Package
com.sparsity.sparksee.algorithms
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

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ConnectedComponents(Session s, String materializedattribute)</td>
</tr>
<tr>
<td>Creates a new instance of ConnectedComponents.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>Closes the ConnectedComponents instance.</td>
</tr>
<tr>
<td>long getConnectedComponent(long idNode)</td>
</tr>
<tr>
<td>Returns the connected component where the given node belongs to.</td>
</tr>
<tr>
<td>long getCount()</td>
</tr>
<tr>
<td>Returns the number of connected components found in the graph.</td>
</tr>
<tr>
<td>Objects getNodes(long idConnectedComponent)</td>
</tr>
<tr>
<td>Returns the collection of nodes contained in the given connected component.</td>
</tr>
<tr>
<td>long getSize(long idConnectedComponent)</td>
</tr>
<tr>
<td>Returns the number of nodes contained in the given connected component.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
</tr>
<tr>
<td>Gets if ConnectedComponents instance has been closed or not.</td>
</tr>
</tbody>
</table>
Constructors

**ConnectedComponents**

```java
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**

```java
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**

```java
public long getCount()
```

Returns the number of connected components found in the graph.

**Returns:**

The number of connected components found in the graph.
getConnectedComponent

public long getConnectedComponent(long idNode)

Returns the connected component where the given node belongs to.

Parameters:
   idNode - [in] The node identifier for which the connected component identifier where it belongs will be returned.

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

getNodes

public Objects getNodes(long idConnectedComponent)

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

Parameters:
   idConnectedComponent - The connected component for which the collection of nodes contained in it will be returned.

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

isClosed

public boolean isClosed()

Gets if ConnectedComponents instance has been closed or not.

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

See Also:
   close()

close

public void close()

Closes the ConnectedComponents instance.

It must be called to ensure the integrity of all data.
com.sparsity.sparksee.algorithms
Class Connectivity

java.lang.Object
   +-com.sparsity.sparksee.algorithms.Connectivity

All Implemented Interfaces:
   Closeable

Direct Known Subclasses:
   WeakConnectivity, StrongConnectivity

public class Connectivity
extends Object
implements Closeable

Connectivity class.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllNodeTypes()</td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void close()</td>
<td>Closes the Connectivity instance.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>ConnectedComponents getConnectedComponents()</td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Connectivity instance has been closed or not.</td>
</tr>
<tr>
<td>void run()</td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
<tr>
<td>void setMaterializedAttribute(String attributeName)</td>
<td>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.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object

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

Methods inherited from interface java.io.Closeable

close

Methods

addAllNodeTypes

public void addAllNodeTypes()

Allows connectivity through all node types of the graph.

run

public void run()

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

public void excludeNodes(Objects nodes)

Set which nodes can't be used.

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

Parameters:

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

addNodeType

public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:

t - null

setMaterializedAttribute

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

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

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

Parameters:

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

**excludeEdges**

public void **excludeEdges**(Objects edges)

Set which edges can't be used.

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

Parameters:

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

**isClosed**

public boolean **isClosed**()

Gets if Connectivity instance has been closed or not.

**Returns:**

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

**See Also:**

**close**

**close**

public void **close**()

Closes the Connectivity instance.

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

**getConnectedComponents**

public **ConnectedComponents** **getConnectedComponents**()

Returns the results generated by the execution of the algorithm.

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

**Returns:**

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

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
</tr>
<tr>
<td>Context</td>
</tr>
<tr>
<td>(Session session, long node)</td>
</tr>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td>addAllEdgeTypes (EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>addAllNodeTypes ()</td>
</tr>
<tr>
<td>Allows for traversing all node types of the graph.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>addEdgeType (int t, EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>addNodeType (int t)</td>
</tr>
<tr>
<td>Allows for traversing nodes of the given type.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>close ()</td>
</tr>
<tr>
<td>Closes the Context instance.</td>
</tr>
<tr>
<td>static Objects</td>
</tr>
<tr>
<td>compute ()</td>
</tr>
<tr>
<td>Gets the resulting collection of nodes.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>compute (Session session, long node, Typelist nodeTypes, Typelist edgetypes, EdgesDirection dir, int maxhops, boolean include)</td>
</tr>
<tr>
<td>Helper method to easily compute a context from a node.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>excludeEdges (Objects edges)</td>
</tr>
<tr>
<td>Set which edges can’t be used.</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>void</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface `java.io.Closeable`
- `close`

Constructors

**Context**

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

Creates a new instance.

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

Methods

**addEdgeType**

```java
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**

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

**Returns:**
Returns an Objects with the computed context of a node.

### excludeNodes

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

Set which nodes can't be used.

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

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

### compute

```java
public Objects compute()
```

Gets the resulting collection of nodes.

**Returns:**
The resulting collection of nodes.

### addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

**Parameters:**
- `d` - [in] Edge direction.

### addNodeType

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

Allows for traversing nodes of the given type.
close
public void close()

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

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.

setMaximumHops
public void setMaximumHops(int maxhops, boolean include)

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

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

excludeEdges
public void excludeEdges(Objects edges)

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

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

isClosed
public boolean isClosed()

Gets if Context instance has been closed or not.

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

See Also:
close()
**com.sparsity.sparksee.algorithms**

**Class ShortestPath**

`java.lang.Object`

```java
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](http://www.sparsity-technologies.com)

---

**Method Summary**

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

Methods inherited from interface java.io.Closeable
close

Methods

addNodeType
public void addNodeType(int type)

Allows for traversing nodes of the given type.

Parameters:
   type - null

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.

run
public void run()

Runs the algorithm.

This method can only be called once.

excludeNodes
public void excludeNodes(Objects nodes)
Set which nodes can't be used.

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

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

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

Allows for traversing edges of the given type.

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

excludeEdges

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

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

setMaximumHops

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

Sets the maximum hops restriction.

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

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

isClosed

```java
public boolean isClosed()
```

Gets if ShortestPath instance has been closed or not.

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

See Also:
- `close()`

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

Parameters:
- `dir` - [in] Edge direction.
close

public void close()

    Closes the ShortestPath instance.

    It must be called to ensure the integrity of all data.
com.sparsity.sparksee.algorithms
Class SinglePairShortestPath

java.lang.Object
   +-com.sparsity.sparksee.algorithms.ShortestPath
      +-com.sparsity.sparksee.algorithms.SinglePairShortestPath

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

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

**exists**

```java
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

**addNodeType**

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

Allows for traversing nodes of the given type.

**Parameters:**
- `type` - null

**excludeNodes**

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

Set which nodes can't be used.

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

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

```java
public void run()
```

Runs the algorithm.

This method can only be called once.

---

**getPathAsEdges**

```java
public OIDList getPathAsEdges()
```

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

**Returns:**

Ordered set of edge identifiers.

---

**setMaximumHops**

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

Sets the maximum hops restriction.

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

**Parameters:**

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

---

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

**Parameters:**

- `dir` - [in] Edge direction.

---

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

---

**addEdgeType**

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

Allows for traversing edges of the given type.

**Parameters:**
type - [in] Edge type.
dir - [in] Edge direction.

**getCost**

```java
public double getCost()
```

Gets the cost of the shortest path.

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

**Returns:**
The cost of the shortest path.

**getPathAsNodes**

```java
public OIDList getPathAsNodes()
```

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

**Returns:**
Ordered set of node identifiers.

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

**Parameters:**
edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
com.sparsity.sparksee.algorithms
Class SinglePairShortestPathBFS

java.lang.Object
   +-com.sparsity.sparksee.algorithms.ShortestPath
      +-com.sparsity.sparksee.algorithms.SinglePairShortestPath
         +-com.sparsity.sparksee.algorithms.SinglePairShortestPathBFS

All Implemented Interfaces:
   Closeable

public class SinglePairShortestPathBFS
extends SinglePairShortestPath

SinglePairShortestPathBFS class.
It solves the single-pair shortest path problem using a BFS-based implementation.
It is a unweighted algorithm, that is it assumes all edges have the same cost.
Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.
Author:
   Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

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

Method Summary

| void addAllEdgeTypes(EdgesDirection dir) |
| Allows for traversing all edge types of the graph. |

| void addAllNodeTypes() |
| Allows for traversing all node types of the graph. |

| void addEdgeType(int type, EdgesDirection dir) |
| Allows for traversing edges of the given type. |

| void addNodeType(int type) |
| Allows for traversing nodes of the given type. |

| void checkOnlyExistence() |
| Set that only the path existence must be calculated and not the path itself. |

| void excludeEdges(Objects edges) |
| Set which edges can't be used. |

| void excludeNodes(Objects nodes) |
| Set which nodes can't be used. |

| boolean exists() |
| Returns TRUE If a path exists or FALSE otherwise. |
double getCost()  
Gets the cost of the shortest path.

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

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

void run()  
Executes the algorithm.

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

Methods inherited from class com.sparsity.sparksee.algorithms.SinglePairShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

Methods inherited from class com.sparsity.sparksee.algorithms.ShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, isClosed, run, setMaximumHops

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

Methods inherited from interface java.io.Closeable
close

Constructors

SinglePairShortestPathBFS

public SinglePairShortestPathBFS (Session session,  
long src,  
long dst)

Creates a new instance.

Parameters:
  session - [in] Session to get the graph from and perform traversal.
  src - [in] Source node.
  dst - [dst] Destination node.

Methods
exists

public boolean exists()

Returns TRUE If a path exists or FALSE otherwise.

addNodeType

public void addNodeType(int type)

Allows for traversing nodes of the given type.

Parameters:

  type - null

excludeNodes

public void excludeNodes(Objects nodes)

Set which nodes can't be used.

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

Parameters:

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

getPathAsEdges

public OIDList getPathAsEdges()

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

Returns:

  Ordered set of edge identifiers.

getPathAsNodes

public OIDList getPathAsNodes()

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

Returns:

  Ordered set of node identifiers.

setMaximumHops

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

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

Parameters:

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

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

Parameters:

- \[ \text{dir} - \text{[in]} \] Edge direction.

**getCost**

```java
public double getCost()
```

Gets the cost of the shortest path.

The cost is the number of hops of the shortest path.

Returns:

- The cost of the shortest path.

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

**addEdgeType**

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

Allows for traversing edges of the given type.

Parameters:

- \[ \text{type} - \text{[in]} \] Edge type.
- \[ \text{dir} - \text{[in]} \] Edge direction.

**run**

```java
public void run()
```

Executes the algorithm.
checkOnlyExistence

public void checkOnlyExistence()

Set that only the path existence must be calculated and not the path itself.

That method should improve the performance of the algorithm, but a call to GetPathAsNodes or GetPathAsEdges will generate an exception even if the path exists.

excludeEdges

public void excludeEdges(Objects edges)

Set which edges can't be used.

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

Parameters:

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

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>SinglePairShortestPathDijkstra(Session session, long src, long dst)</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>addAllEdgeTypes(EdgesDirection dir)</td>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
<tr>
<td>void</td>
<td>addAllNodeTypes()</td>
<td>Allows for traversing all node types of the graph.</td>
</tr>
<tr>
<td>void</td>
<td>addEdgeType(int type, EdgesDirection dir)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void</td>
<td>addNodeType(int type)</td>
<td>Allows for traversing nodes of the given type.</td>
</tr>
<tr>
<td>void</td>
<td>addWeightedEdgeType(int type, EdgesDirection dir, int attr)</td>
<td>Allows for traversing edges of the given type using the given attribute as the weight.</td>
</tr>
<tr>
<td>void</td>
<td>excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void</td>
<td>excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>Method Type</td>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>boolean</td>
<td><code>exists()</code></td>
<td>Returns TRUE if a path exists or FALSE otherwise.</td>
</tr>
<tr>
<td>double</td>
<td><code>getCost()</code></td>
<td>Gets the cost of the shortest path.</td>
</tr>
<tr>
<td>OIDList</td>
<td><code>getPathAsEdges()</code></td>
<td>Gets the shortest path between the source node and the destination node as an ordered set of edges.</td>
</tr>
<tr>
<td>OIDList</td>
<td><code>getPathAsNodes()</code></td>
<td>Gets the shortest path between the source node and the destination node as an ordered set of nodes.</td>
</tr>
<tr>
<td>void</td>
<td><code>run()</code></td>
<td>Executes the algorithm.</td>
</tr>
<tr>
<td>void</td>
<td><code>setMaximumHops(int maxhops)</code></td>
<td>Sets the maximum hops restriction.</td>
</tr>
<tr>
<td>void</td>
<td><code>setUnweightedEdgeCost(double weight)</code></td>
<td>Sets the weight assigned to the unweighted edges.</td>
</tr>
</tbody>
</table>

Methods inherited from class `com.sparsity.sparksee.algorithms.SinglePairShortestPath`

- `addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `excludeEdges`, `excludeNodes`, `exists`, `getCost`, `getPathAsEdges`, `getPathAsNodes`, `run`, `setMaximumHops`

Methods inherited from class `com.sparsity.sparksee.algorithms.ShortestPath`

- `addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `isClosed`, `run`, `setMaximumHops`

Methods inherited from class `java.lang.Object`

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

Methods inherited from interface `java.io.Closeable`

- `close`

## Constructors

**SinglePairShortestPathDijkstra**

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

Creates a new instance.

**Parameters:**

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

### Methods

#### exists

```java
public boolean exists()
```

Returns TRUE if a path exists or FALSE otherwise.

#### addNodeType

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

Allows for traversing nodes of the given type.

**Parameters:**
- `type` - null

#### excludeNodes

```java
public void excludeNodes(Object nodes)
```

Set which nodes can't be used.

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

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

#### getPathAsEdges

```java
public OIDList getPathAsEdges()
```

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

**Returns:**
- Ordered set of edge identifiers.

#### getPathAsNodes

```java
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.
setUnweightedEdgeCost

```java
public void setUnweightedEdgeCost(double weight)
```
Sets the weight assigned to the unweighted edges.

All the edges from the types added without an explicit weight attribute will get this weight. The default weight for this edges is 1.

**Parameters:**
- `weight` - [in] The weight value for unweighted edges.

setMaximumHops

```java
public void setMaximumHops(int maxhops)
```
Sets the maximum hops restriction.

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

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

addAllEdgeTypes

```java
public void addAllEdgeTypes(EdgesDirection dir)
```
Allows for traversing all edge types of the graph.

**Parameters:**
- `dir` - [in] Edge direction.

gCost

```java
public double getCost()
```
Gets the cost of the shortest path.

The cost is the sum of the weights of the edges in the shortest path.

**Returns:**
- The cost of the shortest path.

addWeightedEdgeType

```java
public void addWeightedEdgeType(int type,
                                 EdgesDirection dir,
                                 int attr)
```
Allows for traversing edges of the given type using the given attribute as the weight.

**Parameters:**
- `type` - [in] Edge type.
- `dir` - [in] Edge direction.
- `attr` - [in] Attribute to be used as the weight. It must be a global attribute or an attribute of the given edge type.
**addAllNodeTypes**

public void **addAllNodeTypes**()

Allows for traversing all node types of the graph.

---

**addEdgeType**

public void **addEdgeType**(int type,

    EdgesDirection dir)

Allows for traversing edges of the given type.

**Parameters:**

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

---

**run**

public void **run**()

Executes the algorithm.

---

**excludeEdges**

public void **excludeEdges**(Objects edges)

Set which edges can't be used.

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

**Parameters:**

- edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
com.sparsity.sparksee.algorithms
Class StrongConnectivity

java.lang.Object
   +-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

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection dir)</td>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int type, EdgesDirection dir)</td>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>ConnectedComponents getConnectedComponents()</td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>run()</td>
<td>Runs the algorithm in order to find the connected components. This method can be called only once.</td>
</tr>
<tr>
<td>setMaterializedAttribute(String attributeName)</td>
<td>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.</td>
</tr>
</tbody>
</table>

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`

Methods inherited from interface `java.io.Closeable`

- `close`
Allows connectivity through edges of the given type.

**Parameters:**

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

### addNodeType

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

Allows connectivity through nodes of the given type.

**Parameters:**

- **t** - null

### setMaterializedAttribute

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

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

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

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

**Parameters:**

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

### excludeEdges

```java
public void excludeEdges(Object edges)
```

Set which edges can't be used.

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

**Parameters:**

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

### addAllEdgeTypes

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

Allows connectivity through all edge types of the graph.

**Parameters:**

- **dir** - [in] Edge direction.
getConnectedComponents

public ConnectedComponents getConnectedComponents()

    Returns the results generated by the execution of the algorithm.

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

Returns:

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

<p>| void | excludeNodes(Objects nodes) |
|      | Set which nodes can't be used. |</p>
<table>
<thead>
<tr>
<th>ConnectedComponents</th>
<th>getConnectedComponents()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>run()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Executes the algorithm.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setMaterializedAttribute(String attributeName)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.io.Closeable

- close

Constructors

**StrongConnectivityGabow**

public **StrongConnectivityGabow**(Session session)

Creates a new instance of StrongConnectivityGabow.

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

**Parameters:**

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

Methods

**addAllNodeTypes**

public void **addAllNodeTypes**()

Allows connectivity through all node types of the graph.
excludeNodes

public void excludeNodes(Objects nodes)

Set which nodes can't be used.

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

Parameters:

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

addEdgeType

public void addEdgeType(int type,
                        EdgesDirection dir)

Allows connectivity through edges of the given type.

Parameters:

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

run

public void run()

Executes the algorithm.

addNodeType

public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:

- t - null

setMaterializedAttribute

public void setMaterializedAttribute(String attributeName)

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

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

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

Parameters:

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

public void excludeEdges(Objects edges)

Set which edges can't be used.

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

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

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection dir)

Allows connectivity through all edge types of the graph.

Parameters:
dir - [in] Edge direction.

getConnectedComponents

public ConnectedComponents getConnectedComponents()

Returns the results generated by the execution of the algorithm.

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

Returns:
Returns an instance of the class ConnectedComponents which contain information related to the connected components found.
com.sparsity.sparksee.algorithms
Class Traversal

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

All Implemented Interfaces:
   Closeable

Direct Known Subclasses:
   TraversalDFS, TraversalBFS

public class Traversal
extends Object
implements Closeable

Traversal class.

Any class implementing this abstract class can be used to traverse a graph.

Once the instance has been created and the allowed node and edge types has been set, it can be used as an iterator, retrieving the
next object identifier of the traversal until there are no more.

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

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

Method Summary

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

#### addNodeType

**public void addNodeType(int type)**

Allows for traversing nodes of the given type.

**Parameters:**

- type - null

#### hasNext

**public boolean hasNext()**

Gets if there are more objects to be traversed.

**Returns:**

TRUE if there are more objects, FALSE otherwise.

#### excludeNodes

**public void excludeNodes(Objects nodes)**

Set which nodes can't be used.

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

**Parameters:**

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

public void **setMaximumHops**(int maxhops)

Sets the maximum hops restriction.

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

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

---

**addAllEdgeTypes**

public void **addAllEdgeTypes**(EdgesDirection dir)

Allows for traversing all edge types of the graph.

**Parameters:**

- dir - [in] Edge direction.

---

**getCurrentDepth**

public int **getCurrentDepth**()

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

**Returns:**
The depth of the current node.

---

**close**

public void **close**()

Closes the Traversal instance.

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

---

**addAllNodeTypes**

public void **addAllNodeTypes**()

Allows for traversing all node types of the graph.

---

**addEdgeType**

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

Allows for traversing edges of the given type.

**Parameters:**

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

next

public long next()

Gets the next object of the traversal.

Returns:
A node or edge identifier.

excludeEdges

public void excludeEdges(Objects edges)

Set which edges can't be used.

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

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

isClosed

public boolean isClosed()

Gets if Traversal instance has been closed or not.

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

See Also:
close()
public class TraversalBFS
extends Traversal

Breadth-First Search implementation of Traversal.

Starting from a source node, it visits all its neighbors at distance 1, then all its neighbors at distance 2, and so on.

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

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

Constructor Summary

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

Method Summary

void addAllEdgeTypes(EdgesDirection dir)
Allows for traversing all edge types of the graph.

void addAllNodeTypes()
Allows for traversing all node types of the graph.

void addEdgeType(int type, EdgesDirection dir)
Allows for traversing edges of the given type.

void addNodeType(int type)
Allows for traversing nodes of the given type.

void excludeEdges(Objects edges)
Set which edges can't be used.

void excludeNodes(Objects nodes)
Set which nodes can't be used.

int getCurrentDepth()
Returns the depth of the current node.

boolean hasNext()
Gets if there are more objects to be traversed.

long next()
Gets the next object of the traversal.
void setMaximumHops(int maxhops)
Sets the maximum hops restriction.

Methods inherited from class com.sparsity.sparksee.algorithms.Traversal
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, getCurrentDepth, hasNext, isClosed, next, setMaximumHops

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

Methods inherited from interface java.io.Closeable
close

Constructors

TraversalBFS
public TraversalBFS(Session session,
long node)

Creates a new instance.

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

Methods

addNodeType
public void addNodeType(int type)

Allows for traversing nodes of the given type.

