Contents

1 Class Index
  1.1 Class Hierarchy ........................................... 1

2 Class Index
  2.1 Class List .............................................. 4

3 Class Documentation
  3.1 sparksee.Attribute Class Reference .......................... 7
      3.1.1 Detailed Description ................................ 8
      3.1.2 Member Function Documentation ........................ 8
  3.2 sparksee.AttributeKind Class Reference ...................... 9
      3.2.1 Detailed Description ................................ 10
      3.2.2 Member Data Documentation .......................... 10
  3.3 sparksee.AttributeList Class Reference ...................... 10
      3.3.1 Detailed Description ................................ 10
      3.3.2 Member Function Documentation ........................ 11
  3.4 sparksee.AttributeListIterator Class Reference .......... 11
      3.4.1 Detailed Description ................................ 12
      3.4.2 Member Function Documentation ........................ 12
  3.5 sparksee.AttributeStatistics Class Reference .......... 12
      3.5.1 Detailed Description ................................ 13
      3.5.2 Member Function Documentation ........................ 13
  3.6 sparksee.BooleanList Class Reference ...................... 16
      3.6.1 Detailed Description ................................ 16
      3.6.2 Member Function Documentation ........................ 16
  3.7 sparksee.BooleanListIterator Class Reference .......... 17
      3.7.1 Detailed Description ................................ 17
      3.7.2 Member Function Documentation ........................ 18
  3.8 sparksee.CommunitiesSCD Class Reference .............. 18
      3.8.1 Detailed Description ................................ 19
      3.8.2 Member Function Documentation ........................ 20
  3.9 sparksee.CommunityDetection Class Reference .......... 22
      3.9.1 Detailed Description ................................ 23
      3.9.2 Member Function Documentation ........................ 23
  3.10 sparksee.Condition Class Reference .................... 24
      3.10.1 Detailed Description ................................ 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Class Reference</th>
<th>Detailed Description</th>
<th>Member Function Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.2</td>
<td>Member Data Documentation</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>sparksee.ConnectedComponents Class Reference</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3.11.1</td>
<td>Detailed Description</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3.11.2</td>
<td>Member Function Documentation</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3.12</td>
<td>sparksee.Connectivity Class Reference</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>3.12.1</td>
<td>Detailed Description</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3.12.2</td>
<td>Member Function Documentation</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3.13</td>
<td>sparksee.Context Class Reference</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>3.13.1</td>
<td>Detailed Description</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>3.13.2</td>
<td>Member Function Documentation</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>3.14</td>
<td>sparksee.CSVReader Class Reference</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>3.14.1</td>
<td>Detailed Description</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>3.14.2</td>
<td>Member Function Documentation</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>3.15</td>
<td>sparksee.CSVWriter Class Reference</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>3.15.1</td>
<td>Detailed Description</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>3.15.2</td>
<td>Member Function Documentation</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>3.16</td>
<td>sparksee.Database Class Reference</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>3.16.1</td>
<td>Detailed Description</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>3.16.2</td>
<td>Member Function Documentation</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>3.17</td>
<td>sparksee.DatabaseStatistics Class Reference</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>3.17.1</td>
<td>Detailed Description</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>3.17.2</td>
<td>Member Function Documentation</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>3.18</td>
<td>sparksee.DataType Class Reference</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>3.18.1</td>
<td>Detailed Description</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>3.18.2</td>
<td>Member Data Documentation</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>3.19</td>
<td>sparksee.DefaultExport Class Reference</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>3.19.1</td>
<td>Detailed Description</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>3.19.2</td>
<td>Member Function Documentation</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>3.20</td>
<td>sparksee.DisjointCommunities Class Reference</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>3.20.1</td>
<td>Detailed Description</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>3.20.2</td>
<td>Member Function Documentation</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>3.21</td>
<td>sparksee.DisjointCommunityDetection Class Reference</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>3.21.1</td>
<td>Detailed Description</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>3.21.2</td>
<td>Member Function Documentation</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>3.22</td>
<td>sparksee.EdgeData Class Reference</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>3.22.1</td>
<td>Detailed Description</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>Section Number</td>
<td>Class Reference</td>
<td>Page Number</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3.35</td>
<td>sparksee.NodeShape Class Reference</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>3.35.1</td>
<td>Detailed Description</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>3.36</td>
<td>sparksee.NodeTypeExporter Class Reference</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>3.36.1</td>
<td>Detailed Description</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>3.36.2</td>
<td>Member Function Documentation</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>3.37</td>
<td>sparksee.NodeTypeLoader Class Reference</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>3.37.1</td>
<td>Detailed Description</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>3.37.2</td>
<td>Member Function Documentation</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>3.38</td>
<td>sparksee.Objects Class Reference</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>3.38.1</td>
<td>Detailed Description</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>3.38.2</td>
<td>Member Function Documentation</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>3.39</td>
<td>sparksee.ObjsList Class Reference</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>3.39.1</td>
<td>Detailed Description</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>3.39.2</td>
<td>Member Function Documentation</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>3.40</td>
<td>sparksee.ObjsListIterator Class Reference</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>3.40.1</td>
<td>Detailed Description</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>3.41</td>
<td>sparksee.ObjsList Class Reference</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>3.41.1</td>
<td>Detailed Description</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>3.41.2</td>
<td>Member Function Documentation</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>3.42</td>
<td>sparksee.ObjsListIterator Class Reference</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>3.42.1</td>
<td>Detailed Description</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>3.42.2</td>
<td>Member Function Documentation</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>3.43</td>
<td>sparksee.Order Class Reference</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>3.43.1</td>
<td>Detailed Description</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>3.44</td>
<td>sparksee.Platform Class Reference</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>3.44.1</td>
<td>Detailed Description</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>3.44.2</td>
<td>Member Function Documentation</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>3.45</td>
<td>sparksee.PlatformStatistics Class Reference</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>3.45.1</td>
<td>Detailed Description</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>3.45.2</td>
<td>Member Function Documentation</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>3.46</td>
<td>sparksee.Query Class Reference</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>3.46.1</td>
<td>Detailed Description</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>3.46.2</td>
<td>Member Function Documentation</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>3.47</td>
<td>sparksee.QueryContext Class Reference</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>3.47.1</td>
<td>Detailed Description</td>
<td>129</td>
<td></td>
</tr>
<tr>
<td>3.48</td>
<td>sparksee.QueryStream Class Reference</td>
<td>129</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Class Reference</td>
<td>Detailed Description</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>3.60.1</td>
<td>sparksee.SparkseeConfig</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>3.60.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.61</td>
<td>sparksee.SparkseeProperties</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>3.61.1</td>
<td></td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>3.61.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.62</td>
<td>sparksee.StringList</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>3.62.1</td>
<td></td>
<td>174</td>
<td></td>
</tr>
<tr>
<td>3.62.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.63</td>
<td>sparksee.StringListIterator</td>
<td>175</td>
<td></td>
</tr>
<tr>
<td>3.63.1</td>
<td></td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>3.63.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.64</td>
<td>sparksee.StrongConnectivity</td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>3.64.1</td>
<td></td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>3.64.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.65</td>
<td>sparksee.StrongConnectivityGabow</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>3.65.1</td>
<td></td>
<td>179</td>
<td></td>
</tr>
<tr>
<td>3.65.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.66</td>
<td>sparksee.TextStream</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>3.66.1</td>
<td></td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>3.66.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.67</td>
<td>sparkseeTraversal</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>3.67.1</td>
<td></td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>3.67.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.68</td>
<td>sparksee.TraversalBFS</td>
<td>191</td>
<td></td>
</tr>
<tr>
<td>3.68.1</td>
<td></td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>3.68.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.69</td>
<td>sparksee.TraversalDFS</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>3.69.1</td>
<td></td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>3.69.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.70</td>
<td>sparksee.Type</td>
<td>199</td>
<td></td>
</tr>
<tr>
<td>3.70.1</td>
<td></td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>3.70.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.71</td>
<td>sparksee.TypeExporter</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>3.71.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.71.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Class Index

### 1.1 Class Hierarchy

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

- `sparksee.Attribute` 7
- `sparksee.AttributeKind` 9
- `sparksee.AttributeList` 10
- `sparksee.AttributeListIterator` 11
- `sparksee.AttributeStatistics` 12
- `sparksee.BooleanList` 16
- `sparksee.BooleanListIterator` 17
- `sparksee.CommunityDetection` 22
- `sparksee.DisjointCommunityDetection` 54
- `sparksee.CommunitiesSCD` 18
- `sparksee.Condition` 24
- `sparksee.ConnectedComponents` 27
- `sparksee.Connectivity` 29
- `sparksee.StrongConnectivity` 178
- `sparksee.StrongConnectivityGabow` 182
- `sparksee.WeakConnectivity` 230
- `sparksee.WeakConnectivityDFS` 234
- `sparksee.Context` 32
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sparksee.Database</code></td>
<td>43</td>
</tr>
<tr>
<td><code>sparksee.DatabaseStatistics</code></td>
<td>45</td>
</tr>
<tr>
<td><code>sparksee.DataType</code></td>
<td>47</td>
</tr>
<tr>
<td><code>sparksee.DisjointCommunities</code></td>
<td>51</td>
</tr>
<tr>
<td><code>sparksee.EdgeData</code></td>
<td>57</td>
</tr>
<tr>
<td><code>sparksee.EdgeExport</code></td>
<td>58</td>
</tr>
<tr>
<td><code>sparksee.EdgesDirection</code></td>
<td>61</td>
</tr>
<tr>
<td><code>sparksee.ExportManager</code></td>
<td>72</td>
</tr>
<tr>
<td><code>sparksee.DefaultExport</code></td>
<td>48</td>
</tr>
<tr>
<td><code>sparksee.ExportType</code></td>
<td>75</td>
</tr>
<tr>
<td><code>sparksee.Graph</code></td>
<td>76</td>
</tr>
<tr>
<td><code>sparksee.GraphExport</code></td>
<td>95</td>
</tr>
<tr>
<td><code>sparksee.Int32List</code></td>
<td>96</td>
</tr>
<tr>
<td><code>sparksee.Int32ListIterator</code></td>
<td>98</td>
</tr>
<tr>
<td><code>sparksee.LogLevel</code></td>
<td>99</td>
</tr>
<tr>
<td><code>sparksee.NodeExport</code></td>
<td>100</td>
</tr>
<tr>
<td><code>sparksee.NodeShape</code></td>
<td>104</td>
</tr>
<tr>
<td><code>sparksee.Objects</code></td>
<td>112</td>
</tr>
<tr>
<td><code>sparksee.ObjectsIterator</code></td>
<td>119</td>
</tr>
<tr>
<td><code>sparksee.ObjectType</code></td>
<td>121</td>
</tr>
<tr>
<td><code>sparksee.OIDList</code></td>
<td>121</td>
</tr>
<tr>
<td><code>sparksee.OIDListIterator</code></td>
<td>123</td>
</tr>
<tr>
<td><code>sparksee.Order</code></td>
<td>124</td>
</tr>
<tr>
<td><code>sparksee.Platform</code></td>
<td>125</td>
</tr>
<tr>
<td><code>sparksee.PlatformStatistics</code></td>
<td>125</td>
</tr>
<tr>
<td><code>sparksee.Query</code></td>
<td>127</td>
</tr>
<tr>
<td><code>sparksee.QueryContext</code></td>
<td>128</td>
</tr>
<tr>
<td><code>sparksee.QueryStream</code></td>
<td>129</td>
</tr>
<tr>
<td><code>sparksee.ResultSet</code></td>
<td>130</td>
</tr>
<tr>
<td>Class Name</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>sparksee.ResultSetList</td>
<td>133</td>
</tr>
<tr>
<td>sparksee.ResultSetListIterator</td>
<td>135</td>
</tr>
<tr>
<td>sparksee.RowReader</td>
<td>136</td>
</tr>
<tr>
<td>sparksee.CSVReader</td>
<td>36</td>
</tr>
<tr>
<td>sparksee.RowWriter</td>
<td>137</td>
</tr>
<tr>
<td>sparksee.CSVWriter</td>
<td>40</td>
</tr>
<tr>
<td>sparksee.ScriptParser</td>
<td>138</td>
</tr>
<tr>
<td>sparksee.Session</td>
<td>140</td>
</tr>
<tr>
<td>sparksee.ShortestPath</td>
<td>142</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPath</td>
<td>145</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPathBFS</td>
<td>149</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPathDijkstra</td>
<td>154</td>
</tr>
<tr>
<td>sparksee.Sparksee</td>
<td>158</td>
</tr>
<tr>
<td>sparksee.SparkseeConfig</td>
<td>160</td>
</tr>
<tr>
<td>sparksee.SparkseeProperties</td>
<td>173</td>
</tr>
<tr>
<td>sparksee.StringList</td>
<td>175</td>
</tr>
<tr>
<td>sparksee.StringListIterator</td>
<td>177</td>
</tr>
<tr>
<td>sparksee.TextStream</td>
<td>185</td>
</tr>
<tr>
<td>sparksee.Traversal</td>
<td>187</td>
</tr>
<tr>
<td>sparksee.TraversalBFS</td>
<td>191</td>
</tr>
<tr>
<td>sparksee.TraversalDFS</td>
<td>195</td>
</tr>
<tr>
<td>sparksee.Type</td>
<td>199</td>
</tr>
<tr>
<td>sparksee.TypeExporter</td>
<td>202</td>
</tr>
<tr>
<td>sparksee.EdgeTypeExporter</td>
<td>62</td>
</tr>
<tr>
<td>sparksee.NodeTypeExporter</td>
<td>105</td>
</tr>
<tr>
<td>sparksee&gt;TypeExporterEvent</td>
<td>205</td>
</tr>
<tr>
<td>sparksee.TypeExporterListener</td>
<td>206</td>
</tr>
<tr>
<td>sparksee.TypeList</td>
<td>206</td>
</tr>
<tr>
<td>sparksee.TypeListIterator</td>
<td>208</td>
</tr>
</tbody>
</table>
2 Class Index

