Package

com.sparsity.dex.algorithms
com.sparsity.dex.algorithms
Class ConnectedComponents

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.

Each execution of any implementation of the Connectivity abstract class generates a ConnectedComponents instance which stores the following data:
(i) The number of connected components found.
(ii) For each node, the connected component identifier where it belongs to.
(iii) For each connected component, the collection of node identifiers contained in it and the number of nodes.

Furthermore, when executing any implementation of the Connectivity, it is possible to indicate whether the results of the execution must be stored persistently using the 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.

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 ConnectedComponents(Session s, String materializedattribute)</td>
<td>Creates a new instance of ConnectedComponents.</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 ConnectedComponents instance.</td>
</tr>
<tr>
<td>long getConnectedComponent(long idNode)</td>
<td>Returns the connected component where the given node belongs to.</td>
</tr>
<tr>
<td>long getCount()</td>
<td>Returns the number of connected components found in the graph.</td>
</tr>
<tr>
<td>Objects getNodes(long idConnectedComponent)</td>
<td>Returns the collection of nodes contained in the given connected component.</td>
</tr>
</tbody>
</table>
### Constructor

**ConnectedComponents**

```java
class ConnectedComponents {
    public ConnectedComponents(Session s, String materializedAttribute) {
        // constructor implementation
    }
}
```

**Description:**

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 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

#### getCount

```java
public long getCount() {
    // implementation
}
```

**Description:**

Returns the number of connected components found in the graph.

**Returns:**

- The number of connected components found in the graph.

#### getConnectedComponent

```java
public long getConnectedComponent(long idNode) {
    // implementation
}
```

**Description:**

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.

**isClosed**

```java
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:**
```java
close()
```

**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.

**getNodes**

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

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

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

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

**close**

```java
public void close()
```

Closes the ConnectedComponents instance.

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

java.lang.Object
   +-com.sparsity.dex.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.

Any component of a graph is called connected if there is a path between every pair of nodes contained in this component.

(i) Strongly Connected Components: This is the problem to find all the connected components in a directed graph.

(ii) Weakly Connected Components: This is the problem to find all the connected components in an undirected graph.

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

Method Summary

<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>
<tr>
<td>void</td>
<td>addNodeType(int t)</td>
</tr>
<tr>
<td></td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void</td>
<td>close()</td>
</tr>
<tr>
<td></td>
<td>Closes the Connectivity instance.</td>
</tr>
<tr>
<td>void</td>
<td>excludeEdges(Object edges)</td>
</tr>
<tr>
<td></td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void</td>
<td>excludeNodes(Object nodes)</td>
</tr>
<tr>
<td></td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>ConnectedComponents</td>
<td>getConnectedComponents()</td>
</tr>
<tr>
<td></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td>boolean</td>
<td>isClosed()</td>
</tr>
<tr>
<td></td>
<td>Gets if Connectivity instance has been closed or not.</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
</tr>
<tr>
<td></td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
</tbody>
</table>
**void** `setMaterializedAttribute(String attributeName)`

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

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

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

**Methods inherited from interface** `java.io.Closeable`

`close`

---

**Methods**

**addAllNodeTypes**

`public void addAllNodeTypes()`

Allows connectivity through all node types of the graph.

---

**run**

`public void run()`

Runs the algorithm in order to find the connected components.

This method can be called only once.

---

**excludeNodes**

`public void excludeNodes(Objects nodes)`

Set which nodes can't be used.

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

**Parameters:**

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

---

**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.

addNodeType
public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:
t - null

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.

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.dex.algorithms
Class Context

java.lang.Object
   +-com.sparsity.dex.algorithms.Context

All Implemented Interfaces:
   Closeable

public class Context
   extends Object
   implements Closeable

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

The user must set which edge types and node types can be used for the traversal. For the edge types, the user may specify how to
navigate them:
(i) Just navigate through out-going edges from the current node.
(ii) Just navigate through in-going edges from the current node.
(iii) Navigate through both in-going and out-going edges from the current node.

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

Constructor Summary

| public | Context (Session s, long node) |
|        | Creates a new instance.        |

Method Summary

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

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

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

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

| void | close () |
|      | Closes the Context instance. |

| Objects | compute () |
|         | Gets the resulting collection of nodes. |
Helper method to easily compute a context from a node.

Set which edges can't be used.

Set which nodes can't be used.

Gets if Context instance has been closed or not.

Sets the maximum hops restriction.

Creates a new instance.

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

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.
compute

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

Gets the resulting collection of nodes.

**Returns:**

The resulting collection of nodes.

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.

addNodeType

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

Allows for traversing nodes of the given type.

**Parameters:**

- t - null

compute

```java
public static Objects compute(Session s, long node, TypeList nodeTypes, TypeList edgeTypes, EdgesDirection dir, int maxhops, boolean include)
```

Helper method to easily compute a context from a node.

**Parameters:**

- s - [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.
excludeEdges
public void excludeEdges(Objects edges)

Set which edges can't be used.

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

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

close
public void close()

Closes the 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.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
   d - [in] Edge direction.

excludeNodes
public void excludeNodes(Objects nodes)

Set which nodes can't be used.

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

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

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.dex.algorithms

Class ShortestPath

java.lang.Object
   +-com.sparsity.dex.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. For the edge types, the user may specify how to
navigate them:

(i) Just navigate through out-going edges from the current node.
(ii) Just navigate through in-going edges from the current node.
(iii) Navigate through both in-going and out-going edges from the current node.

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</td>
<td>addAllEdgeTypes(EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing all edge types of the graph.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows for traversing all node types of the graph.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>addEdgeType(int t, EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing edges of the given type.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>addNodeType(int t)</td>
</tr>
<tr>
<td>Allows for traversing nodes of the given type.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>close()</td>
</tr>
<tr>
<td>Closes the ShortestPath instance.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>isClosed()</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
</tr>
<tr>
<td>void</td>
<td>setMaximumHops(int maxhops)</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 for traversing all node types of the graph.

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)  

Allows for traversing all edge types of the graph.

Parameters:
  
  d - [in] Edge direction.

run

public void run()  

Runs the algorithm.

This method can only be called once.

excludeNodes

public void excludeNodes(Objects nodes)  

Set which nodes can't be used.

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

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

**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.

**addNodeType**

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

Allows for traversing nodes of the given type.

**Parameters:**
- t - null

**setMaximumHops**

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

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

**Parameters:**
- maxhops - null

**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()

**close**

```java
public void close()
```

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.
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.dex.algorithms
Class SinglePairShortestPath

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

All Implemented Interfaces:
   Closeable

Direct Known Subclasses:
   SinglePairShortestPathDijkstra, SinglePairShortestPathBFS

public class SinglePairShortestPath
extends ShortestPath

SinglePairShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph from a given source node and to a given
destination node.

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 d)</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 t, EdgesDirection d)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</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>
### OIDList

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getPathAsNodes()</td>
<td>Gets the shortest path between the source node and the destination node as an ordered set of nodes.</td>
</tr>
<tr>
<td>run()</td>
<td>Runs the algorithm.</td>
</tr>
<tr>
<td>setMaximumHops(int maxhops)</td>
<td>Sets the maximum hops restriction.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exists()</td>
<td>Returns TRUE If a path exists or FALSE otherwise.</td>
</tr>
<tr>
<td>run()</td>
<td>Runs the algorithm.</td>
</tr>
<tr>
<td>addEdgeType(int t, EdgesDirection d)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
</tbody>
</table>

**Parameters:**
- t - [in] Edge type.
- d - [in] Edge direction.
addNodeType
public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
  t - null

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.

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.

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
  d - [in] Edge direction.

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

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

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

---

**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.

---

**setMaximumHops**

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

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

**Parameters:**
- **maxhops** - null
**com.sparsity.dex.algorithms**  
**Class SinglePairShortestPathBFS**

```java
public class SinglePairShortestPathBFS  
extends SinglePairShortestPath
```

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.

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

---

### Constructor Summary

<table>
<thead>
<tr>
<th>public SinglePairShortestPathBFS(Session s, long src, long dst)</th>
</tr>
</thead>
</table>

Creates a new instance.

---

### Method Summary

<table>
<thead>
<tr>
<th>void addAllEdgeTypes(EdgesDirection d)</th>
</tr>
</thead>
</table>

Allows for traversing all edge types of the graph.

<table>
<thead>
<tr>
<th>void addAllNodeType()</th>
</tr>
</thead>
</table>

Allows for traversing all node types of the graph.

<table>
<thead>
<tr>
<th>void addEdgeType(int t, EdgesDirection d)</th>
</tr>
</thead>
</table>

Allows for traversing edges of the given type.

<table>
<thead>
<tr>
<th>void addNodeType(int t)</th>
</tr>
</thead>
</table>

Allows for traversing nodes of the given type.

<table>
<thead>
<tr>
<th>void excludeEdges(Objects edges)</th>
</tr>
</thead>
</table>

Set which edges can't be used.

<table>
<thead>
<tr>
<th>void excludeNodes(Objects nodes)</th>
</tr>
</thead>
</table>

Set which nodes can't be used.

<table>
<thead>
<tr>
<th>boolean exists()</th>
</tr>
</thead>
</table>

Returns TRUE if a path exists or FALSE otherwise.

<table>
<thead>
<tr>
<th>double getCost()</th>
</tr>
</thead>
</table>

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.dex.algorithms.SinglePairShortestPath](#)
- `addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `excludeEdges`, `excludeNodes`, `exists`, `getCost`, `getPathAsEdges`, `getPathAsNodes`, `run`, `setMaximumHops`

### Methods inherited from class [com.sparsity.dex.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**

```java
public SinglePairShortestPathBFS(Session s,
    long src,
    long dst)
```

Creates a new instance.

**Parameters:**
- `s` - [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.

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.

addEdgeType
public void addEdgeType(int t, EdgesDirection d)

Allows for traversing edges of the given type.

Parameters:
t - [in] Edge type.
d - [in] Edge direction.

addNodeType
public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
t - null

getCost
public double getCost()

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

Returns:
The cost of the shortest path.
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.

addAllNodeTypes

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

Allows for traversing all node types of the graph.

addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

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

setMaximumHops

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

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
Parameters:

maxhops - null
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).

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

Constructor Summary

<table>
<thead>
<tr>
<th>public SinglePairShortestPathDijkstra(Session s, long src, long dst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void addAllEdgeTypes (EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>void addEdgeType(int t, EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows for traversing edges of the given type.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>void addWeightedEdgeType(int t, EdgesDirection d, int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows for traversing edges of the given type using the given attribute as the weight.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void excludeEdges(Objects edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set which edges can't be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void excludeNodes(Objects nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>boolean exists()</td>
</tr>
<tr>
<td>double getCost()</td>
</tr>
<tr>
<td>OIDList getPathAsEdges()</td>
</tr>
<tr>
<td>OIDList getPathAsNodes()</td>
</tr>
<tr>
<td>void run()</td>
</tr>
<tr>
<td>void setMaximumHops(int maxhops)</td>
</tr>
<tr>
<td>void setUnweightedEdgeCost(double weight)</td>
</tr>
</tbody>
</table>

Methods inherited from class [com.sparsity.dex.algorithms.SinglePairShortestPath](com.sparsity.dex.algorithms.SinglePairShortestPath)
- addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

Methods inherited from class [com.sparsity.dex.algorithms.ShortestPath](com.sparsity.dex.algorithms.ShortestPath)
- addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, isClosed, run, setMaximumHops

Methods inherited from class [java.lang.Object](java.lang.Object)
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

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

**Constructors**

**SinglePairShortestPathDijkstra**

public SinglePairShortestPathDijkstra(Session s, Long src, Long dst)

Creates a new instance.

**Parameters:**
- s - [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.

### addWeightedEdgeType

```java
public void addWeightedEdgeType(int t, EdgesDirection d, int attr)
```

Allows for traversing edges of the given type using the given attribute as the weight.

**Parameters:**
- **t** - [in] Edge type.
- **d** - [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.

### 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.

### 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.

### addNodeType

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

Allows for traversing nodes of the given type.

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

### getCost

```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.

### 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.

### addAllNodeTypes

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

Allows for traversing all node types of the graph.

### addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

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

---

**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** - null

---

**setMaximumHops**

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

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

**Parameters:**

- **maxhops** - null
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. Those restrictions are:

(i) The set of edge types which will be navigated through while traversing the graph. The method is AddEdgeType for adding one edge type and AddAllEdgeTypes for adding all edge types with a specified direction.

(ii) The set of node types which will be navigated through while traversing the graph. The method is AddNodeType for adding one node type and AddAllNodeTypes for adding all node types.

For each instance of this class, it is only allowed to run the algorithm once.

After the execution, we can retrieve the results stored in an instance of the class ConnectedComponents using the GetConnectedComponents() method.

**NOTE:** It is required to indicate the set of edge types and node types before running this algorithm.

**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 d)</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 t, EdgesDirection d)</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>
</tbody>
</table>
void excludeEdges(Objects edges)
Set which edges can't be used.

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

ConnectedComponents getConnectedComponents()
Returns the results generated by the execution of the algorithm.

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

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

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

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

Methods inherited from interface java.io.Closeable
close

Methods

addAllNodeTypes
public void addAllNodeTypes()
Allows connectivity through all node types of the graph.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)
Allows connectivity through all edge types of the graph.

Parameters:
d - [in] Edge direction.

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

This method can be called only once.

### excludeNodes

```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 t, EdgesDirection d)
```

Allows connectivity through edges of the given type.

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

### 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.

### addNodeType

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

Allows connectivity through nodes of the given type.

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

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.dex.algorithms**  
**Class StrongConnectivityGabow**

```java
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. Those restrictions are:

(i) The set of edge types which will be navigated through while traversing the graph. The method is **AddEdgeType** for adding one edge type and **AddAllEdgeTypes** for adding all edge types with a specified direction.

(ii) The set of node types which will be navigated through while traversing the graph. The method is **AddNodeType** for adding one node type and **AddAllNodeTypes** for adding all node types.

For each instance of this class, it is only allowed to run the algorithm once.

After the execution, we can retrieve the results stored in an instance of the class **ConnectedComponents** using the **GetConnectedComponents()** method.

**NOTE:** It is required to indicate the set of edge types and node types before running this algorithm.

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

---

### Constructor Summary

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

### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>addAllEdgeTypes(EdgesDirection d)</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 t, EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</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>void run()</td>
<td>Executes the algorithm.</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 `com.sparsity.dex.algorithms.StrongConnectivity`:
- addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, getConnectedComponents, run, setMaterializedAttribute

Methods inherited from class `com.sparsity.dex.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**

```java
public StrongConnectivityGabow(Session s)
```

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:**

- `s` - [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.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

 Allows connectivity through all edge types of the graph.

Parameters:
   d - [in] Edge direction.

excludeNodes
public void excludeNodes(Objects nodes)

 Set which nodes can't be used.

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

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

run
public void run()

 Executes the algorithm.

addEdgeType
public void addEdgeType(int t,
   EdgesDirection d)

 Allows connectivity through edges of the given type.

Parameters:
   t - [in] Edge type.
   d - [in] Edge direction.

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.

---

### addNodeType

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

Allows connectivity through nodes of the given type.

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

---

### 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.

---

### 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.dex.algorithms
Class Traversal

java.lang.Object
   +- com.sparsity.dex.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.

The user must set which edge types and node types can be used for the traversal. For the edge types, the user may specify how to
navigate them:

(i) Just navigate through out-going edges from the current node.
(ii) Just navigate through in-going edges from the current node.
(iii) Navigate through both in-going and out-going edges from the current node.

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.

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 d)</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 t, EdgesDirection d)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</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>
</tbody>
</table>
### getCurrentDepth()

Returns the depth of the current node.

### hasNext()

Gets if there are more objects to be traversed.

### isClosed()

Gets if Traversal instance has been closed or not.

### next()

Gets the next object of the traversal.

### 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

#### hasNext

**public boolean hasNext ()**

Gets if there are more objects to be traversed.

**Returns:**

TRUE if there are more objects, FALSE otherwise.

#### addEdgeType

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

Allows for traversing edges of the given type.

**Parameters:**

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

#### addNodeType

**public void addNodeType(int t)**

Allows for traversing nodes of the given type.
Parameters:
\[ t \] - null

gGetCurrentDepth

public int \texttt{getCurrentDepth}()

Returns the depth of the current node.

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

Returns:

The depth of the current node.

excludeEdges

public void \texttt{excludeEdges} (\texttt{Objects} edges)

Set which edges can't be used.

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

Parameters:

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

close

public void \texttt{close}()

Closes the Traversal instance.

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

addAllNodeTypes

public void \texttt{addAllNodeTypes}()

Allows for traversing all node types of the graph.

addAllEdgeTypes

public void \texttt{addAllEdgeTypes} (\texttt{EdgesDirection} d)

Allows for traversing all edge types of the graph.

Parameters:

d - [in] Edge direction.

excludeNodes

public void \texttt{excludeNodes} (\texttt{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.

---

**next**

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

Gets the next object of the traversal.

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

---

**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 Traversal instance has been closed or not.

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

**See Also:**
- `close()`
Com.sparsity.dex.algorithms

Class TraversalBFS

java.lang.Object
   +- com.sparsity.dex.algorithms.Traversal
      +- com.sparsity.dex.algorithms.TraversalBFS

All Implemented Interfaces:
   Closeable

public class TraversalBFS
extends Traversal

Breadth-First Search implementation of Traversal.

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

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

Constructor Summary

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public TraversalBFS(Session s, long node)</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 addAllEdgeTypes(EdgesDirection d)</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 t, EdgesDirection d)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</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>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>
<tr>
<td>long next()</td>
<td>Gets the next object of the traversal.</td>
</tr>
</tbody>
</table>
void setMaximumHops(int maxhops)

Sets the maximum hops restriction.

Methods inherited from class com.sparsity.dex.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

class public TraversalBFS(Session s, long node)

Creates a new instance.

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

Methods

addAllNodeTypes

class public void addAllNodeTypes()

Allows for traversing all node types of the graph.

addAllEdgeTypes

class public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
d - [in] Edge direction.

hasNext

class 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.

---

**next**

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

Gets the next object of the traversal.

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

---

**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.

---

**addNodeType**

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

Allows for traversing nodes of the given type.

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

---

**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.

---

**setMaximunHops**

public void `setMaximunHops(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**

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.dex.algorithms
Class TraversalDFS

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

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

Constructor Summary

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

Method Summary

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

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

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

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

void 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.dex.algorithms.Traversal
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges,
excludeNodes, getCurrentDepth, hasNext, isClosed, next, setMaximumHops

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

Methods inherited from interface java.io.Closeable
close

Constructors

TraversalDFS

public TraversalDFS(Session s,
long node)

Creates a new instance.

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

Methods

addAllNodeTypes

public void addAllNodeTypes()

Allows for traversing all node types of the graph.

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
d - [in] Edge direction.

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.

### next

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

Gets the next object of the traversal.

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

### 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.

### addNodeType

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

Allows for traversing nodes of the given type.

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

### 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

```
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

```
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.dex.algorithms
Class WeakConnectivity

java.lang.Object
   +-com.sparsity.dex.algorithms.Connectivity
      +-com.sparsity.dex.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.
Those restrictions are:

(i) The set of edge types which will be navigated through while traversing the graph. The method is AddEdgeType for adding one
edge type and AddAllEdgeTypes for adding all edge types.

(ii) The set of node types which will be navigated through while traversing the graph. The method is AddNodeType for adding one
node type and AddAllNodeTypes for adding all node types.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
</tbody>
</table>
### ConnectedComponents

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getConnectedComponents()</code></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td><code>run()</code></td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
<tr>
<td><code>setMaterializedAttribute(String attributeName)</code></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

**Methods inherited from class** `com.sparsity.dex.algorithms.Connectivity`

- `addAllNodeTypes`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `isClosed`, `run`, `setMaterializedAttribute`

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

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

**Methods inherited from interface** `java.io.Closeable`

- `close`

---

### Methods

**addAllNodeTypes**

```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. `d[in] Edge direction`.