Parameters:
  type - null

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.

hasNext
public boolean hasNext()
Gets if there are more objects to be traversed.

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

### excludeNodes

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

Set which nodes can't be used.

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

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

### addEdgeType

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

Allows for traversing edges of the given type.

**Parameters:**
- `type` - [in] Edge type.
- `dir` - [in] Edge direction.

### next

```java
public long next()
```

Gets the next object of the traversal.

**Returns:**
A node or edge identifier.

### getCurrentDepth

```java
public int getCurrentDepth()
```

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

**Returns:**
The depth of the current node.

### setMaximumHops

```java
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.

---

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

**Parameters:**

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

---

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

**Parameters:**

dir - [in] Edge direction.
com.sparsity.sparksee.algorithms

Class TraversalDFS

java.lang.Object
   +-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.
setMaximumHops(int maxhops)
Sets the maximum hops restriction.

Constructors

TraversalDFS

public TraversalDFS(Session session,
    long node)

Creates a new instance.

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

Methods

addNodeType

public void addNodeType(int type)

Allows for traversing nodes of the given type.

Parameters:
- type - null

addAllNodeTypes

public void addAllNodeTypes()

Allows for traversing all node types of the graph.

hasNext

public boolean hasNext()
Gets if there are more objects to be traversed.

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

### excludeNodes

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

Set which nodes can't be used.

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

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

### addEdgeType

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

Allows for traversing edges of the given type.

**Parameters:**
- `type` - [in] Edge type.
- `dir` - [in] Edge direction.

### next

```java
public long next()
```

Gets the next object of the traversal.

**Returns:**
A node or edge identifier.

### getCurrentDepth

```java
public int getCurrentDepth()
```

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

**Returns:**
The depth of the current node.

### setMaximumHops

```java
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.

### excludeEdges

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

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

### addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

**Parameters:**
- `dir` - [in] Edge direction.
com.sparsity.sparksee.algorithms

Class WeakConnectivity

java.lang.Object
   +-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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>ConnectedComponents</td>
</tr>
</tbody>
</table>
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

addEdgeType

public void addEdgeType(int type)

  Allows connectivity through edges of the given type.

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

  Parameters:

  type - [in] Edge type.

addAllNodeTypes

public void addAllNodeTypes()

  Allows connectivity through all node types of the graph.

addAllEdgeTypes

public void addAllEdgeTypes()

  Allows connectivity through all edge types of the graph.

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

run

public void run()
Runs the algorithm in order to find the connected components.

This method can be called only once.

**excludeNodes**

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

Set which nodes can't be used.

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

**Parameters:**

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

**addNodeType**

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

Allows connectivity through nodes of the given type.

**Parameters:**

- `t` - null

**setMaterializedAttribute**

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

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

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

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

**Parameters:**

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

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

**Parameters:**

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

**getConnectedComponents**

```java
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.
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

<table>
<thead>
<tr>
<th>public</th>
<th>WeakConnectivityDFS(Session session)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance of WeakConnectivityDFS.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>addAllEdgeTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addAllNodeTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addEdgeType(int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows connectivity through edges of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addNodeType(int t)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>excludeEdges(Objects edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set which edges can't be used.</td>
</tr>
</tbody>
</table>
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`, `addAllNodeType`, `addEdgeType`, `addNodeType`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `run`, `setMaterializedAttribute`

Methods inherited from class `com.sparsity.sparksee.algorithms.Connectivity`
- `addAllNodeTypes`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `isClosed`, `run`, `setMaterializedAttribute`

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

Methods inherited from interface `java.io.Closeable`
- `close`

Constructors

**WeakConnectivityDFS**

```java
public WeakConnectivityDFS(Session session)
```

Creates a new instance of WeakConnectivityDFS.

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

Parameters:
- `session` - [in] Session to get the graph from and calculate the connectivity

Methods

**addEdgeType**

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

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.
**Parameters:**
- `type` - [in] Edge type.

---

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

---

**addAllEdgeTypes**

```java
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

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

---

**excludeNodes**

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

Set which nodes can't be used.

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

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

---

**run**

```java
public void run()
```

Executes the algorithm.

---

**addNodeType**

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

Allows connectivity through nodes of the given type.

**Parameters:**
- `t` - null

---

**setMaterializedAttribute**

```java
public void setMaterializedAttribute(String attributeName)
```
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

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

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

**Parameters:**
- `attributeName` - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

---

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

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

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

---

**getConnectedComponents**

```java
public ConnectedComponents getConnectedComponents()
```

Returns the results generated by the execution of the algorithm.

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

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

com.sparsity.sparksee.gdb
public class Attribute
extends Object

Attribute data class.

It contains information about an attribute.

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

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidAttribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid attribute identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>long</th>
<th>getCount()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of non-NULL values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataType</th>
<th>getDataType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getId()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee attribute identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AttributeKind</th>
<th>getKind()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the attribute kind.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getName()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the unique attribute name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of different values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getTypeID()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee type identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isSessionAttribute()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Check if it’s a session attribute or a persistent one.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

public static int InvalidAttribute

Invalid attribute identifier constant.

Methods

getKind

public AttributeKind getKind()

Gets the attribute kind.

Returns:
The AttributeKind.

getCount

public long getCount()

Gets the number of non-NULL values.

Returns:
The number of non-NULL values.

isSessionAttribute

public boolean isSessionAttribute()

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

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

getSize

public long getSize()

Gets the number of different values.

Returns:
The number of different values.

getTypeId

public int getId()

Returns the type identifier.
Gets the Sparksee type identifier.

**Returns:**
The Sparksee type identifier.

---

**getDataType**

```java
public DataType getDataType()
```

Gets the data type.

**Returns:**
The DataType.

---

**getId**

```java
public int getId()
```

Gets the Sparksee attribute identifier.

**Returns:**
The Sparksee attribute identifier.

---

**getName**

```java
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

<table>
<thead>
<tr>
<th>public static final</th>
<th>Basic</th>
</tr>
</thead>
</table>
| Basic               | Basic attribute (non indexed attribute).

<table>
<thead>
<tr>
<th>public static final</th>
<th>Indexed</th>
</tr>
</thead>
</table>
| Indexed             | Indexed attribute.

<table>
<thead>
<tr>
<th>public static final</th>
<th>Unique</th>
</tr>
</thead>
</table>
| Unique              | Unique attribute (indexed + unique restriction).

Method Summary

static AttributeKind.valueOf(String name)

static AttributeKind[] values()

Methods inherited from class java.lang.Enum

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

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Comparable

compareTo
Fields

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

Basic attribute (non indexed attribute).

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

Indexed attribute.

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

Unique attribute (indexed + unique restriction).

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

Methods

values
public static AttributeKind[] values()

valueOf
public static AttributeKind valueOf(String name)
com.sparsity.sparksee.gdb
Class AttributeList

java.lang.Object
    +-com.sparsity.sparksee.gdb.AttributeList

All Implemented Interfaces:
    Iterable

public class AttributeList
    extends Object
    implements Iterable

Sparksee attribute identifier list.

It stores a Sparksee attribute identifier list.

Use AttributeListIterator to access all elements into this collection.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td><strong>AttributeList</strong> (Collection col)</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public</td>
<td><strong>AttributeList</strong> ()</td>
</tr>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
<tr>
<td>public</td>
<td><strong>AttributeList</strong> (int[] list)</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance from an integer array.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><strong>add</strong>(int attr)</td>
</tr>
<tr>
<td></td>
<td>Adds a Sparksee attribute identifier at the end of the list.</td>
</tr>
<tr>
<td>void</td>
<td><strong>clear</strong> ()</td>
</tr>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td><strong>count</strong> ()</td>
</tr>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td><strong>AttributeListIterator</strong></td>
<td><strong>iterator</strong> ()</td>
</tr>
<tr>
<td></td>
<td>Gets a new AttributeListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Iterable

iterator
### Constructors

**AttributeList**

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

Creates a new instance from an integer collection.

**Parameters:**

- `col` - Collection to initialize the instance.

**AttributeList**

```java
public AttributeList()
```

Constructor.

This creates an empty list.

**AttributeList**

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

Creates a new instance from an integer array.

**Parameters:**

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

### Methods

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public AttributeListIterator iterator()
```

Gets a new AttributeListIterator.

**Returns:**

AttributeListIterator instance.

**count**

```java
public int count()
```
Number of elements in the list.

**Returns:**
Number of elements in the list.

### add

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

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

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean hasNext()</td>
</tr>
<tr>
<td>Integer next()</td>
</tr>
<tr>
<td>int nextAttribute()</td>
</tr>
<tr>
<td>void remove()</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.util.Iterator
hasNext, next, remove |
Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()

Operation not supported.

next
public Integer next()

See nextAttribute().

nextAttribute
public int nextAttribute()

Gets the next element.
com.sparsity.sparksee.gdb
Class AttributeStatistics

text 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>getAvgLengthString()</td>
<td>Gets the average length.</td>
</tr>
<tr>
<td>long</td>
<td>getDistinct()</td>
<td>Gets the number of distinct values (BASIC statistics).</td>
</tr>
<tr>
<td>Value</td>
<td>getMax()</td>
<td>Gets the maximum existing value (BASIC statistics).</td>
</tr>
<tr>
<td>int</td>
<td>getMaxLengthString()</td>
<td>Gets the maximum length.</td>
</tr>
<tr>
<td>double</td>
<td>getMean()</td>
<td>Gets the mean or average.</td>
</tr>
<tr>
<td>double</td>
<td>getMedian()</td>
<td>Gets the median.</td>
</tr>
<tr>
<td>Value</td>
<td>getMin()</td>
<td>Gets the minimum existing value (BASIC statistics).</td>
</tr>
<tr>
<td>int</td>
<td>getMinLengthString()</td>
<td>Gets the minimum length.</td>
</tr>
<tr>
<td>Value</td>
<td>getMode()</td>
<td>Gets the mode.</td>
</tr>
<tr>
<td>long</td>
<td>getModeCount()</td>
<td>Gets the number of objects with a Value equal to the mode.</td>
</tr>
<tr>
<td>long</td>
<td>getNull()</td>
<td>Gets the number of objects NULL a Value (BASIC statistics).</td>
</tr>
</tbody>
</table>
### Methods

#### getMin

```java
public Value getMin()
```

Gets the minimum existing value (BASIC statistics).

**Returns:**
- The minimum existing value.

#### getMinLengthString

```java
public int getMinLengthString()
```

Gets the minimum length.

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

**Returns:**
- The minimum length.

#### getVariance

```java
public double getVariance()
```

Gets the variance.

It is computed just for numerical attributes.

**Returns:**
- The variance.

#### getMode

```java
public Value getMode()
```

Gets the mode.

Mode: Most frequent Value.

**Returns:**
- The mode.
getNull
public long getNull()

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

Returns:
The number of objects NULL a Value.

getDistinct
public long getDistinct()

Gets the number of distinct values (BASIC statistics).

Returns:
The number of distinct values.

getMean
public double getMean()

Gets the mean or average.

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

It is computed just for numerical attributes.

Returns:
The mean.

getMax
public Value getMax()

Gets the maximum existing value (BASIC statistics).

Returns:
The maximum existing value.

getMedian
public double getMedian()

Gets the median.

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

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

It is computed just for numerical attributes.

Returns:
The median.

**getTotal**

```java
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**

```java
public int getMaxLengthString()
```

Gets the maximum length.

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

**Returns:**

The maximum length.

**getAvgLengthString**

```java
public double getAvgLengthString()
```

Gets the average length.

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

**Returns:**

The average length.

**getModeCount**

```java
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.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public BooleanList(Collection col)</td>
<td>Creates a new instance from a boolean collection.</td>
</tr>
<tr>
<td>public BooleanList(boolean[] list)</td>
<td>Creates a new instance from a boolean array.</td>
</tr>
<tr>
<td>public BooleanList()</td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add(boolean value)</td>
<td>Adds a Boolean at the end of the list.</td>
</tr>
<tr>
<td>void clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int count()</td>
<td>Number of elements in the list.</td>
</tr>
</tbody>
</table>

BooleanListIterator iterator() Gets a new BooleanListIterator.

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**BooleanList**

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

Creates a new instance from a boolean collection.

**Parameters:**

- `col` - Collection to initialize the instance.

**BooleanList**

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

Creates a new instance from a boolean array.

**Parameters:**

- `list` - Boolean array to initialize the instance.

**BooleanList**

```java
public BooleanList()
```

Constructor.

This creates an empty list.

Methods

**add**

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

Adds a Boolean at the end of the list.

**Parameters:**


**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public BooleanListIterator iterator()
```
Gets a new BooleanListIterator.

**Returns:**
BooleanListIterator instance.

count

```java
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

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if there are more elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean</th>
<th>next()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See nextBoolean().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>nextBoolean()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the next element.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>remove()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext() |

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

**remove**

public void remove()

Operation not supported.

**next**

public Boolean next()

See nextBoolean().

**nextBoolean**

public boolean nextBoolean()

Gets the next element.
**com.sparsity.sparksee.gdb**

**Class Condition**

```java
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
<td>Between</td>
</tr>
<tr>
<td></td>
<td>In range operator condition ([x,y]).</td>
</tr>
<tr>
<td>public static final</td>
<td>Equal</td>
</tr>
<tr>
<td></td>
<td>Equal condition (==).</td>
</tr>
<tr>
<td>public static final</td>
<td>GreaterEqual</td>
</tr>
<tr>
<td></td>
<td>Greater or equal condition (&gt;=).</td>
</tr>
<tr>
<td>public static final</td>
<td>GreaterThan</td>
</tr>
<tr>
<td></td>
<td>Greater than condition (&gt;).</td>
</tr>
<tr>
<td>public static final</td>
<td>LessEqual</td>
</tr>
<tr>
<td></td>
<td>Less or equal condition (&lt;=).</td>
</tr>
<tr>
<td>public static final</td>
<td>LessThan</td>
</tr>
<tr>
<td></td>
<td>Less than condition (&lt;).</td>
</tr>
<tr>
<td>public static final</td>
<td>Like</td>
</tr>
<tr>
<td></td>
<td>Substring condition.</td>
</tr>
<tr>
<td>public static final</td>
<td>LikeNoCase</td>
</tr>
<tr>
<td></td>
<td>Substring (no case sensitive) condition.</td>
</tr>
<tr>
<td>public static final</td>
<td>NotEqual</td>
</tr>
<tr>
<td></td>
<td>Not equal condition (!=).</td>
</tr>
<tr>
<td>public static final</td>
<td>RegExp</td>
</tr>
<tr>
<td></td>
<td>Regular expression condition.</td>
</tr>
</tbody>
</table>

### Method Summary

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>static Condition</td>
<td>valueOf(String name)</td>
</tr>
</tbody>
</table>
static Condition[] values()
NotEqual

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

Not equal condition (!=).
Null values can be used together with this condition to retrieve all objects having a non-null value for an attribute.

Like

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

Substring condition.
Null values cannot be used together with this condition.
This condition can just be used together with String values. It allows for searching substrings (case sensitive). Ex:
'AAABBBCCCD' Like 'BBB' returns TRUE
'AAABBBCCCD' Like 'bbb' returns FALSE
'AAABBBCCCD' Like 'E' returns FALSE

LikeNoCase

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

Substring (no case sensitive) condition.
Null values cannot be used together with this condition.
This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:
'AAABBBCCCD' LikeNoCase 'BBB' returns TRUE
'AAABBBCCCD' LikeNoCase 'bbb' returns TRUE
'AAABBBCCCD' LikeNoCase 'E' returns FALSE

Between

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

In range operator condition ([x,y]).
Null values cannot be used together with this condition.

RegExp

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

Regular expression condition.
Null values cannot be used together with this condition.
This condition can just be used together with String values.
Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.
See the 'Regular expressions' section in the 'SPARKSEE User Manual' for details.
Methods

values
public static Condition[] values()

valueOf
public static Condition valueOf(String name)
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>Closes the Database instance.</td>
</tr>
<tr>
<td>disableRollback()</td>
<td>Disables the rollback mechanism.</td>
</tr>
<tr>
<td>enableRollback()</td>
<td>Enables the rollback mechanism.</td>
</tr>
<tr>
<td>fixCurrentCacheMaxSize()</td>
<td>Sets the cache maximum size to the current cache size in use.</td>
</tr>
<tr>
<td>getAlias()</td>
<td>Gets the alias of the Database.</td>
</tr>
<tr>
<td>getCacheMaxSize()</td>
<td>Gets the cache maximum size (in MB).</td>
</tr>
<tr>
<td>getPath()</td>
<td>Gets the path of the Database.</td>
</tr>
<tr>
<td>getStatistics(DatabaseStatistics stats)</td>
<td>Gets Database statistics.</td>
</tr>
<tr>
<td>isClosed()</td>
<td>Gets if Database instance has been closed or not.</td>
</tr>
<tr>
<td>newSession()</td>
<td>Creates a new Session.</td>
</tr>
</tbody>
</table>
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

Methods inherited from interface java.io.Closeable
close

Methods

disableRollback
public void disableRollback()

Disables the rollback mechanism.

fixCurrentCacheMaxSize
public void fixCurrentCacheMaxSize()

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

Returns:
Returns true if successful or false otherwise.

getAlias
public String getAlias()

Gets the alias of the Database.

Returns:
The alias of the Database.

getPath
public String getPath()

Gets the path of the Database.

Returns:
The path of the Database.
enableRollback
public void enableRollback()

    Enables the rollback mechanism.

newSession
public Session newSession()

    Creates a new Session.

getCacheMaxSize
public int getCacheMaxSize()

    Gets the cache maximum size (in MB).

    Returns:
    Returns the current cache max size.

isClosed
public boolean isClosed()

    Gets if Database instance has been closed or not.

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

    See Also:
    close()

setCacheMaxSize
public void setCacheMaxSize(int megaBytes)

    Sets the cache maximum size (in MB).

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

    Parameters:

close
public void close()

    Closes the Database instance.

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

public void getStatistics(DatabaseStatistics stats)

    Gets Database statistics.

Parameters:
    stats - [out] The DatabaseStatistics instance.
com.sparsity.sparksee.gdb
Class DatabaseStatistics

public class DatabaseStatistics
extends Object

Database statistics.

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCache()</td>
<td>Gets cache size in KBytes.</td>
</tr>
<tr>
<td>getData()</td>
<td>Gets database size in KBytes.</td>
</tr>
<tr>
<td>getRead()</td>
<td>Gets total read data in KBytes.</td>
</tr>
<tr>
<td>getSessions()</td>
<td>Gets the number of sessions.</td>
</tr>
<tr>
<td>getTemp()</td>
<td>Gets temporary storage file size in KBytes.</td>
</tr>
<tr>
<td>getWrite()</td>
<td>Gets total written data in KBytes.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

getWrite

public long getWrite()

Gets total written data in KBytes.

Returns:
Total read written in KBytes.
getSessions
public long getSessions()

    Gets the number of sessions.

    Returns:
    The number of sessions.

data
public long getData()

    Gets database size in KBytes.

    Returns:
    Database size in KBytes.

temp
public long getTemp()

    Gets temporary storage file size in KBytes.

    Returns:
    Temporary storage file size in KBytes.

read
public long getRead()

    Gets total read data in KBytes.

    Returns:
    Total read data in KBytes.

cache
public long getCache()

    Gets cache size in KBytes.

    Returns:
    Cache size in KBytes.
public final class **DataType**

extends Enum

Data type (domain) enumeration.

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

### Field Summary

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

### Method Summary

<table>
<thead>
<tr>
<th>static <strong>DataType</strong></th>
<th><strong>valueOf</strong>(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>DataType[]</strong></td>
<td><strong>values</strong>( )</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.lang.Comparable

compareTo

Fields

Boolean

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

Boolean data type.

Integer

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

32-bit signed integer data type.

Long

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

64-bit signed integer data type.

Double

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

64-bit signed double data type.

Timestamp

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

Distance from Epoch (UTC) time in milliseconds precision.

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

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

  Unicode string data type.

  2048 characters maximum length.

Text

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

  Large unicode character object (CLOB) data type.

  TextStream

OID

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

  Object identifier (OID) data type.

Methods

values

public static DataType[] values()

valueOf

public static DataType.valueOf(String name)
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

<table>
<thead>
<tr>
<th>public</th>
<th>DefaultExport()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>enableType(int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method EnableType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getEdge(long edge, EdgeExport edgeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method GetEdge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getEdgeType(int type, EdgeExport edgeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method GetEdgeType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getGraph(GraphExport graphExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method GetGraph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getNode(long node, NodeExport nodeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method GetNode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getNodeType(int type, NodeExport nodeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default implementation of the ExportManager class method GetNodeType.</td>
</tr>
</tbody>
</table>

| void | prepare(Graph graph) |
|     | Default implementation of the ExportManager class method Prepare. |

| void | release() |
|     | Default implementation of the ExportManager class method Release. |

Methods inherited from class com.sparsity.sparksee.gdb.ExportManager

enableType, getEdge, getEdgeType, getGraph, getNode, getNodeType, prepare, release

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

Constructors

DefaultExport

public DefaultExport()

 Creates a new instance.

Methods

enableType

public boolean enableType(int type)

Default implementation of the ExportManager class method EnableType.

This enables all node and edge types to be exported.

Parameters:

  type - [in] The type to enable.

Returns:

  TRUE.

getEdge

public boolean getEdge(long edge, EdgeExport edgeExport)

Default implementation of the ExportManager class method GetEdge.

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

Parameters:

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

Returns:

  TRUE.

getGraph

public boolean getGraph(GraphExport graphExport)

Default implementation of the ExportManager class method GetGraph.

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

Parameters:

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

**getEdgeType**

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

Default implementation of the ExportManager class method GetEdgeType.

This sets the default EdgeExport values.

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

**Returns:**
TRUE.

**getNodeType**

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

Default implementation of the ExportManager class method GetNodeType.

This sets the default NodeExport values.

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

**Returns:**
TRUE.

**release**

```java
public void release()
```

Default implementation of the ExportManager class method Release.

**getNode**

```java
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.
prepare

public void prepare(Graph graph)

Default implementation of the ExportManager class method Prepare.