2.1 Class List

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

- sparksee.Attribute (Attribute data class) 7
- sparksee.AttributeKind (Attribute kind enumeration) 9
- sparksee.AttributeList (Sparksee attribute identifier list) 10
- sparksee.AttributeListIterator (AttributeList iterator class) 11
- sparksee.AttributeStatistics (Attribute statistics class) 12
- sparksee.BooleanList (Boolean list) 16
- sparksee.BooleanListIterator (BooleanList iterator class) 17
- sparksee.CommunitiesSCD (CommunitiesSCD class) 18
- sparksee.CommunityDetection (CommunityDetection class) 22
- sparksee.Condition (Condition operators enumeration) 24
- sparksee.ConnectedComponents (ConnectedComponents class) 27
- sparksee.Connectivity (Connectivity class) 29
- sparksee.Context (Context class) 32
- sparksee.CSVReader (CSVReader interface) 36
- sparksee.CSVWriter (CSVWriter interface) 40
2.1 Class List

sparksee.Database (Database class) 43
sparksee.DatabaseStatistics (Database statistics) 45
sparksee.DataType (Data type (domain) enumeration) 47
sparksee.DefaultExport (Default implementation for ExportManager class) 48
sparksee.DisjointCommunities (DisjointCommunities class) 51
sparksee.DisjointCommunityDetection (DisjointCommunityDetection class) 54
sparksee.EdgeData (Edge data class) 57
sparksee.EdgeExport (Stores edge exporting values) 58
sparksee.EdgesDirection (Edges direction enumeration) 61
sparksee.EdgeTypeExporter (EdgeTypeExporter class) 62
sparksee.EdgeTypeLoader (EdgeTypeLoader class) 66
sparksee.ExportManager (Defines how to export a graph to an external format) 72
sparksee.ExportType (Export type) 75
sparksee.Graph (Graph class) 76
sparksee.GraphExport (Stores the graph exporting values) 95
sparksee.Int32List (Sparksee 32-bit signed integer list) 96
sparksee.Int32ListIterator (Int32List iterator class) 98
sparksee.LogLevel (Log level enumeration) 99
sparksee.NodeExport (Stores the node exporting values) 100
sparksee.NodeShape (Node shape) 104
sparksee.NodeTypeExporter (NodeTypeExporter class) 105
sparksee.NodeTypeLoader (NodeTypeLoader class) 108
sparksee.Objects (Object identifier set class) 112
sparksee.ObjectsIterator (ObjectsIterator class) 119
sparksee.ObjectType (Object type enumeration) 121
sparksee.OIDList (Sparksee object identifier list) 121
sparksee.OIDListIterator (OIDList iterator class) 123
sparksee.Order (Order enumeration) 124
sparksee.Platform (Platform class) 125
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>sparksee.PlatformStatistics (Platform data and statistics)</td>
<td>125</td>
</tr>
<tr>
<td>sparksee.Query (Query class)</td>
<td>127</td>
</tr>
<tr>
<td>sparksee.QueryContext (Query context interface)</td>
<td>128</td>
</tr>
<tr>
<td>sparksee.QueryStream (Query stream interface)</td>
<td>129</td>
</tr>
<tr>
<td>sparksee.ResultSet (ResultSet class)</td>
<td>130</td>
</tr>
<tr>
<td>sparksee.ResultSetList (ResultSet list)</td>
<td>133</td>
</tr>
<tr>
<td>sparksee.ResultSetListIterator (ResultSetList iterator class)</td>
<td>135</td>
</tr>
<tr>
<td>sparksee.RowReader (RowReader interface)</td>
<td>136</td>
</tr>
<tr>
<td>sparksee.RowWriter (RowWriter interface)</td>
<td>137</td>
</tr>
<tr>
<td>sparksee.ScriptParser (ScriptParser)</td>
<td>138</td>
</tr>
<tr>
<td>sparksee.Session (Session class)</td>
<td>140</td>
</tr>
<tr>
<td>sparksee.ShortestPath (ShortestPath class)</td>
<td>142</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPath (SinglePairShortestPath class)</td>
<td>145</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPathBFS (SinglePairShortestPathBFS class)</td>
<td>149</td>
</tr>
<tr>
<td>sparksee.SinglePairShortestPathDijkstra (SinglePairShortestPathDijkstra class)</td>
<td>154</td>
</tr>
<tr>
<td>sparksee.Sparksee (Sparksee class)</td>
<td>158</td>
</tr>
<tr>
<td>sparksee.SparkseeConfig (Sparksee configuration class)</td>
<td>160</td>
</tr>
<tr>
<td>sparksee.SparkseeProperties (Sparksee properties file)</td>
<td>173</td>
</tr>
<tr>
<td>sparksee.StringList (String list)</td>
<td>175</td>
</tr>
<tr>
<td>sparksee.StringListIterator (StringList iterator class)</td>
<td>177</td>
</tr>
<tr>
<td>sparksee.StrongConnectivity (StrongConnectivity class)</td>
<td>178</td>
</tr>
<tr>
<td>sparksee.StrongConnectivityGabow (This class can be used to solve the problem of finding strongly connected components in a directed graph)</td>
<td>182</td>
</tr>
<tr>
<td>sparksee.TextStream (TextStream class)</td>
<td>185</td>
</tr>
<tr>
<td>sparksee.Traversal (Traversal class)</td>
<td>187</td>
</tr>
<tr>
<td>sparksee.TraversalBFS (Breadth-First Search implementation of Traversal)</td>
<td>191</td>
</tr>
<tr>
<td>sparksee.TraversalDFS (Depth-First Search (DFS) implementation of Traversal)</td>
<td>195</td>
</tr>
<tr>
<td>sparksee.Type (Type data class)</td>
<td>199</td>
</tr>
<tr>
<td>sparksee.TypeExporter (Base TypeExporter class)</td>
<td>202</td>
</tr>
</tbody>
</table>
sparksee.TypeExporterEvent (Provides information about the progress of a TypeExporter instance) 205

sparksee.TypeExporterListener (Interface to be implemented to receive TypeExporterEvent events from a TypeExporter) 206

sparksee.TypeList (Sparksee type identifier list) 206

sparksee.TypeListIterator (TypeList iterator class) 208

sparksee.TypeLoader (Base TypeLoader class) 209

sparksee.TypeLoaderEvent (Provides information about the progress of a TypeLoader instance) 213

sparksee.TypeLoaderListener (Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader) 215

sparksee.Value (Value class) 216

sparksee.ValueList (Value list) 224

sparksee.ValueListIterator (ValueList iterator class) 226

sparksee.Values (Value set class) 227

sparksee.ValuesIterator (Values iterator class) 229

sparksee.WeakConnectivity (WeakConnectivity class) 230

sparksee.WeakConnectivityDFS (WeakConnectivityDFS class) 234

3 Class Documentation

3.1 sparksee.Attribute Class Reference

Attribute data class.

Public Member Functions

- def get_kind
  
  Gets the attribute kind.

- def get_count
  
  Gets the number of non-NULL values.

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

- def get_size
  
  Gets the number of different values.
3.1 sparksee.Attribute Class Reference

- def get_type_id
  Gets the Sparksee type identifier.

- def get_data_type
  Gets the data type.

- def get_id
  Gets the Sparksee attribute identifier.

- def get_name
  Gets the unique attribute name.

Static Public Attributes

- int INVALID_ATTRIBUTE = 0
  Invalid attribute identifier constant.

3.1.1 Detailed Description

Attribute data class.
It contains information about an attribute.

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

3.1.2 Member Function Documentation

3.1.2.1 def sparksee.Attribute.get_kind (self)
Gets the attribute kind.

Returns:
The AttributeKind.

3.1.2.2 def sparksee.Attribute.get_count (self)
Gets the number of non-NULL values.

Returns:
The number of non-NULL values.

3.1.2.3 def sparksee.Attribute.is_session_attribute (self)
Check if it’s a session attribute or a persistent one.

Returns:
True if it’s a session attribute, or false otherwise.
3.1.2.4  def sparksee.Attribute.get_size ( self)
Gets the number of different values.

Returns:
   The number of different values.

3.1.2.5  def sparksee.Attribute.get_type_id ( self)
Gets the Sparksee type identifier.

Returns:
   The Sparksee type identifier.

3.1.2.6  def sparksee.Attribute.get_data_type ( self)
Gets the data type.

Returns:
   The DataType.

3.1.2.7  def sparksee.Attribute.get_id ( self)
Gets the Sparksee attribute identifier.

Returns:
   The Sparksee attribute identifier.

3.1.2.8  def sparksee.Attribute.get_name ( self)
Gets the unique attribute name.

Returns:
   The unique attribute name.

3.2  sparksee.AttributeKind Class Reference

Attribute kind enumeration.

Static Public Attributes
   • int BASIC = 0
     Basic attribute (non indexed attribute).
   • int INDEXED = 1
     Indexed attribute.
3.2.1 Detailed Description

**Attribute** kind enumeration.
It determines the indexing-capabilities of an attribute.

**Author:**
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.2.2 Member Data Documentation

3.2.2.1 int sparksee.AttributeKind.UNIQUE = 2  [static]
Unique attribute (indexed + unique restriction).
Unique restriction sets two objects cannot have the same value for an attribute but NULL.

3.3 sparksee.AttributeList Class Reference

*Sparksee* attribute identifier list.

**Public Member Functions**

- **def clear**
  Clears the list.

- **def __iter__**
  Gets a new *TypeListIterator*.

- **def __init__**
  Constructor.

- **def iterator**
  Gets a new *AttributeListIterator*.

- **def count**
  Number of elements in the list.

- **def add**
  Adds a *Sparksee* attribute identifier at the end of the list.
3.3.1 Detailed Description

Sparksee attribute identifier list.
It stores a Sparksee attribute identifier list.
Use AttributeListIterator to access all elements into this collection.

Author:
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.3.2 Member Function Documentation

3.3.2.1 def sparksee.AttributeList.__iter__ (self)

Gets a new TypeListIterator.

Returns:
TypeListIterator instance

3.3.2.2 def sparksee.AttributeList.__init__ (self)

Constructor.
This creates an empty list.

3.3.2.3 def sparksee.AttributeList.iterator (self)

Gets a new AttributeListIterator.

Returns:
AttributeListIterator instance.

3.3.2.4 def sparksee.AttributeList.count (self)

Number of elements in the list.

Returns:
Number of elements in the list.

3.3.2.5 def sparksee.AttributeList.add (self, attr)

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

Parameters:
attr [in] Sparksee attribute identifier.

3.4 sparksee.AttributeListIterator Class Reference

AttributeList iterator class.
Public Member Functions

- def has_next
  
  Gets if there are more elements.

- def next
  
  Moves to the next element.

- def __next__
  
  Used in next().

3.4.1 Detailed Description

AttributesList iterator class.

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

Author:

Sparsity Technologies \url{http://www.sparsity-technologies.com}

3.4.2 Member Function Documentation

3.4.2.1 def sparksee.AttributeListIterator.has_next (self)

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

3.4.2.2 def sparksee.AttributeListIterator.next (self)

Moves to the next element.

Returns:

The next element.

3.4.2.3 def sparksee.AttributeListIterator.__next__ (self)

Used in next().

Returns:

The next element

3.5 sparksee.AttributeStatistics Class Reference

Attributes statistics class.
Public Member Functions

- `def get_min`
  Gets the minimum existing value (BASIC statistics).

- `def get_min_length_string`
  Gets the minimum length.

- `def get_variance`
  Gets the variance.

- `def get_mode`
  Gets the mode.

- `def get_null`
  Gets the number of objects NULL a Value (BASIC statistics).

- `def get_distinct`
  Gets the number of distinct values (BASIC statistics).

- `def get_mean`
  Gets the mean or average.

- `def get_max`
  Gets the maximum existing value (BASIC statistics).

- `def get_median`
  Gets the median.

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

- `def get_max_length_string`
  Gets the maximum length.

- `def get_avg_length_string`
  Gets the average length.

- `def get_mode_count`
  Gets the number of objects with a Value equal to the mode.

3.5.1 Detailed Description

Attribute statistics class.

It contains statistic data about an attribute.

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

3.5.2.1 `def sparksee.AttributeStatistics.get_min (self)`
Gets the minimum existing value (BASIC statistics).

**Returns:**

The minimum existing value.

3.5.2.2 `def sparksee.AttributeStatistics.get_min_length_string (self)`
Gets the minimum length.
If the attribute is not an string attribute, it just returns 0.

**Returns:**

The minimum length.

3.5.2.3 `def sparksee.AttributeStatistics.get_variance (self)`
Gets the variance.
It is computed just for numerical attributes.

**Returns:**

The variance.

3.5.2.4 `def sparksee.AttributeStatistics.get_mode (self)`
Gets the mode.
Mode: Most frequent Value.

**Returns:**

The mode.

3.5.2.5 `def sparksee.AttributeStatistics.get_null (self)`
Gets the number of objects NULL a Value (BASIC statistics).

**Returns:**

The number of objects NULL a Value.
3.5.2.6 def sparksee.AttributeStatistics.get_distinct (self)
Gets the number of distinct values (BASIC statistics).

Returns:
   The number of distinct values.

3.5.2.7 def sparksee.AttributeStatistics.get_mean (self)
Gets the mean or average.
Mean or average: Sum of all Values divided by the number of observations.
It is computed just for numerical attributes.

Returns:
   The mean.

3.5.2.8 def sparksee.AttributeStatistics.get_max (self)
Gets the maximum existing value (BASIC statistics).

Returns:
   The maximum existing value.

3.5.2.9 def sparksee.AttributeStatistics.get_median (self)
Gets the median.
Median: Middle value that separates the higher half from the lower.
If a < b < c, then the median of the list {a, b, c} is b, and if a < b < c < d, then the median of the list {a, b, c, d} is the mean of b and c, i.e. it is (b + c)/2
It is computed just for numerical attributes.

Returns:
   The median.

3.5.2.10 def sparksee.AttributeStatistics.get_total (self)
Gets the number of objects with a non-NULL Value (BASIC statistic).

Returns:
   The number of objects with a non-NULL Value.
3.5.2.11 def sparksee.AttributeStatistics.get_max_length_string (self)

Gets the maximum length.

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

**Returns:**

The maximum length.

3.5.2.12 def sparksee.AttributeStatistics.get_avg_length_string (self)

Gets the average length.

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

**Returns:**

The average length.

3.5.2.13 def sparksee.AttributeStatistics.get_mode_count (self)

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

**Returns:**

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

3.6 sparksee.BooleanList Class Reference

Boolean list.

**Public Member Functions**

- def add

  Adds a Boolean at the end of the list.

- def clear

  Clears the list.

- def __iter__

  Gets a new TypeListIterator.

- def iterator

  Gets a new BooleanListIterator.

- def count

  Number of elements in the list.

- def __init__

  Constructor.
3.6.1 Detailed Description

Boolean list.
It stores a Boolean list.
Use BooleanListIterator to access all elements into this collection.

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

3.6.2 Member Function Documentation

3.6.2.1 def sparksee.BooleanList.add (self, value)
Adds a Boolean at the end of the list.

Parameters:
value [in] Boolean.

3.6.2.2 def sparksee.BooleanList.__iter__ (self)
Gets a new TypeListIterator.