**run**

```java
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.

---

**addEdgeType**

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

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction. d[in] Edge direction.

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

---

**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.

---

**addNodeType**

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

Allows connectivity through nodes of the given type.

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

---

**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 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.dex.algorithms
Class WeakConnectivityDFS

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. Those restrictions are:

(i) The set of edge types which will be navigated through while traversing the graph. The method is AddEdgeType for adding one edge type and AddAllEdgeTypes for adding all edge types.

(ii) The set of node types which will be navigated through while traversing the graph. The method is AddNodeType for adding one node type and AddAllNodeTypes for adding all node types.

For each instance of this class, it is only allowed to run the algorithm once.

After the execution, we can retrieve the results stored in an instance of the class ConnectedComponents using the getConnectedComponents() method.

Please, see edu.upc.dama.dex.algorithms for further information related to how to use this class.

NOTE: It is required to indicate the set of edge types and node types before running this algorithm.

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

Constructor Summary

| public WeakConnectivityDFS(Session s) |
| Creates a new instance of WeakConnectivityDFS. |

Method Summary

| void addAllEdgeTypes() |
| Allows connectivity through all edge types of the graph. |
| void addAllNodeTypes() |
| Allows connectivity through all node types of the graph. |
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addEdgeType(int t)</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>void run()</td>
<td>Executes the algorithm.</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** `com.sparsity.dex.algorithms.WeakConnectivity`  
`addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, getConnectedComponents, run, setMaterializedAttribute`  

**Methods inherited from class** `com.sparsity.dex.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

**WeakConnectivityDFS**

```java
public WeakConnectivityDFS(Session s)
```

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:**

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

### Methods
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. 

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

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.

addEdgeType

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

Allows connectivity through edges of the given type.

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

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

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.

addNodeType

public void addNodeType(int t)

   Allows connectivity through nodes of the given type.

Parameters:
   t - null

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.

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.
Package

com.sparsity.dex.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>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static</td>
<td>InvalidAttribute</td>
<td>Invalid attribute identifier constant.</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>getCount()</td>
<td>Gets the number of non-NULL values.</td>
</tr>
<tr>
<td></td>
<td>getDataType()</td>
<td>Gets the data type.</td>
</tr>
<tr>
<td>int</td>
<td>getId()</td>
<td>Gets the Dex attribute identifier.</td>
</tr>
<tr>
<td></td>
<td>getKind()</td>
<td>Gets the attribute kind.</td>
</tr>
<tr>
<td></td>
<td>getName()</td>
<td>Gets the unique attribute name.</td>
</tr>
<tr>
<td>long</td>
<td>getSize()</td>
<td>Gets the number of different values.</td>
</tr>
<tr>
<td>int</td>
<td>getType()</td>
<td>Gets the Dex type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<td>isSessionAttribute()</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

Fields
InvalidAttribute
public static int InvalidAttribute

Invalid attribute identifier constant.

Methods

getKind
public AttributeKind getKind()

Gets the attribute kind.

Returns:
The AttributeKind.

count
public long getCount()

Gets the number of non-NULL values.

Returns:
The number of non-NULL values.

isSessionAttribute
public boolean isSessionAttribute()

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

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

getSize
public long getSize()

Gets the number of different values.

Returns:
The number of different values.

getTypeId
public int getTypeId()
Gets the Dex type identifier.

**Returns:**
The Dex type identifier.

---

### getDataType

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

Gets the data type.

**Returns:**
The DataType.

---

### getId

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

Gets the Dex attribute identifier.

**Returns:**
The Dex attribute identifier.

---

### getName

```java
public String getName()
```

Gets the unique attribute name.

**Returns:**
The unique attribute name.
public final class AttributeKind extends Enum

Attribute kind enumeration.

All the attributes can be used for query operations. But the performance in query operations with the Basic attributes will be worse than with Indexed and Unique attributes. Unique attributes works as a primary key, that is two objects cannot have the same value for an attribute (but NULL).

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

Field Summary

| public static final | Basic
|---------------------|--------------------------------------------------|
| Basic attribute kind.
| public static final | Indexed
|---------------------|--------------------------------------------------|
| Indexed attribute kind.
| public static final | Unique
|---------------------|--------------------------------------------------|
| Unique attribute kind.

Method Summary

static AttributeKind valueOf(String name)

static AttributeKind[] values()

Methods inherited from class java.lang.Enum

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

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Comparable

compareTo
### Fields

**Basic**

```java
public static final com.sparsity.dex.gdb.AttributeKind Basic
```

Basic attribute kind.

**Indexed**

```java
public static final com.sparsity.dex.gdb.AttributeKind Indexed
```

Indexed attribute kind.

**Unique**

```java
public static final com.sparsity.dex.gdb.AttributeKind Unique
```

Unique attribute kind.

### Methods

**values**

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

**valueOf**

```java
public static AttributeKind valueOf(String name)
```
public class AttributeList
extends Object
implements Iterable

Dex attribute identifier list.
It stores a Dex 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>public</th>
<th>AttributeList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public</td>
<td>AttributeList()</td>
</tr>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
<tr>
<td>public</td>
<td>AttributeList(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>void</th>
<th>add(int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a Dex attribute identifier at the end of the list.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
</tr>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td>count()</td>
</tr>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>AttributeListIterator</td>
<td>iterator()</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, wait

Methods inherited from interface java.lang.Iterable

iterator
Constructors

AttributeList

public AttributeList(Collection col)

Creates a new instance from an integer collection.

Parameters:
  col - Collection to initialize the instance.

AttributeList

public AttributeList()

Constructor.

This creates an empty list.

AttributeList

public AttributeList(int[] list)

Creates a new instance from an integer array.

Parameters:
  list - Integer array to initialize the instance.

Methods

clear

public void clear()

Clears the list.

iterator

public AttributeListIterator iterator()

Gets a new AttributeListIterator.

Returns:
  AttributeListIterator instance.

count

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

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

### add

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

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

**Parameters:**
- `attr` - [in] Dex attribute identifier.
com.sparsity.dex.gdb
Class AttributeListIterator

java.lang.Object  
   +com.sparsity.dex.gdb.AttributeListIterator

All Implemented Interfaces:
   Iterator

public class AttributeListIterator
extends Object
implements Iterator

AttributeListIterator class.

Iterator to traverse all the Dex attribute identifier into a AttributeList 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>hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next()</td>
<td>See nextAttribute().</td>
</tr>
<tr>
<td>nextAttribute()</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>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, 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**

```java
public void remove()
```

Operation not supported.

**next**

```java
public Integer next()
```

See `nextAttribute()`.

**nextAttribute**

```java
public int nextAttribute()
```

Gets the next element.
public class AttributeStatistics
extends Object

Attribute statistics class.
It contains statistic data about an attribute.

Some fields are valid just for numerical attributes.

Some statistics are considered BASIC because computing them do not require to traverse all the different values of the attribute.
The getter for each statistic will tell you if the statistic is BASIC or not.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>double</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>double</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Methods

getMin

public Value getMin()

Gets the minimum existing value (BASIC statistics).

**Returns:**

The minimum existing value.

getMinLengthString

public int getMinLengthString()

Gets the minimum length.

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

**Returns:**

The minimum length.

getVariance

public double getVariance()

Gets the variance.

It is computed just for numerical attributes.

**Returns:**

The variance.

gemode

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:
getTotal

public long getTotal()

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

Returns:
The number of objects with a non-NULL Value.

getMaxLengthString

public int getMaxLengthString()

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

Returns:
The maximum length.

getAvgLengthString

public double getAvgLengthString()

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

Returns:
The average length.

getModeCount

public long getModeCount()

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

Returns:
The number of objects with a Value equal to the mode.
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 b)</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>
<tr>
<td>BooleanListIterator iterator()</td>
<td>Gets a new BooleanListIterator.</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

**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 b)
```

Adds a Boolean at the end of the list.

**Parameters:**
- `b` - [in] Boolean.

**clear**

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

Clears the list.

**iterator**

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

**Returns:**
BooleanListIterator instance.

## count

```
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
com.sparsity.dex.gdb
Class BooleanListIterator

java.lang.Object
   +-com.sparsity.dex.gdb.BooleanListIterator

All Implemented Interfaces:
   Iterator

public class BooleanListIterator
extends Object
implements Iterator

BooleanListIterator 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, 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.
public final class Condition
extends Enum

Condition operators enumeration.

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

Field Summary

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Condition</td>
<td>valueOf(String name)</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**

**Equal**

```java
public static final com.sparsity.dex.gdb.Condition Equal
```

Equal condition (==).

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

**GreaterEqual**

```java
public static final com.sparsity.dex.gdb.Condition GreaterEqual
```

Greater or equal condition (>=).

Null values cannot be used together with this condition.

**GreaterThan**

```java
public static final com.sparsity.dex.gdb.Condition GreaterThan
```

Greater than condition (>).

Null values cannot be used together with this condition.

**LessEqual**

```java
public static final com.sparsity.dex.gdb.Condition LessEqual
```

Less or equal condition (<=).

Null values cannot be used together with this condition.

**LessThan**

```java
public static final com.sparsity.dex.gdb.Condition LessThan
```

Less than condition (<).

Null values cannot be used together with this condition.
**NotEqual**

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

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

Null values cannot be used together with this condition.

**RegExp**

public static final com.sparsity.dex.gdb.Condition RegExp
Regular expression condition.

Null values cannot be used together with this condition.

This is condition can just be used together with String values. It allows for searching objects matching a given regular expression.

It is case sensitive.

Regular expression format conforms most of the POSIX Extended Regular Expressions.

Implemented operators are:

- letter: expects that letter. Example: "a" true if it includes an a.
- E1E2: two or more consecutive expressions are concatenated. Example: "b" true if it includes a substring ab.
- E1|E2: expects one of both expressions. Example: "cat|dog" true if it includes cat or dog.
- (E): groups an expression. Example: "r(u|a)n" true if it includes run or ran.
- E{m,M}: an expression is repeated from a minimum times, m, to a maximum times, M. If maximum, M, is zero, then unlimited number of repetitions is accepted. Example: "ab{1,3}" true if includes ab, abb or abbb.
- E* E+ E?: equivalent to: E{0,0} E{1,0} E{0,1}.
- [letters]: equivalent to: (letter1|letter2)...[letters]. Example: "[abc]" true if there is any of a or b or c.
-[^letters]: expects none of those letters. Example: "[^abc]" false if there is any of a or b or c.
- .: accepts any letter, equivalent to [^
- ^E: expects an expression at the beginning. Example: "^a" true if it starts with a.
- E$: expects an expression at the end. Example: "a$" true if it ends with a.

Some simple examples:

'AAABBBCCCD' ERE 'A+B*C+' returns true
'AAACCCD' ERE 'B*C+' returns true
'AAACCCD' ERE 'B+C+' returns false
'AAACCCD' ERE '^A[^a]*D$' returns true
'AAACCCD' ERE 'B*C+$' returns false

Methods

values

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

valueOf

```
public static Condition valueOf(String name)
```
public class Database
    extends Object
    implements Closeable

Database class.

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

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 Database instance.</td>
</tr>
<tr>
<td>String getAlias()</td>
<td>Gets the alias of the Database.</td>
</tr>
<tr>
<td>String getPath()</td>
<td>Gets the path of the Database.</td>
</tr>
<tr>
<td>void getStatistics(stats)</td>
<td>Gets Database statistics.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Database instance has been closed or not.</td>
</tr>
<tr>
<td>Session newSession()</td>
<td>Creates a new Session.</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

getAlias
public String getAlias()

Gets the alias of the Database.

Returns:
The alias of the Database.

gPath
public String getPath()

Gets the path of the Database.

Returns:
The path of the Database.

newSession
public Session newSession()

Creates a new Session.

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()

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:**

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

gWrite

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.

getData

public long getData()

    Gets database size in KBytes.

    Returns:
    Database size in KBytes.

getTemp

public long getTemp()

    Gets temporary storage file size in KBytes.

    Returns:
    Temporary storage file size in KBytes.

getRead

public long getRead()

    Gets total read data in KBytes.

    Returns:
    Total read data in KBytes.

getCache

public long getCache()

    Gets cache size in KBytes.

    Returns:
    Cache size in KBytes.
com.sparsity.dex.gdb
Class DataType

public final class DataType
extends Enum

Data type enumeration.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boolean data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Double</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64-bit signed double data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32-bit signed integer data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Long</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64-bit signed integer data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>OID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object identifier data type (oid_t).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unicode string data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Large unicode character object data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance from Epoch (UTC) time in milliseconds precision.</td>
</tr>
</tbody>
</table>

Method Summary

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

| static DataType[]   | values()             |

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</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>

### Fields

**Boolean**

```java
public static final com.sparsity.dex.gdb.DataType Boolean
```

Boolean data type.

**Integer**

```java
public static final com.sparsity.dex.gdb.DataType Integer
```

32-bit signed integer data type.

**Long**

```java
public static final com.sparsity.dex.gdb.DataType Long
```

64-bit signed integer data type.

**Double**

```java
public static final com.sparsity.dex.gdb.DataType Double
```

64-bit signed double data type.

**Timestamp**

```java
public static final com.sparsity.dex.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.dex.gdb.DataType String

  Unicode string data type.
  2048 characters maximum length.

Text

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

  Large unicode character object data type.

  TextStream

OID

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

  Object identifier data type (oid_t).

Methods

values

public static DataType[] values()

valueOf

public static DataType valueOf(String name)
public class DefaultExport extends ExportManager

Default implementation for ExportManager class.

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

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

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public DefaultExport ()</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>boolean enableType (int type)</td>
<td>ExportManager::EnableType.</td>
</tr>
<tr>
<td>boolean getEdge (long edge, EdgeExport edgeExport)</td>
<td>ExportManager::GetEdge.</td>
</tr>
<tr>
<td>boolean getEdgeType (int type, EdgeExport edgeExport)</td>
<td>ExportManager::GetEdgeType.</td>
</tr>
<tr>
<td>boolean getGraph (GraphExport graphExport)</td>
<td>ExportManager::GetGraph.</td>
</tr>
<tr>
<td>boolean getNode (long node, NodeExport nodeExport)</td>
<td>ExportManager::GetNode.</td>
</tr>
<tr>
<td>boolean getNodeType (int type, NodeExport nodeExport)</td>
<td>ExportManager::GetNodeType.</td>
</tr>
<tr>
<td>void prepare (Graph graph)</td>
<td>ExportManager::Prepare.</td>
</tr>
<tr>
<td>void release ()</td>
<td>ExportManager::Release.</td>
</tr>
</tbody>
</table>

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

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

Methods inherited from class java.lang.Object
Constructors

DefaultExport

public DefaultExport()

Creates a new instance.

Methods

enableType

public boolean enableType(int type)

ExportManager::EnableType.
This enables all node and edge types to be exported.

Parameters:
  type - null

Returns:
  TRUE.

getEdge

public boolean getEdge(long edge,
           EdgeExport edgeExport)

ExportManager::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 - null
  edgeExport - null

Returns:
  TRUE.

getGraph

public boolean getGraph(GraphExport graphExport)

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

Parameters:
  graphExport - null
**getEdgeType**

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

ExportManager::GetEdgeType.

This sets the default EdgeExport values.

**Parameters:**
- type - null
- edgeExport - null

**getNodeType**

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

ExportManager::GetNodeType.

This sets the default NodeExport values.

**Parameters:**
- type - null
- nodeExport - null

**release**

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

ExportManager::Release.

**getNode**

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

ExportManager::GetNode.

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

**Parameters:**
- node - null
- nodeExport - null

**Returns:**
- TRUE.

**prepare**

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

ExportManager::Prepare.

**Parameters:**
graph - null
com.sparsity.dex.gdb
Class Dex

java.lang.Object
   +-com.sparsity.dex.gdb.Dex

All Implemented Interfaces:
   Closeable

public class Dex
extends Object
implements Closeable

Dex class.

All Dex programs must have one single Dex 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>Dex version.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Dex(DexConfig 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 Dex 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 Dex instance has been closed or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>open(String path, boolean read)</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

Methods inherited from interface java.io.Closeable
Fields

Version
public static java.lang.String Version

Dex version.

Constructors

Dex
public Dex(DexConfig config)

Creates a new instance.

Parameters:
config - [in] Dex 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


Restores a Database from a backup file.

See Graph::Backup.

**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 exists.
- `java.lang.RuntimeException` - null

---

**isClosed**

*public boolean isClosed()*

Gets if Dex instance has been closed or not.

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

**See Also:**
`close()`

---

**close**

*public void close()*

Closes the Dex instance.

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

---

**open**

*public Database open(String path, boolean read)*

*throws FileNotFoundException, RuntimeException*

Opens an existing Database instance.

**Parameters:**
- `read` - [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
public class DexConfig extends Object

Dex 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 DexProperties 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 ('dex.storage.extentsize' at DexProperties).

Pages per extent: 1 page ('dex.storage.extentpages' at DexProperties).