Parameters:

graph - null
public class EdgeData extends Object

Edge data class.

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

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

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEdge()</td>
<td>long</td>
<td>Gets the edge identifier.</td>
</tr>
<tr>
<td>getHead()</td>
<td>long</td>
<td>Gets the head of the edge.</td>
</tr>
<tr>
<td>getTail()</td>
<td>long</td>
<td>Gets the tail of the edge.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

getHead

public long getHead()

Gets the head of the edge.

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

getTail

public long getTail()

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

**getEdge**

```java
public long getEdge()
```

Gets the edge identifier.

**Returns:**
The Sparksee edge identifier.
com.sparsity.sparksee.gdb
Class EdgeExport

java.lang.Object
     +-com.sparsity.sparksee.gdb.EdgeExport

public class EdgeExport
extends Object

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

As directed: TRUE.

Color: 13882323 (0xD3D3D3, Light gray).

Label color: 0 (0x000000, 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.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int getLabelColorRGB()</code></td>
<td>Gets the edge label color.</td>
</tr>
<tr>
<td><code>int getWidth()</code></td>
<td>Gets the edge width.</td>
</tr>
<tr>
<td><code>void setAsDirected(boolean directed)</code></td>
<td>Sets if the edge should be managed as directed.</td>
</tr>
<tr>
<td><code>void setColor(java.awt.Color c)</code></td>
<td>Sets the color of the edge.</td>
</tr>
<tr>
<td><code>void setColorRGB(int color)</code></td>
<td>Sets the edge color.</td>
</tr>
<tr>
<td><code>void setDefaults()</code></td>
<td>Sets to default values.</td>
</tr>
<tr>
<td><code>void setFontSize(int size)</code></td>
<td>Sets the edge label font size.</td>
</tr>
<tr>
<td><code>void setLabel(String label)</code></td>
<td>Sets the edge label.</td>
</tr>
<tr>
<td><code>void setLabelColor(java.awt.Color c)</code></td>
<td>Sets the color of the label.</td>
</tr>
<tr>
<td><code>void setLabelColorRGB(int color)</code></td>
<td>Sets the edge label color.</td>
</tr>
<tr>
<td><code>void setWidth(int width)</code></td>
<td>Sets the edge width.</td>
</tr>
</tbody>
</table>

Methods inherited from class `java.lang.Object`

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

## Constructors

### EdgeExport

```java
public EdgeExport()
```

Creates a new instance.

## Methods

### getColor

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

Gets the color of the edge.
**setColorRGB**

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

Sets the edge color.

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

**setAsDirected**

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

Sets if the edge should be managed as directed.

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

**getFontSize**

```java
public int getFontSize()
```

Gets the edge label font size.

**Returns:**
- The edge label font size.

**setDefaults**

```java
public void setDefaults()
```

Sets to default values.

**getColorRGB**

```java
public int getColorRGB()
```

Gets the edge color.

**Returns:**
- The edge color.
Gets the edge label color.

**Returns:**

The edge label color.

---

**getWidth**

```java
public int getWidth()
```

Gets the edge width.

**Returns:**

The edge width.

---

**setLabel**

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

Sets the edge label.

**Parameters:**

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

---

**getLabelColor**

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

Gets the color of the label.

---

**setColor**

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

Sets the color of the edge.

**Parameters:**

- `c` - New value.

---

**asDirected**

```java
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.
public String getLabel()

    Gets the edge label.

    Returns:
    The edge label.

public void setLabelColorRGB(int color)

    Sets the edge label color.

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

public void setWidth(int width)

    Sets the edge width.

    Parameters:
    width - [in] The edge width.

public void setFontSize(int size)

    Sets the edge label font size.

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

public void setLabelColor(java.awt.Color c)

    Sets the color of the label.

    Parameters:
    c - New value.
public final class EdgesDirection
extends Enum

Edges direction enumeration.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Any</th>
<th>In-going or out-going edges.</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
<td>Ingoing</td>
<td>In-going edges.</td>
</tr>
<tr>
<td>public static final</td>
<td>Outgoing</td>
<td>Out-going edges.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static EdgesDirection.valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static EdgesDirection[] values()</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.lang.Comparable

compareTo

Fields
Ingoing

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

In-going edges.

Outgoing

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

Out-going edges.

Any

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

In-going or out-going edges.

Methods

values

public static EdgesDirection[] values()

valueOf

public static EdgesDirection valueOf(String name)
com.sparsity.sparksee.gdb
Class ExportManager

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

Direct Known Subclasses:
    DefaultExport

public class ExportManager
extends Object

Defines how to export a graph to an external format.

This is an interface which must be implemented by the user. While the export process, 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 different format. Nowadays, available formats are defined in the ExportType enum.

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

Method Summary

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

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

Methods
getNodeType

public boolean getNodeType(int type,
            NodeExport nodeExport)

GETS THE DEFAULT NODE EXPORT DEFINITION FOR THE GIVEN NODE TYPE.

GETNODE HAS A HIGHER PRIORITY THAN THIS FUNCTION. THAT IS, ONLY IF GETNODE RETURNS FALSE, THE NODEEXPORT OF THIS FUNCTION WILL BE USED.

Parameters:
    type - [in] Node type identifier.
    nodeExport - [out] The NodeExport which defines how to export the nodes of the given type.

Returns:
    TRUE.

getEdge

public boolean getEdge(long edge,
            EdgeExport edgeExport)

GETS THE EDGE EXPORT DEFINITION FOR THE GIVEN EDGE.

Parameters:
    edge - Edge identifier.
    edgeExport - [out] The EdgeExport which defines how to export given edge.

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

getGraph

public boolean getGraph(GraphExport graphExport)

GETS THE GRAPH EXPORT DEFINITION.

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

Returns:
    TRUE.

getEdgeType

public boolean getEdgeType(int type,
            EdgeExport edgeExport)

GETS THE DEFAULT NODE EXPORT DEFINITION FOR THE GIVEN EDGE TYPE.

GETEDGE HAS A HIGHER PRIORITY THAN THIS FUNCTION. THAT IS, ONLY IF GETEDGE RETURNS FALSE, THE EDGEEXPORT OF THIS FUNCTION WILL BE USED.

Parameters:
    type - [in] Edge type identifier.
edgeExport - [out] The EdgeExport which defines how to export the edges of the given type.

**Returns:**
TRUE.

---

**prepare**

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

Prepares the graph for the export process.

It is called once before the export process.

**Parameters:**
graph - Graph to be exported.

---

**getNode**

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

Gets the node export definition for the given node.

**Parameters:**
node - Node identifier.
nodeExport - [out] The NodeExport which defines how to export given node.

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

---

**release**

```java
public void release()
```

Ends the export process.

It is called once after the export process.

---

**enableType**

```java
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:**

- TRUE if all objects of the given type will be exported, otherwise they will not be exported.
com.sparsity.sparksee.gdb

Class ExportType

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

All Implemented Interfaces:
   Serializable, Comparable

public final class ExportType
extends Enum

Export type.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>GraphML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Export to GraphML format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Graphviz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Export to Graphviz format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>YGraphML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Export to YGRAPHML format.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static ExportType</th>
<th>valueOf(String name)</th>
</tr>
</thead>
</table>

| 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

Methods inherited from interface java.lang.Comparable

compareTo

Fields

Page 108 of 305
**Graphviz**

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

Export to Graphviz format.

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

---

**GraphML**

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

Export to GraphML format.

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

---

**YGraphML**

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

Export to YGRAPHML format.

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

---

### Methods

#### values

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

---

#### valueOf

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>long</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>long</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>long</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>void</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Objects</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Objects</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Objects</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td><code>void export(String file, ExportType type, ExportManager em)</code></td>
</tr>
<tr>
<td><code>int findAttribute(int type, String name)</code></td>
</tr>
<tr>
<td><code>AttributeList findAttributes(int type)</code></td>
</tr>
<tr>
<td><code>long findEdge(int etype, long tail, long head)</code></td>
</tr>
<tr>
<td><code>TypeList findEdgeTypes()</code></td>
</tr>
<tr>
<td><code>TypeList findNodeTypes()</code></td>
</tr>
<tr>
<td><code>long findObject(int attr, Value value)</code></td>
</tr>
<tr>
<td><code>long findOrCreateEdge(int etype, long tail, long head)</code></td>
</tr>
<tr>
<td><code>long findOrCreateObject(int attr, Value value)</code></td>
</tr>
<tr>
<td><code>int findType(String name)</code></td>
</tr>
<tr>
<td><code>TypeList findTypes()</code></td>
</tr>
<tr>
<td><code>Attribute getAttribute(int attr)</code></td>
</tr>
<tr>
<td><code>Value getAttribute(long oid, int attr)</code></td>
</tr>
<tr>
<td><code>void getAttribute(long oid, int attr, Value value)</code></td>
</tr>
<tr>
<td><code>long getAttributeIntervalCount(int attr, Value lower, boolean includeLower, Value higher, boolean includeHigher)</code></td>
</tr>
<tr>
<td><code>AttributeList getAttributes(long oid)</code></td>
</tr>
<tr>
<td><code>AttributeStatistics getAttributeStatistics(int attr, boolean basic)</code></td>
</tr>
<tr>
<td><code>TextStream getAttributeText(long oid, int attr)</code></td>
</tr>
<tr>
<td><code>EdgeData getEdgeData(long edge)</code></td>
</tr>
<tr>
<td><code>long getEdgePeer(long edge, long node)</code></td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td><code>int getObjectType(long oid)</code></td>
</tr>
<tr>
<td><code>Type getType(int type)</code></td>
</tr>
<tr>
<td><code>Values getValues(int attr)</code></td>
</tr>
<tr>
<td><code>Objects heads(Objects edges)</code></td>
</tr>
<tr>
<td><code>void indexAttribute(int attr, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>Objects neighbors(long oid, int etype, EdgesDirection dir)</code></td>
</tr>
<tr>
<td><code>Objects neighbors(Objects objs, int etype, EdgesDirection dir)</code></td>
</tr>
<tr>
<td><code>int newAttribute(int type, String name, DataType dt, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>int newAttribute(int type, String name, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
</tr>
<tr>
<td><code>long newEdge(int type, int tailAttr, Value tailV, int headAttr, Value headV)</code></td>
</tr>
<tr>
<td><code>long newEdge(int type, long tail, long head)</code></td>
</tr>
<tr>
<td><code>int newEdgeType(String name, boolean directed, boolean neighbors)</code></td>
</tr>
<tr>
<td><code>long newNode(int type)</code></td>
</tr>
<tr>
<td><code>int newNodeType(String name)</code></td>
</tr>
<tr>
<td><code>int newRestrictedEdgeType(String name, int tail, int head, boolean neighbors)</code></td>
</tr>
<tr>
<td><code>int newSessionAttribute(int type, DataType dt, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>int newSessionAttribute(int type, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
</tr>
<tr>
<td><code>void removeAttribute(int attr)</code></td>
</tr>
<tr>
<td><code>void removeType(int type)</code></td>
</tr>
</tbody>
</table>
### Rename Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>renameAttribute(int attr, String newName)</code></td>
<td>Renames an attribute.</td>
</tr>
<tr>
<td><code>renameType(int type, String newName)</code></td>
<td>Renames a type.</td>
</tr>
<tr>
<td><code>renameType(String oldName, String newName)</code></td>
<td>Renames a type.</td>
</tr>
</tbody>
</table>

### Select Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>select(int type)</code></td>
<td>Selects all OIDs belonging to the given type.</td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value value)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value value, Objects restriction)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value lower, Value higher)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value lower, Value higher, Objects restriction)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
</tbody>
</table>

### Attribute Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setAttribute(long oid, int attr, Value value)</code></td>
<td>Sets the Value for the given attribute and OID.</td>
</tr>
<tr>
<td><code>setAttributeDefaultValue(int attr, Value value)</code></td>
<td>Sets a default value for an attribute.</td>
</tr>
<tr>
<td><code>setAttributeText(long oid, int attr, TextStream tstream)</code></td>
<td>Sets the writable TextStream for the given text attribute and OID.</td>
</tr>
</tbody>
</table>

### Graph Collections Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tails(Objects edges)</code></td>
<td>Gets all the tails from the given edges collection.</td>
</tr>
<tr>
<td><code>tailsAndHeads(Objects edges, Objects tails, Objects heads)</code></td>
<td>Gets all the tails and heads from the given edges collection.</td>
</tr>
</tbody>
</table>

### Methods Inherited from `java.lang.Object`

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`
oid - [in] Sparksee OID.
attr - [in] Sparksee attribute identifier.
tstream - [in] New Text value. This corresponds to a TextStream to write.

## dumpData

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

Dumps logical data to a file.

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

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

## findTypes

```java
public TypeList findTypes()
```

Gets all existing Sparksee node and edge type identifiers.

**Returns:**
- Sparksee node and edge type identifier list.

## renameAttribute

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

Renames an attribute.

The new name must be available.

**Parameters:**
- attr - [in] Sparksee attribute identifier.
- newName - [in] The new name for the attribute.

## select

```java
public void 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.

getIntervalCount

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

Gets how many objects have a value into the given range for the given attribute.
This only works for the attributes with the AttributeKind Indexed or Unique.
Given values must belong to the same DataType than the attribute.

Parameters:
attr - [in] Sparksee attribute identifier.
lower - [in] Lower bound Value of the range.
includeLower - [in] If TRUE, include lower bound Value of the range.
higher - [in] Higher bound Value of the range.
includeHigher - [in] If TRUE, include higher bound Value of the range.

Returns:
Number of objects having a value into the given range.

tailsAndHeads

public void tailsAndHeads(Objects edges, Objects tails, Objects heads)

Gets all the tails and heads from the given edges collection.

Parameters:
edges - [in] Sparksee edge identifier collection.
tails - [in|out] If not NULL, all the tails will be stored here.
heads - [in|out] If not NULL, all the heads will be stored here.

degree

public long degree(long oid, int etype, EdgesDirection dir)

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

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

Returns:
The number of edges.
renameType

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

Renames a type.

The new name must be available.

**Parameters:**

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

dumpStorage

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

Dumps internal storage data to a file.

**Parameters:**

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

**Throws:**

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

neighbors

```java
public Object neighbors(Objects objs,
            int etype,
            EdgesDirection dir)
```

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

**Parameters:**

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

**Returns:**

Objects instance.

getAttributes

```java
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.

---

**getAttributeStatistics**

```java
class getAttributeStatistics
```

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

Gets statistics from the given attribute.

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

**Returns:**
An AttributeStatistics instance.

---

**newNode**

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

Creates a new node instance.

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

**Returns:**
Unique OID of the new node instance.

---

**getAttributeText**

```java
public TextStream getAttributeText(long oid, int attr)
```

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

**Parameters:**
- oid - [in] Sparksee OID.
- attr - [in] Sparksee attribute identifier.

**Returns:**
A TextStream. This returns a TextStream to read.

---

**countEdges**

```java
public long countEdges()"
```

Gets the number of edges.

**Returns:**
The number of edges.

**findEdgeTypes**

```java
public TypeList findEdgeTypes()
```

Gets all existing Sparksee edge type identifiers.

**Returns:**
Sparksee edge type identifier list.

**select**

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

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

This allows to perform the Between operation, thus it has two Value arguments.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `cond` - [in] Condition to be satisfied. It must be the Between Condition.
- `lower` - [in] Lower-bound Value to be satisfied.
- `higher` - [in] Higher-bound Value to be satisfied.

**Returns:**
Objects instance.

**indexAttribute**

```java
public void indexAttribute(int attr, 
AttributeKind kind)
```

Updates the index of the given attribute.

This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one is Indexed or Unique.

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

**getType**

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

Gets information about the given type.

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

**Returns:**
The Type for the given Sparksee type identifier.
findAttribute

```java
public int findAttribute(int type,
    String name)
```

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

**Parameters:**
- `type` - [in] Sparksee type identifier.
- `name` - [in] Unique attribute name.

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

newAttribute

```java
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.

dges

```java
public Objects edges(int etype,
    long tail,
    long head)
```

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

**Parameters:**
- `etype` - [in] Sparksee edge type identifier.
- `tail` - [in] Tail node identifier.
- `head` - [in] Head node identifier.

**Returns:**
Objects instance.

select

```java
public Objects select(int type)
```
Selects all OIDs belonging to the given type.

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

**Returns:**
- Objects instance.

---

**select**

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

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

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `cond` - [in] Condition to be satisfied.
- `value` - [in] Value to be satisfied.

**Returns:**
- Objects instance.

---

**findOrCreateObject**

```java
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.

**Returns:**
- Sparksee OID or the Objects InvalidOID.

---

**findNodeTypes**

```java
public TypeList findNodeTypes()
```

Gets all existing Sparksee node type identifiers.

**Returns:**
- Sparksee node type identifier list.
**getAttribute**

```java
public Value getAttribute(long oid, int attr)
```

Gets the Value for the given attribute and OID.

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

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

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

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

---

**removeAttribute**

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

Removes the given attribute.

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

---

**setAttributeDefaultValue**

```java
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:**
- `value` - [in] The default value to use for this attribute.

---

**backup**

```java
public void backup(String file)
```

Dumps all the data to a backup file.

See the Sparksee class Restore method.

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

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

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

Creates a new Session attribute with a default value.

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

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

Parameters:
    type - [in] Sparksee node or edge type identifier.
    dt - [in] Data type for the new attribute.
    kind - [in] Attribute kind.
    defaultValue - [in] The default value to use in each new node/edge.

Returns:
    Unique Sparksee attribute identifier.

findAttributes

public AttributeList findAttributes(int type)

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

Parameters:
    type - [in] Sparksee type identifier.

Returns:
    Sparksee attribute identifier list.

getAttribute

public void getAttribute(long oid,
    int attr,
    Value value)

Gets the Value for the given attribute and OID.

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

countNodes

public long countNodes()

Gets the number of nodes.
Returns:
The number of nodes.

setAttribute

public void setAttribute(long oid, int attr, Value value)

Sets the Value for the given attribute and OID.

Parameters:
oid - [in] Sparksee OID.
attr - [in] Sparksee attribute identifier.
value - [in] Value for the given attribute and for the given OID.

getEdgeData

public EdgeData getEdgeData(long edge)

Gets information about an edge.

Parameters:
edge - [in] Sparksee edge identifier.

Returns:
An EdgeData instance.

neighbors

public Objects neighbors(long oid, int etype, EdgesDirection dir)

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

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

Returns:
Objects instance.

renameType

public void renameType(int type, String newName)

 Renames a type.
The new name must be available.

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

explode

```java
public Objects explode(Objects objs,
                      int etype,
                      EdgesDirection dir)
```

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

Parameters:
- `objs` - [in] Sparksee node OID collection.
- `etype` - [in] Sparksee edge type identifier.
- `dir` - [in] Direction.

Returns:
- Objects instance.

newNodeType

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

Creates a new node type.

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

Returns:
- Unique Sparksee type identifier.

newSessionAttribute

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

Creates a new Session attribute.

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

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

Parameters:
- `type` - [in] Sparksee node or edge type identifier.
- `dt` - [in] Data type for the new attribute.
- `kind` - [in] Attribute kind.

Returns:
- Unique Sparksee attribute identifier.

tails

```java
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.

**findOrCreateEdge**

public long findOrCreateEdge(int etype, long tail, long head)

Gets any of the edges of the specified type between two given nodes (tail and head).
If it can not find any edge of this type between them it tries to create a new one.

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

Returns:
Any of the edges or the Objects InvalidOID.

**drop**

public void drop(long oid)

Drops the given OID.
It also removes its egdges as well as its attribute values.

Parameters:
oid - [in] Sparksee OID to be removed.

**newEdgeType**

public int newEdgeType(String name, boolean directed, boolean neighbors)

Creates a new edge type.

Parameters:
name - [in] Unique name for the new edge type.
directed - [in] If TRUE, this creates a directed edge type, otherwise this creates a undirected edge type.
neighbors - [in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns:
Unique Sparksee type identifier.