Returns:
TypeListIterator instance

3.6.2.3 def sparksee.BooleanList.iterator (self)
Gets a new BooleanListIterator.

Returns:
BooleanListIterator instance.

3.6.2.4 def sparksee.BooleanList.count (self)
Number of elements in the list.

Returns:
Number of elements in the list.

3.6.2.5 def sparksee.BooleanList.__init__ (self)
Constructor.
This creates an empty list.

3.7 sparksee.BooleanListIterator Class Reference

BooleanList iterator class.
Public Member Functions

- def has_next
  
  Gets if there are more elements.

- def next
  
  Moves to the next element.

- def __next__
  
  Used in next().

3.7.1 Detailed Description

**BooleanList** iterator class.

Iterator to traverse all the strings into a **BooleanList** instance.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.7.2 Member Function Documentation

3.7.2.1 def sparksee.BooleanListIterator.has_next (self)

Gets if there are more elements.

**Returns:**

TRUE if there are more elements, FALSE otherwise.

3.7.2.2 def sparksee.BooleanListIterator.next (self)

Moves to the next element.

**Returns:**

The next element.

3.7.2.3 def sparksee.BooleanListIterator.__next__ (self)

Used in next().

**Returns:**

The next element
3.8 sparksee.CommunitiesSCD Class Reference

CommunitiesSCD class.

Inheritance diagram for sparksee.CommunitiesSCD:

Collaboration diagram for sparksee.CommunitiesSCD:

Public Member Functions

• def add_node_type
  Allows connectivity through nodes of the given type.

• def add_edge_type
  Allows connectivity through edges of the given type.

• def add_all_node_types
  Allows connectivity through all node types of the graph.

• def add_all_edge_types
  Allows connectivity through all edge types of the graph.

• def __init__
  Creates a new instance of CommunitiesSCD.

• def exclude_nodes
  Set which nodes can’t be used.

• def run
  Executes the algorithm.

• def set_materialized_attribute
3.8 sparksee.CommunitiesSCD Class Reference

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

- def get_communities
  Returns the results generated by the execution of the algorithm.

- def exclude_edges
  Set which edges can’t be used.

- def set_look_ahead
  Sets the size of the lookahead iterations to look.

- def is_closed
  Gets if CommunityDetection has been closed or not.

- def close
  Closes the CommunityDetection instance.

3.8.1 Detailed Description

CommunitiesSCD class.


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

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

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

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

Author:

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

3.8.2 Member Function Documentation

3.8.2.1 def sparksee.CommunitiesSCD.add_node_type (self, type)

Allows connectivity through nodes of the given type.

Parameters:

- type  null

Reimplemented from sparksee.DisjointCommunityDetection.
3.8.2.2  def sparksee.CommunitiesSCD.add_edge_type (self, type)
Allows connectivity through edges of the given type.
The edges can be used in Any direction.

Parameters:
  type  [in] Edge type.

Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.3  def sparksee.CommunitiesSCD.add_all_edge_types (self)
Allows connectivity through all edge types of the graph.
The edges can be used in Any direction.
Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.4  def sparksee.CommunitiesSCD.__init__ (self, session)
Creates a new instance of CommunitiesSCD.
After creating this instance is required to indicate the set of edge types and the set of node types which will
be navigated through while traversing the graph in order to find the communities.

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

3.8.2.5  def sparksee.CommunitiesSCD.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage
of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.6  def sparksee.CommunitiesSCD.set_materialized_attribute (self, attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the
results related to the disjoint communities found while executing this algorithm.
Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it
is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the
name of the common attribute type which stores the results. This instance will have all the information
related to the disjoint communities found in the moment of the execution of the algorithm that stored this
data.
It is possible to run the algorithm without specifying this parameter in order to avoid materializing the
results of the execution.
Parameters:

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

Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.7   def sparksee.CommunitiesSCD.get_communities ( self)

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

Returns:

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

Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.8   def sparksee.CommunitiesSCD.exclude_edges ( self, edges)

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

Parameters:

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

Reimplemented from sparksee.DisjointCommunityDetection.

3.8.2.9   def sparksee.CommunitiesSCD.set_look_ahead ( self, lookahead)

Sets the size of the lookahead iterations to look.

Parameters:

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

3.8.2.10  def sparksee.CommunityDetection.is_closed ( self)  [inherited]

Gets if CommunityDetection has been closed or not.

See also:

close()

Returns:

TRUE if the CommunityDetection instance has been closed, FALSE otherwise.
3.8.2.11  def sparksee.CommunityDetection.close (self)  [inherited]
Closes the CommunityDetection instance.
It must be called to ensure the integrity of all data.

3.9  sparksee.CommunityDetection Class Reference

CommunityDetection class.
Inheritance diagram for sparksee.CommunityDetection:

Public Member Functions

• def add_node_type
  Allows connectivity through nodes of the given type.

• def add_all_node_types
  Allows connectivity through all node types of the graph.

• def run
  Runs the algorithm in order to find the connected components.

• def exclude_nodes
  Set which nodes can’t be used.

• def is_closed
  Gets if CommunityDetection has been closed or not.

• def exclude_edges
  Set which edges can’t be used.

• def close
  Closes the CommunityDetection instance.

3.9.1  Detailed Description

CommunityDetection class.
Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.
Check out the ’Algorithms’ section in the SPARKSEE User Manual for more details on this.
3.9 sparksee.CommunityDetection Class Reference

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

3.9.2 Member Function Documentation

3.9.2.1 def sparksee.CommunityDetection.add_node_type (self, type)
Allows connectivity through nodes of the given type.

Parameters:
  type null

Reimplemented in sparksee.CommunitiesSCD, and sparksee.DisjointCommunityDetection.

3.9.2.2 def sparksee.CommunityDetection.run (self)
Runs the algorithm in order to find the connected components.
This method can be called only once.
Reimplemented in sparksee.CommunitiesSCD, and sparksee.DisjointCommunityDetection.

3.9.2.3 def sparksee.CommunityDetection.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented in sparksee.CommunitiesSCD, and sparksee.DisjointCommunityDetection.

3.9.2.4 def sparksee.CommunityDetection.is_closed (self)
Gets if CommunityDetection has been closed or not.

See also:
  close()

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

3.9.2.5 def sparksee.CommunityDetection.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.
Parameters:

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

Reimplemented in sparksee.ComunitiesSCD, and sparksee.DisjointCommunityDetection.

3.9.2.6  def sparksee.CommunityDetection.close ( self)
Closes the CommunityDetection instance.
It must be called to ensure the integrity of all data.

3.10 sparksee.Condition Class Reference

Condition operators enumeration.

Static Public Attributes

• int EQUAL = 0
  Equal condition (==).

• int GREATER_EQUAL = 1
  Greater or equal condition (>=).

• int GREATER_THAN = 2
  Greater than condition (>).

• int LESS_EQUAL = 3
  Less or equal condition (<=).

• int LESS_THAN = 4
  Less than condition (<).

• int NOT_EQUAL = 5
  Not equal condition (!=).

• int LIKE = 6
  Substring condition.

• int LIKE_NO_CASE = 7
  Substring (no case sensitive) condition.

• int BETWEEN = 8
  In range operator condition ([x,y]).

• int REG_EXP = 9
  Regular expression condition.
3.10 sparksee.Condition Class Reference

3.10.1 Detailed Description

**Condition** operators enumeration.

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.10.2 Member Data Documentation

3.10.2.1 `int` `sparksee.Condition.EQUAL = 0` [static]

Equal condition (==).

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

3.10.2.2 `int` `sparksee.Condition.GREATER_EQUAL = 1` [static]

Greater or equal condition (>=).

Null values cannot be used together with this condition.

3.10.2.3 `int` `sparksee.Condition.GREATER_THAN = 2` [static]

Greater than condition (>).

Null values cannot be used together with this condition.

3.10.2.4 `int` `sparksee.Condition.LESS_EQUAL = 3` [static]

Less or equal condition (<=).

Null values cannot be used together with this condition.

3.10.2.5 `int` `sparksee.Condition.LESS_THAN = 4` [static]

Less than condition (<).

Null values cannot be used together with this condition.

3.10.2.6 `int` `sparksee.Condition.NOT_EQUAL = 5` [static]

Not equal condition (!=).

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

3.10.2.7 `int` `sparksee.Condition.LIKE = 6` [static]

Substring condition.

Null values cannot be used together with this condition.

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

`'AAABBBCCCD' Like 'BBB' returns TRUE
'AAABBBCCCD' Like 'bbb' returns FALSE
'AAABBBCCCD' Like 'E' returns FALSE

3.10.2.8 int sparksee.Condition.LIKE_NO_CASE = 7 [static]
Substring (no case sensitive) condition.
Null values cannot be used together with this condition.
This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:
'AAABBBCCCD' LikeNoCase 'BBB' returns TRUE
'AAABBBCCCD' LikeNoCase 'bbb' returns TRUE
'AAABBBCCCD' LikeNoCase 'E' returns FALSE

3.10.2.9 int sparksee.Condition.BETWEEN = 8 [static]
In range operator condition ([x,y]).
Null values cannot be used together with this condition.

3.10.2.10 int sparksee.Condition.REG_EXP = 9 [static]
Regular expression condition.
Null values cannot be used together with this condition.
This condition can just be used together with String values.
Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.
See the 'Regular expressions' section in the 'SPARKSEE User Manual' for details.

3.11 sparksee.ConnectedComponents Class Reference

`ConnectedComponents` class.

**Public Member Functions**

- `def __init__`
  `Creates a new instance of ConnectedComponents.``

- `def get_size`
  `Returns the number of nodes contained in the given connected component.``

- `def get_count`
  `Returns the number of connected components found in the graph.``

- `def is_closed`
  `Gets if ConnectedComponents has been closed or not.`
• def get_connected_component
  Returns the connected component where the given node belongs to.

• def get_nodes
  Returns the collection of nodes contained in the given connected component.

• def close
  Closes the ConnectedComponents instance.

3.11.1 Detailed Description

ConnectedComponents class.

This class contains the results processed on a Connectivity algorithm.

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

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

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

Author:

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

3.11.2 Member Function Documentation

3.11.2.1 def sparksee.ConnectedComponents.__init__ (self, s, materializedattribute)

Creates a new instance of ConnectedComponents.

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

Parameters:

  s [in] Session to get the graph Graph on which the information will be retrieved just by getting the
  values contained in the given common attribute type for all the nodes in the graph and processing
  them.

  materializedattribute [in] The common attribute type for all the nodes in the graph where data will be
  retrieved in order to process the results related to the connected components found in the graph.

3.11.2.2 def sparksee.ConnectedComponents.get_size (self, idConnectedComponent)

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

\textit{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.

3.11.2.3 \textbf{def sparksee.ConnectedComponents.get_count (self)}

Returns the number of connected components found in the graph.

Returns:

The number of connected components found in the graph.

3.11.2.4 \textbf{def sparksee.ConnectedComponents.is_closed (self)}

Gets if \texttt{ConnectedComponents} has been closed or not.

\textbf{See also:}

\texttt{close()}

Returns:

TRUE if the \texttt{ConnectedComponents} instance has been closed, FALSE otherwise.

3.11.2.5 \textbf{def sparksee.ConnectedComponents.get_connected_component (self, idNode)}

Returns the connected component where the given node belongs to.

Parameters:

\textit{idNode} \texttt{[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.

3.11.2.6 \textbf{def sparksee.ConnectedComponents.get_nodes (self, idConnectedComponent)}

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

Parameters:

\textit{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.
3.12 sparksee.Connectivity Class Reference

Connectivity class.

Inheritance diagram for sparksee.Connectivity:

```
sparksee.Connectivity
   `-- sparksee.StrongConnectivity
   `-- sparksee.WeakConnectivity
       `-- sparksee.StrongConnectivityGabow
       `-- sparksee.WeakConnectivityDFS
```

Public Member Functions

- `def add_all_node_types`
  
  Allows connectivity through all node types of the graph.

- `def run`
  
  Runs the algorithm in order to find the connected components.

- `def exclude_nodes`
  
  Set which nodes can’t be used.

- `def is_closed`
  
  Gets if Connectivity has been closed or not.

- `def add_node_type`
  
  Allows connectivity through nodes of the given type.

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

- `def exclude_edges`
  
  Set which edges can’t be used.

- `def close`
3.12 sparksee.Connectivity Class Reference

Closes the Connectivity instance.

- def get_connected_components
  Returns the results generated by the execution of the algorithm.

3.12.1 Detailed Description

Connectivity class.

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

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

Author:

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

3.12.2 Member Function Documentation

3.12.2.1 def sparksee.Connectivity.run (self)

Runs the algorithm in order to find the connected components.

This method can be called only once.

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.12.2.2 def sparksee.Connectivity.exclude_nodes (self, nodes)

Set which nodes can’t be used.

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

Parameters:

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

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.12.2.3 def sparksee.Connectivity.is_closed (self)

Gets if Connectivity has been closed or not.

See also:

close()

Returns:

TRUE if the Connectivity instance has been closed, FALSE otherwise.
3.12.2.4  def sparksee.Connectivity.add_node_type (self, t)
Allows connectivity through nodes of the given type.

Parameters:

  t  null

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.12.2.5  def sparksee.Connectivity.set_materialized_attribute (self, 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.

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.12.2.6  def sparksee.Connectivity.exclude_edges (self, edges)
Set which edges can’t be used.

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

Parameters:

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

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.12.2.7  def sparksee.Connectivity.close (self)
Closes the Connectivity instance.

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

3.12.2.8  def sparksee.Connectivity.get_connected_components (self)
Returns the results generated by the execution of the algorithm.

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

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

Reimplemented in sparksee.WeakConnectivityDFS, sparksee.WeakConnectivity, sparksee.StrongConnectivity, and sparksee.StrongConnectivityGabow.

3.13 sparksee.Context Class Reference

Context class.

Public Member Functions

- def add_edge_type
  
  Allows for traversing edges of the given type.

- def compute
  
  Helper method to easily compute a context from a node.

- def exclude_nodes
  
  Set which nodes can’t be used.

- def compute
  
  Gets the resulting collection of nodes.

- def is_closed
  
  Gets if Context has been closed or not.

- def add_all_edge_types
  
  Allows for traversing all edge types of the graph.

- def add_node_type
  
  Allows for traversing nodes of the given type.

- def close
  
  Closes the Context instance.

- def __init__
  
  Creates a new instance.

- def add_all_node_types
  
  Allows for traversing all node types of the graph.

- def set_maximum_hops
  
  Sets the maximum hops restriction.

- def exclude_edges
  
  Set which edges can’t be used.
3.13.1 Detailed Description

The `Context` class.

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

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

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.13.2 Member Function Documentation

3.13.2.1 `def sparksee.Context.add_edge_type (self, t, d)`

Allows for traversing edges of the given type.

