the type to be exported.

Parameters:

`type` - [in] Type identifier.

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.

setRowWriter

```
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

Parameters:

`rw` - [in] Input RowWriter.

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph that will be exported.

Parameters:

`graph` - [in] Graph.

(continued from last page)

register

`public void register(TypeExporterListener tel)`

Registers a new listener.

Parameters:

`tel` - [in] TypeExporterListener to be registered.

setAttributes

`public void setAttributes(AttributeList attrs)`

Sets the list of Attributes.

Parameters:

`attrs` - [in] Attribute identifiers to be exported

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

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

getTypeId

```
public int getTypeId()
```

Gets the type identifier.

Returns:
The type identifier.

getTotal

```
public long getTotal()
```

(continued from last page)

Gets the total number of objects exported.

Returns:
The total number of objects exported.

getCount

```
public long getCount()
```

Gets the current number of objects exported.

Returns:
The current number of objects exported.

isLast

```
public boolean isLast()
```

Gets if this is the last event or not.

Returns:
TRUE if this is the last event, FALSE otherwise.

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

run

```
public void run()
  throws RuntimeException,
         IOException
```

Run the loader.

runTwoPhases

```
public void runTwoPhases()
  throws RuntimeException,
         IOException
```

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.

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.

(continued from last page)

setType

`public void setType(int type)`

Sets the type to be loaded.

Parameters:
`type` - [in] Type identifier.

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.

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

register

`public void register(TypeLoaderListener tel)`

Registers a new listener.

Parameters:
`tel` - TypeLoaderListener to be registered.

setFrequency

`public void setFrequency(int freq)`

(continued from last page)

Sets the frequency of listener notification.

Parameters:

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

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.

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

Parameters:

`attrsPos` - [in] Attribute positions (column index >=0).

runNPhases

```
public void runNPhases(int partitions)
  throws RuntimeException,
         IOException
```

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.lang.RuntimeException` - null

`java.io.IOException` - null

setGraph

```
public void setGraph(Graph graph)
```

Sets the graph where the data will be loaded.

Parameters:

`graph` - [in] Graph.

(continued from last page)

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.

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

getTypeId

`public int` **`getTypeId`**`()`

(continued from last page)

Gets the type identifier.

Returns:
The type identifier.

getPhase

`public int getPhase()`

Gets the current phase.

Returns:
The current phase.

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.

getPartition

`public int getPartition()`

Gets the current partition.

Returns:
The current partition.

getTotalPartitions

`public int getTotalPartitions()`

Gets the total number of partitions.

Returns:

(continued from last page)

The total number of partitions.

isLast

```
public boolean isLast()
```

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

getTotalPhases

```
public int getTotalPhases()
```

Gets the total number of phases.

Returns:

The total number of phases.

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, wait`

Constructors

(continued from last page)

ScriptParser

```
public ScriptParser()
```

Constructor.

Methods

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 writen.
`db` - [in] Database.

Throws:

`java.io.IOException` - If bad things happen opening or writing the file.

---

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:

(continued from last page)

TRUE if ok, FALSE in case of error.

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.

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.

Index

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y