**heads**

public Objects heads(Objects edges)
Gets all the heads from the given edges collection.

**Parameters:**

**Returns:**
The heads collection.

### findEdge

```java
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.
- head [in] Head node identifier.

**Returns:**
Any of the edges or the Objects InvalidOID.

### explode

```java
public Objects explode(long oid, int etype, EdgesDirection dir)
```

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

**Parameters:**
- oid [in] Sparksee node OID.
- etype [in] Sparksee edge type identifier.
- dir [in] Direction.

**Returns:**
Objects instance.

### findObject

```java
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:**
- value [in] Value.
Returns:
Sparksee OID or the Objects InvalidOID.

**drop**

```java
drop(Objects objs)
```

Drops all the OIDs from the given collection.

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

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

**newAttribute**

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

Creates a new attribute with a default value.

**Parameters:**
- `type` - [in] Sparksee node or edge type identifier.
- `name` - [in] Unique name for the new attribute.
- `dt` - [in] Data type for the new attribute.
- `kind` - [in] Attribute kind.
- `defaultValue` - [in] The default value to use in each new node/edge.

**Returns:**
Unique Sparksee attribute identifier.

**getObjectType**

```java
getObjectType(long oid)
```

Gets the Sparksee node or edge type identifier for the given OID.

**Parameters:**
- `oid` - [in] Sparksee OID.

**Returns:**
Sparksee node or edge type identifier.

**getAttribute**

```java
getAttribute(int attr)
```

Gets information about the given attribute.

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

Returns:
The Attribute for the given Sparksee attribute identifier.

---

export

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

Exports the Graph.

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

Throws:
- `java.io.IOException` - null

---

newEdge

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

Returns:
Unique OID of the new edge instance.

---

getValues

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

Gets the Value collection for the given attribute.

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

Returns:
Returns a Values object.
**getEdgePeer**

```java
public long getEdgePeer(long edge, long node)
```

Gets the other end for the given edge.

**Parameters:**
- `edge` - [in] Sparksee edge identifier.
- `node` - [in] Sparksee node identifier. It must be one of the ends of the edge.

**Returns:**
The other end of the edge.

**newEdge**

```java
public long newEdge(int type, long tail, long head)
```

Creates a new edge instance.

**Parameters:**
- `type` - [in] Sparksee type identifier.
- `tail` - [in] Source Sparksee OID.
- `head` - [in] Target Sparksee OID.

**Returns:**
Unique OID of the new edge instance.

**select**

```java
public Objects select(int attr, 
                     Condition cond, 
                     Value lower, 
                     Value higher, 
                     Objects restriction)
```

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

This allows to perform the Between operation, thus it has two Value arguments.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `cond` - [in] Condition to be satisfied. It must be the Between Condition.
- `lower` - [in] Lower-bound Value to be satisfied.
- `higher` - [in] Higher-bound Value to be satisfied.
- `restriction` - [in] Objects to limit the select in this set of objects.

**Returns:**
Objects instance.

**findType**

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

com.sparsity.sparksee.gdb.Graph
Gets the Sparksee type identifier for the given type name.

**Parameters:**
- **name** - [in] Unique type name.

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

### removeType

**public void removeType(int type)**

Removes the given type.

This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

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

### newRestrictedEdgeType

**public int newRestrictedEdgeType(String name, int tail, int head, boolean neighbors)**

Creates a new restricted edge type.

**Parameters:**
- **name** - [in] Unique name for the new edge type.
- **tail** - [in] Tail Sparksee node type identifier.
- **head** - [in] Head Sparksee node type identifier.
- **neighbors** - [in] If TRUE, this indexes neighbor nodes, otherwise not.

**Returns:**
Unique Sparksee type identifier.
public class GraphExport
extends Object

Stores the graph exporting values.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>GraphExport()</td>
</tr>
</tbody>
</table>

Creates a new GraphExport instance.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>getLabel()</td>
</tr>
</tbody>
</table>

Gets the graph label.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>setDefaults()</td>
</tr>
</tbody>
</table>

Sets to default values.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>setLabel(String label)</td>
</tr>
</tbody>
</table>

Sets the graph label.

### Methods inherited from class java.lang.Object

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

### Constructors

**GraphExport**

**public** GraphExport()

Creates a new GraphExport instance.

### Methods

**getLabel**

**public** String getLabel()
Gets the graph label.

Returns:
   The graph label.

setDefaults
public void setDefaults()

Sets to default values.

setLabel
public void setLabel(String label)

Sets the graph label.

Parameters:
   label - [in] The graph label.
com.sparsity.sparksee.gdb
Class Int32List

java.lang.Object
   +-com.sparsity.sparksee.gdb.Int32List

All Implemented Interfaces:
   Iterable

public class Int32List
extends Object
implements Iterable

Sparksee 32-bit signed integer list.
It stores a 32-bit signed integer list.

Use Int32ListIterator to access all elements into this collection.

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

Constructor Summary

| public | Int32List(Collection col) |
|        | Creates a new instance from an integer collection. |

| public | Int32List() |
|        | Constructor. |

| public | Int32List(int[] list) |
|        | Creates a new instance from an integer array. |

Method Summary

| void | add(int value) |
|      | Adds an 32-bit signed integer at the end of the list. |

| void | clear() |
|      | Clears the list. |

| int | count() |
|     | Number of elements in the list. |

| Int32ListIterator | iterator() |
|                   | Gets a new Int32ListIterator. |

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

Int32List
public Int32List(Collection col)

Creates a new instance from an integer collection.

Parameters:
col - Collection to initialize the instance.

Int32List
public Int32List()

Constructor.
This creates an empty list.

Int32List
public Int32List(int[] list)

Creates a new instance from an integer array.

Parameters:
list - Integer array to initialize the instance.

Methods

add
public void add(int value)

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

Parameters:
value - [in] The integer.

clear
public void clear()

Clears the list.

iterator
public Int32ListIterator iterator()
Gets a new Int32ListIterator.

**Returns:**
Int32ListIterator instance.

count

```java
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
**com.sparsity.sparksee.gdb**  
**Class Int32ListIterator**

```java
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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean hasNext()</code></td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td><code>Integer next()</code></td>
<td>See <code>nextInt32()</code></td>
</tr>
<tr>
<td><code>Integer nextInt32()</code></td>
<td>Gets the next element.</td>
</tr>
<tr>
<td><code>void remove()</code></td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class `java.lang.Object`

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

Methods inherited from interface `java.util.Iterator`

- `hasNext, next, remove`

---

**Methods**

**nextInt32**

```java
public Integer nextInt32()
```

Gets the next element.
hasNext
public boolean hasNext()

Gets if there are more elements.

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

remove
public void remove()

Operation not supported.

next
public Integer next()

See nextInt32().
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

<table>
<thead>
<tr>
<th>public static final</th>
<th>Config</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Debug log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fine log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Info log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disable log.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Severe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Severe log level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Warning log level.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static LogLevel</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LogLevel[]</td>
<td>values()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf
<table>
<thead>
<tr>
<th>Methods inherited from class java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods inherited from interface java.lang.Comparable</th>
</tr>
</thead>
<tbody>
<tr>
<td>compareTo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Off</strong></td>
</tr>
<tr>
<td>public static final com.sparsity.sparksee.gdb.LogLevel Off</td>
</tr>
<tr>
<td>Disable log.</td>
</tr>
</tbody>
</table>

| **Severe**                             |
| public static final com.sparsity.sparksee.gdb.LogLevel Severe |
| Severe log level.                      |
| This is the lower log level, just errors are shown. |

| **Warning**                             |
| public static final com.sparsity.sparksee.gdb.LogLevel Warning |
| Warning log level.                      |
| Errors and warnings are shown.          |

| **Info**                               |
| public static final com.sparsity.sparksee.gdb.LogLevel Info |
| Info log level.                         |
| Errors, warnings and information messages are shown. |

| **Config**                             |
| public static final com.sparsity.sparksee.gdb.LogLevel Config |
| Config log level.                       |
| Errors, warnings, information messages and configuration details of the different components are shown. |

| **Fine**                               |
| public static final com.sparsity.sparksee.gdb.LogLevel Fine |
Fine log level.

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

---

**Debug**

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

Debug log level.

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

---

### Methods

**values**

public static LogLevel[] values()

**valueOf**

public static LogLevel valueOf(String name)
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

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NodeExport()</td>
</tr>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.awt.Color</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>java.awt.Color</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>NodeShape</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class** java.lang.Object

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

**Constructors**
**NodeExport**

```java
public NodeExport()
```

Creates a new instance.

---

## Methods

### getShape

```java
public NodeShape getShape()
```

Gets the node shape.

**Returns:**

The node shape.

### getColor

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

Gets the color of the node.

### setColorRGB

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

Sets the node color.

**Parameters:**

- `color` - The node color.

### setHeight

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

Sets the node height.

**Parameters:**


### getFontSize

```java
public int getFontSize()
```

Gets the node label font size.
Returns:
The node label font size.

```java
setDefaults
public void setDefaults()

Sets to default values.
```

```java
getColorRGB
public int getColorRGB()

Gets the node color.

Returns:
The node color.
```

```java
isFit
public boolean isFit()

Gets whether the node size is fitted to the label or not.

Returns:
If TRUE, then the node size is fitted to the label, otherwise the size is fixed with the values of 'height' and 'width'.
```

```java
getLabelColorRGB
public int getLabelColorRGB()

Gets the node label color.

Returns:
The node label color.
```

```java
getWidth
public int getWidth()

Gets the node width.

Returns:
The node width in pixels.
```

```java
setLabel
public void setLabel(String label)
```
Sets the node label.

**Parameters:**

- **label** - [in] The node label.

---

### getLabelColor

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

Gets the color of the label.

---

### setColor

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

Sets the color of the node.

**Parameters:**

- **c** - New value.

---

### getLabel

```java
public String getLabel()
```

Gets the node label.

**Returns:**

- The node label.

---

### getHeight

```java
public int getHeight()
```

Gets the node height.

**Returns:**

- The node height in pixels.

---

### setLabelColorRGB

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

Sets the node label color.

**Parameters:**

- **color** - [in] The node label color.
setWidth

public void setWidth(int width)

Gets the node width.

Parameters:

width - The node width in pixels.

setShape

public void setShape(NodeShape shape)

Sets the node shape.

Parameters:

shape - [in] The node shape.

setFit

public void setFit(boolean fit)

Sets the node fit property.

Parameters:

fit - [in] If TRUE, then the node size is fitted to the label ('height' and 'width' will be ignored), otherwise the size is fixed with the values of 'height' and 'width'.

setFontSize

public void setFontSize(int size)

Sets the node label font size.

Parameters:

size - [in] The node label font size.

setLabelColor

public void setLabelColor(java.awt.Color c)

Sets the color of the label.

Parameters:

c - New value.
com.sparsity.sparksee.gdb
Class NodeShape

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

All Implemented Interfaces:
   Serializable, Comparable

public final class NodeShape
extends Enum

Node shape.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Box</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Box shape.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Round shape.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static NodeShape</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static NodeShape[]</th>
<th>values()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Enum

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

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Comparable

closeTo

Fields

<table>
<thead>
<tr>
<th>public static final</th>
<th>Box</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Box shape.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Round shape.</td>
</tr>
</tbody>
</table>
**Box**

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

Box shape.

**Round**

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

Round shape.

## Methods

**values**

public static NodeShape[] values()

**valueOf**

public static NodeShape valueOf(String name)
com.sparsity.sparksee.gdb
Class Objects

java.lang.Object
    +-com.sparsity.sparksee.gdb.Objects

All Implemented Interfaces:
    Iterable, Closeable, Set

public class Objects
extends Object
implements Set, Closeable, Iterable

Object identifier set class.
It stores a collection of Sparksee object identifiers as a set. As a set, there is no order and no duplicated elements.

This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.

This class is not thread-safe.

ObjectsIterator must be used to traverse all the elements into the set.

When the Objects instance is closed, it closes all existing and non-closed ObjectsIterator instances too.

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

Field Summary

public static InvalidOID
    Invalid object identifier constant.

Method Summary

boolean add(long e)
    Adds an element into the collection.

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

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

long any()
    Gets an element from the collection.

void clear()
    Clears the collection removing all its elements.

void close()
    Closes the Objects instance.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>Objects</strong> combineDifference(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the difference of the two given.</td>
</tr>
<tr>
<td>static <strong>Objects</strong> combineIntersection(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the intersection of the two given.</td>
</tr>
<tr>
<td>static <strong>Objects</strong> combineUnion(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the union of the two given.</td>
</tr>
<tr>
<td>boolean contains(Object o)</td>
<td>Returns true if this collections contains the specified element or Objects.</td>
</tr>
<tr>
<td>boolean contains(Objects objs)</td>
<td>Check if this objects contains the other one.</td>
</tr>
<tr>
<td>boolean containsAll(Collection clctn)</td>
<td>Returns true if this set contains all of the elements of the specified collection.</td>
</tr>
<tr>
<td><strong>Objects</strong> copy()</td>
<td>Creates a new Objects instance as a copy of the given one.</td>
</tr>
<tr>
<td>long copy(Objects objs)</td>
<td>Performs the copy operation.</td>
</tr>
<tr>
<td>long count()</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td>long difference(Objects objs)</td>
<td>Performs the difference operation.</td>
</tr>
<tr>
<td>boolean equals(Object o)</td>
<td>Returns true if the collection is equal to the object.</td>
</tr>
<tr>
<td>boolean equals(Objects objs)</td>
<td>Checks if the given Objects contains the same information.</td>
</tr>
<tr>
<td>boolean exists(long e)</td>
<td>Gets if the given element exists into the collection.</td>
</tr>
<tr>
<td>long intersection(Objects objs)</td>
<td>Performs the intersection operation.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Objects instance has been closed or not.</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
<td>Returns true if this Objects contains no elements.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong> iterator()</td>
<td>Gets an ObjectsIterator.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong> iteratorFromElement(long e)</td>
<td>Gets an ObjectsIterator starting from the given element.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong> iteratorFromIndex(long index)</td>
<td>Gets an ObjectsIterator skipping index elements.</td>
</tr>
<tr>
<td>boolean remove(long e)</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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](https://docs.oracle.com/en/java/javase/11/docs/api/java/lang/Object.html)  
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Methods inherited from interface** [java.util.Set](https://docs.oracle.com/en/java/javase/11/docs/api/java/util/Set.html)  
add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray

**Methods inherited from interface** [java.util.Collection](https://docs.oracle.com/en/java/javase/11/docs/api/java/util/Collection.html)  
add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray

**Methods inherited from interface** [java.lang.Iterable](https://docs.oracle.com/en/java/javase/11/docs/api/java/lang/Iterable.html)  
iterator

**Methods inherited from interface** [java.io.Closeable](https://docs.oracle.com/en/java/javase/11/docs/api/java/io/Closeable.html)  
close

**Fields**
InvalidOID

public static int InvalidOID

Invalid object identifier constant.

Methods

any

public long any()
    throws NoSuchElementException,
           RuntimeException

    Gets an element from the collection.

Returns:
    Any element from the collection.

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

contains

public boolean contains(Object o)

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

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

Returns:
    true if this set contains the specified element or Objects.

remove

public boolean remove(Object o)

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

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

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

Returns:
    true if the set contained the specified element.
equals

public boolean equals(Object o)

Returns true if the collection is equal to the object.

Parameters:
   o - object to compare with the collection.

Returns:
   true if the objects are equal or false otherwise.

difference

public long difference(Objects objs)

Performs the difference operation.

This updates the Objects calling instance removing those existing elements at the given Objects instance.

Parameters:
   objs - [in] Objects instance.

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

combineUnion

public static Objects combineUnion(Objects objs1, Objects objs2)

Creates a new Objects instance which is the union of the two given.

Two given Objects belong to the same Session.

Parameters:
   objs1 - [in] Objects instance.
   objs2 - [in] Objects instance.

Returns:
   New Objects instance.

containsAll

public boolean containsAll(Collection clctn)

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

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

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

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

```java
public ObjectsIterator iteratorFromElement(long e)
```

Gets an ObjectsIterator starting from the given element.

Objects collection has no order, so this method is implementation-dependent. e[in] The first element to traverse in the resulting

**Parameters:**
- e - [in] The first element to traverse in the resulting ObjectsIterator instance.

**Returns:**
- ObjectsIterator instance.

---

**equals**

```java
public boolean equals(Objects objs)
```

Checks if the given Objects contains the same information.

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

**Returns:**
- True if the objects are equal or false otherwise.

---

**add**

```java
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 elements that they may contain.

**Parameters:**
- e - element to be added to this set.

**Returns:**
- true if this set did not already contain the specified element.

---

**copy**

```java
public long copy(Objects objs)
```

Performs the copy operation.

This updates the Objects calling instance and copies the given Objects instance.

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

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

public static Objects combineIntersection(Objects objs1, Objects objs2)

Creates a new Objects instance which is the intersection of the two given.

Two given Objects belong to the same Session.

Parameters:
- objs1 - [in] Objects instance.
- objs2 - [in] Objects instance.

Returns:
New Objects instance.

close

public void close()

Closes the Objects instance.

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

isEmpty

public boolean isEmpty()

Returns true if this Objects contains no elements.

Returns:
true if the collection contains no elements.

contains

public boolean contains(Objects objs)

Check if this objects contains the other one.

Parameters:
- objs - Objects collection.

Returns:
True if it contains the given object.

clear

public void clear()

Clears the collection removing all its elements.
count

public long count()

    Gets the number of elements into the collection.

    Returns:
    The number of elements into the collection.

retainAll

public boolean retainAll(Collection clctn)

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

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

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

remove

public boolean remove(long e)

    Removes an element from the collection.

    Parameters:
    e - [in] Element to be removed.

    Returns:
    TRUE if the element is removed, FALSE if the element was not into the collection.

iteratorFromIndex

public ObjectsIterator iteratorFromIndex(long index)

    Gets an ObjectsIterator skipping index elements.
    Objects collection has no order, so this method is implementation-dependent.

    Parameters:
    index - [in] The number of elements to skip from the beginning. It must be in the range [0..Size).

    Returns:
    ObjectsIterator instance.

union

public long union(Objects objs)
Performs the union operation.
This adds all existing elements of the given Objects instance to the Objects calling instance

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

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

---

### isClosed

**public boolean isClosed()**

Gets if Objects instance has been closed or not.

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

**See Also:**
- [close()](#)
Returns:
an array containing all of the elements in this set.

**toArray**

```java
class com.sparsity.sparksee.gdb.Objects{
    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.

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

**Parameters:**

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

**Returns:**
an array containing the elements of this set.

**addAll**

```java
class com.sparsity.sparksee.gdb.Objects{
    public boolean addAll(Collection clctn)
}
```

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

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

**Parameters:**

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

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

**exists**

```java
class com.sparsity.sparksee.gdb.Objects{
    public boolean exists(long e)
}
```

Gets if the given element exists into the collection.

**Parameters:**


**Returns:**
TRUE if the element exists into the collection, FALSE otherwise.

**combineDifference**

```java
class com.sparsity.sparksee.gdb.Objects{
    public static Objects combineDifference(Objects obj1, Objects obj2)
}
```

Creates a new Objects instance which is the difference of the two given.

Two given Objects belong to the same Session.

**Parameters:**

- `obj1` - [in] Objects instance.
objs2 - [in] Objects instance.

**Returns:**
New Objects instance.

---

### sample

```java
public Objects sample(Objects exclude, long samples)
```

Creates a new Objects instance which is a sample of the calling one.

**Parameters:**
- `exclude` - [in] If not NULL, elements into this collection will be excluded from the resulting one.
- `samples` - [in] Number of elements into the resulting collection.

**Returns:**
Sample collection.

---

### size

```java
public int size()
```

Gets the size of the collection.

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

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

---

### intersection

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

Performs the intersection operation.

Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

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

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

---

### iterator

```java
public ObjectsIterator iterator()
```

Gets an ObjectsIterator.

**Returns:**
ObjectsIterator instance.
copy

public Objects copy()

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

Returns:
The new Objects instance.
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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.io.Closeable

close |

Methods inherited from interface java.util.Iterator

hasNext, next, remove |
Methods

nextObject
public long nextObject()

Gets the next element.

hasNext
public boolean hasNext()

Gets if there are more elements to traverse.

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

remove
public void remove()

Operation not supported.

next
public Long next()

See nextObject().

isClosed
public boolean isClosed()

Gets if ObjectsIterator instance has been closed or not.

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

See Also:
close()

close
public void close()

Closes the ObjectsIterator instance.

It must be called to ensure the integrity of all data.
com.sparsity.sparksee.gdb
Class ObjectType

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

All Implemented Interfaces:
   Serializable, Comparable

public final class ObjectType
extends Enum

Object type enumeration.

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

Field Summary

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

Method Summary

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

Methods inherited from class java.lang.Enum

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

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Comparable

compareTo

Fields
Node
public static final com.sparsity.sparksee.gdb.ObjectType Node
    Node object type.

Edge
public static final com.sparsity.sparksee.gdb.ObjectType Edge
    Edge object type.