Pool frame size: 1 extent ('dex.io.pool.frame.size' at DexProperties).

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

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

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

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

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

License code: "" ('dex.license' at DexProperties). No license code means evaluation license.

Log level: LogLevel::Info ('dex.log.level' at DexProperties).

Log file: "dex.log" ('dex.log.file' at DexProperties).


Recovery enabled: false ('dex.io.recovery' at DexProperties).

Recovery log file: "" ('dex.io.recovery.logfile' at DexProperties).

Recovery cache max size: 1MB ('dex.io.recovery.cachesize' at DexProperties).

Recovery checkpoint time: 60 seconds [TimeUnit] ('dex.io.recovery.checkpointTime' at DexProperties).

High-availability: false (disabled) ('dex.ha' at DexProperties).

High-availability coordinators: "" ('dex.ha.coordinators' at DexProperties).
High-availability IP: "" ('dex.ha.ip' at DexProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] ('dex.ha.sync' at DexProperties).

High-availability master history: 1D (1 day) [TimeUnit] ('dex.ha.master.history' at DexProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

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

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

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

## Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>DexConfig()</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>getCacheMaxSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the maximum size for the cache (all pools) in MB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getCacheStatisticsEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets whether cache statistics are enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getCacheStatisticsFile()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the cache statistics log file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getCacheStatisticsSnapshotTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the cache statistics snapshot time in microseconds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getExtentPages()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of pages per extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getExtentSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the size of a extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getHighAvailabilityCoordinators()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the coordinators address and port list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getHighAvailabilityEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets whether high availability mode is enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getHighAvailabilityIP()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the IP address and port of the instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getHighAvailabilityMasterHistory()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the master's history log.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getHighAvailabilitySynchronization()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the synchronization polling time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getLicense()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the license code.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>String getLogFile()</code></td>
<td>Gets the log file.</td>
</tr>
<tr>
<td><code>LogLevel getLogLevel()</code></td>
<td>Gets the log level.</td>
</tr>
<tr>
<td><code>int getPoolFrameSize()</code></td>
<td>Gets the size of a pool frame in number of extents.</td>
</tr>
<tr>
<td><code>int getPoolPersistentMaxSize()</code></td>
<td>Gets the maximum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td><code>int getPoolPersistentMinSize()</code></td>
<td>Gets the minimum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td><code>int getPoolTemporaryMaxSize()</code></td>
<td>Gets the maximum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td><code>int getPoolTemporaryMinSize()</code></td>
<td>Gets the minimum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td><code>int getRecoveryCacheMaxSize()</code></td>
<td>Gets the maximum size for the recovery log cache in extents.</td>
</tr>
<tr>
<td><code>long getRecoveryCheckpointTime()</code></td>
<td>Gets the delay time (in microseconds) between automatic checkpoints.</td>
</tr>
<tr>
<td><code>boolean getRecoveryEnabled()</code></td>
<td>Gets whether the recovery is enabled or disabled.</td>
</tr>
<tr>
<td><code>String getRecoveryLogFile()</code></td>
<td>Gets the recovery log file.</td>
</tr>
<tr>
<td><code>void setCacheMaxSize(int v)</code></td>
<td>Sets the maximum size for the cache (all pools) in MB.</td>
</tr>
<tr>
<td><code>void setCacheStatisticsEnabled(boolean v)</code></td>
<td>Enables or disables cache statistics.</td>
</tr>
<tr>
<td><code>void setCacheStatisticsFile(String v)</code></td>
<td>Sets the cache statistics log file.</td>
</tr>
<tr>
<td><code>void setCacheStatisticsSnapshotTime(long v)</code></td>
<td>Sets the cache statistics snapshot time.</td>
</tr>
<tr>
<td><code>void setExtentPages(int v)</code></td>
<td>Sets the number of pages per extent.</td>
</tr>
<tr>
<td><code>void setExtentSize(int v)</code></td>
<td>Sets the size of a pool frame in number of extents.</td>
</tr>
<tr>
<td><code>void setHighAvailabilityCoordinators(String v)</code></td>
<td>Sets the coordinators address and port list.</td>
</tr>
<tr>
<td><code>void setHighAvailabilityEnabled(boolean v)</code></td>
<td>Enables or disables high availability mode.</td>
</tr>
<tr>
<td><code>void setHighAvailabilityIP(String v)</code></td>
<td>Sets the IP address and port of the instance.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void setHighAvailabilityMasterHistory(long v)</code></td>
<td>Sets the master's history log.</td>
</tr>
<tr>
<td><code>void setHighAvailabilitySynchronization(long v)</code></td>
<td>Sets the synchronization polling time.</td>
</tr>
<tr>
<td><code>void setLicense(String v)</code></td>
<td>Sets the license code.</td>
</tr>
<tr>
<td><code>void setLogFile(String v)</code></td>
<td>Sets the log file.</td>
</tr>
<tr>
<td><code>void setLogLevel(LogLevel v)</code></td>
<td>Sets the log level.</td>
</tr>
<tr>
<td><code>void setPoolFrameSize(int v)</code></td>
<td>Sets the size of a pool frame in number of extents.</td>
</tr>
<tr>
<td><code>void setPoolPersistentMaxSize(int v)</code></td>
<td>Sets the maximum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td><code>void setPoolPersistentMinSize(int v)</code></td>
<td>Sets the minimum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td><code>void setPoolTemporaryMaxSize(int v)</code></td>
<td>Sets the maximum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td><code>void setPoolTemporaryMinSize(int v)</code></td>
<td>Sets the minimum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td><code>void setRecoveryCacheMaxSize(int v)</code></td>
<td>Sets the maximum size for the recovery log cache in extents.</td>
</tr>
<tr>
<td><code>void setRecoveryCheckpointTime(long v)</code></td>
<td>Sets the delay time (in microseconds) between automatic checkpoints.</td>
</tr>
<tr>
<td><code>void setRecoveryEnabled(boolean v)</code></td>
<td>Enables or disables the recovery.</td>
</tr>
<tr>
<td><code>void setRecoveryLogFile(String v)</code></td>
<td>Sets the recovery log file.</td>
</tr>
</tbody>
</table>

### Constructor

**DexConfig**

```java
public DexConfig()
```

Creates a new instance.

Values are set with default values.
Methods

**getHighAvailabilitySynchronization**

```java
public long getHighAvailabilitySynchronization()
```

Gets the synchronization polling time.

**Returns:**

The Synchronization polling time.

**setRecoveryCacheMaxSize**

```java
public void setRecoveryCacheMaxSize(int v)
```

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

**Parameters:**

- `v` - [in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

**setHighAvailabilityIP**

```java
public void setHighAvailabilityIP(String v)
```

Sets the IP address and port of the instance.

**Parameters:**

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

**setPoolPersistentMinSize**

```java
public void setPoolPersistentMinSize(int v)
```

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

**Parameters:**

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

**getExtentSize**

```java
public int getExtentSize()
```

Gets the size of a extent.

**Returns:**

The size of a extent in KB.
setHighAvailabilityEnabled

```java
public void setHighAvailabilityEnabled(boolean v)
```

Enables or disables high availability mode.

**Parameters:**

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

setHighAvailabilityCoordinators

```java
public void setHighAvailabilityCoordinators(String v)
```

Sets the coordinators address and port list.

**Parameters:**

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

getLogFile

```java
public String getLogFile()
```

Gets the log file.

**Returns:**

The log file.

getRecoveryCheckpointTime

```java
public long getRecoveryCheckpointTime()
```

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

**Returns:**

The delay time (in microseconds) between automatic checkpoints.

setCacheStatisticsSnapshotTime

```java
public void setCacheStatisticsSnapshotTime(long v)
```

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

**Parameters:**

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

getCacheStatisticsEnabled

```java
public boolean getCacheStatisticsEnabled()
```
Gets whether cache statistics are enabled or disabled.

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

---

### getPoolPersistentMaxSize

```java
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.

---

### setLogLevel

```java
public void setLogLevel(LogLevel v)
```

Sets the log level.

**Parameters:**
- v - [in] The LogLevel.

---

### setExtentSize

```java
public void setExtentSize(int v)
```

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

**Parameters:**
- v - [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.

---

### getRecoveryLogFile

```java
public String getRecoveryLogFile()
```

Gets the recovery log file.

**Returns:**
- The recovery log file.

---

### getExtentPages

```java
public int getExtentPages()
```

Gets the number of pages per extent.
Returns:
The number of pages per extent.

setPoolPersistentMaxSize

public void setPoolPersistentMaxSize(int v)

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

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

setCacheMaxSize

public void setCacheMaxSize(int v)

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

Parameters:
  v - [in] The maximum size for the cache (all pools) in MB. 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.

setRecoveryLogFile

public void setRecoveryLogFile(String v)

Sets the recovery log file.

Parameters:
  v - [in] The recovery log file. Left it empty for the default log file (same as .log)

getRecoveryEnabled

public boolean getRecoveryEnabled()

Gets whether the recovery is enabled or disabled.

Returns:
  TRUE if the recovery is enabled, FALSE otherwise.
getHighAvailabilityCoordinators

```java
public String getHighAvailabilityCoordinators()
```

Gets the coordinators address and port list.

**Returns:**
The coordinators address and port list.

setLicense

```java
public void setLicense(String v)
```

Sets the license code.

**Parameters:**

- `v` [in] The license code.

setRecoveryEnabled

```java
public void setRecoveryEnabled(boolean v)
```

Enables or disables the recovery.

**Parameters:**

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

setPoolTemporaryMaxSize

```java
public void setPoolTemporaryMaxSize(int v)
```

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

**Parameters:**

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

setHighAvailabilitySynchronization

```java
public void setHighAvailabilitySynchronization(long v)
```

Sets the synchronization polling time.

**Parameters:**

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

setPoolFrameSize

```java
public void setPoolFrameSize(int v)
```
Sets the size of a pool frame in number of extents.

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

getCacheStatisticsFile
public String getCacheStatisticsFile()

Gets the cache statistics log file.
Useless if cache statistics are disabled.

Returns:
The cache statistics log file.

setCacheStatisticsFile
public void setCacheStatisticsFile(String v)

Sets the cache statistics log file.
Useless if cache statistics are disabled.

Parameters:

getPoolFrameSize
public int getPoolFrameSize()

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

Returns:
The size of a pool frame in number of extents.

getCacheMaxSize
public int getCacheMaxSize()

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

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

getCacheStatisticsSnapshotTime
public long getCacheStatisticsSnapshotTime()

Gets the cache statistics snapshot time in microseconds.
Useless if cache statistics are disabled.

Returns:
The cache statistics snapshot time in microseconds.

### getPoolTemporaryMaxSize

```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.

### setExtentPages

```java
public void setExtentPages(int v)
```

Sets the number of pages per extent.

**Parameters:**
- v - [in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

### setPoolTemporaryMinSize

```java
public void setPoolTemporaryMinSize(int v)
```

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

**Parameters:**
- v - [in] The minimum size for the temporary pool in number of frames. It must be non-negative.

### getLicense

```java
public String getLicense()
```

Gets the license code.

**Returns:**
The license code.

### 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.
**setHighAvailabilityMasterHistory**

```java
public void setHighAvailabilityMasterHistory(long v)
```

Sets the master's history log.

**Parameters:**
- `v` - [in] The master's history log.

---

**setCacheStatisticsEnabled**

```java
public void setCacheStatisticsEnabled(boolean v)
```

Enables or disables cache statistics.

**Parameters:**
- `v` - [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

---

**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.

---

**getHighAvailabilityIP**

```java
public String getHighAvailabilityIP()
```

Gets the IP address and port of the instance.

**Returns:**
- The IP address and port of the instance.

---

**getLogLevel**

```java
public LogLevel getLogLevel()
```


Gets the log level.

**Returns:**
The LogLevel.

---

**setRecoveryCheckpointTime**

```java
public void setRecoveryCheckpointTime(long v)
```

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

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

---

**setLogFile**

```java
public void setLogFile(String v)
```

Sets the log file.

**Parameters:**

---

**getPoolPersistentMinSize**

```java
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.
public class DexProperties extends Object

Dex 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 './dex.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</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, wait

### Methods

**get**

public static String get(String key, String def)

  Gets a property.
**getTimeUnit**

```java
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] 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**

```java
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**

```java
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.
Class EdgeData

public class EdgeData
extends Object

Edge data class.

It stores the tail and the head of an edge.

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

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

Methods

getHead

public long getHead()

Gets the head of the edge.

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

getTail

public long getTail()

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

getEdge

public long getEdge()

Gets the edge identifier.

Returns:
The Dex edge identifier.
public class EdgeExport
extends Object

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:
Label: "" (empty string).
As directed: TRUE.
Color: 13882323 (OxD3D3D3, Light gray).
Label color: 0 (Ox000000, Black).
Width: 5px.
Font size: 10.

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

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean asDirected ()</td>
</tr>
<tr>
<td>Gets if the edge should be managed as directed.</td>
</tr>
<tr>
<td>java.awt.Color getColor ()</td>
</tr>
<tr>
<td>Gets the color of the edge.</td>
</tr>
<tr>
<td>int getColorRGB ()</td>
</tr>
<tr>
<td>Gets the edge color.</td>
</tr>
<tr>
<td>int getFontSize ()</td>
</tr>
<tr>
<td>Gets the edge label font size.</td>
</tr>
<tr>
<td>String getLabel ()</td>
</tr>
<tr>
<td>Gets the edge label.</td>
</tr>
<tr>
<td>java.awt.Color getLabelColor ()</td>
</tr>
<tr>
<td>Gets the color of the label.</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><code>int getLabelColorRGB()</code></td>
</tr>
<tr>
<td><code>int getWidth()</code></td>
</tr>
<tr>
<td><code>void setAsDirected(boolean b)</code></td>
</tr>
<tr>
<td><code>void setColor(java.awt.Color c)</code></td>
</tr>
<tr>
<td><code>void setColorRGB(int c)</code></td>
</tr>
<tr>
<td><code>void setDefaults()</code></td>
</tr>
<tr>
<td><code>void setFontSize(int s)</code></td>
</tr>
<tr>
<td><code>void setLabel(String l)</code></td>
</tr>
<tr>
<td><code>void setLabelColor(java.awt.Color c)</code></td>
</tr>
<tr>
<td><code>void setLabelColorRGB(int c)</code></td>
</tr>
<tr>
<td><code>void setWidth(int w)</code></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.
setLabelColorRGB

public void setLabelColorRGB(int c)

Sets the edge label color.

Parameters:

c - [in] The edge label color.

getFontSize

public int getFontSize()

Gets the edge label font size.

Returns:
The edge label font size.

setWidth

public void setWidth(int w)

Sets the edge width.

Parameters:

w - [in] The edge width.

setDefault

public void setDefaults()

Sets to default values.

colorRGB

public int getColorRGB()

Gets the edge color.

Returns:
The edge color.

setFontSize

public void setFontSize(int s)
Sets the edge label font size.

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

**setAsDirected**

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

Sets if the edge should be managed as directed.

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

**getLabelColorRGB**

```java
public int getLabelColorRGB()
```

Gets the edge label color.

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

**getWidth**

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

Gets the edge width.

**Returns:**
- The edge width.

**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.
**setLabel**

public void `setLabel(String l)`

Sets the edge label.

**Parameters:**

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

---

**asDirected**

public boolean `asDirected()`

Gets if the edge should be managed as directed.

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

**Returns:**

The edge direction.

---

**setColorRGB**

public void `setColorRGB(int c)`

Sets the edge color.

**Parameters:**

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

---

**getLabel**

public String `getLabel()`

Gets the edge label.

**Returns:**

The edge label.

---

**setLabelColor**

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

Sets the color of the label.

**Parameters:**

- `c` - New value.
com.sparsity.dex.gdb
Class EdgesDirection

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

All Implemented Interfaces:
    Serializable, Comparable

public final class EdgesDirection
extends Enum

Edges direction enumeration.

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

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static EdgesDirection valueOf(String name)</td>
</tr>
<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.dex.gdb.EdgesDirection Ingoing

In-going edges.

---

**Outgoing**

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

Out-going edges.

---

**Any**

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

In-going or out-going edges.

---

**Methods**

**values**

public static EdgesDirection[] values()

---

**valueOf**

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

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, these are the available formats: (i) ExportType.Graphviz, (ii) ExportType.GraphML, and (iii) ExportType.YGraphML

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean enableType(int type)</td>
</tr>
<tr>
<td>boolean getEdge(long edge, EdgeExport edgeExport)</td>
</tr>
<tr>
<td>boolean getEdgeType(int type, EdgeExport edgeExport)</td>
</tr>
<tr>
<td>boolean getGraph(GraphExport graphExport)</td>
</tr>
<tr>
<td>boolean getNode(long node, NodeExport nodeExport)</td>
</tr>
<tr>
<td>boolean getNodeType(int type, NodeExport nodeExport)</td>
</tr>
<tr>
<td>void prepare(Graph graph)</td>
</tr>
<tr>
<td>void release()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

**getNodeType**

```java
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**

```java
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**

```java
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**

```java
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:
  If TRUE all objects of the given type will be exported, otherwise they will not be exported.
com.sparsity.dex.gdb
Class ExportType

java.lang.Object
   +-java.lang.Enum
      +-com.sparsity.dex.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>
<tbody>
<tr>
<td>ExportType[]</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
Graphviz

public static final com.sparsity.dex.gdb.ExportType Graphviz

   Export to Graphviz format.
   Graphviz home page: http://www.graphviz.org

GraphML

public static final com.sparsity.dex.gdb.ExportType GraphML

   Export to GraphML format.
   GraphML home page: http://graphml.graphdrawing.org/

YGraphML

public static final com.sparsity.dex.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.

It allows for manipulating the Database as a graph. Especially, it allows for the manipulation of an attributed labeled and directed multigraph (labels represent object types).

A Graph has the following characteristics: (i) It belongs to a Database and a Session. (ii) It has nodes which belong to node types and edges which belong to a edge types. (iii) Nodes and edges both can have attributes. (iv) There is no limit in the number of edges between two nodes, even if they belong to the same edge type. (v) Nodes and edges have an unique identifier (object identifier, OID) in the Graph. (vi) Object types (node and edge types) and attributes have also an unique identifier in the Graph.

Edge types

There are different kind of edge types. They are classified as undirected or directed.

Directed edges have a node which is the tail (the source of the edge) and a node which is the head (the destination of the edge). In case of undirected edges, two nodes at the extreme of the edge play the two roles, head and tail. Whereas undirected edges allow for any kind of navigation, directed edges restrict the navigation to the direction of the edge.