Parameters:

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

3.13.2.2 `def sparksee.Context.compute (self, session, node, nodeTypes, edgeTypes, dir, maxhops, include)`

Helper method to easily compute a context from a node.

Parameters:

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

Returns:

Returns an `Objects` with the computed context of a node.

3.13.2.3 `def sparksee.Context.exclude_nodes (self, nodes)`

Set which nodes can’t be used.

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

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

3.13.2.4 def sparksee.Context.compute (self)
Gets the resulting collection of nodes.

Returns:
The resulting collection of nodes.

3.13.2.5 def sparksee.Context.is_closed (self)
Gets if Context has been closed or not.

See also:
close()

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

3.13.2.6 def sparksee.Context.add_all_edge_types (self, d)
Allows for traversing all edge types of the graph.

Parameters:
d [in] Edge direction.

3.13.2.7 def sparksee.Context.add_node_type (self, t)
Allows for traversing nodes of the given type.

Parameters:
t null

3.13.2.8 def sparksee.Context.close (self)
Closes the Context instance.
It must be called to ensure the integrity of all data.

3.13.2.9 def sparksee.Context.__init__ (self, session, node)
Creates a new instance.

Parameters:

session [in] Session to get the graph from and perform operation.
node [in] Node to start traversal from.
3.13.2.10  def sparksee.Context.set_maximum_hops (self, maxhops, include)
Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

Parameters:

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

include [in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

3.13.2.11  def sparksee.Context.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

Parameters:

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

3.14  sparksee.CSVReader Class Reference

CSVReader interface.
Inheritance diagram for sparksee.CSVReader:

Collaboration diagram for sparksee.CSVReader:

Public Member Functions

• def reset
  
  Moves the reader to the beginning.

• def close
Closes the reader.

- **def set_num_lines**
  Used to limit the number of lines that will be read.

- **def set_multilines**
  Allows the use of fields with more than one line.

- **def set_separator**
  Sets the character used to separate fields in the file.

- **def set_quotes**
  Sets the character used to quote fields.

- **def open**
  Opens the source file path.

- **def set_single_line**
  Only allows single line fields.

- **def read**
  Reads the next row as a string array.

- **def set_start_line**
  Sets the number of lines to be skipped from the beginning.

- **def set_locale**
  Sets the locale that will be used to read the file.

- **def get_row**
  The row number for the current row.

- **def __init__**
  Constructs CSVReader.

### 3.14.1 Detailed Description

CSVReader interface.

A very simple CSV reader.

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

Using the format RFC 4180.

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

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

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

The locale string can be used to set the language, country and the file encoding. The format must be "[language_territory][.codeset]". But only the file encoding is being used in the current version.
The languages supported are: "en_US", "es_ES" and "ca_ES". 
The file encodings supported are: "utf8" and "iso88591". 
For example: 
To don’t change the default locales, use an empty string: "". 
To read a file in utf8 with the default language settings use ".utf8". 
To read a file in iso88591 with English language use: "en_US.iso88591". 
Check out the 'Data import’ section in the SPARKSEE User Manual for more details on this.

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

3.14.2 Member Function Documentation

3.14.2.1 def sparksee.CSVReader.reset (self) 
Moves the reader to the beginning. 
Restarts the reader. 

Returns:
true if the reader can be restarted, false otherwise.

Exceptions:

IOError If bad things happen during the restart.

Reimplemented from sparksee.RowReader.

3.14.2.2 def sparksee.CSVReader.close (self) 
Closes the reader. 

Exceptions:

IOError If the close fails.

Reimplemented from sparksee.RowReader.

3.14.2.3 def sparksee.CSVReader.set_num_lines (self, numLines) 
Used to limit the number of lines that will be read. 

Parameters:

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

3.14.2.4 def sparksee.CSVReader.set_multilines (self, numExtralines) 
Allows the use of fields with more than one line.
3.14 sparksee.CSVReader Class Reference

Parameters:

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

3.14.2.5  def sparksee.CSVReader.set_separator ( self, sep)
Sets the character used to separate fields in the file.

Parameters:

   sep  [in] Separator character.

Exceptions:

   RuntimeError  null

3.14.2.6  def sparksee.CSVReader.set_quotes ( self, quotes)
Sets the character used to quote fields.

Parameters:

   quotes  [in] Quote character.

Exceptions:

   RuntimeError  null

3.14.2.7  def sparksee.CSVReader.open ( self, filePath)
Opens the source file path.
File can be optionally compressed in GZIP format.

Parameters:

   filePath  [in] CSV file path.

Exceptions:

   IOError  If bad things happen opening the file.

3.14.2.8  def sparksee.CSVReader.read ( self, row)
Reads the next row as a string array.

Parameters:

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

Returns:

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

\textit{IOException} If bad things happen during the read.

Reimplemented from \texttt{sparksee.RowReader}.

3.14.2.9 \texttt{def sparksee.CSVReader.set_start_line (self, startLine)}

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

**Parameters:**

- \texttt{startLine} [in] The line number to skip for start reading

3.14.2.10 \texttt{def sparksee.CSVReader.set_locale (self, localeStr)}

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

**Parameters:**

- \texttt{localeStr} [in] The locale string for the file encoding.

3.14.2.11 \texttt{def sparksee.CSVReader.get_row (self)}

The row number for the current row.

**Returns:**

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

Exceptions:

\textit{IOException} If it fails.

Reimplemented from \texttt{sparksee.RowReader}.

3.15 \texttt{sparksee.CSVWriter Class Reference}

\texttt{CSVWriter} interface.

Inheritance diagram for \texttt{sparksee.CSVWriter}:
Collaboration diagram for sparksee.CSVWriter:

Public Member Functions

- def set_auto_quotes
  Sets on/off the automatic quote mode.

- def set_separator
  Sets the character used to separate fields in the file.

- def set_quotes
  Sets the character used to quote fields.

- def set_locale
  Sets the locale that will be used to write the file.

- def write
  Writes the next row.

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

- def close
  Closes the writer.

- def __init__
  Creates a new instance.

- def open
  Opens the output file path.

3.15.1 Detailed Description

CSVWriter interface.

A very simple CSV writer implementing RowWriter.

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

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

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

See the CSVReader locale documentation or the SPARKSEE User Manual.

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

3.15.2 Member Function Documentation

3.15.2.1 def sparksee.CSVWriter.set_auto_quotes (self, autoquotes)
Sets on/off the automatic quote mode.
If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced
quotes are provided, there will not be any quote.

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

3.15.2.2 def sparksee.CSVWriter.set_separator (self, sep)
Sets the character used to separate fields in the file.

Parameters:
sep [in] Separator character.

Exceptions:
RuntimeError null

3.15.2.3 def sparksee.CSVWriter.set_quotes (self, quotes)
Sets the character used to quote fields.

Parameters:
quotes [in] Quote character.

Exceptions:
RuntimeError null

3.15.2.4 def sparksee.CSVWriter.set_locale (self, localeStr)
Sets the locale that will be used to write the file.

Parameters:
localeStr [in] The locale string for the file encoding.

3.15.2.5 def sparksee.CSVWriter.write (self, row)
Writes the next row.
Parameters:

   row [in] Row of data.

Exceptions:

   IOError If bad things happen during the write.
   RuntimeError null

Reimplemented from sparksee.RowWriter.

3.15.2.6 def sparksee.CSVWriter.set_forced_quotes (self, forcequotes)
Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

Parameters:

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

3.15.2.7 def sparksee.CSVWriter.close (self)
Closes the writer.

Exceptions:

   IOError If the close fails.
   RuntimeError null

Reimplemented from sparksee.RowWriter.

3.15.2.8 def sparksee.CSVWriter.open (self, f)
Opens the output file path.

Parameters:

   f [in] Output file path.

Exceptions:

   IOError If bad things happen opening the file.

3.16 sparksee.Database Class Reference

Database class.

Public Member Functions

   • def disable_rollback
     Disables the rollback mechanism.
   • def fix_current_cache_max_size
Sets the cache maximum size to the current cache size in use.

- def get_alias
  Gets the alias of the Database.

- def get_path
  Gets the path of the Database.

- def enable_rollback
  Enables the rollback mechanism.

- def is_closed
  Gets if Database instance has been closed or not.

- def new_session
  Creates a new Session.

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

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

- def close
  Closes the Database instance.

- def get_statistics
  Gets Database statistics.

3.16.1 Detailed Description

Database class.

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

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

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

Author:

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

3.16.2 Member Function Documentation

3.16.2.1 def sparksee.Database.fix_current_cache_max_size (self)

Sets the cache maximum size to the current cache size in use.
Returns:
Returns true if successful or false otherwise.

3.16.2.2  def sparksee.Database.get_alias ( self)
Gets the alias of the Database.

Returns:
The alias of the Database.

3.16.2.3  def sparksee.Database.get_path ( self)
Gets the path of the Database.

Returns:
The path of the Database.

3.16.2.4  def sparksee.Database.is_closed ( self)
Gets if Database instance has been closed or not.
See also:
close()

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

3.16.2.5  def sparksee.Database.get_cache_max_size ( self)
Gets the cache maximum size (in MB).

Returns:
Returns the current cache max size.

3.16.2.6  def sparksee.Database.set_cache_max_size ( self, megaBytes)
Sets the cache maximum size (in MB).
0 means unlimited which is all the physical memory of the computer minus a small margin.

Parameters:


3.16.2.7  def sparksee.Database.close ( self)
Closes the Database instance.
It must be called to ensure the integrity of all data.
3.17 sparksee.DatabaseStatistics Class Reference

3.16.2.8 def sparksee.Database.get_statistics (self, stats)

Gets Database statistics.

Parameters:


3.17 sparksee.DatabaseStatistics Class Reference

Database statistics.

Public Member Functions

• def get_write
  Gets total written data in KBytes.

• def get_sessions
  Gets the number of sessions.

• def get_data
  Gets database size in KBytes.

• def get_temp
  Gets temporary storage file size in KBytes.

• def get_read
  Gets total read data in KBytes.

• def get_cache
  Gets cache size in KBytes.

3.17.1 Detailed Description

Database statistics.

Author:

Sparsity Technologieshttp://www.sparsity-technologies.com

3.17.2 Member Function Documentation

3.17.2.1 def sparksee.DatabaseStatistics.get_write (self)

Gets total written data in KBytes.

Returns:

Total read written in KBytes.
3.18 sparksee.DataType Class Reference

Data type (domain) enumeration.

Static Public Attributes

- int BOOLEAN = 0
  
  Boolean data type.

- int INTEGER = 1
  
  32-bit signed integer data type.
• int **LONG** = 2
  64-bit signed integer data type.

• int **DOUBLE** = 3
  64-bit signed double data type.

• int **TIMESTAMP** = 4
  Distance from Epoch (UTC) time in milliseconds precision.

• int **STRING** = 5
  Unicode string data type.

• int **TEXT** = 6
  Large unicode character object (CLOB) data type.

• int **OID** = 7
  Object identifier (OID) data type.

### 3.18.1 Detailed Description

Data type (domain) enumeration.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

### 3.18.2 Member Data Documentation

#### 3.18.2.1 int sparksee.DataType.TIMESTAMP = 4  [static]

Distance from Epoch (UTC) time in milliseconds precision.

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

#### 3.18.2.2 int sparksee.DataType.STRING = 5  [static]

Unicode string data type.

2048 characters maximum length.

#### 3.18.2.3 int sparksee.DataType.TEXT = 6  [static]

Large unicode character object (CLOB) data type.

TextStream
3.19  sparksee.DefaultExport Class Reference

Default implementation for ExportManager class.

Inheritance diagram for sparksee.DefaultExport:

Collaboration diagram for sparksee.DefaultExport:

Public Member Functions

- def enable_type
  Default implementation of the ExportManager class method EnableType.

- def get_edge
  Default implementation of the ExportManager class method GetEdge.

- def get_graph
  Default implementation of the ExportManager class method GetGraph.

- def get_edge_type
  Default implementation of the ExportManager class method GetEdgeType.

- def get_node_type
  Default implementation of the ExportManager class method GetNodeType.

- def release
  Default implementation of the ExportManager class method Release.

- def get_node
  Default implementation of the ExportManager class method GetNode.

- def prepare
  Default implementation of the ExportManager class method Prepare.

- def __init__
  Creates a new instance.
3.19 sparksee.DefaultExport Class Reference

3.19.1 Detailed Description

Default implementation for ExportManager class.

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.19.2 Member Function Documentation

3.19.2.1 def sparksee.DefaultExport.enable_type (self, type)

Default implementation of the ExportManager class method EnableType.

This enables all node and edge types to be exported.

**Parameters:**

- `type` [in] The type to enable.

**Returns:**

TRUE.

Reimplemented from sparksee.ExportManager.

3.19.2.2 def sparksee.DefaultExport.get_edge (self, edge, edgeExport)

Default implementation of the ExportManager class method GetEdge.

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

**Parameters:**

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

**Returns:**

TRUE.

Reimplemented from sparksee.ExportManager.

3.19.2.3 def sparksee.DefaultExport.get_graph (self, graphExport)

Default implementation of the ExportManager class method GetGraph.

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

**Parameters:**

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

TRUE.

Reimplemented from sparksee.ExportManager.

3.19.2.4 def sparksee.DefaultExport.get_edge_type (self, type, edgeExport)

Default implementation of the ExportManager class method GetEdgeType.
This sets the default EdgeExport values.

Parameters:

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

Returns:

TRUE.

Reimplemented from sparksee.ExportManager.

3.19.2.5 def sparksee.DefaultExport.get_node_type (self, type, nodeExport)

Default implementation of the ExportManager class method GetNodeType.
This sets the default NodeExport values.

Parameters:

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

Returns:

TRUE.

Reimplemented from sparksee.ExportManager.

3.19.2.6 def sparksee.DefaultExport.get_node (self, node, nodeExport)

Default implementation of the ExportManager class method GetNode.
This sets the default NodeExport values and sets the OID as the label.

Parameters:

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

Returns:

TRUE.

Reimplemented from sparksee.ExportManager.
3.19.2.7 `def sparksee.DefaultExport.prepare (self, graph)`
Default implementation of the ExportManager class method Prepare.

**Parameters:**

- `graph`: null

Reimplemented from sparksee.ExportManager.

3.20 sparksee.DisjointCommunities Class Reference

DisjointCommunities class.

**Public Member Functions**

- `def __init__`
  Creates a new instance of DisjointCommunities.

- `def get_size`
  Returns the number of nodes contained in the given community.

- `def get_count`
  Returns the number of communities found in the graph.

- `def is_closed`
  Gets if DisjointCommunities has been closed or not.