Methods

values
public static ObjectType[] values()

valueOf
public static ObjectType valueOf(String name)
com.sparsity.sparksee.gdb
Class OIDList

java.lang.Object
   +-com.sparsity.sparksee.gdb.OIDList

All Implemented Interfaces:
   Iterable

public class OIDList
extends Object
implements Iterable

Sparksee object identifier list.
It stores a Sparksee object identifier list.

Use OIDListIterator to access all elements into this collection.

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

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList(long[] list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance from a long array.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance from a long collection.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList(int numInvalidOIDs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(long attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds a Sparksee object identifier at the end of the list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clears the list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of elements in the list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OIDListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets a new OIDListIterator.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>set(int pos, long oid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets a Sparksee object identifier at the specified position of the list.</td>
<td></td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.lang.Iterable
iterator

Constructors

OIDList
public OIDList(long[] list)

Creates a new instance from a long array.

Parameters:
list - Long array to initialize the instance.

OIDList
public OIDList(Collection col)

Creates a new instance from a long collection.

Parameters:
col - Collection to initialize the instance.

OIDList
public OIDList(int numInvalidOIDs)

Constructor.
This creates a list with N invalid oids.

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

OIDList
public OIDList()

Constructor.
This creates an empty list.

Methods

add
public void add(long attr)
Adds a Sparksee object identifier at the end of the list.

**Parameters:**
- **attr** - [in] Sparksee object identifier.

### clear

```java
public void clear()
```

Clears the list.

### set

```java
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

```java
public OIDListIterator iterator()
```

Gets a new OIDListIterator.

**Returns:**
- OIDListIterator instance.

### count

```java
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext</td>
<td>boolean</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next</td>
<td>Long</td>
<td>See nextOID().</td>
</tr>
<tr>
<td>nextOID</td>
<td>long</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>remove</td>
<td>void</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext ()

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

remove
public void remove()

Operation not supported.

next
public Long next()

See nextOID().

nextOID
public long nextOID()

Gets the next element.
com.sparsity.sparksee.gdb

Class Order

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

All Implemented Interfaces:
   Serializable, Comparable

public final class Order
extends Enum

Order enumeration.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Ascendent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From lower to higher.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Descendent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From higher to lower.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static Order</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Order[]</td>
<td>values()</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.lang.Comparable

compareTo

Fields
Ascendent
public static final com.sparsity.sparksee.gdb.Order Ascendent
   From lower to higher.

Descendent
public static final com.sparsity.sparksee.gdb.Order Descendent
   From higher to lower.

Methods

values
public static Order[] values()

valueOf
public static Order valueOf(String name)
com.sparsity.sparksee.gdb
Class Platform

java.lang.Object
   +-com.sparsity.sparksee.gdb.Platform

public class Platform extends Object

Platform class.

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

Method Summary

<table>
<thead>
<tr>
<th>static void</th>
<th>getStatistics(PlatformStatistics stats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets platform data and statistics.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

getStatistics

public static void getStatistics(PlatformStatistics stats)

  Gets platform data and statistics.

  Parameters:
  stats - [in|out] This updates the given PlatformStatistics.
public class **PlatformStatistics**
extends Object

Platform data and statistics.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PlatformStatistics()</strong></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td><strong>getAvailableMem()</strong></td>
<td>Gets available (free) memory size in Bytes.</td>
</tr>
<tr>
<td>int</td>
<td><strong>getNumCPUs()</strong></td>
<td>Gets the number of CPUs.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getRealTime()</strong></td>
<td>Gets time in microseconds (since epoch).</td>
</tr>
<tr>
<td>long</td>
<td><strong>getSystemTime()</strong></td>
<td>Gets CPU system time.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getTotalMem()</strong></td>
<td>Gets physical memory size in Bytes.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getUserTime()</strong></td>
<td>Gets CPU user time.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

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

### Constructors

**PlatformStatistics**

<table>
<thead>
<tr>
<th>public</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PlatformStatistics()</strong></td>
</tr>
</tbody>
</table>

---

**com.sparsity.sparksee.gdb**

**Class PlatformStatistics**

java.lang.Object

---

**Platform data and statistics.**

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

**Constructor Summary**

<table>
<thead>
<tr>
<th>public</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PlatformStatistics()</strong></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td><strong>getAvailableMem()</strong></td>
<td>Gets available (free) memory size in Bytes.</td>
</tr>
<tr>
<td>int</td>
<td><strong>getNumCPUs()</strong></td>
<td>Gets the number of CPUs.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getRealTime()</strong></td>
<td>Gets time in microseconds (since epoch).</td>
</tr>
<tr>
<td>long</td>
<td><strong>getSystemTime()</strong></td>
<td>Gets CPU system time.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getTotalMem()</strong></td>
<td>Gets physical memory size in Bytes.</td>
</tr>
<tr>
<td>long</td>
<td><strong>getUserTime()</strong></td>
<td>Gets CPU user time.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

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

---

**Constructors**

**PlatformStatistics**

<table>
<thead>
<tr>
<th>public</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PlatformStatistics()</strong></td>
</tr>
</tbody>
</table>
Creates a new instance setting all values to 0.

## Methods

### getRealTime

```
public long getRealTime()
```

Gets time in microseconds (since epoch).

**Returns:**
Time in microseconds (since epoch).

### getAvailableMem

```
public long getAvailableMem()
```

Gets available (free) memory size in Bytes.

**Returns:**
Available (free) memory size in Bytes.

### getTotalMem

```
public long getTotalMem()
```

Gets physical memory size in Bytes.

**Returns:**
Physical memory size in Bytes.

### getSystemTime

```
public long getSystemTime()
```

Gets CPU system time.

**Returns:**
CPU system time.

### getUserTime

```
public long getUserTime()
```

Gets CPU user time.

**Returns:**
CPU user time.

**getNumCPUs**

```java
public int getNumCPUs()
```

Gets the number of CPUs.

**Returns:**

The number of CPUs.
com.sparsity.sparksee.gdb

Class Query

java.lang.Object

public class Query
extends Object

Query class.

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResultSet</td>
<td>execute(String stmt)</td>
<td>Executes the given statement.</td>
</tr>
<tr>
<td>void</td>
<td>setDynamic(String name, Value value)</td>
<td>Sets the value for a dynamic parameter.</td>
</tr>
<tr>
<td>QueryStream</td>
<td>setStream(String stream, QueryStream handler)</td>
<td>Sets a query stream handler.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

setDynamic

public void setDynamic(String name, Value value)

Sets the value for a dynamic parameter.

Parameters:
name - [in] Parameter name
value - [in] Parameter value

setStream

public QueryStream setStream(String stream, QueryStream handler)

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.
Parameters:
    stream - [in] The stream name
    handler - [in] Query stream handler

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

execute

public ResultSet execute(String stmt)

Executes the given statement.

Parameters:
    stmt - [in] Query statement.

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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>QueryContext()</td>
<td>Default constructor.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>newQuery()</td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><code>fetch(ValueList list)</code></td>
<td>Gets the next row and moves the iterator forward.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>prepare(ValueList list)</code></td>
<td>Prepares the stream before it is started.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>start(ResultSetList list)</code></td>
<td>Starts the stream.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

start

public boolean `start(ResultSetList list)`

Starts the stream.

Parameters:

list - [in] Optional list of input ResultSets

Returns:

FALSE on error
fetch

public boolean fetch(ValueList list)

    Gets the next row and moves the iterator forward.

    The end of sequence is denoted by returning TRUE with an empty row. A valid row must contain as many values (even NULL) as expected by the query.

    Parameters:
        list - [out] Storage for the new rows

    Returns:
        TRUE if there is a row or end of sequence, FALSE on error

prepare

public boolean prepare(ValueList list)

    Prepares the stream before it is started.

    Parameters:
        list - [in] Optional list of arguments

    Returns:
        FALSE on error
## com.sparsity.sparksee.gdb ResultSet

### class ResultSet

extends Object

ResultSet class.

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

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>close()</td>
<td>Closes the iterator.</td>
</tr>
<tr>
<td>Value</td>
<td>getColumn(int index)</td>
<td>Gets the value for the given column.</td>
</tr>
<tr>
<td>void</td>
<td>getColumn(int index, Value value)</td>
<td>Gets the value for the given column.</td>
</tr>
<tr>
<td>DataType</td>
<td>getColumnDataType(int index)</td>
<td>Gets the datatype for the given column.</td>
</tr>
<tr>
<td>int</td>
<td>getColumnIndex(String name)</td>
<td>Gets the column index for the given column name.</td>
</tr>
<tr>
<td>String</td>
<td>getColumnName(int index)</td>
<td>Gets the name for the given column.</td>
</tr>
<tr>
<td>String</td>
<td>getJSON(int rows)</td>
<td>Returns rows in JSON format.</td>
</tr>
<tr>
<td>int</td>
<td>getNumColumns()</td>
<td>Gets the number of columns.</td>
</tr>
<tr>
<td>boolean</td>
<td>next()</td>
<td>Fetches the next row.</td>
</tr>
<tr>
<td>void</td>
<td>rewind()</td>
<td>Positions the cursor before the first row.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

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

### Methods
**rewind**

```java
public void rewind()
```

Positions the cursor before the first row.

**getColumn**

```java
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.

```java
public Value getColumn(int index)
```

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

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

**Returns:**
The Value of the given column.

**getColumnIndex**

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

Gets the column index for the given column name.

**Parameters:**
- `name` - [in] Column name.

**Returns:**
Column index.

**next**

```java
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.

Parameters:
rows - [in] Maximum number of rows

Returns:
JSON representation of the next rows in the resultset

close

public void close()

Closes the iterator.

getColumnName

public String getColumnName(int index)

Gets the name for the given column.

Parameters:
index - [in] Column index.

Returns:
Column name.

columnDataType

public DataType getColumnDataType(int index)

Gets the datatype for the given column.

Parameters:
index - [in] Column index.

Returns:
DataType for the given column.

getNumColumns

public int getNumColumns()

Gets the number of columns.
Columns are in the range [0...COLUMNS).
Returns:
The number of columns.
### com.sparsity.sparksee.gdb

**Class ResultSetList**

java.lang.Object  
  +com.sparsity.sparksee.gdb.ResultSetList

**public class ResultSetList extends Object**

It stores a ResultSet list.  
**Author:**  
  Sparsity Technologies http://www.sparsity-technologies.com

#### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ResultSetList()</td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

#### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>ResultSet get(int index)</td>
<td>Returns the ResultSet at the specified position in the list.</td>
</tr>
<tr>
<td>ResultSetListIterator iterator()</td>
<td>Gets a new ResultSetListIterator.</td>
</tr>
</tbody>
</table>

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
get

public ResultSet get(int index)

Returns the ResultSet at the specified position in the list.

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

clear

public void clear()

Clears the list.

iterator

public ResultSetListIterator iterator()

Gets a new ResultSetListIterator.

Returns:
ResultSetListIterator instance.

count

public int count()

Number of elements in the list.

Returns:
Number of elements in the list.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>ResultSet next()</td>
<td>Moves to the next element.</td>
</tr>
</tbody>
</table>

### Methods

**hasNext**

public boolean hasNext()  

Gets if there are more elements.

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

**next**

public ResultSet next()  

Moves to the next element.

Returns:  
The next element.
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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Graph</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Objects</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Query</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rollback</strong></td>
<td>Rollbacks a transaction.</td>
</tr>
</tbody>
</table>

#### Methods inherited from class `java.lang.Object`

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

#### Methods inherited from interface `java.io.Closeable`

- close

---

### rollback

```java
public void rollback()
```

Rollbacks a transaction.

---

### beginUpdate

```java
public void beginUpdate()
```

Begins an update transaction.

---

### newQuery

```java
public Query newQuery()
```

Creates a new Query.

---

### isClosed

```java
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

```java
public void commit()
```
Commits a transaction.

---

**getGraph**

```java
public Graph getGraph()
```

Gets the Graph instance.

**Returns:**
The Graph instance.

---

**close**

```java
public void close()
```

Closes the Session instance.

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

---

**begin**

```java
public void begin()
```

Begins a transaction.

---

**newObjects**

```java
public Objects newObjects()
```

Creates a new Objects instance.

**Returns:**
The new Objects instance.
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

<table>
<thead>
<tr>
<th>public static</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sparksee version.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Sparksee(SparkseeConfig config)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the Sparksee instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>create(String path, String alias)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new Database instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isClosed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if Sparksee instance has been closed or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>open(String path, boolean readOnly)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opens an existing Database instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>restore(String path, String backupFile)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Restores a Database from a backup file.</td>
</tr>
</tbody>
</table>

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
Fields

Version

public static java.lang.String Version

Sparksee version.

Constructors

Sparksee

public Sparksee(SparkseeConfig config)

Creates a new instance.

Parameters:
  config - [in] Sparksee configuration.

Methods

create

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

Creates a new Database instance.

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

Returns:
  A Database instance.

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

restore

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

Page 192 of 305
Restores a Database from a backup file.

See the Graph class Backup method.

**Parameters:**
- `backupFile` - [in] The Backup file to be restored.

**Returns:**
A Database instance.

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

---

**isClosed**

```java
public boolean isClosed()
```

Gets if Sparksee instance has been closed or not.

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

**See Also:**
- `close()`

---

**open**

```java
public Database open(String path,
                      Boolean readOnly)
```

**Throws:**
- `java.io.FileNotFoundException` - If the given file does not exist.
- `java.lang.RuntimeException` - null

Opens an existing Database instance.

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

**Returns:**
A Database instance.

**Throws:**
- `java.io.FileNotFoundException` - If the given file does not exist.
- `java.lang.RuntimeException` - null

---

**close**

```java
public void close()
```

Closes the Sparksee instance.

It must be called to ensure the integrity of all data.
com.sparsity.sparksee.gdb
Class SparkseeConfig

public class SparkseeConfig extends Object

Sparksee configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overridden with values from a properties file (see SparkseeProperties class). Also, this settings can be overridden calling a specific setter.

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

Extent size: 4KB ('sparksee.storage.extentsize' at SparkseeProperties).
Pages per extent: 1 page ('sparksee.storage.extentpages' at SparkseeProperties).
Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at SparkseeProperties).
Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at SparkseeProperties).
Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at SparkseeProperties).
Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at SparkseeProperties).
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).
Recovery enabled: false ('sparksee.io.recovery' at SparkseeProperties).
Recovery cache max size: 1MB ('sparksee.io.recovery.cachesize' at SparkseeProperties).
Recovery checkpoint time: 60 seconds [TimeUnit] ('sparksee.io.recovery.checkpointTime' at SparkseeProperties).
High-availability: false (disabled) ('sparksee.ha' at SparkseeProperties).
High-availability coordinators: "" ('sparksee.ha.coordinators' at SparkseeProperties).
High-availability IP: "" (‘sparksee.ha.ip’ at SparkseeProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] (‘sparksee.ha.sync’ at SparkseeProperties).

High-availability master history: 1D (1 day) [TimeUnit] (‘sparksee.ha.master.history’ at SparkseeProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

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

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

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

---

### Constructor Summary

<table>
<thead>
<tr>
<th>public SparkseeConfig()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>int getCacheMaxSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the maximum size for the cache (all pools) in MB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getCacheStatisticsEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets whether cache statistics are enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String getCacheStatisticsFile()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the cache statistics log file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long getCacheStatisticsSnapshotTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the cache statistics snapshot time in microseconds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int getExtentPages()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the number of pages per extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int getExtentSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the size of a extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String getHighAvailabilityCoordinators()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the coordinators address and port list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getHighAvailabilityEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets whether high availability mode is enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String getHighAvailabilityIP()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the IP address and port of the instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long getHighAvailabilityMasterHistory()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the master's history log.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long getHighAvailabilitySynchronization()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the synchronization polling time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String getLicense()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the license code.</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>LogLevel</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td><code>setHighAvailabilityIP(String ip)</code></td>
</tr>
<tr>
<td><code>setHighAvailabilityMasterHistory(long filePath)</code></td>
</tr>
<tr>
<td><code>setHighAvailabilitySynchronization(long microSeconds)</code></td>
</tr>
<tr>
<td><code>setLicense(String key)</code></td>
</tr>
<tr>
<td><code>setLogFile(String filePath)</code></td>
</tr>
<tr>
<td><code>setLogLevel(LogLevel level)</code></td>
</tr>
<tr>
<td><code>setPoolFrameSize(int extents)</code></td>
</tr>
<tr>
<td><code>setPoolPersistentMaxSize(int frames)</code></td>
</tr>
<tr>
<td><code>setPoolPersistentMinSize(int frames)</code></td>
</tr>
<tr>
<td><code>setPoolTemporaryMaxSize(int frames)</code></td>
</tr>
<tr>
<td><code>setPoolTemporaryMinSize(int frames)</code></td>
</tr>
<tr>
<td><code>setRecoveryCacheMaxSize(int extents)</code></td>
</tr>
<tr>
<td><code>setRecoveryCheckpointTime(long microSeconds)</code></td>
</tr>
<tr>
<td><code>setRecoveryEnabled(boolean status)</code></td>
</tr>
<tr>
<td><code>setRecoveryLogFile(String filePath)</code></td>
</tr>
<tr>
<td><code>setRollbackEnabled(boolean status)</code></td>
</tr>
</tbody>
</table>

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

public SparkseeConfig()

Creates a new instance.

Values are set with default values.

Methods

setLicense

public void setLicense(String key)

Sets the license code.

Parameters:

key - [in] The license code.

getHighAvailabilitySynchronization

public long getHighAvailabilitySynchronization()

Gets the synchronization polling time.

Returns:

The Synchronization polling time.

setExtentPages

public void setExtentPages(int pages)

Sets the number of pages per extent.

Parameters:

pages - [in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

setHighAvailabilityCoordinators

public void setHighAvailabilityCoordinators(String ip)

Sets the coordinators address and port list.

Parameters:

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

gExtentSize

public int getExtentSize()
Gets the size of a extent.

**Returns:**
The size of a extent in KB.

---

### setLogFile

```java
class SparkseeConfig {
    public void setLogFile(String filePath)
    {
        Sets the log file.

        **Parameters:**
        
    }
}
```

---

### setLogLevel

```java
class SparkseeConfig {
    public void setLogLevel(LogLevel level)
    {
        Sets the log level.

        **Parameters:**
        
        level - [in] The LogLevel.
    }
}
```

---

### setCacheStatisticsEnabled

```java
class SparkseeConfig {
    public void setCacheStatisticsEnabled(boolean status)
    {
        Enables or disables cache statistics.

        **Parameters:**
        
        status - [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.
    }
}
```

---

### getLogFile

```java
class SparkseeConfig {
    public String getLogFile()
    {
        Gets the log file.

        **Returns:**
        
        The log file.
    }
}
```

---

### setCacheStatisticsSnapshotTime

```java
class SparkseeConfig {
    public void setCacheStatisticsSnapshotTime(long microSeconds)
    {
        Sets the cache statistics snapshot time.

        Useless if cache statistics are disabled.

        **Parameters:**
    }
}
```
getRecoveryCheckpointTime

public long getRecoveryCheckpointTime()

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

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

cacheStatisticsEnabled

public boolean cacheStatisticsEnabled()

Gets whether cache statistics are enabled or disabled.

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

getPoolPersistentMaxSize

public int getPoolPersistentMaxSize()

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

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

setPoolPersistentMaxSize

public void setPoolPersistentMaxSize(int frames)

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

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

getRecoveryLogFile

public String getRecoveryLogFile()

Gets the recovery log file.

Returns:
The recovery log file.
setHighAvailabilityMasterHistory

public void setHighAvailabilityMasterHistory(long filePath)

Sets the master's history log.

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

setCacheStatisticsFile

public void setCacheStatisticsFile(String filePath)

Sets the cache statistics log file.
Useless if cache statistics are disabled.

Parameters:

cacheStatisticsFile

public String getCacheStatisticsFile()

Gets the cache statistics log file.
Useless if cache statistics are disabled.

Returns:
The cache statistics log file.

cacheStatisticsSnapshotTime

public long getCacheStatisticsSnapshotTime()

Gets the cache statistics log file.

Returns:
The cache statistics log file.
Gets the cache statistics snapshot time in microseconds.
Useless if cache statistics are disabled.

**Returns:**
The cache statistics snapshot time in microseconds.

---

**getPoolTemporaryMaxSize**

```java
public int getPoolTemporaryMaxSize()
```

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

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

---

**setRecoveryEnabled**

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

Enables or disables the recovery.

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

---

**getLicense**

```java
public String getLicense()
```

Gets the license code.

**Returns:**
The license code.

---

**setPoolTemporaryMinSize**

```java
public void setPoolTemporaryMinSize(int frames)
```

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

**Parameters:**
- **frames** - [in] The minimum size for the temporary pool in number of frames. It must be non-negative.

---

**getHighAvailabilityIP**

```java
public String getHighAvailabilityIP()
```

Gets the IP address and port of the instance.

**Returns:**
The IP address and port of the instance.

### getLogLevel

`public LogLevel getLogLevel()`

Gets the log level.

**Returns:**

The LogLevel.

### setHighAvailabilitySynchronization

`public void setHighAvailabilitySynchronization(long microSeconds)`

Sets the synchronization polling time.

**Parameters:**

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

### setCacheMaxSize

`public void setCacheMaxSize(int megaBytes)`

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

**Parameters:**

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

### getPoolPersistentMinSize

`public int getPoolPersistentMinSize()`

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

**Returns:**

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

### setHighAvailabilityEnabled

`public void setHighAvailabilityEnabled(boolean status)`

Enables or disables high availability mode.

**Parameters:**

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

public void setRecoveryCacheMaxSize(int extents)

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

Parameters:
    extents - [in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

setHighAvailabilityIP

public void setHighAvailabilityIP(String ip)

Sets the IP address and port of the instance.

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

setExtentSize

public void setExtentSize(int kBytes)

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

Parameters:
    kBytes - [in] The size of a extent size in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

setRollbackEnabled

public void setRollbackEnabled(boolean status)

Enables or disables the rollback.

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

getExtentPages

public int getExtentPages()

Gets the number of pages per extent.

Returns:
    The number of pages per extent.
setPoolTemporaryMaxSize

public void setPoolTemporaryMaxSize(int frames)

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

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

getHighAvailabilityEnabled

public boolean getHighAvailabilityEnabled()

Gets whether high availability mode is enabled or disabled.

Returns:
TRUE if high availability mode is enabled, FALSE otherwise.

getRecoveryEnabled

public boolean getRecoveryEnabled()

Gets whether the recovery is enabled or disabled.

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

setPoolPersistentMinSize

public void setPoolPersistentMinSize(int frames)

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

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

cacheMaxSize

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

**Returns:**
The size of a pool frame in number of extents.

---

**getRollbackEnabled**

```java
public boolean getRollbackEnabled()
```

Gets whether the rollback is enabled or disabled.

**Returns:**
TRUE if the rollback is enabled, FALSE otherwise.

---

**getPoolTemporaryMinSize**

```java
public int getPoolTemporaryMinSize()
```

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

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

---

**setRecoveryCheckpointTime**

```java
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 committed transaction.