Also, edges can be classified as restricted or unrestricted. Restricted edges define which must be the type of the tail nodes and the type of the head nodes, thus edges only will be allowed between nodes belonging to the specified types in the restriction. In case of unrestricted edges, there is no restriction and edges are allowed between nodes belonging to any type.

It is important to note that restricted edges are directed edges.

Indexed neighbors

Neighbors for each OID can be indexed in order to significantly increase the performance of the neighbors methods.

Unfortunately, it slightly decreases the performance of the construction of new edge instances. So, only if you will make intensive use of neighbors method should force materialize neighbors. This can be done by means of a parameter at the definition of new edge types.

Attributes

Basically attributes are restricted to a given node or edge type when they are created. That is, only node or edge objects of the corresponding restricting type can set and get values from that attribute identifier.

But also, attributes can be global. That is, all node or edge objects (no matters which type they belong to) can set and get values from that attribute identifier. To do that, Type::GlobalType must be used when creating the attribute.

There are three kind of attributes, but all of them allows for storing a value and run query methods (such as Select): (i) Basic: It allows the same usage as any other type, but the performance at query methods is not the best. (ii) Indexed: Like Basic, but have better performance at query methods. (iii) Unique: it works as Indexed but additionally they have a constraint: two different objects cannot have the same value (but NULL).

Virtual edges

Previous version of Dex (v4.1.x and older) have another type of edges called virtual edges. Check out the documentation of previous version of Dex to have a detailed explanation about the functionality of virtual edges.
Although virtual edges have been removed, the functionality they provided can be simulated as follows.

Let's suppose we have a virtual edge defined between the attributes ATTR_A and ATTR_B. These edges allow for navigating from any object OBJ having an attribute value for the attribute ATTR_A to those objects having the same attribute value for the attribute ATTR_B. That is, this can be done as follows: (i) Getting the value of OBJ for the attribute ATTR_A. (ii) Selecting the objects having the retrieved value for attribute ATTR_B.

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>backup(String file)</code></td>
<td>Dumps all the data to a backup file.</td>
</tr>
<tr>
<td><code>countEdges()</code></td>
<td>Gets the number of edges.</td>
</tr>
<tr>
<td><code>countNodes()</code></td>
<td>Gets the number of nodes.</td>
</tr>
<tr>
<td><code>degree(long oid, int etype, EdgesDirection dir)</code></td>
<td>Gets the number of edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td><code>drop(long oid)</code></td>
<td>Drops the given OID.</td>
</tr>
<tr>
<td><code>drop(Objects objs)</code></td>
<td>Drops all the OIDs from the given collection.</td>
</tr>
<tr>
<td><code>dumpData(String file)</code></td>
<td>Dumps logical data to a file.</td>
</tr>
<tr>
<td><code>dumpStorage(String file)</code></td>
<td>Dumps internal storage data to a file.</td>
</tr>
<tr>
<td><code>edges(int etype, long tail, long head)</code></td>
<td>Gets all the edges of the given type between two given nodes (tail and head).</td>
</tr>
<tr>
<td><code>explode(long oid, int etype, EdgesDirection dir)</code></td>
<td>Selects all edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td><code>explode(Objects objs, int etype, EdgesDirection dir)</code></td>
<td>Selects all edges from or to each of the node OID in the given collection and for the given edge type.</td>
</tr>
<tr>
<td><code>export(String file, ExportType type, ExportManager e)</code></td>
<td>Exports the Graph.</td>
</tr>
<tr>
<td><code>findAttribute(int type, String name)</code></td>
<td>Gets the Dex attribute identifier for the given type identifier and attribute name.</td>
</tr>
<tr>
<td><code>findAttributes(int type)</code></td>
<td>Gets all existing Dex attribute identifiers for the given type identifier.</td>
</tr>
<tr>
<td><code>findEdge(int etype, long tail, long head)</code></td>
<td>Gets any of the edges of the given type between two given nodes (tail and head).</td>
</tr>
<tr>
<td>TypeList</td>
<td>findEdgeTypes()</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>Gets all existing Dex edge type identifiers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TypeList</th>
<th>findNodeTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets all existing Dex node type identifiers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>findObject(int attr, Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Finds one object having the given Value for the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>findType(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Dex type identifier for the given type name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TypeList</th>
<th>findTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets all existing Dex node and edge type identifiers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>getAttribute(int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets information about the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>getAttribute(long oid, int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Value for the given attribute and OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>getAttribute(long oid, int attr, Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Value for the given attribute and OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getAttributeIntervalCount(int attr, Value lower, boolean includeLower, Value higher, boolean includeHigher)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets how many objects have a value into the given range for the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AttributeList</th>
<th>getAttributes(long oid)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets all Dex attribute identifiers with a non-NULL value for the given Dex OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AttributeStatistics</th>
<th>getAttributeStatistics(int attr, boolean basic)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets statistics from the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TextStream</th>
<th>getAttributeText(long oid, int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the read-only TextStream for the given text attribute and OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EdgeData</th>
<th>getEdgeData(long edge)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets information about an edge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getEdgePeer(long edge, long node)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the other end for the given edge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getObjectType(long oid)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Dex node or edge type identifier for the given OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>getType(int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets information about the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values</th>
<th>getValues(int attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Value collection for the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>heads(Objects edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets all the heads from the given edges collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>indexAttribute(int attr, AttributeKind kind)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Updates the index of the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>neighbors(long oid, int etype, EdgesDirection dir)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selects all neighbor nodes from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td>Objects</td>
<td><strong>neighbors</strong>(<em>Objects</em> <code>objs</code>, int <code>etype</code>, <code>EdgesDirection</code> <code>dir</code>)</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newAttribute</strong>(<em>int type</em>, String <code>name</code>, <code>DataType</code> <code>dt</code>, <code>AttributeKind</code> <code>kind</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newAttribute</strong>(<em>int type</em>, String <code>name</code>, <code>DataType</code> <code>dt</code>, <code>AttributeKind</code> <code>kind</code>, <code>Value</code> <code>defaultValue</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new attribute with a default value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th><strong>newEdge</strong>(<em>int type</em>, int <code>tailAttr</code>, <code>Value</code> <code>tailV</code>, int <code>headAttr</code>, <code>Value</code> <code>headV</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new edge instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th><strong>newEdge</strong>(<em>int type</em>, long <code>tail</code>, long <code>head</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new edge instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newEdgeType</strong>(<em>String name</em>, <code>boolean</code> <code>directed</code>, <code>boolean</code> <code>neighbors</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th><strong>newNode</strong>(<em>int type</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new node instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newNodeType</strong>(<em>String name</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new node type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newRestrictedEdgeType</strong>(<em>String name</em>, int <code>tail</code>, int <code>head</code>, <code>boolean</code> <code>neighbors</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new restricted edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newSessionAttribute</strong>(<em>int type</em>, <code>DataType</code> <code>dt</code>, <code>AttributeKind</code> <code>kind</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new Session attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>newSessionAttribute</strong>(<em>int type</em>, <code>DataType</code> <code>dt</code>, <code>AttributeKind</code> <code>kind</code>, <code>Value</code> <code>defaultValue</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new Session attribute with a default value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>removeAttribute</strong>(<em>int attr</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>removeType</strong>(<em>int type</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th><strong>select</strong>(<em>int type</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selects all OIDs belonging to the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th><strong>select</strong>(<em>int attr</em>, <code>Condition</code> <code>cond</code>, <code>Value</code> <code>v</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th><strong>select</strong>(<em>int attr</em>, <code>Condition</code> <code>cond</code>, <code>Value</code> <code>lower</code>, <code>Value</code> <code>higher</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setAttribute</strong>(<em>long oid</em>, int <code>attr</code>, <code>Value</code> <code>v</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the Value for the given attribute and OID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setAttributeDefaultValue</strong>(<em>int attr</em>, <code>Value</code> <code>v</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets a default value for an attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th><strong>setAttributeText</strong>(<em>long oid</em>, int <code>attr</code>, <code>TextStream</code> <code>tstream</code>)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the writable TextStream for the given text attribute and OID.</td>
</tr>
</tbody>
</table>
### Methods

**setAttributeText**

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

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

**Parameters:**
- `oid` - [in] Dex OID.
- `attr` - [in] Dex 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 Dex node and edge type identifiers.

**Returns:**
- Dex node and edge type identifier list.
setAttribute

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

Sets the Value for the given attribute and OID.

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

select

public Objects select(int attr,
    Condition cond,
    Value v)

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

Parameters:
attr - [in] Dex attribute identifier.
cond - [in] Condition to be satisfied.
v - [in] Value to be satisfied.

Returns:
Objects instance.

getAttributeIntervalCount

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

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

This only works for AttributeKind::Indexed or AttributeKind::Unique attributes.

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

Parameters:
attr - [in] Dex 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] Dex 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] Dex node OID.
   etype - [in] Dex edge type identifier.
   dir - [in] Direction.

Returns:
   The number of edges.

dumpStorage

public void dumpStorage(String file)
   throws FileNotFoundException,
        RuntimeException

Dumps internal storage data to a file.

Parameters:
   file - [in] Output file path.

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

neighbors

public Objects neighbors(Objects objs,
                          int etype,
                          EdgesDirection dir)

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

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

Returns:
Objects instance.

export
public void export(String file,
                    ExportType type,
                    ExportManager e)

Exports the Graph.

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

getAttributes
public AttributeList getAttributes(long oid)

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

Parameters:
oid - [in] Dex OID.

Returns:
Dex attribute identifier list.

getAttributeStatistics
public AttributeStatistics getAttributeStatistics(int attr,
                                                  boolean basic)

Gets statistics from the given attribute.

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

Returns:
An AttributeStatistics instace.

newNode
public long newNode(int type)
Creates a new node instance.

**Parameters:**
- `type` - [in] Dex 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] Dex OID.
- `attr` - [in] Dex 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.

---

**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 `Condition::Between` operation, thus it has two `Value` arguments.

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

**Returns:**
- Objects instance.

---

**findEdgeTypes**

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

---

(completed from last page)
Gets all existing Dex edge type identifiers.

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

### indexAttribute

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

Updates the index of the given attribute.

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

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

### getType

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

Gets information about the given type.

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

**Returns:**
The Type for the given Dex type identifier.

### setAttributeDefaultValue

```
public void setAttributeDefaultValue(int attr, Value v)
```

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

### findAttribute

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

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

**Parameters:**
newAttribute

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

Creates a new attribute.

Parameters:
  type - [in] Dex 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 Dex attribute identifier.

edges

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

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

type[in] Dex edge type identifier.

Parameters:
  etype - null
tail - [in] Tail node identifier.
head - [in] Head node identifier.

Returns:
  Objects instance.

select

public Objects select(int type)

Selects all OIDs belonging to the given type.

Parameters:
  type - [in] Dex type identifier.

Returns:
  Objects instance.
findNodeTypes
public TypeList findNodeTypes()

Gets all existing Dex node type identifiers.

Retruns:
Dex node type identifier list.

defAttribute
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] Dex OID.
attr - [in] Dex attribute identifier.

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

deleteAttribute
public void deleteAttribute(int attr)

Removes the given attribute.

Parameters:
attr - [in] Dex attribute identifier.

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

Dumps all the data to a backup file.

See Dex::Restore.

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

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

```java
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] Dex 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 Dex attribute identifier.

**findAttributes**

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

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

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

**Returns:**
- Dex attribute identifier list.

**countNodes**

```java
public long countNodes()
```

Gets the number of nodes.

**Returns:**
- The number of nodes.

**getEdgeData**

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

Gets information about an edge.

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

Returns:
  Objects instance.

eplode

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] Dex node OID collection.
  etype - [in] Dex edge type identifier.
  dir - [in] Direction.

Returns:
  Objects instance.

newNodeType

public int newNodeType(String name)

Creates a new node type.

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

Returns:
  Unique Dex type identifier.

newSessionAttribute

public int newSessionAttribute(int type, DataType dt, AttributeKind kind)
Creates a new Session attribute.

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

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

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

**Returns:**
Unique Dex attribute identifier.

tails

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

Gets all the tails from the given edges collection.

**Parameters:**

**Returns:**
The tails collection.

drop

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

Drops the given OID.

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

**Parameters:**
- `oid` - [in] Dex OID to be removed.

newEdgeType

```java
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 Dex type identifier.
heads

public Objects heads(Objects edges)

Gets all the heads from the given edges collection.

Parameters:
edges - [in] Dex edge identifier collection.

Returns:
The heads collection.

findEdge

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

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

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

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

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

explode

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

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

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

Returns:
Objects instance.

drop

public void drop(Objects objs)

Drops all the OIDs from the given collection.

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

Parameters:
newAttribute

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

Creates a new attribute with a default value.

Parameters:
  type - [in] Dex 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 Dex attribute identifier.

getObjectType

public int getObjectType(long oid)

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

Parameters:
  oid - [in] Dex OID.

Returns:
  Dex node or edge type identifier.

getAttribute

public Attribute getAttribute(int attr)

Gets information about the given attribute.

Parameters:
  attr - [in] Dex attribute identifier.

Returns:
  The Attribute for the given Dex attribute identifier.

newEdge

public long newEdge(int type,
                     int tailAttr,
                     Value tailV,
                     int headAttr,
                     Value headV)
Creates a new edge instance.

The tail of the edge will be any node having the given tailV Value for the given tailAttr attribute identifier, and the head of the edge will be any node having the given headV Value for the given headAttr attribute identifier.

**Parameters:**
- `type` - [in] Dex type identifier.
- `headAttr` - [in] Dex 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] Dex attribute identifier.

---

**getEdgePeer**

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

Gets the other end for the given edge.

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

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

---

**findObject**

```java
public long findObject(int attr, Value v)
```

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, `Objects::InvalidOID` will be returned.

**Parameters:**
- `attr` - [in] Dex attribute identifier.
- `v` - [in] Value.

**Returns:**
Dex OID or `Objects::InvalidOID`.
newEdge

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

Creates a new edge instance.

Parameters:
  type - [in] Dex type identifier.
  tail - [in] Source Dex OID.
  head - [in] Target Dex OID.

Returns:
  Unique OID of the new edge instance.

findType

public int findType(String name)

Gets the Dex type identifier for the given type name.

Parameters:
  name - [in] Unique type name.

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

getAttribute

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

Gets the Value for the given attribute and OID.

Parameters:
  oid - [in] Dex OID.
  attr - [in] Dex attribute identifier.
  v - [in/out] Value for the given attribute and for the given OID.

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] Dex 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 Dex node type identifier.
- head - [in] Head Dex node type identifier.
- neighbors - [in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns:
Unique Dex type identifier.
com.sparsity.dex.gdb
Class GraphExport

java.lang.Object
   +-com.sparsity.dex.gdb.GraphExport

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>String</td>
<td>String l</td>
<td>Set the graph label.</td>
</tr>
<tr>
<td>void</td>
<td>setDefaults()</td>
<td></td>
<td>Set to default values.</td>
</tr>
<tr>
<td>void</td>
<td>setLabel(String l)</td>
<td></td>
<td>Set the graph label.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Constructors

GraphExport

public GraphExport()

Creates a new GraphExport instance.

Methods

setLabel

public void setLabel(String l)
Sets the graph label.

**Parameters:**

- `l` - [in] The graph label.

---

**getLabel**

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

 Gets the graph label.

**Returns:**

The graph label.

---

**setDefaults**

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

 Sets to default values.
com.sparsity.dex.gdb
Class Int32List

dex32-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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Int32List(Collection col)</td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public</td>
<td>Int32List()</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public</td>
<td>Int32List(int[] list)</td>
<td>Creates a new instance from an integer array.</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 value)</td>
<td>Adds an 32-bit signed integer 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>Int32ListIterator</td>
<td>iterator()</td>
<td>Gets a new Int32ListIterator.</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

**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** - null

**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.dex.gdb**  
**Class Int32ListIterator**  

```java
public class Int32ListIterator extends Object implements Iterator
```

**Int32ListIterator 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>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>Integer</td>
<td>See nextInt32().</td>
</tr>
<tr>
<td>nextInt32()</td>
<td>Integer</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

**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().
class LogLevel

extends Enum

Log level enumeration.

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

Field Summary

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

Method Summary

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

Methods inherited from class java.lang.Enum

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

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

Methods inherited from interface java.lang.Comparable
compareTo

Fields

**Off**

```java
public static final com.sparsity.dex.gdb.LogLevel Off
```

Disable log.

**Severe**

```java
public static final com.sparsity.dex.gdb.LogLevel Severe
```

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

**Warning**

```java
public static final com.sparsity.dex.gdb.LogLevel Warning
```

Warning log level.

**Info**

```java
public static final com.sparsity.dex.gdb.LogLevel Info
```

Info log level.

**Config**

```java
public static final com.sparsity.dex.gdb.LogLevel Config
```

Config log level.

**Fine**

```java
public static final com.sparsity.dex.gdb.LogLevel Fine
```

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

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

Debug log level.
This is for Dex development purposes and just works on 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 (0x000000, Black).
Height: 25px.
Width: 25px.
Fit: TRUE.
Font size: 10.