- `def get_nodes`
  Returns the collection of nodes contained in the given community.

- `def get_community`
  Returns the disjoint community where the given node belongs to.

- `def close`
  Closes the DisjointCommunities instance.

3.20.1 Detailed Description

DisjointCommunities class.

This class contains the results processed on a DisjointCommunityDetection algorithm.

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

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

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

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

3.20.2 Member Function Documentation

3.20.2.1 def sparksee.DisjointCommunities.__init__ (self, session, materializedattribute)

Creates a new instance of DisjointCommunities.

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

Parameters:

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

3.20.2.2 def sparksee.DisjointCommunities.get_size (self, idCommunity)

Returns the number of nodes contained in the given community.

Parameters:

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

Returns:

The number of nodes contained in the given community.

3.20.2.3 def sparksee.DisjointCommunities.get_count (self)

Returns the number of communities found in the graph.

Returns:

The number of communities found in the graph.

3.20.2.4 def sparksee.DisjointCommunities.is_closed (self)

Gets if DisjointCommunities has been closed or not.

See also:

close()
Returns: TRUE if the DisjointCommunities instance has been closed, FALSE otherwise.

3.20.2.5 `def sparksee.DisjointCommunities.get_nodes (self, idCommunity)`

Returns the collection of nodes contained in the given community.

**Parameters:**

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

**Returns:**

The collection of node identifiers contained in the given community.

3.20.2.6 `def sparksee.DisjointCommunities.get_community (self, idNode)`

Returns the disjoint community where the given node belongs to.

**Parameters:**

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

**Returns:**

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

3.20.2.7 `def sparksee.DisjointCommunities.close (self)`

Closes the DisjointCommunities instance.

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

3.21 sparksee.DisjointCommunityDetection Class Reference

DisjointCommunityDetection class.

Inheritance diagram for sparksee.DisjointCommunityDetection:
Collaboration diagram for sparksee.DisjointCommunityDetection:

Public Member Functions

- **def add_node_type**
  
  Allows connectivity through nodes of the given type.

- **def add_edge_type**
  
  Allows connectivity through edges of the given type.

- **def add_all_node_types**
  
  Allows connectivity through all node types of the graph.

- **def add_all_edge_types**
  
  Allows connectivity through all edge types of the graph.

- **def exclude_nodes**
  
  Set which nodes can’t be used.

- **def run**
  
  Runs the algorithm in order to find the communities.

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

- **def get_communities**
  
  Returns the results generated by the execution of the algorithm.

- **def exclude_edges**
  
  Set which edges can’t be used.

- **def is_closed**
  
  Gets if CommunityDetection has been closed or not.

- **def close**
  
  Closes the CommunityDetection instance.

### 3.21.1 Detailed Description

**DisjointCommunityDetection** class.

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

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

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

3.21.2 Member Function Documentation

3.21.2.1 def sparksee.DisjointCommunityDetection.add_node_type ( self, type)
 Allows connectivity through nodes of the given type.

Parameters:
  type  null

Reimplemented from sparksee.CommunityDetection.
Reimplemented in sparksee.CommunitiesSCD.

3.21.2.2 def sparksee.DisjointCommunityDetection.add_edge_type ( self, type)
 Allows connectivity through edges of the given type.
The edges can be used in Any direction.

Parameters:
  type  [in] Edge type.

Reimplemented in sparksee.CommunitiesSCD.

3.21.2.3 def sparksee.DisjointCommunityDetection.add_all_edge_types ( self)
 Allows connectivity through all edge types of the graph.
The edges can be used in Any direction.
Reimplemented in sparksee.CommunitiesSCD.

3.21.2.4 def sparksee.DisjointCommunityDetection.exclude_nodes ( self, nodes)
 Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented from sparksee.CommunityDetection.
Reimplemented in sparksee.CommunitiesSCD.
3.21.2.5 def sparksee.DisjointCommunityDetection.run (self)
Runs the algorithm in order to find the communities.
This method can be called only once.
Reimplemented from sparksee.CommunityDetection.
Reimplemented in sparksee.CommunitiesSCD.

3.21.2.6 def sparksee.DisjointCommunityDetection.set_materialized_attribute (self, attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the
results related to the disjoint communities found while executing this algorithm.
Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it
is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the
name of the common attribute type which stores the results. This instance will have all the information
related to the disjoint communities found in the moment of the execution of the algorithm that stored this
data.
It is possible to run the algorithm without specifying this parameter in order to avoid materializing the
results of the execution.

Parameters:

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

Reimplemented in sparksee.CommunitiesSCD.

3.21.2.7 def sparksee.DisjointCommunityDetection.get_communities (self)
Returns the results generated by the execution of the algorithm.
These results contain information related to the disjoint communities found as the number of different
components, the set of nodes contained in each component or many other data.

Returns:

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

Reimplemented in sparksee.CommunitiesSCD.

3.21.2.8 def sparksee.DisjointCommunityDetection.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage
of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

Parameters:

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

Reimplemented from sparksee.CommunityDetection.
Reimplemented in sparksee.CommunitiesSCD.
3.21.2.9  def sparksee.CommunityDetection.is_closed (self)  [inherited]

Gets if CommunityDetection has been closed or not.

See also:
  close()

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

3.21.2.10  def sparksee.CommunityDetection.close (self)  [inherited]

Closes the CommunityDetection instance.

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

3.22  sparksee.EdgeData Class Reference

Edge data class.

Public Member Functions

• def get_head
  Gets the head of the edge.

• def get_tail
  Gets the tail of the edge.

• def get_edge
  Gets the edge identifier.

3.22.1  Detailed Description

Edge data class.

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

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

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

3.22.2  Member Function Documentation

3.22.2.1  def sparksee.EdgeData.get_head (self)

Gets the head of the edge.

Returns:
  The Sparksee edge identifier of the head of the edge.
3.22.2.2 def sparksee.EdgeData.get_tail (self)
Gets the tail of the edge.

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

3.22.2.3 def sparksee.EdgeData.get_edge (self)
Gets the edge identifier.

Returns:
The Sparksee edge identifier.

3.23 sparksee.EdgeExport Class Reference
Stores edge exporting values.

Public Member Functions
• def set_color_rgb
  Sets the edge color.

• def set_as_directed
  Sets if the edge should be managed as directed.

• def get_font_size
  Gets the edge label font size.

• def set_defaults
  Sets to default values.

• def get_color_rgb
  Gets the edge color.

• def get_labelcolor_rgb
  Gets the edge label color.

• def get_width
  Gets the edge width.

• def set_label
  Sets the edge label.

• def as_directed
  Gets if the edge should be managed as directed.

• def __init__
3.23 sparksee.EdgeExport Class Reference

Creates a new instance.

- def get_label
  Gets the edge label.

- def set_labelcolor_rgb
  Sets the edge label color.

- def set_width
  Sets the edge width.

- def set_font_size
  Sets the edge label font size.

3.23.1 Detailed Description

Stores edge exporting values.
Some properties may be ignored depending on the exportation type.
Default values are:
Label: "" (empty string).
As directed: TRUE.
Color: 13882323 (OxD3D3D3, Light gray).
Label color: 0 (Ox000000, Black).
Width: 5px.
Font size: 10.

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

3.23.2 Member Function Documentation

3.23.2.1 def sparksee.EdgeExport.set_color_rgb (self, color)
Sets the edge color.

Parameters:
  color [in] The edge color.

3.23.2.2 def sparksee.EdgeExport.set_as_directed (self, directed)
Sets if the edge should be managed as directed.

Parameters:
  directed [in] If TRUE, use as directed, otherwise use as undirected.
3.23.2.3  def sparksee.EdgeExport.get_font_size ( self)
Gets the edge label font size.

Returns:
    The edge label font size.

3.23.2.4  def sparksee.EdgeExport.get_color_rgb ( self)
Gets the edge color.

Returns:
    The edge color.

3.23.2.5  def sparksee.EdgeExport.get_labelcolor_rgb ( self)
Gets the edge label color.

Returns:
    The edge label color.

3.23.2.6  def sparksee.EdgeExport.get_width ( self)
Gets the edge width.

Returns:
    The edge width.

3.23.2.7  def sparksee.EdgeExport.set_label ( self, label)
Sets the edge label.

Parameters:
    label [in] The edge label.

3.23.2.8  def sparksee.EdgeExport.as_directed ( self)
Gets if the edge should be managed as directed.
TRUE is the default value. If TRUE, use as directed, otherwise use as undirected.

Returns:
    The edge direction.
3.23.2.9  def sparksee.EdgeExport.get_label (self)
Gets the edge label.

Returns:

The edge label.

3.23.2.10  def sparksee.EdgeExport.set_labelcolor_rgb (self, color)
Sets the edge label color.

Parameters:

    color  [in] The edge label color.

3.23.2.11  def sparksee.EdgeExport.set_width (self, width)
Sets the edge width.

Parameters:

    width  [in] The edge width.

3.23.2.12  def sparksee.EdgeExport.set_font_size (self, size)
Sets the edge label font size.

Parameters:

    size  [in] The edge label font size.

3.24  sparksee.EdgesDirection Class Reference

Edges direction enumeration.

Static Public Attributes

- int INGOING = 0
  In-going edges.

- int OUTGOING = 1
  Out-going edges.

- int ANY = 2
  In-going or out-going edges.
3.24.1 Detailed Description

Edges direction enumeration.

Author:

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

3.25 sparksee.EdgeTypeExporter Class Reference

**EdgeTypeExporter** class.

Inheritance diagram for sparksee.EdgeTypeExporter:

```
sparksee.TypeExporter
  ↓
sparksee.EdgeTypeExporter
```

Collaboration diagram for sparksee.EdgeTypeExporter:

```
sparksee.TypeExporter
  ↓
sparksee.EdgeTypeExporter
```

Public Member Functions

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

- **def set_frequency**
  
  Sets the frequency of listener notification.

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

- **def set_type**
  
  Sets the type to be exported.

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

- **def __init__**
  
  Creates a new instance.

- **def set_row_writer**
  
  Sets the output data destination.
• def register
  Registers a new listener.

• def run
  See the TypeExporter class Run method.

• def __init__
  Creates a new instance.

• def set_graph
  Sets the graph that will be exported.

• def set_header
  Sets the presence of a header row.

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

• def set_attributes
  Sets the list of Attributes.

3.25.1 Detailed Description

EdgeTypeExporter class.
Specific TypeExporter implementation for edge types.
Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

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

3.25.2 Member Function Documentation

3.25.2.1 def sparksee.EdgeTypeExporter.set_tail_attribute ( self, attr)
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

Parameters:
  attr [in] Tail Attribute

3.25.2.2 def sparksee.EdgeTypeExporter.set_frequency ( self, freq)
Sets the frequency of listener notification.

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

Reimplemented from sparksee.TypeExporter.
3.25.2.3  \texttt{def sparksee.EdgeTypeExporter.set\_head\_attribute ( self, attr)}
Sets the attribute that will be used to get the value to be dumped for the head of the edge.

\textbf{Parameters:}

\begin{itemize}
\item \textit{attr} [in] Head Attribute
\end{itemize}

3.25.2.4  \texttt{def sparksee.EdgeTypeExporter.set\_type ( self, type)}
Sets the type to be exported.

\textbf{Parameters:}

\begin{itemize}
\item \textit{type} [in] Type identifier.
\end{itemize}

Reimplemented from \texttt{sparksee.TypeExporter}.

3.25.2.5  \texttt{def sparksee.EdgeTypeExporter.set\_tail\_position ( self, pos)}
Sets the position (index column) of the tail attribute in the exported data.

\textbf{Parameters:}

\begin{itemize}
\item \textit{pos} [in] Tail position
\end{itemize}

3.25.2.6  \texttt{def sparksee.EdgeTypeExporter.__init__ ( self, rowWriter, graph, type, attrs, hPos, tPos, hAttr, tAttr)}
Creates a new instance.

\textbf{Parameters:}

\begin{itemize}
\item \textit{rowWriter} [in] Output \texttt{RowWriter}.
\item \textit{graph} [in] Graph.
\item \textit{type} [in] Type identifier.
\item \textit{attrs} [in] Attribute identifiers to be exported.
\item \textit{hPos} [in] The position (index column) for the head value.
\item \textit{tPos} [in] The position (index column) for the tail value.
\item \textit{hAttr} [in] The attribute identifier to get the value to be dumped for the head.
\item \textit{tAttr} [in] The attribute identifier to get the value to be dumped for the tail.
\end{itemize}

3.25.2.7  \texttt{def sparksee.EdgeTypeExporter.set\_row\_writer ( self, rw)}
Sets the output data destination.

\textbf{Parameters:}

\begin{itemize}
\item \textit{rw} [in] Input \texttt{RowWriter}.
\end{itemize}

Reimplemented from \texttt{sparksee.TypeExporter}.
3.25.2.8  def sparksee.EdgeTypeExporter.register (self, tel)
Registers a new listener.

Parameters:
    tel [in] TypeExporterListener to be registered.

Reimplemented from sparksee.TypeExporter.

3.25.2.9  def sparksee.EdgeTypeExporter.run (self)
See the TypeExporter class Run method.

Exceptions:
    IOError null
    RuntimeError null

Reimplemented from sparksee.TypeExporter.

3.25.2.10 def sparksee.EdgeTypeExporter.set_graph (self, graph)
Sets the graph that will be exported.

Parameters:
    graph [in] Graph.

Reimplemented from sparksee.TypeExporter.

3.25.2.11 def sparksee.EdgeTypeExporter.set_header (self, header)
Sets the presence of a header row.

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

Reimplemented from sparksee.TypeExporter.

3.25.2.12 def sparksee.EdgeTypeExporter.set_head_position (self, pos)
Sets the position (index column) of the head attribute in the exported data.

Parameters:
    pos [in] Head position

3.25.2.13 def sparksee.EdgeTypeExporter.set_attributes (self, attrs)
Sets the list of Attributes.

Parameters:
    attrs [in] Attribute identifiers to be exported

Reimplemented from sparksee.TypeExporter.
3.26 sparksee.EdgeTypeLoader Class Reference

EdgeTypeLoader class.

Inheritance diagram for sparksee.EdgeTypeLoader:

```
sparksee.EdgeTypeLoader
```

Collaboration diagram for sparksee.EdgeTypeLoader:

```
sparksee.EdgeTypeLoader
```

Public Member Functions

- `def set_tail_attribute`
  
  Sets the attribute that will be used to find the tail of the edge.

- `def set_frequency`
  
  Sets the frequency of listener notification.

- `def set_log_off`
  
  Turns off all the error reporting.

- `def set_log_error`
  
  Sets a log error file.