---

**getRecoveryCacheMaxSize**

```java
public int getRecoveryCacheMaxSize()
```

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

**Returns:**
The maximum size for the recovery log cache in extents.

---

**getHighAvailabilityMasterHistory**

```java
public long getHighAvailabilityMasterHistory()
```

Gets the master's history log.
Returns:
The master's history log.

**setRecoveryLogFile**

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

Sets the recovery log file.

**Parameters:**

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

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String get(String key, String def)</td>
<td>Gets a property.</td>
</tr>
<tr>
<td>static boolean getBoolean(String key, boolean def)</td>
<td>Gets a property as a boolean.</td>
</tr>
<tr>
<td>static int getInteger(String key, int def)</td>
<td>Gets a property as an integer.</td>
</tr>
<tr>
<td>static long getTimeUnit(String key, long def)</td>
<td>Gets a property as a time unit.</td>
</tr>
<tr>
<td>static void load(String path)</td>
<td>Loads properties from the given file path.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

### Methods

**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.

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.

load

public static void load(String path)

Loads properties from the given file path.

Parameters:
path - [in] File path to load properties from.
getInteger

public static int getInteger(String key,
               int def)

    Gets a property as an integer.

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

Returns:
    The property value, or def if the key is not found or in case of error.
com.sparsity.sparksee.gdb
Class StringList

does not extend Object
implements 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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public StringList(Collection col)</td>
<td>Creates a new instance from an string collection.</td>
</tr>
<tr>
<td>public StringList()</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public StringList(String[] list)</td>
<td>Creates a new instance from an string array.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add(String str)</td>
<td>Adds a String at the end of the list.</td>
</tr>
<tr>
<td>void clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>StringListIterator iterator()</td>
<td>Gets a new StringListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**StringList**

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

Creates a new instance from an string collection.

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

**StringList**

```java
public StringList()
```

Constructor.

This creates an empty list.

**StringList**

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

Creates a new instance from an string array.

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

Methods

**clear**

```java
public void clear()
```

Cleats the list.

**iterator**

```java
public StringListIterator iterator()
```

Gets a new StringListIterator.

**Returns:**
- StringListIterator instance.

**count**

```java
public int count()
```
Number of elements in the list.

**Returns:**
Number of elements in the list.

---

**add**

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

Adds a String at the end of the list.

**Parameters:**
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext()</td>
<td>boolean</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next()</td>
<td>String</td>
<td>See nextString().</td>
</tr>
<tr>
<td>nextString()</td>
<td>String</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>remove()</td>
<td>void</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()

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

remove
public void remove()

Operation not supported.

next
public String next()

See nextString().

nextString
public String nextString()

Gets the next element.
public class TextStream
extends Object
implements Closeable

TextStream class.

It allows for reading and writing Text attribute values.

It is very important to close the stream once no more reading or writing 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) execute 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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public TextStream(boolean append)</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>Closes the stream.</td>
</tr>
<tr>
<td>boolean isNull()</td>
<td>Returns TRUE if the stream is not available.</td>
</tr>
<tr>
<td>int read(char[] dataOUT, int length)</td>
<td>Read data from the stream.</td>
</tr>
<tr>
<td>void write(char[] dataIN, int length)</td>
<td>Write data to the stream.</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.io.Closeable

close

Constructors

TextStream

public TextStream(boolean append)

Creates a new instance.

A TextStream only can be created by the user to write data.

Parameters:
append - [in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to
write from the begining of the stream.

Methods

read

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

Read data from the stream.

Parameters:
  dataOUT - [out] Buffer to read data to. It must be allocated by the user.
  length - [in] Length of the given data buffer. It must be > 0.

Returns:
  Amount of read data (<= length). If 0, there is no more data to be read from the stream.

isNull

public boolean isNull()

Returns TRUE if the stream is not available.

It returns for reading or writing data.

Returns:
  FALSE if the stream is ready

write

public void write(char[] dataIN,
                int length)
Write data to the stream.

**Parameters:**

- **dataIn** - [in] Buffer to write data from.
- **length** - [in] Length of the data buffer. It must be > 0.

### close

```java
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.
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

<table>
<thead>
<tr>
<th>public static</th>
<th>EdgesType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifier for all edgeType attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static</th>
<th>GlobalType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global type identifier constant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid type identifier constant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static</th>
<th>NodesType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifier for all nodeType attributes.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>getAreNeighborsIndexed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is an edge type with neighbors index.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getId()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee type identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getIsDirected()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is a directed edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getIsRestricted()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is a restricted edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getName()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the unique type name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getNumObjects()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of objects belonging to the type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ObjectType</th>
<th>getObjectType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the object type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getRestrictedFrom()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the tail or source type identifier for restricted edge types.</td>
</tr>
</tbody>
</table>
### 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

### Fields

#### EdgesType

public static int **EdgesType**

Identifier for all edgeType attributes.

#### GlobalType

public static int **GlobalType**

Global type identifier constant.

#### InvalidType

public static int **InvalidType**

Invalid type identifier constant.

#### NodesType

public static int **NodesType**

Identifier for all nodeType attributes.

### Methods

#### getRestrictedFrom

public int **getRestrictedFrom()**

Gets the tail or source type identifier for restricted edge types.

**Returns:**

For restricted edge types, the tail or source type identifier, the Type InvalidType otherwise.
getAreNeighborsIndexed
public boolean getAreNeighborsIndexed()

    Gets if this is an edge type with neighbors index.

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

getObjectType
public ObjectType getObjectType()

    Gets the object type.

    Returns:
    The object type.

getRestrictedTo
public int getRestrictedTo()

    Gets the head or target type identifier for restricted edge types.

    Returns:
    For restricted edge types, the head or target type identifier, the Type InvalidType otherwise.

getIsRestricted
public boolean getIsRestricted()

    Gets if this is a restricted edge type.

    Returns:
    TRUE for restricted edge types, FALSE otherwise.

getNumObjects
public long getNumObjects()

    Gets the number of objects belonging to the type.

    Returns:
    The number of objects belonging to the type.

getId
public int getId()
Gets the Sparksee type identifier.

**Returns:**
  The Sparksee type identifier.

---

**getIsDirected**

class com.sparsity.sparksee.gdb

public boolean getIsDirected()

  Gets if this is a directed edge type.

**Returns:**
  TRUE for directed edge types, FALSE otherwise.

---

**getName**

class com.sparsity.sparksee.gdb

public String getName()

  Gets the unique type name.

**Returns:**
  The unique type name.
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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>TypeList(int[] list)</td>
<td>Creates a new instance from an integer array.</td>
</tr>
<tr>
<td>public</td>
<td>TypeList(Collection col)</td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public</td>
<td>TypeList()</td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add(int type)</td>
<td>Adds a Sparksee type identifier at the end of the list.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td>count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>TypeListIterator</td>
<td>iterator()</td>
<td>Gets a new TypeListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**TypeList**

```java
public TypeList(int[] list)
```

Creates a new instance from an integer array.

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

```java
public TypeList(Collection col)
```

Creates a new instance from an integer collection.

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

```java
public TypeList()
```

Constructor.

This creates an empty list.

Methods

**add**

```java
public void add(int type)
```

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

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

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public TypeListIterator iterator()
```
Gets a new TypeListIterator.

**Returns:**
TypeListIterator instance.

count

```java
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean hasNext()</td>
</tr>
<tr>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>Integer next()</td>
</tr>
<tr>
<td>See nextType().</td>
</tr>
<tr>
<td>int nextType()</td>
</tr>
<tr>
<td>Gets the next element.</td>
</tr>
<tr>
<td>void remove()</td>
</tr>
<tr>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext() 

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

---

**remove**

def public void remove()  

Operation not supported.

---

**next**

def public Integer next()  

See nextType().

---

**nextType**

def public int nextType()  

Gets the next element.
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

<table>
<thead>
<tr>
<th>public static</th>
<th>MaxLengthString</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum number of characters allowed for a String.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Value(Value value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Copy constructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>Value()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>compare(Value value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares with the given Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Object value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See compare().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Value value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See compare().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Object other)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Value value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares with the given Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getBoolean()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets Boolean Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataType</th>
<th>getDataType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the DataType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>double</th>
<th>getDouble()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets Double Value.</td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>int</td>
<td>getInteger()</td>
</tr>
<tr>
<td>long</td>
<td>getLong()</td>
</tr>
<tr>
<td>long</td>
<td>getOID()</td>
</tr>
<tr>
<td>String</td>
<td>getString()</td>
</tr>
<tr>
<td>long</td>
<td>getTimestamp()</td>
</tr>
<tr>
<td>Calendar</td>
<td>getTimestampAsCalendar()</td>
</tr>
<tr>
<td>Date</td>
<td>getTimestampAsDate()</td>
</tr>
<tr>
<td>int</td>
<td>hashCode()</td>
</tr>
<tr>
<td>boolean</td>
<td>isNull()</td>
</tr>
<tr>
<td>Value</td>
<td>set(Value value)</td>
</tr>
<tr>
<td>Value</td>
<td>setBoolean(boolean value)</td>
</tr>
<tr>
<td>void</td>
<td>setBooleanVoid(boolean value)</td>
</tr>
<tr>
<td>Value</td>
<td>setDouble(double value)</td>
</tr>
<tr>
<td>void</td>
<td>setDoubleVoid(double value)</td>
</tr>
<tr>
<td>Value</td>
<td>setInteger(int value)</td>
</tr>
<tr>
<td>void</td>
<td>setIntegerVoid(int value)</td>
</tr>
<tr>
<td>Value</td>
<td>setLong(long value)</td>
</tr>
<tr>
<td>void</td>
<td>setLongVoid(long value)</td>
</tr>
<tr>
<td>Value</td>
<td>setNull()</td>
</tr>
<tr>
<td>void</td>
<td>setNullVoid()</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Value setOID(long value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>void setOIDVoid(long value)</code></td>
<td>Sets the OID Value.</td>
</tr>
<tr>
<td><code>Value setString(String value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>void setStringVoid(String value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>Value setTimestamp(Calendar value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>Value setTimestamp(Date value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>Value setTimestamp(int year, int month, int day, int hour, int minutes, int seconds, int millisec)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>void setTimestampVoid(int year, int month, int day, int hour, int minutes, int seconds, int millisecs)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>void setTimestampVoid(long value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>void setVoid(Value value)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>String toString()</code></td>
<td>Gets a String representation of the Value.</td>
</tr>
<tr>
<td><code>String toString(String str)</code></td>
<td>Gets a string representation of the Value.</td>
</tr>
</tbody>
</table>

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
Value
public Value(Value value)

Copy constructor.

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

Value
public Value()

Creates a new instance.
It creates a NULL Value.

Methods

setLongVoid
public void setLongVoid(long value)

Sets the Value.

Parameters:

setTimestamp
public Value setTimestamp(Date value)

Sets the Value.

Parameters:
value - New value.

Returns:
The calling instance.

getTimestampAsCalendar
public Calendar getTimestampAsCalendar()

Gets the Value as a Calendar instance.

Returns:
The returning Calendar instance.
equals
public boolean equals(Object other)

Parameters:
other - null

getOID
public long getOID()

Gets OID Value.
This must be an non-NULL OID Value.

Returns:
The OID Value.

set
public Value set(Value value)

Sets the Value.

Parameters:
value - New value.

Returns:
The calling instance.

setString
public Value setString(String value)

Sets the Value.

Parameters:
value - New value.

Returns:
The calling instance.

hashCode
public int hashCode()
setBoolean

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

Sets the Value.

**Parameters:**
- `value` - New value.

**Returns:**
- The calling instance.

getBoolean

```java
public boolean getBoolean()
```

Gets Boolean Value.

This must be a non-NULL Boolean Value.

**Returns:**
- The Boolean Value.

setDouble

```java
public Value setDouble(double value)
```

Sets the Value.

**Parameters:**
- `value` - New value.

**Returns:**
- The calling instance.

setNullVoid

```java
public void setNullVoid()
```

Sets the Value to NULL.

setTimestampVoid

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

Sets the Value.

**Parameters:**
**setStringVoid**

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

Sets the Value.

**Parameters:**

---

**setIntegerVoid**

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

Sets the Value.

**Parameters:**

---

**getDataType**

```java
public DataType getDataType()
```

Gets the DataType.

Value cannot be NULL.

**Returns:**
- The DataType.

---

**setLong**

```java
public Value setLong(long value)
```

Sets the Value.

**Parameters:**
- value - New value.

**Returns:**
- The calling instance.

---

**getTimestampAsDate**

```java
public Date getTimestampAsDate()
```

Gets the Value as a Date instance.

**Returns:**
- The returning Date instance.
getString

public String getString()

Gets String Value.

This must be a non-NULL String Value.

Returns:
The String Value.

setDoubleVoid

public void setDoubleVoid(double value)

Sets the Value.

Parameters:
value - [in] New Double value.

equals

public boolean equals(Value value)

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

Parameters:
value - Given value to compare to.

Returns:
TRUE if this Value is equal to the given one; FALSE otherwise.

isNull

public boolean isNull()

Gets if this is a NULL Value.

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

setVoid

public void setVoid(Value value)

Sets the Value.

Parameters:
value - [in] New value.
setTimestamp

public Value setTimestamp(int year, 
    int month, 
    int day, 
    int hour, 
    int minutes, 
    int seconds, 
    int millisec)

Sets the Value.

Parameters:
    year - The year (>=1970).
    month - The month ([1..12]).
    day - The day of the month ([1..31]).
    hour - The hour ([0..23]).
    minutes - The minutes ([0..59]).
    seconds - The seconds ([0..59]).
    millisec - The milliseconds ([0..999]).

Returns:
    The calling instance.

setTimestampVoid

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

Sets the Value.

Parameters:
    month - [in] The month ([1..12]).
    day - [in] The day of the month ([1..31]).
    hour - [in] The hour ([0..23]).
    minutes - [in] The minutes ([0..59]).
    seconds - [in] The seconds ([0..59]).
    millisecs - [in] The milliseconds ([0..999]).

setOIDVoid

public void setOIDVoid(long value)

Sets the OID Value.

Parameters:
    value - [in] New OID value.
**getDouble**

```java
public double getDouble()
```

Gets Double Value.

This must be a non-NULL Double Value.

**Returns:**

The Double Value.

---

**toString**

```java
public String toString(String str)
```

Gets a string representation of the Value.

**Parameters:**

str - String to be used. It is cleared and set with the string representation of the Value.

**Returns:**

The given string which has been updated.

---

**setNull**

```java
public Value setNull()
```

Sets the Value to NULL.

**Returns:**

The calling instance.

---

**setOID**

```java
public Value setOID(long value)
```

Sets the Value.

**Parameters:**

value - New value.

**Returns:**

The calling instance.

---

**compare**

```java
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.

**getLong**

```java
generic public long getLong()
```

Gets Long Value.

This must be a non-NULL Long Value.

**Returns:**
The Long Value.

**compareTo**

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

See compare().

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

**Parameters:**
- value - null

**setBooleanVoid**

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

Sets the Value.

**Parameters:**

**toString**

```java
generic public String toString()
```

Gets a String representation of the Value.

**getTimestamp**

```java
generic public long getTimestamp()
```

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

**Returns:**
The Timestamp Value.

**setTimestamp**

```java
generic public Value setTimestamp(Calendar value)
```
Sets the Value.

**Parameters:**
  - `value` - New value.

**Returns:**
  - The calling instance.

---

### setInteger

```java
public Value setInteger(int value)
```

Sets the Value.

**Parameters:**
  - `value` - New value.

**Returns:**
  - The calling instance.

---

### getInteger

```java
public int getInteger()
```

Gets Integer Value.

This must be a non-NULL Integer Value.

**Returns:**
  - The Integer Value.

---

### compareTo

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

See `compare()`.

**Parameters:**
  - `value` - null
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

<table>
<thead>
<tr>
<th>Constructor</th>
<th>ValueList ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add (Value value)</td>
<td>Adds a value to the end of the list.</td>
</tr>
<tr>
<td>void clear ()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int count ()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>Value get (int index)</td>
<td>Returns the Value at the specified position in the list.</td>
</tr>
<tr>
<td>ValueListIterator iterator ()</td>
<td>Gets a new ValueListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Constructors

ValueList

public ValueList ()

Constructor.
This creates an empty list.
Methods

get
public Value get(int index)

Returns the Value at the specified position in the list.

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

clear
public void clear()

Clears the list.

add
public void add(Value value)

Adds a value to the end of the list.

Parameters:
value - [in] The value to add

iterator
public ValueListIterator iterator()

Gets a new ValueListIterator.

Returns:
ValueListIterator instance.

count
public int count()

Number of elements in the list.

Returns:
Number of elements in the list.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext()</td>
<td>boolean</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next()</td>
<td>Value</td>
<td>Moves to the next element.</td>
</tr>
</tbody>
</table>

### Methods

**hasNext**

```java
public boolean hasNext()
```

Gets if there are more elements.

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

**next**

```java
public Value next()
```

Moves to the next element.

**Returns:**
The next element.
public class Values
extends Object
implements Iterable, Closeable

Value set class.
This is a set of Value instances, that is there is no duplicated elements.
Use a ValuesIterator to traverse all the elements into the set.
When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>long count()</td>
</tr>
<tr>
<td>boolean isClosed()</td>
</tr>
<tr>
<td>ValuesIterator iterator()</td>
</tr>
<tr>
<td>ValuesIterator iterator(Order order)</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.lang.Iterable
iterator

Methods inherited from interface java.io.Closeable
close
Methods

**iterator**

```java
public ValuesIterator iterator()
```

See iterator().

Creates an Ascendent iterator.

**count**

```java
public long count()
```

Gets the number of elements into the collection.

**Returns:**

The number of elements into the collection.

**iterator**

```java
public ValuesIterator iterator(Order order)
```

Gets a ValuesIterator.

**Parameters:**

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

**Returns:**

ValuesIterator instance.

**isClosed**

```java
public boolean isClosed()
```

Gets if Values instance has been closed or not.

**Returns:**

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

**See Also:**

- `close()`

**close**

```java
public void close()
```

Closes the Values instance.

It must be called to ensure the integrity of all data.
com.sparsity.sparksee.gdb
Class ValuesIterator

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>Closes the ValuesIterator instance.</td>
</tr>
<tr>
<td>boolean hasNext()</td>
<td>Gets if there are more elements to traverse.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if ValuesIterator instance has been closed or not.</td>
</tr>
<tr>
<td>Value next()</td>
<td>Gets the next element to traverse.</td>
</tr>
<tr>
<td>void remove()</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

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

Methods inherited from interface java.io.Closeable
close

Methods inherited from interface java.util.Iterator
hasNext, next, remove

Methods
**hasNext**

```java
public boolean hasNext()
```

Gets if there are more elements to traverse.

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

---

**remove**

```java
public void remove()
```

Operation not supported.

---

**next**

```java
public Value next()
```

Gets the next element to traverse.

**Returns:**
The next element.

---

**isClosed**

```java
public boolean isClosed()
```

Gets if ValuesIterator instance has been closed or not.