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

### Constructor Summary

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

### Method Summary

<table>
<thead>
<tr>
<th>java.awt.Color</th>
<th>getColor ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the color of the node.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getColorRGB ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the node color.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getFontSize ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the node label font size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getHeight ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the node height.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getLabel ()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the node label.</td>
</tr>
<tr>
<td>java.awt.Color</td>
<td>getLabelColor()</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>Gets the color of the label.</td>
</tr>
<tr>
<td>int</td>
<td>getLabelColorRGB()</td>
</tr>
<tr>
<td></td>
<td>Gets the node label color.</td>
</tr>
<tr>
<td>NodeShape</td>
<td>getShape()</td>
</tr>
<tr>
<td></td>
<td>Gets the node shape.</td>
</tr>
<tr>
<td>int</td>
<td>getWidth()</td>
</tr>
<tr>
<td></td>
<td>Gets the node width.</td>
</tr>
<tr>
<td>boolean</td>
<td>isFit()</td>
</tr>
<tr>
<td></td>
<td>Gets whether the node size is fitted to the label or not.</td>
</tr>
<tr>
<td>void</td>
<td>setColor(java.awt.Color c)</td>
</tr>
<tr>
<td></td>
<td>Sets the color of the node.</td>
</tr>
<tr>
<td>void</td>
<td>setColorRGB(int c)</td>
</tr>
<tr>
<td></td>
<td>Sets the node color.</td>
</tr>
<tr>
<td>void</td>
<td>setDefaults()</td>
</tr>
<tr>
<td></td>
<td>Sets to default values.</td>
</tr>
<tr>
<td>void</td>
<td>setFit(boolean f)</td>
</tr>
<tr>
<td></td>
<td>Sets the node fit property.</td>
</tr>
<tr>
<td>void</td>
<td>setFontSize(int s)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label font size.</td>
</tr>
<tr>
<td>void</td>
<td>setHeight(int h)</td>
</tr>
<tr>
<td></td>
<td>Sets the node height.</td>
</tr>
<tr>
<td>void</td>
<td>setLabel(String l)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label.</td>
</tr>
<tr>
<td>void</td>
<td>setLabelColor(java.awt.Color c)</td>
</tr>
<tr>
<td></td>
<td>Sets the color of the label.</td>
</tr>
<tr>
<td>void</td>
<td>setLabelColorRGB(int c)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label color.</td>
</tr>
<tr>
<td>void</td>
<td>setShape(NodeShape s)</td>
</tr>
<tr>
<td></td>
<td>Sets the node shape.</td>
</tr>
<tr>
<td>void</td>
<td>setWidth(int w)</td>
</tr>
<tr>
<td></td>
<td>Gets the node width.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

**setLabelColorRGB**

```java
public void setLabelColorRGB(int c)

Sets the node label color.

**Parameters:**
- c - [in] The node label color.
```

**getFontSize**

```java
public int getFontSize()

Gets the node label font size.

**Returns:**
The node label font size.
```

**setWidth**

```java
public void setWidth(int w)

Gets the node width.
```
Parameters:
\[ w \] - The node width in pixels.

**setDefaults**

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

Sets to default values.

**getColorRGB**

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

Gets the node color.

**Returns:**
The node color.

**setFontSize**

```java
public void setFontSize(int s)
```

Sets the node label font size.

**Parameters:**
\[ s \] - [in] The node label font size.

**setHeight**

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

Sets the node height.

**Parameters:**
\[ h \] - [in] The node height in pixels.

**isFit**

```java
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'.

**setFit**

```java
public void setFit(boolean f)
```
Sets the node fit property.

**Parameters:**

- $f$ [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'.

### getLabelColorRGB

**public int getLabelColorRGB()**

Gets the node label color.

**Returns:**

The node label color.

### getWidth

**public int getWidth()**

Gets the node width.

**Returns:**

The node width in pixels.

### getLabelColor

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

Gets the color of the label.

### setColor

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

Sets the color of the node.

**Parameters:**

- c - New value.

### setLabel

**public void setLabel(String l)**

Sets the node label.

**Parameters:**

- l [in] The node label.
setColorRGB
public void setColorRGB(int c)

Sets the node color.

Parameters:
  c - The node color.

getLabel
public String getLabel()

Gets the node label.

Returns:
  The node label.

setShape
public void setShape(NodeShape s)

Sets the node shape.

Parameters:
  s - [in] The node shape.

getHeight
public int getHeight()

Gets the node height.

Returns:
  The node height in pixels.

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

Sets the color of the label.

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

Node shape.

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

Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
<td>Box</td>
<td>Box shape.</td>
</tr>
<tr>
<td>public static final</td>
<td>Round</td>
<td>Round shape.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static NodeShape.valueOf(String name)</td>
<td></td>
</tr>
<tr>
<td>static NodeShape[] values()</td>
<td></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
Box

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

Box shape.

Round

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

Round shape.

Methods

values

public static NodeShape[] values()

valueOf

public static NodeShape valueOf(String name)
public class Objects
extends Object
implements Set, Closeable, Iterable

Object identifier set class.
It stores a collection of Dex object identifiers as a set. As a set, there is 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.

Object identifiers cannot be added or removed from an Objects instance while iterators are still traversing the collection. Any objects modifying method will invalidate it's open iterators.

Also, in case Objects has been retrieved from the Graph, this collection cannot be updated through its Graph. For example, if we retrieve all the objects belonging to a given node or edge type, we cannot add or drop nodes or edges of this type while traversing the retrieved collection.

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

---

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidOID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid object identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>add(long e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds an element into the collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>add(Long e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds the specified element to this set if it is not already present (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>addAll(Collection clctn)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds all of the elements in the specified collection to this set if they're not already present (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>any()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets an element from the collection.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>void clear()</strong></td>
<td>Clears the collection removing all its elements.</td>
</tr>
<tr>
<td><strong>void close()</strong></td>
<td>Closes the Objects instance.</td>
</tr>
<tr>
<td><strong>static Objects combineDifference(Objects objs1, Objects objs2)</strong></td>
<td>Creates a new Objects instance which is the difference of the two given.</td>
</tr>
<tr>
<td><strong>static Objects combineIntersection(Objects objs1, Objects objs2)</strong></td>
<td>Creates a new Objects instance which is the intersection of the two given.</td>
</tr>
<tr>
<td><strong>static Objects combineUnion(Objects objs1, Objects objs2)</strong></td>
<td>Creates a new Objects instance which is the union of the two given.</td>
</tr>
<tr>
<td><strong>boolean contains(Object o)</strong></td>
<td>Returns true if this collections contains the specified element or Objects.</td>
</tr>
<tr>
<td><strong>boolean contains(Objects objs)</strong></td>
<td>Check if this objects contains the other one.</td>
</tr>
<tr>
<td><strong>boolean containsAll(Collection clctn)</strong></td>
<td>Returns true if this set contains all of the elements of the specified collection.</td>
</tr>
<tr>
<td><strong>Objects copy()</strong></td>
<td>Creates a new Objects instance as a copy of the given one.</td>
</tr>
<tr>
<td><strong>long copy(Objects objs)</strong></td>
<td>Performs the copy operation.</td>
</tr>
<tr>
<td><strong>long count()</strong></td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td><strong>long difference(Objects objs)</strong></td>
<td>Performs the difference operation.</td>
</tr>
<tr>
<td><strong>boolean equals(Object o)</strong></td>
<td>Returns true if the collection is equal to the object.</td>
</tr>
<tr>
<td><strong>boolean equals(Objects objs)</strong></td>
<td>Checks if the given Objects contains the same information.</td>
</tr>
<tr>
<td><strong>boolean exists(long e)</strong></td>
<td>Gets if the given element exists into the collection.</td>
</tr>
<tr>
<td><strong>long intersection(Objects objs)</strong></td>
<td>Performs the intersection operation.</td>
</tr>
<tr>
<td><strong>boolean isClosed()</strong></td>
<td>Gets if Objects instance has been closed or not.</td>
</tr>
<tr>
<td><strong>boolean isEmpty()</strong></td>
<td>Returns true if this Objects contains no elements.</td>
</tr>
<tr>
<td><strong>ObjectsIterator iterator()</strong></td>
<td>Gets an ObjectsIterator.</td>
</tr>
<tr>
<td><strong>ObjectsIterator iteratorFromElement(long e)</strong></td>
<td>Gets an ObjectsIterator starting from the given element.</td>
</tr>
<tr>
<td>ObjectsIterator</td>
<td>iteratorFromIndex(long index)</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td>Gets an ObjectsIterator skipping index elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>remove(long e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes an element from the collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>remove(Object o)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes the specified element from this set if it is present (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>removeAll(Collection clctn)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes from this set all of its elements that are contained in the specified collection (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>retainAll(Collection clctn)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Retains only the elements in this set that are contained in the specified collection (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>sample(Objects exclude, long samples)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new Objects instance which is a sample of the calling one.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>size()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the size of the collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object[]</th>
<th>toArray()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns an array containing all of the object identifiers in this set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object[]</th>
<th>toArray(Object[] ts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>union(Objects objs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Performs the union operation.</td>
</tr>
</tbody>
</table>

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

cl**one**, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Methods inherited from interface java.util.Set**

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

**Methods inherited from interface java.util.Collection**

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

**Methods inherited from interface java.lang.Iterable**

iterator

**Methods inherited from interface java.io.Closeable**

close

**Methods inherited from interface java.lang.Iterable**

iterator
### Fields

**InvalidOID**

public static int InvalidOID

Invalid object identifier constant.

### Methods

**any**

public long any()
    throwsNoSuchElementException,
          NoSuchElementException

    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

**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.

**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**

```java
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**

```java
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 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**

```java
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**

```java
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.

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.

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()

add

public boolean add(long e)

Adds an element into the collection.

Parameters:
e - [in] Element to be added.

Returns:
TRUE if the element is added, FALSE if the element was already into the collection.

removeAll

public boolean removeAll(Collection clctn)

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

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

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

Returns:
true if this set changed as a result of the call

toArray

public Object[] toArray()

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

Obey the general contract of the Collection.toArray method.

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

toArray

public Object[] toArray(Object[] ts)
>Returns an array containing all of the object identifiers in this set; the runtime type of the returned array is that of the specified array.

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

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

**Returns:**
- an array containing the elements of this set.

### addAll

```java
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.

### contains

```java
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.

### exists

```java
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
public static Objects combineDifference(Objects objs1, Objects objs2)
```
Creates a new Objects instance which is the difference of the two given.

Two given Objects belong to the same Session.

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

**Returns:**
New Objects instance.

**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**

```java
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 identifier into a Objects 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>close()</td>
<td>Closes the ObjectsIterator instance.</td>
</tr>
<tr>
<td>hasNext()</td>
<td>Gets if there are more elements to traverse.</td>
</tr>
<tr>
<td>isClosed()</td>
<td>Gets if ObjectsIterator instance has been closed or not.</td>
</tr>
<tr>
<td>long next()</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>long nextObject()</td>
<td>See nextObject().</td>
</tr>
<tr>
<td>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**

**nextObject**

```java
public long nextObject()
```

Gets the next element.

**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 Long next()
```

See nextObject().

**isClosed**

```java
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**

```java
public void close()
```

Closes the ObjectsIterator instance.

It must be called to ensure the integrity of all data.
com.sparsity.dex.gdb
Class ObjectType

java.lang.Object
   +-java.lang.Enum
    +-com.sparsity.dex.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

<table>
<thead>
<tr>
<th>public static final</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge object type.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node object type.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

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

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

public static final com.sparsity.dex.gdb.ObjectType Node

Node object type.

Edge

public static final com.sparsity.dex.gdb.ObjectType Edge

Edge object type.

Methods

values

public static ObjectType[] values()

valueOf

public static ObjectType valueOf(String name)
com.sparsity.dex.gdb
Class OIDList

java.lang.Object
   +-com.sparsity.dex.gdb.OIDList

All Implemented Interfaces:
   Iterable

public class OIDList
extends Object
implements Iterable

Dex object identifier list.
It stores a Dex 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></td>
<td>Creates a new instance from a long array.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from a long collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>OIDList()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(long attr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a Dex object identifier at the end of the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OIDListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets a new OIDListIterator.</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()

    Constructor.
    This creates an empty list.

Methods

add

public void add(long attr)

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

    Parameters:
    attr - [in] Dex object identifier.

clear

public void clear()

    Clears the list.

iterator

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.dex.gdb
Class OIDListIterator

java.lang.Object
  +-com.sparsity.dex.gdb.OIDListIterator

All Implemented Interfaces:
    Iterator

public class OIDListIterator
extends Object
implements Iterator

OIDListIterator class.

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

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

Method Summary

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>Long</td>
<td>next()</td>
<td>See nextOID().</td>
</tr>
<tr>
<td>long</td>
<td>nextOID()</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>void</td>
<td>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.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**

```java
public void remove()
```

Operation not supported.

**next**

```java
public Long next()
```

See nextOID().

**nextOID**

```java
public long nextOID()
```

Gets the next element.
com.sparsity.dex.gdb

Class Order

java.lang.Object
  \--java.lang.Enum
  \--com.sparsity.dex.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>
</table>

<table>
<thead>
<tr>
<th>static Order[]</th>
<th>values()</th>
</tr>
</thead>
</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

```java
public static final com.sparsity.dex.gdb.Order Ascendent
```

From lower to higher.

Descendent

```java
public static final com.sparsity.dex.gdb.Order Descendent
```

From higher to lower.

### Methods

**values**

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

**valueOf**

```java
public static Order valueOf(String name)
```
public class Platform extends Object

Platform class.

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

**Method Summary**

| static void | getStatistics(PlatformStatistics stats) | Gets platform usage statistics. |

**Method**

getStatistics

public static void getStatistics(PlatformStatistics stats)

Gets platform usage statistics.

**Parameters:**
- stats - [in|out] This updates the given PlatformStatistics.
public class **PlatformStatistics**
extends Object

Platform usage statistics.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>public PlatformStatistics()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>long getAvailableMem()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets available (free) memory size in Bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int getNumCPUs()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the number of CPUs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long getRealTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets time in microseconds (since epoch).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long getUserTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets CPU user time.</td>
</tr>
</tbody>
</table>

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

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

### Constructors

<table>
<thead>
<tr>
<th>public PlatformStatistics()</th>
</tr>
</thead>
</table>

**com.sparsity.dex.gdb**

Class **PlatformStatistics**
Creates a new instance setting all values to 0.

**Methods**

**getRealTime**

```java
public long getRealTime()
```

Gets time in microseconds (since epoch).

**Returns:**
Time in microseconds (since epoch).

**getAvailableMem**

```java
public long getAvailableMem()
```

Gets available (free) memory size in Bytes.

**Returns:**
Avialable (free) memory size in Bytes.

**getTotalMem**

```java
public long getTotalMem()
```

Gets physical memory size in Bytes.

**Returns:**
Physical memory size in Bytes.

**getSystemTime**

```java
public long getSystemTime()
```

Gets CPU system time.

**Returns:**
CPU system time.

**getUserTime**

```java
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.
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.

Transactions

A Session allows for enclosing a set of graph operations into a transaction. A transaction defines a granularity level for concurrent execution of Sessions. The explicit use of transactions may improve the performance of concurrent execution of Sessions.

All operations within a transaction are considered an execution unit. By default, if no transactions are defined by the user, all operations behave as autocommit, that is a transaction is created just before each method and closed when the method finishes.

For the moment, transactions have a partial support of the ACID properties.

Transaction type

There are two types of transactions: Read or Shared, and Write or Exclusive. DEX's concurrency model is based in a N-readers 1-writer model, thus read transactions can be executed concurrently whereas write transactions are executed exclusively.

The type of a transaction is defined for the operations it contains. Initially, a transaction starts as a read transaction and just when a method which updates the persistent graph database is found it automatically becomes a write transaction. Obviously, to become a write transaction this must wait until all other read transactions have finished.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>begin ()</td>
</tr>
<tr>
<td></td>
<td>Begins a transaction.</td>
</tr>
<tr>
<td>void</td>
<td>close ()</td>
</tr>
<tr>
<td></td>
<td>Closes the Session instance.</td>
</tr>
<tr>
<td>void</td>
<td>commit ()</td>
</tr>
<tr>
<td></td>
<td>Commits a transaction.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>getGraph()</td>
<td>Gets the Graph instance.</td>
</tr>
<tr>
<td>isClosed()</td>
<td>Gets if Session instance has been closed or not.</td>
</tr>
<tr>
<td>newObjects()</td>
<td>Creates a new Objects instance.</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**

**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.
com.sparsity.dex.gdb
Class StringList

java.lang.Object
   +-com.sparsity.dex.gdb.StringList

All Implemented Interfaces:
   Iterable

public class StringList
extends Object
implements Iterable

String list.
It stores a String (unicode) list.

Use StringListIterator to access all elements into this collection.

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

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>StringList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an string collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>StringList()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>StringList(String[] list)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an string array.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(String str)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a String at the end of the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StringListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></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

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()
```

Clears 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**

public void **add**(String str)

Adds a String at the end of the list.

**Parameters:**

*str* - [in] String.
com.sparsity.dex.gdb
Class StringListIterator

java.lang.Object
    ---com.sparsity.dex.gdb.StringListIterator

All Implemented Interfaces:
    Iterator

public class StringListIterator
extends Object
implements Iterator

StringListIterator 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>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>String</th>
<th>next()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See nextString().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>nextString()</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
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 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.

Use of TextStream for writing: (i) Create a TextStream instance and (ii) set a Value instance with this TextStream instance, then (iii) use this value to set the Text attribute of a node or edge instance. Once the set attribute 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 text attribute of a node or edge instance and (ii) get the TextStream instance from the retrieved Value instance. Once you have the TextStream instance, 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.

A stream just can be used once, that is if you need to read or write from or to the stream more than once, it is necessary to perform the get or set the attribute for each case.

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 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 if the stream is ready for reading or writing data.</td>
</tr>
<tr>
<td>int read(char[] dataOUT, int length)</td>
<td>Read data from the stream.</td>
</tr>
<tr>
<td>void setSession(Session s)</td>
<td></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 beginning 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 if the stream is ready for reading or writing data.

Returns:
TRUE 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.

```java
public void setSession(Session s)
```

Parameters:

s - null

```java
public void close()
```

Closes the stream.

Once the Stream is closed, it cannot be used again.
### com.sparsity.dex.gdb
#### Class Type

```java
java.lang.Object
   └--com.sparsity.dex.gdb.Type
```

public class Type
extends Object

Type data class.

It contains information about a node or edge type.

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

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public static</strong></td>
</tr>
<tr>
<td>Global type identifier constant.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Invalid type identifier constant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boolean</strong></td>
</tr>
<tr>
<td>Gets if this is an edge type with neighbors index.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets the Dex type identifier.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets if this is a directed edge type.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets if this is a restricted edge type.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets the unique type name.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets the number of objects belonging to the type.</td>
</tr>
</tbody>
</table>
| | **ObjectType**
| **getObjectType()** |
| Gets the object type. |
| | **getRestrictedFrom()** |
| Gets the tail or source type identifier for restricted edge types. |
| | **getRestrictedTo()** |
| Gets the head or target type identifier for restricted edge types. |

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

### Fields

**GlobalType**

```java
public static int GlobalType
```

Global type identifier constant.

**InvalidType**

```java
public static int InvalidType
```

Invalid type identifier constant.

### Methods

**getRestrictedFrom**

```java
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, Type::InvalidType otherwise.

**getAreNeighborsIndexed**

```java
public boolean getAreNeighborsIndexed()
```

Gets if this is an edge type with neighbors index.

**Returns:**

TRUE for edges types with neighbors index, FALSE otherwise.

**getObjectType**

```java
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, 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 Dex type identifier.