- `def set_head_attribute`
  
  Sets the attribute that will be used to find the head of the edge.

- `def set_type`
  
  Sets the type to be loaded.

- `def run_two_phases`
  
  See the TypeLoader class RunTwoPhases method.

- `def set_tail_position`
  
  Sets the position of the tail attribute in the source data.

- `def set_row_reader`
  
  Sets the input data source.

- `def set_attribute_positions`
3.26. sparksee.EdgeTypeLoader Class Reference

Sets the list of attribute positions.

- def register
  Registers a new listener.

- def setlocale
  Sets the locale that will be used to read the data.

- def run
  See the TypeLoader class Run method.

- def setgraph
  Sets the graph where the data will be loaded.

- def __init__
  Creates a new instance.

- def runNphases
  See the TypeLoader class RunNPhases method.

- def __init__
  Creates a new instance.

- def settimestampformat
  Sets a specific timestamp format.

- def setheadposition
  Sets the position of the head attribute in the source data.

- def setattributes
  Sets the list of Attributes.

3.26.1 Detailed Description

EdgeTypeLoader class.
Specific TypeLoader implementation for edge types.
Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

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

3.26.2 Member Function Documentation

3.26.2.1 def sparksee.EdgeTypeLoader.set_tail_attribute ( self, attr)
Sets the attribute that will be used to find the tail of the edge.
This method is protected because only the Edge loaders should have it.
Parameters:

[attr] [in] Tail Attribute

3.26.2.2  def sparksee.EdgeTypeLoader.set_frequency (self, freq)
Sets the frequency of listener notification.
Parameters:

[freq] [in] Frequency in number of rows managed to notify progress to all listeners
Reimplemented from sparksee.TypeLoader.

3.26.2.3  def sparksee.EdgeTypeLoader.set_log_off (self)
Turns off all the error reporting.
The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.
Reimplemented from sparksee.TypeLoader.

3.26.2.4  def sparksee.EdgeTypeLoader.set_log_error (self, path)
Sets a log error file.
By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.
Parameters:

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

IOError  If bad things happen opening the file.
Reimplemented from sparksee.TypeLoader.

3.26.2.5  def sparksee.EdgeTypeLoader.set_head_attribute (self, attr)
Sets the attribute that will be used to find the head of the edge.
This method is protected because only the Edge loaders should have it.
Parameters:

[attr] [in] Head Attribute

3.26.2.6  def sparksee.EdgeTypeLoader.set_type (self, type)
Sets the type to be loaded.
Parameters:

[type] [in] Type identifier.
Reimplemented from sparksee.TypeLoader.
3.26.2.7 def sparksee.EdgeTypeLoader.run_two_phases (self)
See the TypeLoader class RunTwoPhases method.

Exceptions:

    IOError  null
    RuntimeError  null

Reimplemented from sparksee.TypeLoader.

3.26.2.8 def sparksee.EdgeTypeLoader.set_tail_position (self, pos)
Sets the position of the tail attribute in the source data.
This method is protected because only the Edge loaders should have it.

Parameters:

    pos  [in] Tail position

3.26.2.9 def sparksee.EdgeTypeLoader.set_row_reader (self, rr)
Sets the input data source.

Parameters:

    rr  [in] Input RowReader.

Reimplemented from sparksee.TypeLoader.

3.26.2.10 def sparksee.EdgeTypeLoader.set_attribute_positions (self, attrsPos)
Sets the list of attribute positions.

Parameters:

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

Reimplemented from sparksee.TypeLoader.

3.26.2.11 def sparksee.EdgeTypeLoader.register (self, tel)
Registers a new listener.

Parameters:

    tel  TypeLoaderListener to be registered.

Reimplemented from sparksee.TypeLoader.
3.26.2.12 `def sparksee.EdgeTypeLoader.set_locale (self, 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.

Reimplemented from `sparksee.TypeLoader`.

3.26.2.13 `def sparksee.EdgeTypeLoader.run (self)`
See the `TypeLoader` class Run method.

**Exceptions:**

- `IOError` null
- `RuntimeError` null

Reimplemented from `sparksee.TypeLoader`.

3.26.2.14 `def sparksee.EdgeTypeLoader.set_graph (self, graph)`
Sets the graph where the data will be loaded.

**Parameters:**

- `graph` [in] Graph.

Reimplemented from `sparksee.TypeLoader`.

3.26.2.15 `def sparksee.EdgeTypeLoader.__init__ (self, rowReader, graph, type, attrs, attrsPos, hPos, tPos, hAttr, tAttr)`
Creates a new instance.

**Parameters:**

- `rowReader` [in] Input RowReader.
- `graph` [in] Graph.
- `type` [in] Type identifier.
- `attrs` [in] Attribute identifiers to be loaded.
- `attrsPos` [in] Attribute positions (column index >=0). to all listeners.
- `hPos` [in] The position (index column) for the head value.
- `tPos` [in] The position (index column) for the tail value.
- `hAttr` [in] The attribute identifier for the head.
- `tAttr` [in] The attribute identifier for the tail.
3.26.2.16  def sparksee.EdgeTypeLoader.run_n-phases (self, partitions)
See the TypeLoader class RunNPhases method.

Parameters:
    partitions  null

Exceptions:
    IOError  null
    RuntimeError  null

Reimplemented from sparksee.TypeLoader.

3.26.2.17  def sparksee.EdgeTypeLoader.set_timestamp_format (self, timestampFormat)
Sets a specific timestamp format.

Parameters:
    timestampFormat  [in] A string with the timestamp format definition.

Reimplemented from sparksee.TypeLoader.

3.26.2.18  def sparksee.EdgeTypeLoader.set_head_position (self, pos)
Sets the position of the head attribute in the source data.
This method is protected because only the Edge loaders should have it.

Parameters:
    pos  [in] Head position

3.26.2.19  def sparksee.EdgeTypeLoader.set_attributes (self, attrs)
Sets the list of Attributes.

Parameters:
    attrs  [in] Attribute identifiers to be loaded

Reimplemented from sparksee.TypeLoader.

3.27  sparksee.ExportManager Class Reference
Defines how to export a graph to an external format.

Inheritance diagram for sparksee.ExportManager:

sparksee.ExportManager

sparksee.DefaultExport

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
Public Member Functions

- def get_node_type
  
  Gets the default node export definition for the given node type.

- def get_edge
  
  Gets the edge export definition for the given edge.

- def get_graph
  
  Gets the graph export definition.

- def get_edge_type
  
  Gets the default node export definition for the given edge type.

- def prepare
  
  Prepares the graph for the export process.

- def get_node
  
  Gets the node export definition for the given node.

- def release
  
  Ends the export process.

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

3.27.1 Detailed Description

Defines how to export a graph to an external format.

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

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

Author:

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

3.27.2 Member Function Documentation

3.27.2.1 def sparksee.ExportManager.get_node_type ( self, type, 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.
3.27 sparksee.ExportManager Class Reference

Returns:

TRUE.

Reimplemented in sparksee.DefaultExport.

3.27.2.2 def sparksee.ExportManager.get_edge (self, edge, 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.

Reimplemented in sparksee.DefaultExport.

3.27.2.3 def sparksee.ExportManager.get_graph (self, graphExport)

 Gets the graph export definition.

Parameters:

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

Returns:

TRUE.

Reimplemented in sparksee.DefaultExport.

3.27.2.4 def sparksee.ExportManager.get_edge_type (self, type, 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.

Reimplemented in sparksee.DefaultExport.
3.27.2.5 def sparksee.ExportManager.prepare (self, graph)
Prepares the graph for the export process.
It is called once before the export process.

Parameters:

    graph Graph to be exported.

Reimplemented in sparksee.DefaultExport.

3.27.2.6 def sparksee.ExportManager.get_node (self, node, 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.

Reimplemented in sparksee.DefaultExport.

3.27.2.7 def sparksee.ExportManager.release (self)
Ends the export process.
It is called once after the export process.
Reimplemented in sparksee.DefaultExport.

3.27.2.8 def sparksee.ExportManager.enable_type (self, type)
Gets whether a node or edge type must be exported or not.

Parameters:

    type Node or edge type identifier.

Returns:

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

Reimplemented in sparksee.DefaultExport.

3.28 sparksee.ExportType Class Reference

Export type.
Static Public Attributes

- int \texttt{GRAPHVIZ} = 0
  
  Export to Graphviz format.

- int \texttt{GRAPHML} = 1
  
  Export to GraphML format.

- int \texttt{YGRAPHML} = 2
  
  Export to YGRAPHML format.

3.28.1 Detailed Description

Export type.

Author:

Sparsity Technologies \url{http://www.sparsity-technologies.com}

3.28.2 Member Data Documentation

3.28.2.1 \texttt{int sparksee.ExportType.GRAPHVIZ = 0} \ [static]

Export to Graphviz format.

Graphviz home page: \url{http://www.graphviz.org}

3.28.2.2 \texttt{int sparksee.ExportType.GRAPHML = 1} \ [static]

Export to GraphML format.

GraphML home page: \url{http://graphml.graphdrawing.org/}

3.28.2.3 \texttt{int sparksee.ExportType.YGRAPHML = 2} \ [static]

Export to YGRAPHML format.

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

3.29 \texttt{sparksee.Graph Class Reference}

Graph class.

Public Member Functions

- def \texttt{set_attribute_text}

  Sets the writable \texttt{TextStream} for the given text attribute and \texttt{OID}.

- def \texttt{dump_data}
Dumps logical data to a file.

- `def find_types
  Gets all existing Sparksee node and edge type identifiers.

- `def rename_attribute
  Renames an attribute.

- `def select
  Selects all OIDs satisfying the given condition for the given attribute.

- `def get_attribute_interval_count
  Gets how many objects have a value into the given range for the given attribute.

- `def tails_and_heads
  Gets all the tails and heads from the given edges collection.

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

- `def rename_type
  Renames a type.

- `def dump_storage
  Dumps internal storage data to a file.

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

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

- `def get_attribute_statistics
  Gets statistics from the given attribute.

- `def new_node
  Creates a new node instance.

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

- `def count_edges
  Gets the number of edges.

- `def find_edge_types
  Gets all existing Sparksee edge type identifiers.

- `def select
  Selects all OIDs satisfying the given condition for the given attribute.
• def index_attribute
  Updates the index of the given attribute.

• def get_type
  Gets information about the given type.

• def find_attribute
  Gets the Sparksee attribute identifier for the given type identifier and attribute name.

• def new_attribute
  Creates a new attribute.

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

• def select
  Selects all OIDs belonging to the given type.

• def select
  Selects all OIDs satisfying the given condition for the given attribute.

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

• def find_node_types
  Gets all existing Sparksee node type identifiers.

• def get_attribute
  Gets the Value for the given attribute and OID.

• def remove_attribute
  Removes the given attribute.

• def set_attribute_default_value
  Sets a default value for an attribute.

• def backup
  Dumps all the data to a backup file.

• def new_session_attribute
  Creates a new Session attribute with a default value.

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

• def get_attribute
  Gets the Value for the given attribute and OID.

• def count_nodes
3.29  sparksee.Graph Class Reference

- def set_attribute
  Sets the Value for the given attribute and OID.

- def get_edge_data
  Gets information about an edge.

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

- def rename_type
  Renames a type.

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

- def new_node_type
  Creates a new node type.

- def new_session_attribute
  Creates a new Session attribute.

- def tails
  Gets all the tails from the given edges collection.

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

- def drop
  Drops the given OID.

- def new_edge_type
  Creates a new edge type.

- def heads
  Gets all the heads from the given edges collection.

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

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

- def find_object
  Finds one object having the given Value for the given attribute.

- def drop
  Drops all the OIDs from the given collection.
3.29 sparksee.Graph Class Reference

- def new_attribute
  Creates a new attribute with a default value.

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

- def get_attribute
  Gets information about the given attribute.

- def export
  Exports the Graph.

- def new_edge
  Creates a new edge instance.

- def get_values
  Gets the Value collection for the given attribute.

- def get_edge_peer
  Gets the other end for the given edge.

- def new_edge
  Creates a new edge instance.

- def select
  Selects all OIDs satisfying the given condition for the given attribute.

- def find_type
  Gets the Sparksee type identifier for the given type name.

- def remove_type
  Removes the given type.

- def new_restricted_edge_type
  Creates a new restricted edge type.

3.29.1 Detailed Description

Graph class.

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

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

Author:

Sparsity Technologies http://www.sparsity-technologies.com
3.29.2 Member Function Documentation

3.29.2.1 def sparksee.Graph.set_attribute_text (self, oid, attr, tstream)
Sets the writable TextStream for the given text attribute and OID.

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

3.29.2.2 def sparksee.Graph.dump_data (self, file)
Dumps logical data to a file.

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

Exceptions:
- IOError If the given file cannot be created.
- RuntimeError null

3.29.2.3 def sparksee.Graph.find_types (self)
Gets all existing Sparksee node and edge type identifiers.

Returns:
- Sparksee node and edge type identifier list.

3.29.2.4 def sparksee.Graph.rename_attribute (self, attr, newName)
Renames an attribute.
The new name must be available.

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

3.29.2.5 def sparksee.Graph.select (self, attr, cond, value, restriction)
Selects all OIDs satisfying the given condition for the given attribute.

Parameters:
- **attr** [in] Sparksee attribute identifier.
- **cond** [in] Condition to be satisfied.
3.29 sparksee.Graph Class Reference

**value** [in] Value to be satisfied.

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

Returns:

- Objects instance.

### 3.29.2.6 def sparksee.Graph.get_attribute_interval_count (self, attr, lower, includeLower, higher, includeHigher)

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

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

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

Parameters:

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

Returns:

- Number of objects having a value into the given range.

### 3.29.2.7 def sparksee.Graph.tails_and_heads (self, edges, tails, heads)

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

Parameters:

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

### 3.29.2.8 def sparksee.Graph.degree (self, oid, etype, dir)

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

Parameters:

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

Returns:

- The number of edges.
3.29.2.9  def sparksee.Graph.rename_type (self, oldName, newName)
Renames a type.
The new name must be available.

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

3.29.2.10 def sparksee.Graph.dump_storage (self, file)
Dumps internal storage data to a file.

Parameters:
    file  [in] Output file path.

Exceptions:
    IOError  If the given file cannot be created.
    RuntimeError  null

3.29.2.11 def sparksee.Graph.neighbors (self, objs, etype, dir)
Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

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

Returns:
    Objects instance.

3.29.2.12 def sparksee.Graph.get_attributes (self, oid)
Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.

Parameters:
    oid  [in] Sparksee OID.

Returns:
    Sparksee attribute identifier list.
3.29.2.13  def sparksee.Graph.get_attribute_statistics (self, attr, basic)

Gets statistics from the given attribute.