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

**See Also:**
`close()`

---

**close**

```java
public void close()
```

Closes the ValuesIterator instance.

It must be called to ensure the integrity of all data.
Package
com.sparsity.sparksee.io
com.sparsity.sparksee.io
Class CSVReader

java.lang.Object
   +-com.sparsity.sparksee.io.RowReader
      +-com.sparsity.sparksee.io.CSVReader

public class CSVReader
   extends RowReader

CSVReader interface.

A very simple CSV reader.

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

Using the format RFC 4180.

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

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

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

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

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

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

For example:

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

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

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

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

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

---

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public CSVReader()</td>
<td></td>
<td>Constructs CSVReader.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td></td>
<td>Closes the reader.</td>
</tr>
<tr>
<td>intgetRow()</td>
<td></td>
<td>The row number for the current row.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>open(String filePath)</td>
<td>Opens the source file path.</td>
<td></td>
</tr>
<tr>
<td>read(StringList row)</td>
<td>Reads the next row as a string array.</td>
<td></td>
</tr>
<tr>
<td>reset()</td>
<td>Moves the reader to the beginning.</td>
<td></td>
</tr>
<tr>
<td>setLocale(String localeStr)</td>
<td>Sets the locale that will be used to read the file.</td>
<td></td>
</tr>
<tr>
<td>setMultilines(int numExtraLines)</td>
<td>Allows the use of fields with more than one line.</td>
<td></td>
</tr>
<tr>
<td>setNumLines(int numLines)</td>
<td>Used to limit the number of lines that will be read.</td>
<td></td>
</tr>
<tr>
<td>setQuotes(String quotes)</td>
<td>Sets the character used to quote fields.</td>
<td></td>
</tr>
<tr>
<td>setSeparator(String sep)</td>
<td>Sets the character used to separate fields in the file.</td>
<td></td>
</tr>
<tr>
<td>setSingleLine()</td>
<td>Only allows single line fields.</td>
<td></td>
</tr>
<tr>
<td>setStartLine(int startLine)</td>
<td>Sets the number of lines to be skipped from the beginning.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class `com.sparsity.sparksee.io.RowReader`
- close, getRow, read, reset

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

Constructors

**CSVReader**

public **CSVReader**()

Constructs CSVReader.

Methods

**reset**

public boolean reset()

throws IOException
Moves the reader to the beginning.

Restarts the reader.

**Returns:**
true if the reader can be restarted, false otherwise.

**Throws:**
java.io.IOException - If bad things happen during the restart.

---

**close**

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

Closes the reader.

---

**setNumLines**

```java
public void setNumLines(int numLines)
```

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

**Parameters:**
numLines - [in] The maximum number of lines to read (0 == unlimited)

---

**setMultilines**

```java
public void setMultilines(int numExtralines)
```

Allows the use of fields with more than one line.

**Parameters:**
numExtralines - [in] Maximum number of extra lines for each column (0==unlimited, N==N+1 total rows).

---

**setSeparator**

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

Sets the character used to separate fields in the file.

**Parameters:**
sep - [in] Separator character.

**Throws:**
java.lang.RuntimeException - null

---

**setQuotes**

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

Page 250 of 305
Sets the character used to quote fields.

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

**Throws:**
- `java.lang.RuntimeException` - null

---

**open**

```java
public void open(String filePath)
throws IOException
```

Opens the source file path.

File can be optionally compressed in GZIP format.

**Parameters:**
- `filePath` - [in] CSV file path.

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

---

**setSingleLine**

```java
public void setSingleLine()
```

Only allows single line fields.

---

**read**

```java
public boolean read(StringList row)
throws IOException
```

Reads the next row as a string array.

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

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

**Throws:**
- `java.io.IOException` - If bad things happen during the read.

---

**setStartLine**

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

Sets the number of lines to be skipped from the beginning.
Parameters:
startLine - [in] The line number to skip for start reading

setLocale
public void setLocale(String localeStr)

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

Parameters:
localeStr - [in] The locale string for the file encoding.

getRow
public int getRow()
throws IOException

The row number for the current row.

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

Throws:
java.io.IOException - If it fails.
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

<table>
<thead>
<tr>
<th>public</th>
<th><strong>CSVWriter()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th><strong>close()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Closes the writer.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>open(String f)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens the output file path.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setAutoQuotes(boolean autoquotes)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets on/off the automatic quote mode.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setForcedQuotes(BooleanList forcequotes)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setLocale(String localeStr)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the locale that will be used to write the file.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setQuotes(String quotes)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the character used to quote fields.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setSeparator(String sep)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the character used to separate fields in the file.</td>
<td></td>
</tr>
</tbody>
</table>
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,

Constructors

CSVWriter

public CSVWriter()
  Creates a new instance.

Methods

setAutoQuotes

public void setAutoQuotes(boolean autoquotes)
  Sets on/off the automatic quote mode.
  
  If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced quotes are
  provided, there will not be any quote.

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

setSeparator

public void setSeparator(String sep)
  throws RuntimeException
  Sets the character used to separate fields in the file.

  Parameters:
    sep - [in] Separator character.

  Throws:
    java.lang.RuntimeException - null

setQuotes

public void setQuotes(String quotes)
  throws RuntimeException
Sets the character used to quote fields.

**Parameters:**

quotes - [in] Quote character.

**Throws:**

java.lang.RuntimeException - null

---

**setLocale**

public void setLocale(String localeStr)

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

**Parameters:**

localeStr - [in] The locale string for the file encoding.

---

**write**

public void write(StringList row)

throws IOException,

RuntimeException

Writes the next row.

**Parameters:**

row - [in] Row of data.

**Throws:**

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

java.lang.RuntimeException - null

---

**setForcedQuotes**

public void setForcedQuotes(BooleanList forcequotes)

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

**Parameters:**

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

---

**close**

public void close()

throws IOException,

RuntimeException

Closes the writer.
open

public void open(String f)
throws IOException

Opens the output file path.

Parameters:
   f - [in] Output file path.

Throws:
   java.io.IOException - If bad things happen opening the file.
com.sparsity.sparksee.io

Class EdgeTypeExporter

public class EdgeTypeExporter
extends TypeExporter

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

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

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

Constructor Summary

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

public EdgeTypeExporter()
Creates a new instance.

Method Summary

void register(TypeExporterListener tel)
 Registers a new listener.

void run()
 See the TypeExporter class Run method.

void setAttributes(AttributeList attrs)
 Sets the list of Attributes.

void setFrequency(int freq)
 Sets the frequency of listener notification.

void setGraph(Graph graph)
 Sets the graph that will be exported.

void setHeadAttribute(int attr)
 Sets the attribute that will be used to get the value to be dumped for the head of the edge.

void setHeader(boolean header)
 Sets the presence of a header row.

void setHeadPosition(int pos)
 Sets the position (index column) of the head attribute in the exported data.
void setRowWriter(RowWriter rw)
Sets the output data destination.

void setTailAttribute(int attr)
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

void setTailPosition(int pos)
Sets the position (index column) of the tail attribute in the exported data.

void setType(int type)
Sets the type to be exported.

Methods inherited from class com.sparsity.sparksee.io.TypeExporter
register, run, setAttributes, setFrequency, setGraph, setHeader, setRowWriter, setType

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

Constructors

EdgeTypeExporter
public EdgeTypeExporter(RowWriter rowWriter,
Graph graph,
int type,
AttributeList attrs,
int hPos,
int tPos,
int hAttr,
int tAttr)

Creates a new instance.

Parameters:
graph - [in] Graph.
type - [in] Type identifier.
attrs - [in] Attribute identifiers to be exported.
hPos - [in] The position (index column) for the head value.
tPos - [in] The position (index column) for the tail value.
hAttr - [in] The attribute identifier to get the value to be dumped for the head.
tAttr - [in] The attribute identifier to get the value to be dumped for the tail.

EdgeTypeExporter
public EdgeTypeExporter()

Creates a new instance.
Methods

**setTailAttribute**

```java
class EdgeTypeExporter {
    public void setTailAttribute(int attr) {
        // Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
        // Parameters:
        //     attr - [in] Tail Attribute
    }
}
```

**setFrequency**

```java
class EdgeTypeExporter {
    public void setFrequency(int freq) {
        // Sets the frequency of listener notification.
        // Parameters:
        //     freq - [in] Frequency in number of rows managed to notify progress to all listeners
    }
}
```

**setHeadAttribute**

```java
class EdgeTypeExporter {
    public void setHeadAttribute(int attr) {
        // Sets the attribute that will be used to get the value to be dumped for the head of the edge.
        // Parameters:
        //     attr - [in] Head Attribute
    }
}
```

**setType**

```java
class EdgeTypeExporter {
    public void setType(int type) {
        // Sets the type to be exported.
        // Parameters:
        //     type - [in] Type identifier.
    }
}
```

**setTailPosition**

```java
class EdgeTypeExporter {
    public void setTailPosition(int pos) {
        // Sets the position (index column) of the tail attribute in the exported data.
        // Parameters:
        //     pos - [in] Tail position
    }
}
```
**setRowWriter**

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

Sets the output data destination.

**Parameters:**
- `rw` - [in] Input RowWriter.

**register**

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

Registers a new listener.

**Parameters:**
- `tel` - [in] TypeExporterListener to be registered.

**run**

```java
public void run()
```

```java
throws IOException, RuntimeException
```

See the TypeExporter class Run method.

**setGraph**

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

Sets the graph that will be exported.

**Parameters:**
- `graph` - [in] Graph.

**setHeader**

```java
public void setHeader(boolean header)
```

Sets the presence of a header row.

**Parameters:**
- `header` - [in] If TRUE, a header row is dumped with the name of the attributes.

**setHeadPosition**

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

(continued from last page)
Sets the position (index column) of the head attribute in the exported data.

**Parameters:**

- `pos` - [in] Head position

---

**setAttribute**

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

Sets the list of Attributes.

**Parameters:**

- `attrs` - [in] Attribute identifiers to be exported
com.sparsity.sparksee.io
Class EdgeTypeLoader

java.lang.Object
   +-com.sparsity.sparksee.io.TypeLoader
      +-com.sparsity.sparksee.io.EdgeTypeLoader

public class EdgeTypeLoader
   extends TypeLoader

   EdgeTypeLoader class.

   Specific TypeLoader implementation for edge types.

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

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

Constructor Summary

| public | \begin{verbatim}EdgeTypeLoader(RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr)\end{verbatim} |
|        | Creates a new instance. |

| public | \begin{verbatim}EdgeTypeLoader()\end{verbatim} |
|        | Creates a new instance. |

Method Summary

| void | \begin{verbatim}register(TypeLoaderListener tel)\end{verbatim} |
|      | Registers a new listener. |

| void | \begin{verbatim}run()\end{verbatim} |
|      | See the TypeLoader class Run method. |

| void | \begin{verbatim}runNPhases(int partitions)\end{verbatim} |
|      | See the TypeLoader class RunNPhases method. |

| void | \begin{verbatim}runTwoPhases()\end{verbatim} |
|      | See the TypeLoader class RunTwoPhases method. |

| void | \begin{verbatim}setAttributePositions(Int32List attrsPos)\end{verbatim} |
|      | Sets the list of attribute positions. |

| void | \begin{verbatim}setAttributes(AttributeList attrs)\end{verbatim} |
|      | Sets the list of Attributes. |

| void | \begin{verbatim}setFrequency(int freq)\end{verbatim} |
|      | Sets the frequency of listener notification. |

| void | \begin{verbatim}setGraph(Graph graph)\end{verbatim} |
|      | 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 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 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

Constructors

EdgeTypeLoader
public EdgeTypeLoader(RowReader rowReader,
Graph graph,
Int Type,
AttributeList attrs,
Int32List attrsPos,
int hPos,
int tPos,
int hAttr,
int tAttr)

Creates a new instance.
Parameters:
- rowReader - [in] Input RowReader.
- graph - [in] Graph.
- type - [in] Type identifier.
- attrs - [in] Attribute identifiers to be loaded.
- attrsPos - [in] Attribute positions (column index >=0) to all listeners.
- hPos - [in] The position (index column) for the head value.
- tPos - [in] The position (index column) for the tail value.
- hAttr - [in] The attribute identifier for the head.
- tAttr - [in] The attribute identifier for the tail.

EdgeTypeLoader

```java
public EdgeTypeLoader()
```

Creates a new instance.

Methods

**setTailAttribute**

```java
public void setTailAttribute(int attr)
```

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

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

**Parameters:**
- attr - [in] Tail Attribute

**setFrequency**

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

Sets the frequency of listener notification.

**Parameters:**
- freq - [in] Frequency in number of rows managed to notify progress to all listeners

**setLogOff**

```java
public void setLogOff()
```

Turns off all the error reporting.

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

**setLogError**

```java
public void setLogError(String path)
throws IOException
```

```java
Page 264 of 305
```
Sets a log error file.

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

**Parameters:**
- **path** - [in] The path to the error log file.

**Throws:**
- java.io.IOException - If bad things happen opening the file.

### setHeadAttribute

```java
public void setHeadAttribute(int attr)
```

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

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

**Parameters:**
- **attr** - [in] Head Attribute

### setType

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

Sets the type to be loaded.

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

### runTwoPhases

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

See the TypeLoader class RunTwoPhases method.

### setTailPosition

```java
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

```java
public void setRowReader(RowReader rr)
```

Sets the input data source.
Parameters:

- **rr** - [in] Input RowReader.

---

**setAttributePositions**

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

Sets the list of attribute positions.

Parameters:

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

---

**register**

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

Registers a new listener.

Parameters:

- **tel** - TypeLoaderListener to be registered.

---

**setLocale**

```java
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

Parameters:

- **localeStr** - [in] The locale string for the read data. See CSVReader.

---

**run**

```java
public void run()
```

throws IOException, RuntimeException

See the TypeLoader class Run method.

---

**setGraph**

```java
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 IOException, RuntimeException

    See the TypeLoader class RunNPhases method.

Parameters:
    partitions - null

Throws:
    java.io.IOException - null
    java.lang.RuntimeException - null

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

    Sets a specific timestamp format.

Parameters:
    timestampFormat - [in] A string with the timestamp format definition.

setHeadPosition

public void setHeadPosition(int pos)

    Sets the position of the head attribute in the source data.
    This method is protected because only the Edge loaders should have it.

Parameters:
    pos - [in] Head position

setAttributes

public void setAttributes(AttributeList attrs)

    Sets the list of Attributes.

Parameters:
    attrs - [in] Attribute identifiers to be loaded
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

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
</tr>
<tr>
<td>NodeTypeExporter(RowWriter rowWriter, Graph graph, int type, AttributeList attrs)</td>
</tr>
<tr>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public</td>
</tr>
<tr>
<td>NodeTypeExporter()</td>
</tr>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
</tr>
<tr>
<td>register(TypeExporterListener tel)</td>
</tr>
<tr>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>run()</td>
</tr>
<tr>
<td>See the TypeExporter class Run method.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setAttributes(AttributeList attrs)</td>
</tr>
<tr>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setFrequency(int freq)</td>
</tr>
<tr>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setGraph(Graph graph)</td>
</tr>
<tr>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setHeader(boolean header)</td>
</tr>
<tr>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setRowWriter(RowWriter rw)</td>
</tr>
<tr>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>setType(int type)</td>
</tr>
<tr>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object

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

Constructors

NodeTypeExporter
public NodeTypeExporter(RowWriter rowWriter, Graph graph, int type, AttributeList attrs)

Creates a new instance.

Parameters:
graph - [in] Graph.
type - [in] Type identifier.
attrs - [in] Attribute identifiers to be exported.

NodeTypeExporter
public NodeTypeExporter()

Creates a new instance.

Methods

setRowWriter
public void setRowWriter(RowWriter rw)

Sets the output data destination.

Parameters:
rw - [in] Input RowWriter.

setFrequency
public void setFrequency(int freq)

Sets the frequency of listener notification.
Parameters:

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

---

**register**

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

Registers a new listener.

**Parameters:**

tel - [in] TypeExporterListener to be registered.

---

**run**

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

See the TypeExporter class Run method.

---

**setGraph**

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

Sets the graph that will be exported.

**Parameters:**

graph - [in] Graph.

---

**setHeader**

```java
public void setHeader(boolean header)
```

Sets the presence of a header row.

**Parameters:**

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

---

**setType**

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

Sets the type to be exported.

**Parameters:**

type - [in] Type identifier.
setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be exported
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public</strong></td>
<td><strong>NodeTypeLoader(RowReader rowReader, Graph graph, int type, AttributeList attrs, Int32List attrsPos)</strong></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td><strong>public</strong></td>
<td><strong>NodeTypeLoader()</strong></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong></td>
<td><strong>register(TypeLoaderListener tel)</strong></td>
</tr>
<tr>
<td></td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>run()</strong></td>
</tr>
<tr>
<td></td>
<td>See the TypeLoader class Run method.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>runNPhases(int partitions)</strong></td>
</tr>
<tr>
<td></td>
<td>See the TypeLoader class RunNPhases method.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>runTwoPhases()</strong></td>
</tr>
<tr>
<td></td>
<td>See the TypeLoader class RunTwoPhases method.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setAttributePositions(Int32List attrsPos)</strong></td>
</tr>
<tr>
<td></td>
<td>Sets the list of attribute positions.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setAttributes(AttributeList attrs)</strong></td>
</tr>
<tr>
<td></td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setFrequency(int freq)</strong></td>
</tr>
<tr>
<td></td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>setGraph(Graph graph)</strong></td>
</tr>
<tr>
<td></td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>setLocale(String localeStr)</code></td>
<td>Sets the locale that will be used to read the data.</td>
</tr>
<tr>
<td><code>setLogError(String path)</code></td>
<td>Sets a log error file.</td>
</tr>
<tr>
<td><code>setLogOff()</code></td>
<td>Turns off all the error reporting.</td>
</tr>
<tr>
<td><code>setRowReader(RowReader rr)</code></td>
<td>Sets the input data source.</td>
</tr>
<tr>
<td><code>setTimestampFormat(String timestampFormat)</code></td>
<td>Sets a specific timestamp format.</td>
</tr>
<tr>
<td><code>setType(int type)</code></td>
<td>Sets the type to be loaded.</td>
</tr>
</tbody>
</table>

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**

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

Creates a new instance.

**Parameters:**
- `rowReader` - [in] Input RowReader.
- `graph` - [in] Graph.
- `type` - [in] Type identifier.
- `attrs` - [in] Attribute identifiers to be loaded.
- `attrsPos` - [in] Attribute positions (column index >=0).

**NodeTypeLoader**

```java
public NodeTypeLoader()
```

Creates a new instance.
## Methods

**setFrequency**

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

Sets the frequency of listener notification.

**Parameters:**

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

**setLogOff**

```java
public void setLogOff()
```

Turns off all the error reporting.

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

**setLogError**

```java
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**

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

Sets the type to be loaded.

**Parameters:**

- `type` - [in] Type identifier.

**runTwoPhases**

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

See the TypeLoader class RunTwoPhases method.
**setRowReader**

```java
public void setRowReader(RowReader rr)
```

Sets the input data source.

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

---

**setAttributePositions**

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

Sets the list of attribute positions.

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

---

**register**

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

Registers a new listener.