Returns:
The Dex type identifier.

getIsDirected

public boolean getIsDirected()

Gets if this is a directed edge type.

Returns:
TRUE for directed edge types, FALSE otherwise.

getName

public String getName()
Gets the unique type name.

**Returns:**

The unique type name.
public class TypeList
extends Object
implements Iterable

Dex type identifier list.
It stores a Dex 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>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public TypeList(int[] list)</td>
<td>Creates a new instance from an integer array.</td>
</tr>
<tr>
<td>public TypeList(Collection col)</td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public TypeList()</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(int type)</td>
<td>Adds a Dex type identifier 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>TypeListIterator 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.

**TypeList**

```java
public TypeList(Collection col)
```

Creates a new instance from an integer collection.

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

**TypeList**

```java
public TypeList()
```

Constructor.

This creates an empty list.

Methods

**add**

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

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

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

**clear**

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

Clears the list.

**iterator**

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

**Returns:**
TypeListIterator instance.

---

**count**

```java
class com.sparsity.dex.gdb.TypeList
{
    public int count()
    {
        Number of elements in the list.
    }
}
```

**Returns:**
Number of elements in the list.
com.sparsity.dex.gdb
Class TypeListIterator

java.lang.Object
   +-com.sparsity.dex.gdb.TypeListIterator

All Implemented Interfaces:
   Iterator

public class TypeListIterator extends Object implements Iterator

TypeListIterator class.
Iterator to traverse all the Dex 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><strong>boolean</strong></td>
</tr>
<tr>
<td><strong>Integer</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
</tr>
<tr>
<td><strong>void</strong></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 Integer next()

See nextType().

nextType
public int nextType()

Gets the next element.
### Field Summary

| public static | MaxLengthString | Maximum number of characters allowed for a String. |

### Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Value(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy constructor.</td>
<td></td>
</tr>
</tbody>
</table>

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

### Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>compare(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compares with the given Value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Object v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>See compare().</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>See compare().</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Object other)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compares with the given Value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getBoolean()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets Boolean Value.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>double</th>
<th>getDouble()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets Double Value.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>int</td>
<td><code>getInteger()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getLong()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getOID()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getString()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getTimestamp()</code></td>
</tr>
<tr>
<td>Calendar</td>
<td><code>getTimestampAsCalendar()</code></td>
</tr>
<tr>
<td>Date</td>
<td><code>getTimestampAsDate()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>hashCode()</code></td>
</tr>
<tr>
<td>boolean</td>
<td><code>isNull()</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>set(Value v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setBoolean(boolean v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setBooleanVoid(boolean v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setDouble(double v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setDoubleVoid(double v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setInteger(int v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setIntegerVoid(int v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setLong(long v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setLongVoid(long v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setNull()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setNullVoid()</code></td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setOID</td>
<td>(long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setOIDVoid</td>
<td>(long v)</td>
<td>Sets the OID Value.</td>
</tr>
<tr>
<td>setString</td>
<td>(String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setStringVoid</td>
<td>(String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setText</td>
<td>(TextStream v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setTimestamp</td>
<td>(Calendar v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setTimestamp</td>
<td>(Date v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setTimestamp</td>
<td>(int year, int month, int day, int hour, int minutes, int seconds, int millisec)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setTimestampVoid</td>
<td>(int year, int month, int day, int hour, int minutes, int seconds, int millisecs)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setTimestampVoid</td>
<td>(long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>setVoid</td>
<td>(Value v)</td>
<td>Sets the Value.</td>
</tr>
</tbody>
</table>

### toString

- **toString()**
  - Gets a String representation of the Value.
- **toString(String str)**
  - Gets a string representation of the Value.

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

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

### Fields

**MaxLengthString**

- public static int `MaxLengthString`
  - Maximum number of characters allowed for a String.

### Constructors
Value

public Value(Value v)

Copy constructor.

Parameters:
  v - [in] Value to be copied.

Value

public Value()

Creates a new instance.

It creates a NULL Value.

Methods

setLongVoid

public void setLongVoid(long v)

Sets the Value.

Parameters:
  v - [in] New Long value.

getTimestampAsCalendar

public Calendar getTimestampAsCalendar()

Gets the Value as a Calendar instance.

Returns:
  The returning Calendar instance.

getOID

public long getOID()

Gets OID Value.

This must be an non-NULL OID Value.

Returns:
  The OID Value.

set

public Value set(Value v)
Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
The calling instance.

**setString**

```java
public Value setString(String v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
The calling instance.

**setBooleanVoid**

```java
public void setBooleanVoid(boolean v)
```

Sets the Value.

**Parameters:**
- `v` - [in] New Boolean value.

**hashCode**

```java
public int hashCode()
```

**compare**

```java
public int compare(Value v)
```

Compares with the given Value.

It does not work for Text or if given Value objects does not have the same DataType.

**Parameters:**
- `v` - 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.
**setDoubleVoid**

public void setDoubleVoid(double v)

Sets the Value.

**Parameters:**

 v - [in] New Double value.

---

**setText**

public Value setText(TextStream v)

Sets the Value.

**Parameters:**

 v - New value.

**Returns:**

 The calling instance.

---

**setBoolean**

public Value setBoolean(boolean v)

Sets the Value.

**Parameters:**

 v - New value.

**Returns:**

 The calling instance.

---

**compareTo**

public int compareTo(Value v)

See compare().

**Parameters:**

 v - null

---

**getBoolean**

public boolean getBoolean()

Gets Boolean Value.

This must be a non-NULL Boolean Value.

**Returns:**

 The Boolean Value.
setTimestampVoid

public void setTimestampVoid(long v)

Sets the Value.

Parameters:

setDouble

public Value setDouble(double v)

Sets the Value.

Parameters:
   v - New value.

Returns:
   The calling instance.

setNullVoid

public void setNullVoid()

Sets the Value to NULL.

equals

public boolean equals(Object other)

Parameters:
   other - null

setOIDVoid

public void setOIDVoid(long v)

Sets the OID Value.

Parameters:
   v - [in] New OID value.

getDataType

public DataType getDataType()

Gets the DataType.

Value cannot be NULL.

**Returns:**

The DataType.

### setLong

```java
public Value setLong(long v)
```

Sets the Value.

**Parameters:****

- `v` - New value.

**Returns:**

The calling instance.

### getTimestampAsDate

```java
public Date getTimestampAsDate()
```

Gets the Value as a Date instance.

**Returns:**

The returning Date instance.

### compareTo

```java
public int compareTo(Object v)
```

See compare().

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

**Parameters:****

- `v` - null

### getString

```java
public String getString()
```

Gets String Value.

This must be a non-NULL String Value.

**Returns:**

The String Value.

### setTimestamp

```java
public Value setTimestamp(Calendar v)
```
Sets the Value.

**Parameters:**

- v - New value.

**Returns:**

The calling instance.

### isNull

```java
public boolean isNull()
```

Gets if this is a NULL Value.

**Returns:**

TRUE if this is a NULL Value, FALSE otherwise.

### setTimestamp

```java
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

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

Sets the Value.

Sets the Value.
**Parameters:**
- **month** - [in] The month ([1..12]),
- **day** - [in] The of the month ([1..31]).
- **hour** - [in] The hour ([0..23]).
- **minutes** - [in] The minutes ([0..59]).
- **seconds** - [in] The seconds ([0..59]).
- **millisecond** - [in] The milliseconds ([0..999]).

**equals**

```
public boolean equals(Value v)
```

Compares with the given Value.

It does not work for Text or if given Value objects does not have the same Data\-Type.

**Parameters:**
- **v** - Given value to compare to.

**Returns:**
- TRUE if this Value is equal to the given one; FALSE otherwise.

**getDouble**

```
public double getDouble()
```

Gets Double Value.

This must be a non-NULL Double Value.

**Returns:**
The Double Value.

**setIntegerVoid**

```
public void setIntegerVoid(int v)
```

Sets the Value.

**Parameters:**
- **v** - [in] New Integer value.

**toString**

```
public String toString(String str)
```

Gets a string representation of the Value.

It does not work for Text.

**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.
**setVoid**

```java
public void setVoid(Value v)
```

Sets the Value.

**Parameters:**

v - [in] New value.

---

**setNull**

```java
public Value setNull()
```

Sets the Value to NULL.

**Returns:**

The calling instance.

---

**setTimestamp**

```java
public Value setTimestamp(Date v)
```

Sets the Value.

**Parameters:**

v - New value.

**Returns:**

The calling instance.

---

**setOID**

```java
public Value setOID(long v)
```

Sets the Value.

**Parameters:**

v - New value.

**Returns:**

The calling instance.

---

**setStringVoid**

```java
public void setStringVoid(String v)
```

Sets the Value.

**Parameters:**
v - [in] New String value.

public long **getLong**()

    Gets Long Value.
    This must be a non-NULL Long Value.

    **Returns:**
    The Long Value.

public String **toString**()

    Gets a String representation of the Value.
    It does not work for Text Value instances.

public long **getTimestamp**()

    Gets Timestamp Value.
    This must be a non-NULL Timestamp Value.

    **Returns:**
    The Timestamp Value.

public **Value** setInteger(int v)

    Sets the Value.

    **Parameters:**
    v - New value.

    **Returns:**
    The calling instance.

public int **getInteger**()

    Gets Integer Value.
    This must be a non-NULL Integer Value.

    **Returns:**
    The Integer Value.
com.sparsity.dex.gdb
Class Values

java.lang.Object
   +-com.sparsity.dex.gdb.Values

All Implemented Interfaces:
   Iterable, Closeable

public class Values
extends Object
implements Closeable, Iterable

Value set class.

This is a set of Value instances, that is there is no duplicated elements.

Use a ValuesIterator to traverse all the elements into the set.

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

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>close()</td>
<td>Closes the Values instance.</td>
</tr>
<tr>
<td>long</td>
<td>count()</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td>boolean</td>
<td>isClosed()</td>
<td>Gets if Values instance has been closed or not.</td>
</tr>
<tr>
<td>ValuesIterator</td>
<td>iterator()</td>
<td>See iterator().</td>
</tr>
<tr>
<td>ValuesIterator</td>
<td>iterator(Order order)</td>
<td>Gets a ValuesIterator.</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.lang.Iterable

iterator
Methods

iterator

public ValuesIterator iterator()

See iterator().

Creates an Ascendent iterator.

count

public long count()

Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

iterator

public ValuesIterator iterator(Order order)

Gets a ValuesIterator.

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

Returns:
ValuesIterator instance.

isClosed

public boolean isClosed()

Gets if Values instance has been closed or not.

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

See Also:
close()

close

public void close()

Closes the Values instance.

It must be called to ensure the integrity of all data.
public class ValuesIterator
extends Object
implements Iterator, Closeable

ValuesIterator 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.util.Iterator

hasNext, next, remove

Methods inherited from interface java.io.Closeable

close
hasNext

public boolean hasNext()

Gets if there are more elements to traverse.

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

remove

public void remove()

Operation not supported.

next

public Value next()

Gets the next element to traverse.

Returns:
The next element.

isClosed

public boolean isClosed()

Gets if ValuesIterator instance has been closed or not.

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

See Also:
close()

close

public void close()

Closes the ValuesIterator instance.

It must be called to ensure the integrity of all data.
Package
com.sparsity.dex.io
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".

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 CSVReader()</td>
<td>Constructs CSVReader.</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 reader.</td>
</tr>
<tr>
<td>int getRow()</td>
<td>The row number for the current row.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void open(String f)</code></td>
<td>Opens the source file path.</td>
</tr>
<tr>
<td><code>boolean read(StringList row)</code></td>
<td>Reads the next row as a string array.</td>
</tr>
<tr>
<td><code>boolean reset()</code></td>
<td>Moves the reader to the beginning.</td>
</tr>
<tr>
<td><code>void setLocale(String localeStr)</code></td>
<td>Sets the locale that will be used to read the file.</td>
</tr>
<tr>
<td><code>void setMultilines(int numExtralines)</code></td>
<td>Allows the use of fields with more than one line.</td>
</tr>
<tr>
<td><code>void setNumLines(int numLines)</code></td>
<td>Used to limit the number of lines that will be read.</td>
</tr>
<tr>
<td><code>void setQuotes(String quotes)</code></td>
<td>Sets the character used to quote fields.</td>
</tr>
<tr>
<td><code>void setSeparator(String sep)</code></td>
<td>Sets the character used to separate fields in the file.</td>
</tr>
<tr>
<td><code>void setSingleLine()</code></td>
<td>Only allows single line fields.</td>
</tr>
<tr>
<td><code>void setStartLine(int startLine)</code></td>
<td>Sets the number of lines to be skipped from the beginning.</td>
</tr>
</tbody>
</table>

Methods inherited from class `com.sparsity.dex.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**

```java
public CSVReader()

Constructs CSVReader.
```

### Methods

**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).

---

### 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)

---

### reset

```java
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.

---

### setStartLine

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

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

**Parameters:**

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

---

### open

```java
public void open(String f) throws IOException
```
Opens the source file path.

**Parameters:**

- f - [in] CSV file path.

**Throws:**

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

---

**setSeparator**

```java
class CSVReader {
    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
class CSVReader {
    public void setQuotes(String quotes) throws RuntimeException {
        Sets the character used to quote fields.
    }
}
```

**Parameters:**

- quotes - [in] Quote character.

**Throws:**

- java.lang.RuntimeException - null

---

**setSingleLine**

```java
class CSVReader {
    public void setSingleLine() {
        Only allows single line fields.
    }
}
```

---

**setLocale**

```java
class CSVReader {
    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 \texttt{getRow}() 
\hspace{1em} \texttt{throws IOException} 

The row number for the current row.

\textbf{Returns:}

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

\textbf{Throws:}

\texttt{java.io.IOException} - If it fails.

read

public boolean \texttt{read(StringList row)} 
\hspace{1em} \texttt{throws IOException} 

Reads the next row as a string array.

\textbf{Parameters:}

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

\textbf{Returns:}

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

\textbf{Throws:}

\texttt{java.io.IOException} - If bad things happen during the read.
**com.sparsity.dex.io**

**Class CSVWriter**

```java
java.lang.Object
   ^---com.sparsity.dex.io.RowWriter
   |   ^--com.sparsity.dex.io.CSVWriter

public class CSVWriter extends RowWriter
```

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.

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

### Constructor Summary

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

### Method Summary

<table>
<thead>
<tr>
<th>void close()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closes the writer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void open(String f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opens the output file path.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void setAutoQuotes(boolean autoquotes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets on/off the automatic quote mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void setForcedQuotes(BooleanList forcequotes)</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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void setLocale(String localeStr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the locale that will be used to write the file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void setQuotes(String quotes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the character used to quote fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void setSeparator(String sep)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the character used to separate fields in the file.</td>
</tr>
</tbody>
</table>
void write(StringList row)
    Writes the next row.

Methods inherited from class com.sparsity.dex.io.RowWriter

close, write

Methods inherited from class java.lang.Object

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

Constructors

CSVWriter

class CSVWriter

    public CSVWriter()
        Creates a new instance.

Methods

setSeparator

public void setSeparator(String sep)
    throws RuntimeException

    Sets the character used to separate fields in the file.

    Parameters:
    sep - [in] Separator character.

    Throws:
    java.lang.RuntimeException - null

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.

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.

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.

close

public void close()
    throws IOException, RuntimeException

Closes the writer.

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
**open**

```java
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.dex.io
Class EdgeTypeExporter

defined in com.sparsity.dex.io

public class EdgeTypeExporter extends TypeExporter

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

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

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public com.sparsity.dex.io.EdgeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs, int hPos, int tPos, int hAttr, int tAttr)</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public com.sparsity.dex.io.EdgeTypeExporter()</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 register(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
<td>See TypeExporter::Run.</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 g)</td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void setHeadAttribute(int attr)</td>
<td>Sets the attribute that will be used to get the value to be dumped for the head of the edge.</td>
</tr>
<tr>
<td>void setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void setHeadPosition(int pos)</td>
<td>Sets the position (index column) of the head attribute in the exported data.</td>
</tr>
<tr>
<td>void setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
</tr>
</tbody>
</table>
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 t)
Sets the type to be exported.

Methods inherited from class com.sparsity.dex.io.TypeExporter
register, run, setAttributes, setFrequency, setGraph, setHeader, setRowWriter, setType

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

Constructors

EdgeTypeExporter
public EdgeTypeExporter(RowWriter rw,
Graph g,
int t,
AttributeList attrs,
int hPos,
int tPos,
int hAttr,
int tAttr)

Creates a new instance.

Parameters:
g - [in] Graph.
t - [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
**setTailPosition**

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

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

**Parameters:**
- `pos` - [in] Tail position

**setAttributes**

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

Sets the list of Attributes.

**Parameters:**
- `attrs` - [in] Attribute identifiers to be exported

**setHeadAttribute**

```java
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

**setRowWriter**

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

Sets the output data destination.

**Parameters:**
- `rw` - null

**register**

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

Registers a new listener.

**Parameters:**
- `tel` - [in] TypeExporterListener to be registered.
run

public void run()
    throws IOException,
            RuntimeException

    See TypeExporter::Run.

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.

setFrequency

public void setFrequency(int freq)

    Sets the frequency of listener notification.

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

    Parameters:
        freq - null

setGraph

public void setGraph(Graph g)

    Sets the graph that will be exported.

    Parameters:
        g - [in] Graph.

setTailAttribute

public void setTailAttribute(int attr)

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

    Parameters:
        attr - [in] Tail Attribute

setType

public void setType(int t)
Sets the type to be exported.

**Parameters:**
- \( t \) - [in] Type identifier.

---

**setHeadPosition**

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

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

**Parameters:**
- \( pos \) - [in] Head position
com.sparsity.dex.io
Class EdgeTypeLoader

public class EdgeTypeLoader
extends TypeLoader

EdgeTypeLoader class.
Specific TypeLoader implementation for edge types.
Author:
  Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public EdgeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr)</td>
</tr>
<tr>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public EdgeTypeLoader()</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 register(TypeLoaderListener tel)</td>
</tr>
<tr>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
</tr>
<tr>
<td>See TypeLoader::Run.</td>
</tr>
<tr>
<td>void runNPhases(int partitions)</td>
</tr>
<tr>
<td>See TypeLoader::RunNPhases.</td>
</tr>
<tr>
<td>void runTwoPhases()</td>
</tr>
<tr>
<td>See TypeLoader::RunTwoPhases.</td>
</tr>
<tr>
<td>void setAttributePositions(Int32List attrsPos)</td>
</tr>
<tr>
<td>Sets the list of attribute positions.</td>
</tr>
<tr>
<td>void setAttributes(AttributeList attrs)</td>
</tr>
<tr>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void setFrequency(int freq)</td>
</tr>
<tr>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void setGraph(Graph g)</td>
</tr>
<tr>
<td>Sets the graph where the data will be loaded.</td>
</tr>
<tr>
<td>void setHeadAttribute(int attr)</td>
</tr>
<tr>
<td>Sets the attribute that will be used to find the head of the edge.</td>
</tr>
</tbody>
</table>
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 t)
Sets the type to be loaded.