Parameters:

attr [in] Sparksee attribute identifier.

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

Returns:

An AttributeStatistics instace.

3.29.2.14  def sparksee.Graph.new_node (self, type)

Creates a new node instance.

Parameters:

type [in] Sparksee type identifier.

Returns:

Unique OID of the new node instance.

3.29.2.15  def sparksee.Graph.get_attribute_text (self, oid, attr)

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

Parameters:

oid [in] Sparksee OID.

attr [in] Sparksee attribute identifier.

Returns:

A TextStream. This returns a TextStream to read.

3.29.2.16  def sparksee.Graph.count_edges (self)

Gets the number of edges.

Returns:

The number of edges.

3.29.2.17  def sparksee.Graph.find_edge_types (self)

Gets all existing Sparksee edge type identifiers.

Returns:

Sparksee edge type identifier list.
3.29.18  def sparksee.Graph.select (self, attr, cond, lower, higher)
Selects all OIDs satisfying the given condition for the given attribute.
This allows to perform the Between operation, thus it has two Value arguments.

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

Returns:
  Objects instance.

3.29.19  def sparksee.Graph.index_attribute (self, attr, kind)
Updates the index of the given attribute.
This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one
is Indexed or Unique.

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

3.29.20  def sparksee.Graph.get_type (self, type)
Gets information about the given type.

Parameters:
  type [in] Sparksee type identifier.

Returns:
  The Type for the given Sparksee type identifier.

3.29.21  def sparksee.Graph.find_attribute (self, type, name)
Gets the Sparksee attribute identifier for the given type identifier and attribute name.

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

Returns:
  The Sparksee attribute identifier for the given type and attribute name or InvalidAttribute if there is no
attribute with the given name for the given type.
3.29.22  `sparksee.Graph.new_attribute (self, type, name, dt, kind)`

Creates a new attribute.

**Parameters:**

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

**Returns:**

Unique Sparksee attribute identifier.

3.29.23  `sparksee.Graph.edges (self, etype, tail, head)`

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

**Parameters:**

- `etype` [in] Sparksee edge type identifier.
- `head` [in] Head node identifier.

**Returns:**

Objects instance.

3.29.24  `sparksee.Graph.select (self, type)`

Selects all OIDs belonging to the given type.

**Parameters:**

- `type` [in] Sparksee type identifier.

**Returns:**

Objects instance.

3.29.25  `sparksee.Graph.select (self, attr, cond, value)`

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

**Parameters:**

- `cond` [in] Condition to be satisfied.
- `value` [in] Value to be satisfied.

**Returns:**

Objects instance.
3.29.2.26  def sparksee.Graph.find_or_create_object ( self, attr, value)

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

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

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

Parameters:

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

Returns:

    Sparksee OID or the Objects InvalidOID.

3.29.2.27  def sparksee.Graph.find_node_types ( self)

Gets all existing Sparksee node type identifiers.

Returns:

    Sparksee node type identifier list.

3.29.2.28  def sparksee.Graph.get_attribute ( self, oid, attr)

Gets the Value for the given attribute and OID.

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

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

Parameters:

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

Returns:

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

3.29.2.29  def sparksee.Graph.remove_attribute ( self, attr)

Removes the given attribute.

Parameters:

    attr [in] Sparksee attribute identifier.
3.29.2.30 `def sparksee.Graph.set_attribute_default_value ( self, attr, value)`

Sets a default value for an attribute.

The default value will be applied to all the new nodes or edges.

The given value must have the same `DataType` as the attribute or be a NULL value to remove the current default value.

**Parameters:**

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

3.29.2.31 `def sparksee.Graph.backup ( self, file)`

Dumps all the data to a backup file.

See the `Sparksee` class Restore method.

**Parameters:**


**Exceptions:**

- `IOError` If the given file cannot be created.
- `RuntimeError` null

3.29.2.32 `def sparksee.Graph.new_session_attribute ( self, type, dt, kind, defaultValue)`

Creates a new `Session` attribute with a default value.

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

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

**Parameters:**

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

**Returns:**

Unique `Sparksee` attribute identifier.

3.29.2.33 `def sparksee.Graph.find_attributes ( self, type)`

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

**Parameters:**

- `type` [in] `Sparksee` type identifier.
Returns:
Sparksee attribute identifier list.

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

Gets the Value for the given attribute and OID.

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

3.29.2.35 def sparksee.Graph.count_nodes ( self)

Gets the number of nodes.

Returns:
The number of nodes.

3.29.2.36 def sparksee.Graph.set_attribute ( self, oid, attr, value)

Sets the Value for the given attribute and OID.

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

3.29.2.37 def sparksee.Graph.get_edge_data ( self, edge)

Gets information about an edge.

Parameters:
edge [in] Sparksee edge identifier.

Returns:
An EdgeData instance.

3.29.2.38 def sparksee.Graph.neighbors ( self, oid, etype, dir)

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

Parameters:
oid [in] Sparksee node OID.
3.29 sparksee.Graph Class Reference

```python
class Graph:

    etype [in] Sparksee edge type identifier.
    dir [in] Direction.

    Returns:
    Objects instance.

3.29.2.39 def sparksee.Graph.rename_type (self, type, newName)
Renames a type.
The new name must be available.

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

3.29.2.40 def sparksee.Graph.explode (self, objs, etype, dir)
Selects all edges from or to each of the node OID in the given collection and for the given edge type.

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

Returns:
    Objects instance.

3.29.2.41 def sparksee.Graph.new_node_type (self, name)
Creates a new node type.

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

Returns:
    Unique Sparksee type identifier.

3.29.2.42 def sparksee.Graph.new_session_attribute (self, type, dt, kind)
Creates a new Session attribute.
Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically
removed when the Session is closed (thus, attribute data is not persistent into the database).
Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.
```
Parameters:

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

Returns:

Unique Sparksee attribute identifier.

### 3.29.2.43 def sparksee.Graph.tails (self, edges)

Gets all the tails from the given edges collection.

Parameters:

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

Returns:

The tails collection.

### 3.29.2.44 def sparksee.Graph.find_or_create_edge (self, etype, tail, head)

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

Parameters:

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

Returns:

Any of the edges or the Objects InvalidOID.

### 3.29.2.45 def sparksee.Graph.drop (self, oid)

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

Parameters:

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

### 3.29.2.46 def sparksee.Graph.new_edge_type (self, name, directed, neighbors)

Creates a new edge type.

Parameters:

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

Returns:
Unique Sparksee type identifier.

3.29.2.47  def sparksee.Graph.heads ( self, edges)
Gets all the heads from the given edges collection.

Parameters:
edges  [in]  Sparksee edge identifier collection.

Returns:
The heads collection.

3.29.2.48  def sparksee.Graph.find_edge ( self, etype, tail, head)
Gets any of the edges of the given type between two given nodes (tail and head).
If there are more than one, then any of them will be returned. And in case there are no edge between the
given tail and head, the Objects InvalidOID will be returned.

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

Returns:
Any of the edges or the Objects InvalidOID.

3.29.2.49  def sparksee.Graph.explode ( self, oid, etype, dir)
Selects all edges from or to the given node OID and for the given edge type.

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

Returns:
Objects instance.
3.29.2.50  def sparksee.Graph.find_object ( self, attr, value)
Finds one object having the given Value for the given attribute.
If there are more than one, then any of them will be returned. And in case there are no object having this
Value, the Objects InvalidOID will be returned.

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

Returns:
    Sparksee OID or the Objects InvalidOID.

3.29.2.51  def sparksee.Graph.drop ( self, objs)
Drops all the OIDs from the given collection.
See Drop method with the single OID parameter. This performs that call for all the elements into the
collection.

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

3.29.2.52  def sparksee.Graph.new_attribute ( self, type, name, dt, kind, defaultValue)
Creates a new attribute with a default value.

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

Returns:
    Unique Sparksee attribute identifier.

3.29.2.53  def sparksee.Graph.get_object_type ( self, oid)
Gets the Sparksee node or edge type identifier for the given OID.

Parameters:
    oid [in] Sparksee OID.

Returns:
    Sparksee node or edge type identifier.
3.29.2.54  def sparksee.Graph.get_attribute (self, attr)
Gets information about the given attribute.

Parameters:
   attr [in] Sparksee attribute identifier.

Returns:
   The Attribute for the given Sparksee attribute identifier.

3.29.2.55  def sparksee.Graph.export (self, file, type, em)
Exports the Graph.

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

Exceptions:
   IOError null

3.29.2.56  def sparksee.Graph.new_edge (self, type, tailAttr, tailV, headAttr, headV)
Creates a new edge instance.
The tail of the edge will be any node having the given tailV Value for the given tailAttr attribute identifier,
and the head of the edge will be any node having the given headV Value for the given headAttr attribute identifier.

Parameters:
   type [in] Sparksee type identifier.
   headAttr [in] Sparksee attribute identifier.
   headV [in] Value.

Returns:
   Unique OID of the new edge instance.

3.29.2.57  def sparksee.Graph.get_values (self, attr)
Gets the Value collection for the given attribute.

Parameters:
   attr [in] Sparksee attribute identifier.

Returns:
   Returns a Values object.
3.29.2.58  def sparksee.Graph.get_edge_peer (self, edge, node)
Gets the other end for the given edge.

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

Returns:
   The other end of the edge.

3.29.2.59  def sparksee.Graph.new_edge (self, type, tail, head)
Creates a new edge instance.

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

Returns:
   Unique OID of the new edge instance.

3.29.2.60  def sparksee.Graph.select (self, attr, cond, lower, higher, restriction)
Selects all OIDs satisfying the given condition for the given attribute.
This allows to perform the Between operation, thus it has two Value arguments.

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

Returns:
   Objects instance.

3.29.2.61  def sparksee.Graph.find_type (self, name)
Gets the Sparksee type identifier for the given type name.

Parameters:
   name  [in] Unique type name.

Returns:
   The Sparksee type identifier for the given type name or the Type InvalidType if there is no type with
   the given name.
3.29.2.62  def sparksee.Graph.remove_type (self, type)
Removes the given type.
This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

Parameters:

  type  [in] Sparksee type identifier.

3.29.2.63  def sparksee.Graph.new_restricted_edge_type (self, name, tail, head, neighbors)
Creates a new restricted edge type.

Parameters:

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

Returns:

  Unique Sparksee type identifier.

3.30 sparksee.GraphExport Class Reference

Stores the graph exporting values.

Public Member Functions

- def get_label
  Gets the graph label.

- def set_defaults
  Sets to default values.

- def __init__
  Creates a new GraphExport instance.

- def set_label
  Sets the graph label.

3.30.1 Detailed Description

Stores the graph exporting values.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
3.30.2 Member Function Documentation

3.30.2.1 def sparksee.GraphExport.get_label ( self)

Gets the graph label.

Returns:
    The graph label.

3.30.2.2 def sparksee.GraphExport.set_label ( self, label)

Sets the graph label.

Parameters:
    label [in] The graph label.

3.31 sparksee.Int32List Class Reference

Sparksee 32-bit signed integer list.

Public Member Functions

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

- def clear
  
  Clears the list.

- def __iter__
  
  Gets a new TypeListIterator.

- def __init__
  
  Constructor.

- def iterator
  
  Gets a new Int32ListIterator.

- def count
  
  Number of elements in the list.

3.31.1 Detailed Description

Sparksee 32-bit signed integer list.

It stores a 32-bit signed integer list.

Use Int32ListIterator to access all elements into this collection.

Author:
    Sparsity Technologies http://www.sparsity-technologies.com
3.31.2 Member Function Documentation

3.31.2.1 def sparksee.Int32List.add (self, value)
Adds an 32-bit signed integer at the end of the list.

Parameters:

value [in] The integer.

3.31.2.2 def sparksee.Int32List.__iter__ (self)
Gets a new TypeListIterator.

Returns:

TypeListIterator instance

3.31.2.3 def sparksee.Int32List.__init__ (self)
Constructor.
This creates an empty list.

3.31.2.4 def sparksee.Int32List.iterator (self)
Gets a new Int32ListIterator.

Returns:

Int32ListIterator instance.

3.31.2.5 def sparksee.Int32List.count (self)
Number of elements in the list.

Returns:

Number of elements in the list.

3.32 sparksee.Int32ListIterator Class Reference

Int32List iterator class.

Public Member Functions

• def has_next
  
  Gets if there are more elements.

• def next
  
  Moves to the next element.

• def __next__
  
  Used in next().
3.32.1 Detailed Description

**Int32List** iterator class.

Iterator to traverse all the integer into a **Int32List** instance.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.32.2 Member Function Documentation

3.32.2.1 **def sparksee.Int32ListIterator.hasNext (self)**

Gets if there are more elements.

**Returns:**

TRUE if there are more elements, FALSE otherwise.

3.32.2.2 **def sparksee.Int32ListIterator.next (self)**

Moves to the next element.

**Returns:**

The next element.

3.32.2.3 **def sparksee.Int32ListIterator.__next__ (self)**

Used in next().

**Returns:**

The next element

3.33 sparksee.LogLevel Class Reference

Log level enumeration.

**Static Public Attributes**

- **int OFF = 0**
  
  Disable log.

- **int SEVERE = 1**

  Severe log level.

- **int WARNING = 2**

  Warning log level.

- **int INFO = 3**
Info log level.

• int CONFIG = 4
  Config log level.

• int FINE = 5
  Fine log level.

• int DEBUG = 6
  Debug log level.

3.33.1 Detailed Description

Log level enumeration.

Log level priority order is as follows, from minimum to maximum log information: Off (log is disabled), Severe, Warning, Info, Config, Fine, Debug.

Author:

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

3.33.2 Member Data Documentation

3.33.2.1 int sparksee.LogLevel.SEVERE = 1 [static]
Severe log level.
This is the lower log level, just errors are shown.

3.33.2.2 int sparksee.LogLevel.WARNING = 2 [static]
Warning log level.
Errors and warnings are shown.

3.33.2.3 int sparksee.LogLevel.INFO = 3 [static]
Info log level.
Errors, warnings and information messages are shown.

3.33.2.4 int sparksee.LogLevel.CONFIG = 4 [static]
Config log level.
Errors, warnings, information messages and configuration details of the different components are shown.

3.33.2.5 int sparksee.LogLevel.FINE = 5 [static]
Fine log level.
This is the higher and finest public log level, everything is dumped to the log.
3.33.2.6  int sparksee.LogLevel.DEBUG = 6  [static]

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

3.34  sparksee.NodeExport Class Reference

Stores the node exporting values.

Public Member Functions

- def get_shape
  Gets the node shape.

- def set_color_rgb
  Sets the node color.