**Parameters:**
- `tel` - TypeLoaderListener to be registered.

---

**setLocale**

```java
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

**Parameters:**
- `localeStr` - [in] The locale string for the read data. See CSVReader.

---

**run**

```java
public void run()
```

**Throws:**
- IOException
- RuntimeException

See the TypeLoader class Run method.

---

**setGraph**

```java
public void setGraph(Graph graph)
```
Sets the graph where the data will be loaded.

**Parameters:**
- graph - [in] Graph.

---

### runNPhases

```java
def runNPhases(int partitions) throws IOException, RuntimeException
```

See the TypeLoader class RunNPhases method.

**Parameters:**
- partitions - null

**Throws:**
- java.io.IOException - null
- java.lang.RuntimeException - null

---

### setTimestampFormat

```java
def setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

**Parameters:**
- timestampFormat - [in] A string with the timestamp format definition.

---

### setAttributes

```java
def setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

**Parameters:**
- attrs - [in] Attribute identifiers to be loaded
com.sparsity.sparksee.io
Class 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>void</td>
<td>Closes the reader.</td>
</tr>
<tr>
<td>int</td>
<td>getRow()</td>
<td>The row number for the current row.</td>
</tr>
<tr>
<td>boolean</td>
<td>read(StringList row)</td>
<td>Reads the next row as a string array.</td>
</tr>
<tr>
<td>boolean</td>
<td>reset()</td>
<td>Moves the reader to the beginning.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
getRow
public int getRow()
    throws IOException

    The row number for the current row.

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

Throws:
    java.io.IOException - If it fails.

reset
public boolean reset()
    throws IOException

    Moves the reader to the beginning.
    Restarts the reader.

Returns:
    true if the reader can be restarted, false otherwise.

Throws:
    java.io.IOException - If bad things happen during the restart.

read
public boolean read(StringList row)
    throws IOException

    Reads the next row as a string array.

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

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

Throws:
    java.io.IOException - If bad things happen during the read.
com.sparsity.sparksee.io
Class RowWriter

java.lang.Object
    +-com.sparsity.sparksee.io.RowWriter

Direct Known Subclasses:
    CSVWriter

public class RowWriter
extends Object

RowWriter interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more
write operations can be executed.

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

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>Closes the writer.</td>
</tr>
<tr>
<td>void write(StringList row)</td>
</tr>
<tr>
<td>Writes the next row.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
</tr>
</tbody>
</table>

public void write(StringList row)
    throws IOException, RuntimeException

    Writes the next row.

Parameters:
    row - [in] Row of data.

Throws:
    java.io.IOException - If bad things happen during the write.
close

public void close()
    throws IOException,
    RuntimeException

    Closes the writer.
com.sparsity.sparksee.io
Class TypeExporter

java.lang.Object
   +- com.sparsity.sparksee.io.TypeExporter

Direct Known Subclasses:
   NodeTypeExporter, EdgeTypeExporter

public class TypeExporter
extends Object

Base TypeExporter class.

Base class to export a node or edge type from a graph using a RowWriter.

TypeExporterListener can be registered to receive information about the progress of the export process by means of
TypeExporterEvent. The default frequency of notification to listeners is 100000.

By default no header row is created.

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

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void register(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
<td>Runs export process.</td>
</tr>
<tr>
<td>void setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void setGraph(Graph graph)</td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td>void setType(int type)</td>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

**setRowWriter**

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

Sets the output data destination.

**Parameters:**
- `rw` - [in] Input RowWriter.

**setFrequency**

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

Sets the frequency of listener notification.

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

**run**

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

Runs export process.

**register**

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

Registers a new listener.

**Parameters:**
- `tel` - [in] TypeExporterListener to be registered.

**setGraph**

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

Sets the graph that will be exported.

**Parameters:**
- `graph` - [in] Graph.
**setHeader**

public void **setHeader**(boolean header)

Sets the presence of a header row.

**Parameters:**

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

---

**setType**

public void **setType**(int type)

Sets the type to be exported.

**Parameters:**

type - [in] Type identifier.

---

**setAttributes**

public void **setAttributes**(AttributeList attrs)

Sets the list of Attributes.

**Parameters:**

attrs - [in] Attribute identifiers to be exported
com.sparsity.sparksee.io
Class TypeExporterEvent

java.lang.Object
   +--com.sparsity.sparksee.io.TypeExporterEvent

public class TypeExporterEvent
extends Object

Provides information about the progress of an TypeExproter instance.

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

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCount()</td>
<td>long</td>
<td>Gets the current number of objects exported.</td>
</tr>
<tr>
<td>getTotal()</td>
<td>long</td>
<td>Gets the total number of objects exported.</td>
</tr>
<tr>
<td>getType()</td>
<td>int</td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>isLast()</td>
<td>boolean</td>
<td>Gets if this is the last event or not.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

g 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.

---

**getTypeId**

```java
public int getTypeId()
```

Gets the type identifier.

**Returns:**
The type identifier.

---

**getTotal**

```java
public long getTotal()
```

Gets the total number of objects exported.

**Returns:**
The total number of objects exported.
com.sparsity.sparksee.io
Class 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>notifyEvent(TypeExporterEvent tee)</td>
<td>Method to be notified from a TypeExporter.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

diagram: 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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
</table>
| 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 \texttt{setLogOff()}

Truns off all the error reporting.

void \texttt{setRowReader(RowReader \texttt{rr})}

Sets the input data source.

void \texttt{setTimestampFormat(String timestampFormat)}

Sets a specific timestamp format.

void \texttt{setType(int type)}

Sets the type to be loaded.

Methods inherited from class \texttt{java.lang.Object}

\texttt{clone}, \texttt{equals}, \texttt{finalize}, \texttt{getClass}, \texttt{hashCode}, \texttt{notify}, \texttt{notifyAll}, \texttt{toString}, \texttt{wait}, \texttt{wait}, \texttt{wait}

Methods

\textbf{runTwoPhases}

\begin{verbatim}
public void \texttt{runTwoPhases()} throws IOException, 
    RuntimeException

    Run the loader for two phases loading.
    Firstly load all objects (and create them if necessary) and secondly loads all the attributes.
    Working on this mode it is necessary to build a temporary file.
\end{verbatim}

\textbf{runNPhases}

\begin{verbatim}
public void \texttt{runNPhases(int partitions)} throws IOException, 
    RuntimeException

    Run the loader for N phases loading.
    Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.
    Working on this mode it is necessary to build a temporary file.

    Parameters:
    \texttt{partitions - [in]} Number of horizontal partitions to perform the load.

    Throws:
    java.io.IOException - null
    java.lang.RuntimeException - null
\end{verbatim}

\textbf{setFrequency}

\begin{verbatim}
public void \texttt{setFrequency(int freq)}

Sets the frequency of listener notification.
\end{verbatim}
Parameters:
  freq - [in] Frequency in number of rows managed to notify progress to all listeners

setLogOff

public void setLogOff()

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

run

public void run() throws IOException, RuntimeException

  Run the loader.

setLogError

public void setLogError(String path) throws IOException

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

Parameters:
  path - [in] The path to the error log file.

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

setType

public void setType(int type)

  Sets the type to be loaded.

Parameters:
  type - [in] Type identifier.

setRowReader

public void setRowReader(RowReader rr)

  Sets the input data source.

Parameters:
  rr - [in] Input RowReader.
**register**

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

Registers a new listener.

**Parameters:**
- `tel` - TypeLoaderListener to be registered.

---

**setAttributePositions**

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

Sets the list of attribute positions.

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

---

**setLocale**

```java
public void setLocale(String localeStr)
```

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

It should match the locale used in the rowreader.

**Parameters:**
- `localeStr` - [in] The locale string for the read data. See CSVReader.

---

**setGraph**

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

Sets the graph where the data will be loaded.

**Parameters:**
- `graph` - [in] Graph.

---

**setTimestampFormat**

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

Sets a specific timestamp format.

**Parameters:**
- `timestampFormat` - [in] A string with the timestamp format definition.

---

**setAttributes**

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


Sets the list of Attributes.

**Parameters:**

- `attrs` - [in] Attribute identifiers to be loaded
## com.sparsity.sparksee.io

### Class TypeLoaderEvent

java.lang.Object

```java
+-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

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td><code>getCount()</code></td>
<td>Gets the current number of objects created.</td>
</tr>
<tr>
<td>int</td>
<td><code>getPartition()</code></td>
<td>Gets the current partition.</td>
</tr>
<tr>
<td>int</td>
<td><code>getPhase()</code></td>
<td>Gets the current phase.</td>
</tr>
<tr>
<td>int</td>
<td><code>getTotalPartitions()</code></td>
<td>Gets the total number of partitions.</td>
</tr>
<tr>
<td>int</td>
<td><code>getTotalPartitionSteps()</code></td>
<td>Gets the total number of steps in the current partition.</td>
</tr>
<tr>
<td>int</td>
<td><code>getTotalPhases()</code></td>
<td>Gets the total number of phases.</td>
</tr>
<tr>
<td>int</td>
<td><code>getTypeId()</code></td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>isLast()</code></td>
<td>Gets if this is the last event or not.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

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

### Methods

#### getTotalPhases

```java
public int getTotalPhases()
```
Gets the total number of phases.

**Returns:**
The total number of phases.

---

`getCount`

```java
public long getCount()
```

Gets the current number of objects created.

**Returns:**
The current number of objects created.

---

`getTotalPartitionSteps`

```java
public int getTotalPartitionSteps()
```

Gets the total number of steps in the current partition.

**Returns:**
The total number of steps in the current partition.

---

`isLast`

```java
public boolean isLast()
```

Gets if this is the last event or not.

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

---

`getPartition`

```java
public int getPartition()
```

Gets the current partition.

**Returns:**
The current partition.

---

`getTypeId`

```java
public int getTypeId()
```

Gets the type identifier.

**Returns:**
getTotalPartitions

public int getTotalPartitions()

    Gets the total number of partitions.

    Returns:
    The total number of partitions.

getPhase

public int getPhase()

    Gets the current phase.

    Returns:
    The current phase.
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void notifyEvent(TypeLoaderEvent ev)</td>
<td>Method to receive events from a Loader.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
</tr>
</tbody>
</table>

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
ScriptParser

public ScriptParser()

   Constructor.

Methods

parse

public boolean parse(String path, boolean execute, String localeStr)
throws IOException

   Parses the given input file.

Parameters:
   path - [in] Input file path.
   execute - [in] If TRUE the script is executed, if FALSE it is just parsed.
   localeStr - [in] The locale string for reading the input file. See CSVReader.

Returns:
   TRUE if ok, FALSE in case of error.

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

setErrorLog

public void setErrorLog(String path)
throws IOException

   Sets the error log.
   If not set, error log corresponds to standard error output.

Parameters:
   path - [in] Path of the error log.

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

generateSchemaScript

public static void generateSchemaScript(String path, Database db)
throws IOException

   Writes an script with the schema definition for the given database.

Parameters:
   path - [in] Filename of the script to be written.
db - [in] Database.

Throws:
java.io.IOException - If bad things happen opening or writing the file.

**setOutputLog**

```java
public void setOutputLog(String path)
throws IOException
```

Sets the output log.

Parameters:
path - [in] Path of the output log.

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

**main**

```java
public static void main()
```

Executes ScriptParser for the given file path.

One argument is required, a file path which contains the script to be parsed.

A second argument may be given, a boolean to set if the script must be executed or just parsed. If not given, the script will be executed.
Index

A

add 67, 75, 134, 154, 157, 166, 213, 224, 241
addAll 158
addAllEdgeTypes 11, 15, 19, 24, 29, 33, 38, 41, 46, 50, 52, 57
addAllNodeTypes 7, 12, 14, 19, 24, 30, 32, 36, 41, 44, 48, 52, 57
addEdgeType 10, 14, 19, 24, 30, 32, 37, 41, 45, 49, 52, 56
addNodeType 7, 11, 14, 18, 23, 28, 33, 37, 40, 44, 48, 53, 57
addWeightedEdgeType 29
Any 104
any 152
Ascendent 170
asDirected 101
AttributeList 66

B

backup 121
Basic 63
begin 190
beginUpdate 189
Between 81
Boolean 90
BooleanList 75
Box 147

C

checkOnlyExistence 24
clear 66, 75, 134, 155, 167, 186, 212, 224, 241
close 5, 8, 12, 16, 41, 85, 155, 162, 183, 190, 193, 218, 244,
246, 250, 255, 277, 280
combineDifference 158
combineIntersection 155
combineUnion 153
commit 189
compare 237
compareTo 238, 239
calculate 10, 11
Config 139
ConnectedComponents 4
contains 152, 155
containsAll 153
Context 10
copy 154, 159
count 66, 76, 135, 155, 167, 186, 212, 225, 241, 244
countEdges 117
countNodes 122
create 192
CSVReader 249
CSVWriter 254

D

Debug 140
DefaultExport 93
degree 115
Descendent 171
difference 153
disableRollback 84
Double 90
drop 125, 127
dumpData 114
dumpStorage 116

E

Edge 164
EdgeExport 99
deges 119
EdgesType 220
EdgeTypeExporter 258
EdgeTypeLoader 263, 264
enableRollback 84
enableType 93, 107
Equal 80
equals 152, 154, 231, 235
excludeEdges 8, 12, 15, 20, 25, 30, 33, 38, 42, 46, 50, 53, 58
excludeNodes 7, 11, 14, 18, 23, 28, 32, 36, 40, 45, 49, 53, 57
execute 177
exists 18, 22, 28, 158
explode 124, 126
export 128

F

fetch 179
findAttribute 119
findAttributes 122
findEdge 126
findEdgeTypes 118
findNodeTypes 120
findObject 126
findOrCreateEdge 125
findOrCreateObject 120
findType 129
findTypes 114
Fine 139
fixCurrentCacheMaxSize 84

G

generateSchemaScript 298
get 185, 208, 240
getAlias 84
getAreNeighborsIndexed 220
getAttribute 120, 122, 127
getAttributeIntervalCount 115
getAttributes 116
getAttributeStatistics 117
getAvailableText 117
getAvailableMem 174
getAvgLengthString 73
getBoolean 209, 233
getCache 88
getCacheMaxSize 85, 205
getCacheStatisticsEnabled 200
getCacheStatisticsFile 201
getCacheStatisticsSnapshotTime 201
ggetColor 99, 143
ggetColorRGB 100, 144
getAddress 182
getColumn 182
getColumnType 183
getColumnIndex 182
getColumnName 183
getConnectedComponent 4
getConnectedComponents 8, 34, 38, 53, 58
getCost 20, 24, 29
getCount 4, 61, 284, 293
g GetCurrentDepth 41, 45, 49
data 88
dataType 62, 234
getDistinct 72
getDouble 236
gEdge 93, 97, 106
gEdgeData 123
gEdgePeer 129
gEdgeType 94, 106
gExtentPages 204
gExtentSize 198
gFontSize 100, 143
gGet 93, 106, 190
gGetHead 96
gGetHeight 145
gGetHighAvailabilityCoordinators 201
gGetHighAvailabilityEnabled 205
gGetHighAvailabilityIP 202
gGetHighAvailabilityMasterHistory 206
gGetHighAvailabilitySynchronization 198
gGetId 62, 221
gGetInteger 209, 239
gGetIsDirected 222
gGetIsRestricted 221
gGetJSON 183
gGetKind 61
gGetLabel 101, 131, 145
gGetLabelColor 101, 145
gGetLabelColorRGB 100, 144
gGetLicense 202
gGetLogFile 199
gGetLogLevel 203
gGetMax 72
gGetMaxLengthString 73
gGetMean 72
gGetMedian 72
gGetMin 71
gGetMinLengthString 71
gGetMode 71
gGetModeCount 73
gGetName 62, 222
ggetNode 94, 107
gGetNodes 5
ggetNodeType 94, 105
gGetNull 72
gGetNumColumns 183
gGetNumCPUs 175
getNumObjects 221
getObjectType 127, 221
g_OID 232
getPartition 293
getPath 84
getPathAsEdges 19, 23, 28
getPathAsNodes 20, 23, 28
getPhase 294
getPoolFrameSize 205
getPoolPersistentMaxSize 200
getPoolPersistentMinSize 203
getPoolTemporaryMaxSize 202
getPoolTemporaryMinSize 206
getRead 88
getRealTime 174
getRecoveryCacheMaxSize 206
getRecoveryCheckpointTime 200
getRecoveryEnabled 205
getRecoveryLogFile 200
getRestrictedFrom 220
getRestrictedTo 221
getRollbackEnabled 206
getRow 252, 278
getShape 143
getSize 4, 61
getStatistics 86, 172
getString 234
getSystemTime 174
getTail 96
getTemp 88
getTimestamp 238
getTimestampAsCalendar 231
getTimestampAsDate 234
getTimeUnit 209
getTotal 73, 285
getTotalMem 174
getTotalPartitions 294
getTotalPartitionSteps 293
getTotalPhases 292
g_type 118
getTypeId 61, 285, 293
getUserTime 174
getValues 128
getVariance 71
getWidth 101, 144
getWrite 87
GlobalType 220
GraphExport 131
GraphML 109
Graphviz 108
GreaterEqual 80
GreaterThan 80
hashCode 232
hasNext 40, 44, 48, 68, 77, 137, 162, 168, 187, 214, 226, 242, 245
heads 125

I

indexAttribute 118
Indexed 64
Info 139
Ingoing 103
Int32List 134
Integer 90
intersection 159
InvalidAttribute 60
InvalidOID 151
InvalidType 220
isClosed 5, 8, 12, 15, 42, 85, 157, 162, 189, 193, 244, 246
isEmpty 155
isFit 144
isLast 284, 293
isNull 217, 235
isSessionAttribute 61
iterator 66, 75, 134, 159, 167, 186, 212, 224, 241, 243, 244
iteratorFromElement 153
iteratorFromIndex 156

L

LessEqual 80
LessThan 80
Like 81
LikeNoCase 81
load 209
Long 90

M

main 299
MaxLengthString 230

N

neighbors 116, 123
newAttribute 119, 127
newEdge 128, 129
newEdgeType 125
newNode 117
newNodeType 124
newObjects 190
newQuery 178, 189
newRestrictedEdgeType 130
newSession 85
newSessionAttribute 122, 124
next 42, 45, 49, 69, 137, 162, 169, 182, 187, 215, 227, 242, 246
nextAttribute 69
nextBoolean 78
nextInt32 136
nextObject 161
nextOID 169
nextString 215
nextType 227
Node 163
NodeExport 142
NodesType 220
NodeTypeExporter 269
NodeTypeLoader 273
NotEqual 81
notifyEvent 286, 295

O

Off 139
OID 91
OIDList 166
open 193, 251, 255
Outgoing 104

P

parse 298
PlatformStatistics 173
prepare 94, 107, 180

Q

QueryContext 178

R

read 217, 251, 278
RegExp 81
register 260, 266, 270, 275, 282, 290
release 94, 107
remove 69, 78, 137, 152, 156, 162, 169, 215, 227, 246
removeAll 157
removeAttribute 121
removeType 130
renameAttribute 114
renameType 116, 123
reset 249, 278
restore 192
ResultSetList 185
retainAll 156
rewind 181
rollback 189
Round 148
run 7, 14, 19, 24, 30, 32, 37, 57, 60, 266, 270, 275, 282, 289
runNPhases 266, 276, 288
runTwoPhases 265, 274, 288

S

sample 159
ScriptParser 297
select 114, 118, 119, 120, 129
set 167, 232
setAsDirected 100
setAttribute 123
setAttributeDefaultValue 121
setAttributePositions 266, 275, 290
setAttributes 261, 267, 270, 276, 283, 290
setAttributeText 113
setAutoQuotes 254
setBoolean 232
setBooleanVoid 238
setCacheMaxSize 85, 203
setCacheStatisticsEnabled 199
setCacheStatisticsFile 201
setCacheStatisticsSnapshotTime 199
setColor 101, 145
setColorRGB 100, 143
setDefaults 100, 132, 144
setDouble 233
setDoubleVoid 235
setDynamic 176
setErrorLog 298
setExtentPages 198
setExtentSize 204
setFit 146
setFontSize 102, 146
setForcedQuotes 255
setFrequency 259, 264, 269, 273, 282, 288
setGraph 260, 266, 270, 275, 282, 290
setHeadAttribute 259, 265
setHeader 260, 270, 282
setHeadPosition 260, 267
setHeight 143
setHighAvailabilityCoordinators 198
setHighAvailabilityEnabled 203
setHighAvailabilityIP 204
setHighAvailabilityMasterHistory 200
setHighAvailabilitySynchronization 203
setInteger 239
setIntegerVoid 234
setLabel 101, 132, 144
setLabelColor 102, 146
setLabelColorRGB 102, 145
setLicense 198
setLocale 252, 255, 266, 275, 290
setLogError 264, 274, 289
setLogFile 199
setLogLevel 199
setLogOff 264, 274, 289
setLong 234
setLongVoid 231
setMaterializedAttribute 7, 33, 37, 53, 57
setMaximumHops 12, 15, 19, 23, 29, 40, 45, 49
setMultilines 250
setNull 237
setNullVoid 233
setNumLines 250
setOID 237
setOIDVoid 236
setOutputLog 299
setPoolFrameSize 201
setPoolPersistentMaxSize 200
setPoolPersistentMinSize 205
setPoolTemporaryMaxSize 204
setPoolTemporaryMinSize 202
setQuotes 250, 254
setRecoveryCacheMaxSize 203
setRecoveryCheckpointTime 206
setRecoveryEnabled 202
setRecoveryLogFile 207
setRollbackEnabled 204
setRowReader 265, 275, 289
setRowWriter 259, 269, 282
setSeparator 250, 254
setShape 146
setSingleLine 251
setStartLine 251
setStream 176
setString 232
setStringValue 233
setTailAttribute 258, 264
setTailPosition 259, 265
setTimestamp 231, 235, 238
setTimestampFormat 267, 276, 290
setTimestampVoid 233, 236
setType 259, 265, 270, 274, 283, 289
setUnweightedEdgeCost 28
setVoid 235
setWidth 102, 146
Severe 139
SinglePairShortestPathBFS 22
SinglePairShortestPathDijkstra 27
size 159
Sparksee 192
SparkseeConfig 197
start 179
String 90
StringList 212
StrongConnectivityGabow 36

T

tails 124
tailsAndHeads 115
Text 91
TextStream 217
Timestamp 90
toArray 157, 158
toString 237, 238
TraversalBFS 44
TraversalDFS 48
TypeList 224

U

union 156
Unique 64

V

Value 230, 231
ValueList 240
valueOf 64, 82, 91, 104, 109, 140, 148, 164, 171
values 64, 81, 91, 104, 109, 140, 148, 164, 171
Version 192

W

Warning 139
WeakConnectivityDFS 56
write 217, 255, 279

Y

YGraphML 109