Methods inherited from class `com.sparsity.dex.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

*Constructor*

**EdgeTypeLoader**

```java
public EdgeTypeLoader(RowReader rr,
Graph g,
int t,
AttributeList attrs,
Int32List attrsPos,
int hPos,
int tPos,
int hAttr,
int tAttr)
```

Creates a new instance.

**Parameters:**

rr - [in] Input RowReader.
g - [in] Graph.
t - [in] Type identifier.
attrs - [in] Attribute identifiers to be loaded.
attrsPos - [in] Attribute positions (column index >=0). to all listeners.
hPos - [in] The position (index column) for the head value.
tPos - [in] The position (index column) for the tail value.
hAttr - [in] The attribute identifier for the head.
tAttr - [in] The attribute identifier for the tail.

**EdgeTypeLoader**

public **EdgeTypeLoader**()

Creates a new instance.

**Methods**

**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.

**setTailPosition**

public void **setTailPosition**(int pos)

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

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

**Parameters:**

pos - [in] Tail position

**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 - null

**Throws:**

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

**setAttributes**

public void **setAttributes**(AttributeList attrs)
Sets the list of Attributes.

**Parameters:**
- `attrs` - [in] Attribute identifiers to be loaded

---

### runTwoPhases

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

See TypeLoader::RunTwoPhases.

---

### 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

---

### setRowReader

```java
public void setRowReader(RowReader rr)
```

Sets the input data source.

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

---

### runNPhases

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

See TypeLoader::RunNPhases.

**Parameters:**
- `partitions` - null

**Throws:**
- `java.io.IOException` - null
- `java.lang.RuntimeException` - null

---

### 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 TypeLoader::Run.

---

**setAttributePositions**

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

Sets the list of attribute positions.

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

**Parameters:**

attrsPos - null

---

**setFrequency**

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

Sets the frequency of listener notification.

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

**Parameters:**

freq - null

---

**setGraph**

```java
public void setGraph(Graph g)
```

Sets the graph where the data will be loaded.

**Parameters:**

g - [in] Graph.
setTimestampFormat

public void setTimestampFormat(String timestampFormat)

Sets a specific timestamp format.

Parameters:
  timestampFormat - null

setType

public void setType(int t)

Sets the type to be loaded.

Parameters:
  t - [in] Type identifier.

setTailAttribute

public void setTailAttribute(int attr)

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

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

Parameters:
  attr - [in] Tail Attribute

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
public class NodeTypeExporter extends TypeExporter

NodeTypeExporter class.

Specific TypeExporter implementation for node types.

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</td>
<td>NodeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs)</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public</td>
<td>NodeTypeExporter()</td>
<td>Creates a new instance.</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>register(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
<td>See TypeExporter::Run.</td>
</tr>
<tr>
<td>void</td>
<td>setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void</td>
<td>setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void</td>
<td>setGraph(Graph g)</td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void</td>
<td>setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void</td>
<td>setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td>void</td>
<td>setType(int t)</td>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>

Methods inherited from class com.sparsity.dex.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

NodeTypeExporter

public NodeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs)

Creates a new instance.

Parameters:

g - [in] Graph.
t - [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 - null

register

public void register(TypeExporterListener tel)

Registers a new listener.

Parameters:

tel - [in] TypeExporterListener to be registered.
run

public void run()
    throws IOException,
            RuntimeException

    See TypeExporter::Run.

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.

setFrequency

public void setFrequency(int freq)

    Sets the frequency of listener notification.

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

    Parameters:
        freq - null

setAttributes

public void setAttributes(AttributeList attrs)

    Sets the list of Attributes.

    Parameters:
        attrs - [in] Attribute identifiers to be exported

setGraph

public void setGraph(Graph g)

    Sets the graph that will be exported.

    Parameters:
        g - [in] Graph.

setType

public void setType(int t)
Sets the type to be exported.

**Parameters:**

\( \tau \) - [in] Type identifier.
public class **NodeTypeLoader**
extends **TypeLoader**

NodeTypesLoader class.

Specific **TypeLoader** implementation for node types.

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

### Constructor Summary

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

<table>
<thead>
<tr>
<th>public</th>
<th>NodeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos)</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>register(TypeLoaderListener tel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers a new listener.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>run()</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <strong>TypeLoader</strong>::Run.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>runNPhases(int partitions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <strong>TypeLoader</strong>::RunNPhases.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>runTwoPhases()</th>
</tr>
</thead>
<tbody>
<tr>
<td>See <strong>TypeLoader</strong>::RunTwoPhases.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAttributePositions(Int32List attrsPos)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the list of attribute positions.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAttributes(AttributeList attrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the list of Attributes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setFrequency(int freq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the frequency of listener notification.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setGraph(Graph g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the graph where the data will be loaded.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setLocale(String localeStr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the locale that will be used to read the data.</td>
<td></td>
</tr>
</tbody>
</table>
void `setLogError(String path)`
Sets a log error file.

void `setLogOff()`
Turns off all the error reporting.

void `setRowReader(RowReader rr)`
Sets the input data source.

void `setTimestampFormat(String timestampFormat)`
Sets a specific timestamp format.

void `setType(int t)`
Sets the type to be loaded.

Methods inherited from class `com.sparsity.dex.io.TypeLoader`

`register`, `run`, `runNPhases`, `runTwoPhases`, `setAttributePositions`, `setAttributes`,
`setFrequency`, `setGraph`, `setLocale`, `setLogError`, `setLogOff`, `setRowReader`,
`setTimestampFormat`, `setType`

Methods inherited from class `java.lang.Object`

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

Constructors

**NodeTypeLoader**

public `NodeTypeLoader()`

Creates a new instance.

**NodeTypeLoader**

public `NodeTypeLoader(RowReader rr, Graph g, `Int T, AttributeList attrs, `Int32List attrsPos)`

Creates a new instance.

Parameters:

- rr - [in] Input RowReader.
- g - [in] Graph.
- t - [in] Type identifier.
- attrs - [in] Attribute identifiers to be loaded.
- attrsPos - [in] Attribute positions (column index >=0).

Methods
setLogOff

public void setLogOff()

Truns off all the error reporting.

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

setLogError

public void setLogError(String path)

throws IOException

Sets a log error file.

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

Parameters:
  path - null

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

setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:
  attrs - [in] Attribute identifiers to be loaded

runTwoPhases

public void runTwoPhases()

throws IOException,
    RuntimeException

See TypeLoader::RunTwoPhases.

setRowReader

public void setRowReader(RowReader rr)

Sets the input data source.

Parameters:
  rr - [in] Input RowReader.
runNPhases
public void runNPhases(int partitions)
    throws IOException,
            RuntimeException

See TypeLoader::RunNPhases.

Parameters:
    partitions - null

Throws:
    java.io.IOException - null
    java.lang.RuntimeException - null

register
public void register(TypeLoaderListener tel)

Registers a new listener.

Parameters:
    tel - TypeLoaderListener to be registered.

setLocale
public void setLocale(String localeStr)

Sets the locale that will be used to read the data.
It should match the locale used in the rowreader.

Parameters:
    localeStr - [in] The locale string for the read data. See CSVReader.

run
public void run()
    throws IOException,
            RuntimeException

See TypeLoader::Run.

setAttributePositions
public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

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

Parameters:
    attrsPos - null
**setFrequency**

public void **setFrequency**(int freq)

Sets the frequency of listener notification.

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

**Parameters:**
freq - null

**setGraph**

public void **setGraph**(Graph g)

Sets the graph where the data will be loaded.

**Parameters:**

g - [in] Graph.

**setTimestampFormat**

public void **setTimestampFormat**(String timestampFormat)

Sets a specific timestamp format.

**Parameters:**

timestampFormat - null

**setType**

public void **setType**(int t)

Sets the type to be loaded.

**Parameters:**

t - [in] Type identifier.
com.sparsity.dex.io
Class RowReader

java.lang.Object
    + com.sparsity.dex.io.RowReader

Direct Known Subclasses:
    CSVReader

public class RowReader
extends Object

RowReader interface.

Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once close is called no more read operations can be executed.

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 close()</td>
<td>Closes the reader.</td>
</tr>
<tr>
<td>getRow</td>
<td>int getRow()</td>
<td>The row number for the current row.</td>
</tr>
<tr>
<td>read</td>
<td>boolean read(StringList row)</td>
<td>Reads the next row as a string array.</td>
</tr>
<tr>
<td>reset</td>
<td>boolean 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

Methods

close

public void close() throws IOException
    Closes the reader.
getRow

public int getRow()
throws IOException

The row number for the current row.

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

Throws:
java.io.IOException - If it fails.

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.

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.
com.sparsity.dex.io

Class 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.

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 writer.</td>
</tr>
<tr>
<td>void write(StringList row)</td>
<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

Methods

close

public void close()
    throws IOException, RuntimeException

    Closes the writer.

write

public void write(StringList row)
    throws IOException, RuntimeException

    Writes the next row.
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
com.sparsity.dex.io
Class TypeExporter

java.lang.Object
   ^-com.sparsity.dex.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.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void register(TypeExporterListener tel) Registers a new listener.</td>
</tr>
<tr>
<td>void run() Runs export process.</td>
</tr>
<tr>
<td>void setAttributes(AttributeList attrs) Sets the list of Attributes.</td>
</tr>
<tr>
<td>void setFrequency(int freq) Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void setGraph(Graph g) Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void setHeader(boolean header) Sets the presence of a header row.</td>
</tr>
<tr>
<td>void setRowWriter(RowWriter rw) Sets the output data destination.</td>
</tr>
<tr>
<td>void setType(int t) 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
## Methods

### setRowWriter

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

Sets the output data destination.

*Input:* RowWriter.

**Parameters:**
- **rw** - null

### 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.

### 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.

### setFrequency

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

Sets the frequency of listener notification.

*Input:* Frequency in number of rows managed to notify progress to all listeners.

**Parameters:**
- **freq** - null
**setAttributes**

public void **setAttributes**(*AttributeList* attrs)

Sets the list of Attributes.

**Parameters:**

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

---

**setGraph**

public void **setGraph**(*Graph* g)

Sets the graph that will be exported.

**Parameters:**

- *g* - [in] Graph.

---

**setType**

public void **setType**(*int* t)

Sets the type to be exported.

**Parameters:**

- *t* - [in] Type identifier.
public class TypeExporterEvent
extends Object

Provides information about the progress of a type export process.

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>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>getTypeId</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

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.dex.io
Class TypeExporterListener

java.lang.Object
    +-com.sparsity.dex.io.TypeExporterListener

public class TypeExporterListener extends Object

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

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

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>notifyEvent(TypeExporterEvent tee)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method to be notified from a TypeExporter.</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

**notifyEvent**

public void notifyEvent(TypeExporterEvent tee)

    Method to be notified from a TypeExporter.

Parameters:
    tee - [in] Notified event.
**com.sparsity.dex.io**

**Class TypeLoader**

```
java.lang.Object
    +-com.sparsity.dex.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.

Timestamp formats accepted by default (see ScriptParser):  
"yyyy-MM-dd hh:mm:ss.SSS"  
"yyyy-MM-dd hh:mm:ss"  
"yyyy-MM-dd"

**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(TypeLoaderListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
<td>Run the loader.</td>
</tr>
<tr>
<td>void runNPhases(int partitions)</td>
<td>Run the loader for N phases loading.</td>
</tr>
<tr>
<td>void runTwoPhases()</td>
<td>Run the loader for two phases loading.</td>
</tr>
<tr>
<td>void setAttributePositions(Int32List attrsPos)</td>
<td>Sets the list of attribute positions.</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 g)</td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
<tr>
<td>void</td>
<td>setLocale(String localeStr)</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td>Sets the locale that will be used to read the data.</td>
</tr>
<tr>
<td>void</td>
<td>setLogError(String path)</td>
</tr>
<tr>
<td></td>
<td>Sets a log error file.</td>
</tr>
<tr>
<td>void</td>
<td>setLogOff()</td>
</tr>
<tr>
<td></td>
<td>Turns off all the error reporting.</td>
</tr>
<tr>
<td>void</td>
<td>setRowReader(RowReader rr)</td>
</tr>
<tr>
<td></td>
<td>Sets the input data source.</td>
</tr>
<tr>
<td>void</td>
<td>setTimestampFormat(String timestampFormat)</td>
</tr>
<tr>
<td></td>
<td>Sets a specific timestamp format.</td>
</tr>
<tr>
<td>void</td>
<td>setType(int t)</td>
</tr>
<tr>
<td></td>
<td>Sets the type to be loaded.</td>
</tr>
</tbody>
</table>

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

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

### Methods

**runTwoPhases**

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

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

**setLogOff**

```java
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**

```java
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 - null

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

setAttributes

public void setAttributes(AttributeList attrs)

    Sets the list of Attributes.

    Parameters:
    attrs - [in] Attribute identifiers to be loaded

setRowReader

public void setRowReader(RowReader rr)

    Sets the input data source.

    Parameters:
    rr - [in] Input RowReader.

register

public void register(TypeLoaderListener tel)

    Registers a new listener.

    Parameters:
    tel - TypeLoaderListener to be registered.

setLocale

public void setLocale(String localeStr)

    Sets the locale that will be used to read the data.
    It should match the locale used in the rowreader.

    Parameters:
    localeStr - [in] The locale string for the read data. See CSVReader.
setAttributePositions

public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

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

Parameters:
   attrsPos - null

runNPhases

public void runNPhases(int partitions)
   throws IOException, RuntimeException

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are
loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

Parameters:
   partitions - [in] Number of horizontal partitions to perform the load.

Throws:
   java.io.IOException - null
   java.lang.RuntimeException - null

setFrequency

public void setFrequency(int freq)

Sets the frequency of listener notification.

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

Parameters:
   freq - null

setGraph

public void setGraph(Graph g)

Sets the graph where the data will be loaded.

Parameters:
   g - [in] Graph.

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

Sets a specific timestamp format.
Parameters:
- `timestampFormat` - null

**setType**

```java
class TypeLoader
{
    public void setType(int t)
    {
        // Sets the type to be loaded.
        //
        // Parameters:
        // t - [in] Type identifier.
    }
}
```
public class TypeLoaderEvent
extends Object

Provides information about the progress of a type load process.

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 created.</td>
</tr>
<tr>
<td>getPartition</td>
<td>int</td>
<td>Gets the current partition.</td>
</tr>
<tr>
<td>getPhase</td>
<td>int</td>
<td>Gets the current phase.</td>
</tr>
<tr>
<td>getTotalPartitions</td>
<td>int</td>
<td>Gets the total number of partitions.</td>
</tr>
<tr>
<td>getTotalPartitionSteps</td>
<td>int</td>
<td>Gets the total number of steps in the current partition.</td>
</tr>
<tr>
<td>getTotalPhases</td>
<td>int</td>
<td>Gets the total number of phases.</td>
</tr>
<tr>
<td>getTypeId</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

getTotalPhases

public int getTotalPhases()
Gets the total number of phases.

**Returns:**
The total number of phases.

---

### getCount
public long getCount()

Gets the current number of objects created.

**Returns:**
The current number of objects created.

---

### getTotalPartitionSteps
public int getTotalPartitionSteps()

Gets the total number of steps in the current partition.

**Returns:**
The total number steps in the current partition.

---

### isLast
public boolean isLast()

Gets if this is the last event or not.

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

---

### getPartition
public int getPartition()

Gets the current partition.

**Returns:**
The current partition.

---

### getTypeId
public int getTypeId()

Gets the type identifier.

**Returns:**
The type identifier.

getTotalPartitions

public int getTotalPartitions()

Gets the total number of partitions.

Returns:
The total number of partitions.

getPhase

public int getPhase()

Gets the current phase.

Returns:
The current phase.
public class TypeLoaderListener extends Object

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

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(ev)</td>
<td>Method to receive events from a Loader.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

**notifyEvent**

public void notifyEvent(TypeLoaderEvent ev)

Method to receive events from a Loader.

Parameters:
ev - Loader.LoaderEvent with information from a running Loader.
Package
com.sparsity.dex.script
com.sparsity.dex.script

Class ScriptParser

class ScriptParser extends Object

ScriptParser.

The ScriptParser can create schemas and load data from a set of commands in a dex script.

A DEX script contains an ordered list of commands. ScriptParser will execute each one of them in order. Commands will create schemas, define nodes and edges, and load data into a previously defined DEX schema.

There are six main commands: (i) database creation 'create dbgraph': creates a new empty schema into a DEX database, and sets this database to perform the following operations; (ii) database usage 'use dbgraph': opens an existing DEX database and sets this database to perform the following operations; (iii) node type creation 'create node': creates a node type into the database; (iv) edge type creation 'create edge': creates an edge type into the database; (v) node data load 'load nodes': loads an CSV file into the database; (vi) edge data load 'load edges': loads an CSV file into the database.

-- Schema definition --

This creates a DEX graph database:

CREATE (GDB|DBGRAPH) alias INTO filename

where alias is the name of the graph database to be created and filename corresponds to the path to store the dex database.

Instead of creating a new database, you can set an existing one as the operation database of the script:

USE (GDB|DBGRAPH) alias INTO filename

All following commands will be performed on the last created or used graph database.

This creates a node type:

CREATE NODE node_type_name "(" [attribute_name (INTEGER|LONG|DOUBLE|STRING|BOOLEAN|TIMESTAMP|TEXT) [INDEXED|UNIQUE|BASIC] [DEFAULT value], ...] ")"

and this an edge type:

CREATE [UNDIRECTED] EDGE edge_type_name [FROM node_type_name TO node_type_name] "(" [attribute_name (INT|LONG|DOUBLE|STRING|BOOLEAN|TIMESTAMP|TEXT) [DEFAULT value], ...] ") [MATERIALIZE NEIGHBORS]

Here are some examples:

create gdb EXAMPLE into 'ex.dex'

use gdb WIKIPEDIA into 'wikipedia.dex'

create node TITLES (ID int unique, 'TEXT' string, NLC string, TITLE string indexed)

create node IMAGES (ID int unique, NLC string, FILENAME string indexed)

create edge REFS (NLC string, "TEXT" string, TYPE string)

create edge IMGS
create dbgraph FAMILY into 'family.dex'
create node PERSON (NAME string indexed, ID int unique, YEAR int)
create node DOG (NAME string indexed, YEAR int default 2012)
create edge CHILD from PERSON to PERSON (YEAR int)
create undirected edge MARRIED from PERSON to PERSON (YEAR int) materialize neighbors
create edge PET from PERSON to DOG () materialize neighbors
create gdb CARMODEL into 'cars.dex'
create node PERSON (NAME string, ID int unique, YEAR int)
create node CAR (MODEL string, ID int, OWNER int indexed)

Note you may quote name identifiers in order to be able to use reserved words.

Attributes can be defined as follows.

CREATE ATTRIBUTE [type.]name (INT|LONG|DOUBLE|STRING|BOOLEAN|TIMESTAMP|TEXT) [INDEXED|UNIQUE|BASIC] [DEFAULT value]

If no node or edge type name is given, it creates a global attribute.

-- Data node load --

Load nodes command creates nodes and sets attributes values for nodes imported from a CSV. For each CSV row a new node is created.

By default a new log file with the node type name is created to keep the invalid data error messages. But you can set a specific log file name (LOG logfile), abort at the first error instead of keeping a log (LOG ABORT) or turn off all the invalid data error reporting (LOG OFF).

This is the command:

LOAD NODES file_name [LOCALE locale_name] COLUMNS attribute_name [alias_name], ... INTO node_type_name [IGNORE (attribute_name|alias_name), ....] [FIELDS [TERMINATED char] [ENCLOSED char] [ALLOW_MULTILINE [maxExtralines]]] [FROM num] [MAX num] [MODE (ROWS|COLUMNS [SPLIT [PARTITIONS num]])] [LOG (OFF|ABORT|logfile)]

Here there are some examples:

load nodes 'titles.csv' columns ID, 'TEXT', NLC, TITLE into TITLES
load nodes 'images.csv' columns ID, NLC, FILENAME into IMAGES from 2 max 10000 mode columns
load nodes 'people.csv' locale 'en_US.utf8' columns *, DNI, NAME, AGE, *, ADDRESS into PEOPLE mode row

-- Data edge load --

Load edges command creates edges between existing nodes and sets attributes values for those edges imported from a CSV. For each CSV row a new edge is created.

By default a new log file with the edge type name is created to keep the invalid data error messages. But you can set a specific log file name (LOG logfile), abort at the first error instead of keeping a log (LOG ABORT) or turn off all the invalid data error reporting (LOG OFF).

LOAD EDGES file_name [LOCALE locale_name] COLUMNS attribute_name [alias_name], ... INTO node_type_name [IGNORE (attribute_name|alias_name), ....] WHERE TAIL (attribute_name|alias_name) = node_type_name.attribute_name HEAD (attribute_name|alias_name) = node_type_name.attribute_name [FIELDS [TERMINATED char] [ENCLOSED char] [ALLOW_MULTILINE [maxExtralines]]] [FROM num] [MAX num] [MODE (ROWS|COLUMNS [SPLIT [PARTITIONS num]])] [LOG (OFF|ABORT|logfile)]
Tail node is defined by tail property, it looks for the node where attribute value is the same than the node of an specific name with the same value at specific attribute name. In the same way, head node is defined by head property.

Here there are some examples:

```sql
load edges 'references.csv' columns NLC, 'TEXT', TYPE, FROM F, TO T into REFS ignore F, T where tail F = TITLES.ID head T = TITLES.ID mode columns split partitions 3
```

```sql
load edges 'imagesReferences.csv' locale 'es_ES.iso88591' columns From, To into IMGS ignore From, To where tail From = TITLES.ID HEAD To = IMAGES.ID mode rows
```

```sql
load edges 'calls.gz' columns From, To, Time, Long into CALLS ignore From, To where tail From = PEOPLE.DNI head To = PEOPLE.DNI mode columns
```

-- Schema update --

Schema update commands allows for updating the schema of a graph database. Nowadays it is possible to remove node or edge types or attributes. The node attribute indexing can also be modified.

```sql
DROP (NODE|EDGE) name
DROP ATTRIBUTE [type_name.]attribute_name
INDEX [type_name.]attribute_name [INDEXED|UNIQUE|BASIC]
```

When no type_name is given, then it references a global attribute.

Examples:

```sql
drop edge REFS
drop node 'TITLES'
drop attribute PEOPLE.DNI
drop attribute GLOBAL_ID
index PEOPLE.NAME indexed
index CAR.ID unique
```

-- Timestamp Format --

The timestamp format can be set with the command:

```sql
SET TIMESTAMP FORMAT timestamp_format_string
```

After this command, all timestamps data are loaded with the format specified.

Valid format fields:

- `yyyy` -> Year
- `yy` -> Year without century (80-, 20+ from current year)
- `MM` -> Month [1..12]
- `dd` -> Day of month [1..31]
- `hh` -> Hour [0..23]
mm -> Minute [0..59]

ss -> Second [0..59]

SSS -> Millisecond [0..999]

For parsing, if the pattern is ‘yy’, the parser determines the full year relative to the current year. The parser assumes that the two-digit year is within 80 years before or 20 years after the time of processing. For example, if the current year is 2007, the pattern MM/dd/yy assigned the value 01/11/12 parses to January 11, 2012, while the same pattern assigned the value 05/04/64 parses to May 4, 1964.

Default formats accepted when this command is not present:

"yyyy-MM-dd hh:mm:ss.SSS"

"yyyy-MM-dd hh:mm:ss"

"yyyy-MM-dd"

-- Default Attribute value --

The default value of an attribute can be set with the command:

SET ATTRIBUTE type.attribute_name DEFAULT value

Where the value should be of the same datatype as the attribute being set or NULL.

After this command, all the new nodes or edges with this attribute will be created with this value for this attribute.

--------

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Public Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScriptParser()</strong></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>generateSchemaScript(String path, Database db)</strong></td>
<td>Writes an script with the schema definition for the given database.</td>
</tr>
<tr>
<td><strong>main()</strong></td>
<td>Executes ScriptParser for the given file path.</td>
</tr>
<tr>
<td><strong>parse(String path, boolean execute, String localeStr)</strong></td>
<td>Parses the given input file.</td>
</tr>
<tr>
<td><strong>setErrorLog(String path)</strong></td>
<td>Sets the error log.</td>
</tr>
<tr>
<td><strong>setOutputLog(String path)</strong></td>
<td>Sets the output log.</td>
</tr>
</tbody>
</table>

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

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

ScriptParser

public ScriptParser()

   Constructor.

Methods

setErrorLog

public void setErrorLog(String path)
   throws IOException

   Sets the error log.

   If not set, error log corresponds to standard error output.

   Parameters:
      path - [in] Path of the error log.

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

generateSchemaScript

public static void generateSchemaScript(String path,
   Database db)
   throws IOException

   Writes an script with the schema definition for the given database.

   Parameters:
      path - [in] Filename of the script to be writen.
      db - [in] Database.

   Throws:
      java.io.IOException - If bad things happen opening or writing the file.

parse

public boolean parse(String path,
   boolean execute,
   String localeStr)
   throws IOException

   Parses the given input file.

   Parameters:
      path - [in] Input file path.
      execute - [in] If TRUE the script is executed, if FALSE it is just parsed.
localeStr - [in] The locale string for reading the input file. See CSVReader.

Returns:
TRUE if ok, FALSE in case of error.

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

public void setOutputLog(String path)
throws IOException

Sets the output log.
If not set, output log corresponds to standard output.

Parameters:
path - [in] Path of the output log.

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

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 69, 77, 153, 173, 176, 185, 200, 211
addAll 177
addAllEdgeTypes 13, 16, 21, 26, 31, 34, 39, 43, 46, 50, 54, 59
addAllNodeTypes 7, 13, 16, 21, 26, 31, 34, 38, 43, 46, 50, 54, 58
addEdgeType 12, 17, 20, 25, 30, 35, 39, 42, 47, 51, 55, 59
addNodeType 8, 12, 17, 20, 25, 31, 35, 40, 42, 47, 51, 55, 60
addWeightedEdgeType 30
Any 124
any 171
Ascendent 189
asDirected 122
AttributeList 68

B

backup 141
Basic 66
begin 197
Between 83
Boolean 91
BooleanList 77
Box 166

C

clear 68, 77, 153, 174, 185, 199, 211
close 5, 8, 13, 17, 43, 86, 99, 173, 181, 196, 205, 228, 230, 234, 239, 261, 263
combineDifference 177
combineIntersection 173
combineUnion 172
commit 196
compare 219
compareTo 220, 222
compute 12
Config 158
ConnectedComponents 4
contains 174, 177
containsAll 172
Context 11
copy 173, 179
count 68, 78, 154, 174, 186, 199, 212, 228
countEdges 138
countNodes 142
create 98
CSVReader 233
CSVWriter 238

D

Debug 159
DefaultExport 94
degree 136
Descendent 190
Dex 98
DexConfig 103
difference 171
Double 91
drop 144, 145
dumpData 134
dumpStorage 136

E

Edge 183
EdgeExport 119
edges 140
EdgeTypeExporter 242
EdgeTypeLoader 247, 248
enableType 94, 127
Equal 82
equals 172, 174, 221, 224
excludeEdges 8, 12, 18, 21, 26, 31, 36, 40, 43, 48, 52, 56, 60
excludeNodes 7, 13, 16, 21, 26, 31, 35, 39, 43, 47, 51, 54, 59
exists 20, 24, 30, 177
explode 143, 145
export 137

F

findAttribute 139
findAttributes 142
findEdge 145
findEdgeTypes 138
findNodeTypes 140
findObject 147
findType 148
findTypes 134
Fine 158

G

generateSchemaScript 285
get 113
getAlias 85
getAreNeighborsIndexed 207
getAttribute 141, 146, 148
getAttributesIntervalCount 135
getAttributes 137
getAttributeStatistics 137
getAttributeText 138
getAvailableMem 193
getAvgLengthString 75
getBoolean 114, 220
getCache 89
getCacheMaxSize 109
getCacheStatisticsEnabled 105
getCacheStatisticsFile 109
getCacheStatisticsSnapshotTime 109
ggetColor 119, 162
ggetColorRGB 120, 163
getConnectedComponent 4
getConnectedComponents 8, 35, 40, 55, 60
gGetCost 22, 25, 31
gGetCount 4, 63, 268, 277
gGetCurrentDepth 43, 47, 51
gGetData 89
gGetDataType 64, 221
gGetDistinct 74
gGetDouble 224
gGetEdge 94, 117, 126
gGetEdgeData 142
gGetEdgePeer 147
gGetEdgeType 95, 126
gGetExtentPages 106
gGetExtentSize 104
gGetFontSize 120, 162
gGetGraph 94, 126, 196
gGetHeight 116
gGetHeight 165

getHighAvailabilityCoordinators 107
getHighAvailabilityEnabled 107
getHighAvailabilityIP 111
getHighAvailabilityMasterHistory 111
getHighAvailabilitySynchronization 103
getId 64, 208
getInteger 114, 226
getIsDirected 208
getIsRestricted 208
getKind 63
getLabel 122, 151, 165
getLabelColor 121, 164
getLabelColorRGB 121, 164
gGetLicense 110
gGetLogFile 105
gGetLogLevel 111
gGetLong 226
gGetMax 74
gGetMaxLengthString 75
gGetMean 74
gGetMedian 74
gGetMin 73
gGetMinLengthString 73
gGetMode 73
gGetModeCount 75
gGetName 64, 208
getNode 95, 127
getNodes 5
gGetNodeType 95, 125
gGetNull 74
gGetNumCPUs 194
gGetNumObjects 208
gGetObjectType 146, 207
gGetOID 218
gGetPartition 277
gGetPath 86
gGetPathAsEdges 21, 25, 30
gGetPathAsNodes 22, 25, 30
gGetPhase 278
gGetPoolFrameSize 109
gGetPoolPersistentMaxSize 106
gGetPoolPersistentMinSize 112
gGetPoolTemporaryMaxSize 110
gGetPoolTemporaryMinSize 110
gGetRead 89
getRealTime 193
getRecoveryCacheMaxSize 111
getRecoveryCheckpointTime 105
getRecoveryEnabled 107
getRecoveryLogFile 106
getRestrictedFrom 207
getRestrictedTo 208
getRow 235, 261
getSessions 88
getShape 162
getSize 5, 63
getStatistics 86, 191
getString 222
getSystemTime 193
getTail 116
getTemp 89
getTimestamp 226
getTimestampAsCalendar 218
getTimestampAsDate 222
g getTimeUnit 114
g getTotal 75, 269
g getTotalMem 193
g getTotalPartitions 278
g getTotalPartitionSteps 277
g getTotalPhases 276
g getType 139
g getTypeId 63, 269, 277
g getUserTime 193
g getValues 147
g getVariances 73
g getWidth 121, 164
g getWrite 88
GlobalType 207
GraphExport 150
GraphML 129
Graphviz 128
GreaterEqual 82
GreaterThan 82

I

indexAttribute 139
Indexed 66
Info 158
Ingoing 123
Int32List 153
Integer 91
intersection 178
InvalidAttribute 62
Invalid_OID 171
InvalidType 207
isClosed 5, 8, 13, 17, 44, 86, 99, 175, 181, 196, 228, 230
isEmpty 173
isFit 163
isLast 268, 277
isNull 204, 223
isSessionAttribute 63
iterator 68, 77, 153, 178, 185, 199, 211, 227, 228
iteratorFromElement 172
iteratorFromIndex 175

L

LessEqual 82
LessThan 82
Like 83
LikeNoCase 83
load 114
Long 91

M

main 286
MaxLengthString 217

N

neighbors 136, 143
newAttribute 140, 146
newEdge 146, 147
newEdgeType 144
newNode 137
newNodeType 143
newObjects 197

H

hashCode 219
hasNext 42, 46, 50, 70, 79, 156, 181, 187, 201, 213, 229
heads 144
newRestrictedEdgeType 148
newSession 86
newSessionAttribute 141, 143
next 44, 47, 51, 71, 80, 156, 181, 188, 202, 214, 230
nextAttribute 71
nextBoolean 80
nextInt32 155
nextObject 180
nextOID 188
nextString 202
nextType 214
Node 182
NodeExport 161
NodeTypeExporter 253
NodeTypeLoader 257
NotEqual 83
notifyEvent 270, 279

O

Off 158
OID 92
OIDList 185
open 99, 234, 240
Outgoing 124

P

parse 285
PlatformStatistics 192
prepare 95, 127

R

read 204, 236, 262
RegExp 83
register 243, 249, 253, 259, 266, 273
release 95, 127
remove 71, 80, 156, 171, 175, 181, 188, 202, 214, 230
removeAll 176
removeAttribute 141
removeType 148
reset 234, 262
restore 98
retainAll 174

Round 167
run 7, 16, 20, 26, 32, 34, 39, 54, 59, 243, 250, 253, 259, 266, 272
runNPPhases 249, 258, 274
runTwoPhases 249, 258, 272

S

sample 178
ScriptParser 285
select 135, 138, 140
set 218
setAsDirected 121
setAttribute 134
setAttributeDefaultValue 139
setAttributePositions 250, 259, 274
setAttributes 243, 248, 254, 258, 266, 273
setAttributeText 134
setAutoQuotes 239
setBoolean 220
setBooleanVoid 219
setCacheMaxSize 107
setCacheStatisticsEnabled 111
setCacheStatisticsFile 109
setCacheStatisticsSnapshotTime 105
setColor 121, 164
setColorRGB 122, 165
setDefaults 120, 151, 163
setDouble 221
setDoubleVoid 219
setErrorLog 285
setExtentPages 110
setExtentSize 106
setFit 163
setFontSize 120, 163
set ForcedQuotes 239
setFrequency 244, 250, 254, 260, 266, 274
setGraph 244, 250, 254, 260, 267, 274
setHeadAttribute 243, 249
setHeader 244, 254, 266
setHeadPosition 245, 251
setHeight 163
setHighAvailabilityCoordinators 105
setHighAvailabilityEnabled 104
setHighAvailabilityIP 104
setHighAvailabilityMasterHistory 110
setHighAvailabilitySynchronization 108
setInteger 226
setIntegerVoid 224
setLabel 122, 150, 164
setLabelColor 122, 165
setLabelColorRGB 120, 162
setLicense 108
setLocale 235, 238, 250, 259, 273
setLogError 248, 258, 272
setLogFile 112
setLogLevel 106
setLogOff 248, 257, 272
setLong 222
setLongVoid 218
setMaterializedAttribute 7, 35, 39, 55, 59
setMaxumHops 11, 17, 22, 26, 32, 44, 48, 52
setMultilines 233
setNull 225
setNullVoid 221
setNumLines 234
setOID 225
setOIDVoid 221
setOutputLog 286
setPoolFrameSize 108
setPoolPersistentMaxSize 107
setPoolPersistentMinSize 104
setPoolTemporaryMaxSize 108
setPoolTemporaryMinSize 110
setQuotes 235, 238
setRecoveryCacheMaxSize 104
setRecoveryCheckpointTime 112
setRecoveryEnabled 108
setRecoveryLogFile 107
setRowReader 249, 258, 273
setRowWriter 243, 253, 266
setSeparator 235, 238
setSession 205
setShape 165
setSingleLine 235
setStartLine 234
setString 219
setStringVoid 225
setTailAttribute 244, 251
setTailPosition 242, 248
setText 220
setTimestamp 222, 223, 225
setTimestampFormat 251, 260, 274
setTimestampVoid 221, 223
setType 244, 251, 254, 260, 267, 275
setUnweightedEdgeCost 32
setVoid 224
setWidth 120, 162
Severe 158
SinglePairShortestPathBFS 24
SinglePairShortestPathDijkstra 29
size 178
String 91
StringList 199
StrongConnectivityGabow 38
T
tails 144
tailsAndHeads 135
Text 92
TextStream 204
Timestamp 91
toArray 176
toString 224, 226
TraversalBFS 46
TraversalDFS 50
TypeList 211
U
union 175
Unique 66
V
Value 217, 218
valueOf 66, 84, 92, 124, 129, 159, 167, 183, 190
values 66, 84, 92, 124, 129, 159, 167, 183, 190
Version 98
W
Warning 158
WeakConnectivityDFS 58
write 204, 239, 263

Y

YGraphML 129