- def set_height
  Sets the node height.

- def get_font_size
  Gets the node label font size.

- def set_defaults
  Sets to default values.

- def get_color_rgb
  Gets the node color.

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

- def get_labelcolor_rgb
  Gets the node label color.

- def get_width
  Gets the node width.

- def set_label
  Sets the node label.

- def get_label
  Gets the node label.

- def get_height
  Gets the node height.

- def __init__
  Creates a new instance.
3.34 sparksee.NodeExport Class Reference

- def set_labelcolor_rgb
  
  Sets the node label color.

- def set_width
  
  Gets the node width.

- def set_shape
  
  Sets the node shape.

- def set_fit
  
  Sets the node fit property.

- def set_font_size
  
  Sets the node label font size.

3.34.1 Detailed Description

Stores the node exporting values.

When 'fit' is set to TRUE, then 'height' and 'width' will be ignored.

Some properties may be ignored depending on the exportation type.

Default values are:

- Label: "" (empty string).
- Shape: Box.
- Color: 10863606 (0xa5c3f6).
- Label color: 0 (0x000000, Black).
- Height: 25px.
- Width: 25px.
- Fit: TRUE.
- Font size: 10.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.34.2 Member Function Documentation

3.34.2.1 def sparksee.NodeExport.get_shape (self)

Gets the node shape.

Returns:

The node shape.
3.34.2.2 def sparksee.NodeExport.set_color_rgb (self, color)
Sets the node color.

Parameters:

  *color* The node color.

3.34.2.3 def sparksee.NodeExport.set_height (self, height)
Sets the node height.

Parameters:

  *height* [in] The node height in pixels.

3.34.2.4 def sparksee.NodeExport.get_font_size (self)
 Gets the node label font size.

Returns:

  The node label font size.

3.34.2.5 def sparksee.NodeExport.get_color_rgb (self)
 Gets the node color.

Returns:

  The node color.

3.34.2.6 def sparksee.NodeExport.is_fit (self)
 Gets whether the node size is fitted to the label or not.

Returns:

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

3.34.2.7 def sparksee.NodeExport.get_labelcolor_rgb (self)
 Gets the node label color.

Returns:

  The node label color.
3.34.2.8  def sparksee.NodeExport.get_width (self)

Gets the node width.

**Returns:**

The node width in pixels.

3.34.2.9  def sparksee.NodeExport.set_label (self, label)

Sets the node label.

**Parameters:**

*label* [in] The node label.

3.34.2.10  def sparksee.NodeExport.get_label (self)

Gets the node label.

**Returns:**

The node label.

3.34.2.11  def sparksee.NodeExport.get_height (self)

Gets the node height.

**Returns:**

The node height in pixels.

3.34.2.12  def sparksee.NodeExport.set_labelcolor_rgb (self, color)

Sets the node label color.

**Parameters:**

*color* [in] The node label color.

3.34.2.13  def sparksee.NodeExport.set_width (self, width)

Gets the node width.

**Parameters:**

*width* The node width in pixels.

3.34.2.14  def sparksee.NodeExport.set_shape (self, shape)

Sets the node shape.

**Parameters:**

*shape* [in] The node shape.
3.34.2.15  def sparksee.NodeExport.set_fit (self, fit)
Sets the node fit property.

Parameters:

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

3.34.2.16  def sparksee.NodeExport.set_font_size (self, size)
Sets the node label font size.

Parameters:

size  [in] The node label font size.

3.35  sparksee.NodeShape Class Reference

Node shape.

Static Public Attributes

• int BOX = 0
  Box shape.

• int ROUND = 1
  Round shape.

3.35.1  Detailed Description

Node shape.

Author:

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

3.36  sparksee.NodeTypeExporter Class Reference

NodeTypeExporter class.

Inheritance diagram for sparksee.NodeTypeExporter:
Collaboration diagram for sparksee.NodeTypeExporter:

Public Member Functions

• def __init__
  Creates a new instance.

• def set_row_writer
  Sets the output data destination.

• def set_frequency
  Sets the frequency of listener notification.

• def register
  Registers a new listener.

• def run
  See the TypeExporter class Run method.

• def set_graph
  Sets the graph that will be exported.

• def set_header
  Sets the presence of a header row.

• def set_type
  Sets the type to be exported.

• def __init__
  Creates a new instance.

• def set_attributes
  Sets the list of Attributes.

3.36.1 Detailed Description

NodeTypeExporter class.
Specific TypeExporter implementation for node types.
Check out the ’Data export’ section in the SPARKSEE User Manual for more details on this.

Author:
  Sparsity Technologies http://www.sparsity-technologies.com
3.36.2 Member Function Documentation

3.36.2.1 def sparksee.NodeTypeExporter.__init__ (self, rowWriter, graph, type, attrs)
Creates a new instance.

Parameters:

graph [in] Graph.
type [in] Type identifier.
attrs [in] Attribute identifiers to be exported.

3.36.2.2 def sparksee NodeTypeExporter.set_row_writer (self, rw)
Sets the output data destination.

Parameters:

rw [in] Input RowWriter.

Reimplemented from sparksee.TypeExporter.

3.36.2.3 def sparksee NodeTypeExporter.set_frequency (self, freq)
Sets the frequency of listener notification.

Parameters:

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

Reimplemented from sparksee.TypeExporter.

3.36.2.4 def sparksee.nodeTypeExporter.register (self, tel)
Registers a new listener.

Parameters:

tel [in] TypeExporterListener to be registered.

Reimplemented from sparksee.TypeExporter.

3.36.2.5 def sparksee.nodeTypeExporter.run (self)
See the TypeExporter class Run method.

Exceptions:

IOError null
RuntimeError null

Reimplemented from sparksee.TypeExporter.
### 3.36.2.6 `def sparksee.NodeTypeExporter.set_graph (self, graph)`
Sets the graph that will be exported.

**Parameters:**

- `graph [in]` Graph.

Reimplemented from `sparksee.TypeExporter`.

### 3.36.2.7 `def sparksee.NodeTypeExporter.set_header (self, header)`
Sets the presence of a header row.

**Parameters:**

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

Reimplemented from `sparksee.TypeExporter`.

### 3.36.2.8 `def sparksee.NodeTypeExporter.set_type (self, type)`
Sets the type to be exported.

**Parameters:**

- `type [in]` Type identifier.

Reimplemented from `sparksee.TypeExporter`.

### 3.36.2.9 `def sparksee.NodeTypeExporter.set_attributes (self, attrs)`
Sets the list of Attributes.

**Parameters:**

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

Reimplemented from `sparksee.TypeExporter`.

### 3.37 `sparksee.NodeTypeLoader Class Reference`

`NodeTypeLoader` class.

Inheritance diagram for `sparksee.NodeTypeLoader`:

```plaintext
sparksee.TypeLoader
    └── sparksee.NodeTypeLoader
```
Collaboration diagram for sparksee.NodeTypeLoader:

Public Member Functions

- def set_frequency
  
  Sets the frequency of listener notification.

- def set_log_off
  
  Turns off all the error reporting.

- def set_log_error
  
  Sets a log error file.

- def __init__
  
  Creates a new instance.

- def set_type
  
  Sets the type to be loaded.

- def run_two_phases
  
  See the TypeLoader class RunTwoPhases method.

- def set_row_reader
  
  Sets the input data source.

- def __init__
  
  Creates a new instance.

- def set_attribute_positions
  
  Sets the list of attribute positions.

- def register
  
  Registers a new listener.

- def set_locale
  
  Sets the locale that will be used to read the data.

- def run
  
  See the TypeLoader class Run method.

- def set_graph
  
  Sets the graph where the data will be loaded.
3.37.1 Detailed Description

NodeTypeLoader class.
Specific TypeLoader implementation for node types.
Check out the ‘Data import’ section in the SPARKSEE User Manual for more details on this.

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

3.37.2 Member Function Documentation

3.37.2.1 def sparksee.NodeTypeLoader.set_frequency (self, freq)
Sets the frequency of listener notification.

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

Reimplemented from sparksee.TypeLoader.

3.37.2.2 def sparksee.NodeTypeLoader.set_log_off (self)
Truns off all the error reporting.
The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.
Reimplemented from sparksee.TypeLoader.

3.37.2.3 def sparksee.NodeTypeLoader.set_log_error (self, path)
Sets a log error file.
By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

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

Exceptions:

IOError If bad things happen opening the file.
Reimplemented from sparksee.TypeLoader.

3.37.2.4 `def sparksee.NodeTypeLoader.__init__(self, rowReader, graph, type, attrs, attrsPos)`

Creates a new instance.

Parameters:
- `rowReader` [in] Input RowReader.
- `graph` [in] Graph.
- `type` [in] Type identifier.
- `attrs` [in] Attribute identifiers to be loaded.
- `attrsPos` [in] Attribute positions (column index >=0).

3.37.2.5 `def sparksee.NodeTypeLoader.set_type(self, type)`

Sets the type to be loaded.

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

Reimplemented from sparksee.TypeLoader.

3.37.2.6 `def sparksee.NodeTypeLoader.run_two_phases(self)`

See the TypeLoader class RunTwoPhases method.

Exceptions:
- `IOError` null
- `RuntimeError` null

Reimplemented from sparksee.TypeLoader.

3.37.2.7 `def sparksee.NodeTypeLoader.set_row_reader(self, rr)`

Sets the input data source.

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

Reimplemented from sparksee.TypeLoader.

3.37.2.8 `def sparksee.NodeTypeLoader.set_attribute_positions(self, attrsPos)`

Sets the list of attribute positions.

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

Reimplemented from sparksee.TypeLoader.
3.37.2.9  def sparksee.NodeTypeLoader.register (self, tel)
Registers a new listener.

Parameters:
   tel  TypeLoaderListener to be registered.

Reimplemented from sparksee.TypeLoader.

3.37.2.10 def sparksee.NodeTypeLoader.set_locale (self, 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.

Reimplemented from sparksee.TypeLoader.

3.37.2.11 def sparksee.NodeTypeLoader.run (self)
See the TypeLoader class Run method.

Exceptions:
   IOError  null
   RuntimeError  null

Reimplemented from sparksee.TypeLoader.

3.37.2.12 def sparksee.NodeTypeLoader.set_graph (self, graph)
Sets the graph where the data will be loaded.

Parameters:
   graph  [in] Graph.

Reimplemented from sparksee.TypeLoader.

3.37.2.13 def sparksee.NodeTypeLoader.run_n_phases (self, partitions)
See the TypeLoader class RunNPhases method.

Parameters:
   partitions  null

Exceptions:
   IOError  null
   RuntimeError  null

Reimplemented from sparksee.TypeLoader.
3.37.2.14  def sparksee.NodeTypeLoader.set_timestamp_format ( self, timestampFormat)
Sets a specific timestamp format.

Parameters:

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

Reimplemented from sparksee.TypeLoader.

3.37.2.15  def sparksee.NodeTypeLoader.set_attributes ( self, attrs)
Sets the list of Attributes.

Parameters:

    attrs  [in]  Attribute identifiers to be loaded

Reimplemented from sparksee.TypeLoader.

3.38  sparksee.Objects Class Reference
Object identifier set class.

Public Member Functions

• def any
  
  Gets an element from the collection.

• def difference
  
  Performs the difference operation.

• def is_closed
  
  Gets if Objects instance has been closed or not.

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

• def iterator_from_element
  
  Gets an ObjectsIterator starting from the given element.

• def exists
  
  Gets if the given element exists into the collection.

• def equals
  
  Checks if the given Objects contains the same information.

• def combine_difference
  
  Creates a new Objects instance which is the difference of the two given.

• def copy
Performs the copy operation.

• def sample
  Creates a new Objects instance which is a sample of the calling one.

• def close
  Closes the Objects instance.

• def combine_intersection
  Creates a new Objects instance which is the intersection of the two given.

• def contains
  Check if this objects contains the other one.

• def count
  Gets the number of elements into the collection.

• def clear
  Clears the collection removing all its elements.

• def __iter__
  Gets a new ObjectsIterator.

• def iterator_from_index
  Gets an ObjectsIterator skipping index elements.

• def remove
  Removes an element from the collection.

• def intersection
  Performs the intersection operation.

• def iterator
  Gets an ObjectsIterator.

• def union
  Performs the union operation.

• def add
  Adds an element into the collection.

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

Static Public Attributes

• int INVALID_OID = 0
  Invalid OID constant.
3.38.1 Detailed Description

Object identifier set class.
It stores a collection of Sparksee object identifiers as a set. As a set, there is no order and no duplicated elements.
This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.
This class is not thread-safe.
ObjectsIterator must be used to traverse all the elements into the set.
When the Objects instance is closed, it closes all existing and non-closed ObjectsIterator instances too.

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

3.38.2 Member Function Documentation

3.38.2.1 def sparksee.Objects.any (self)

Gets an element from the collection.

Returns:
Any element from the collection.

Exceptions:

RuntimeError whether the collection is empty.
RuntimeError null

3.38.2.2 def sparksee.Objects.difference (self, objs)

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

Parameters:
objs [in] Objects instance.

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

3.38.2.3 def sparksee.Objects.is_closed (self)

Gets if Objects instance has been closed or not.

See also:
close()
Returns:

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

3.38.2.4 def sparksee.Objects.combine_union (self, objs1, objs2)

Creates a new Objects instance which is the union of the two given.
Two given Objects belong to the same Session.

Parameters:

objs1 [in] Objects instance.
objs2 [in] Objects instance.

Returns:

New Objects instance.

3.38.2.5 def sparksee.Objects.iterator_from_element (self, e)

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

Parameters:

e [in] The first element to traverse in the resulting ObjectsIterator instance.

Returns:

ObjectsIterator instance.

3.38.2.6 def sparksee.Objects.exists (self, e)

Gets if the given element exists into the collection.

Parameters:

e [in] Element.

Returns:

TRUE if the element exists into the collection, FALSE otherwise.

3.38.2.7 def sparksee.Objects.equals (self, objs)

Checks if the given Objects contains the same information.

Parameters:

objs [in] Objects instance.

Returns:

True if the objects are equal or false otherwise.
3.38.8  def sparksee.Objects.combine_difference (self, objs1, objs2)

Creates a new Objects instance which is the difference of the two given.
Two given Objects belong to the same Session.

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

Returns:
   New Objects instance.

3.38.9  def sparksee.Objects.copy (self, objs)

Performs the copy operation.
This updates the Objects calling instance and copies the given Objects instance.

Parameters:
   objs [in] Objects instance.

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

3.38.10  def sparksee.Objects.sample (self, exclude, samples)

Creates a new Objects instance which is a sample of the calling one.

Parameters:
   exclude [in] If not NULL, elements into this collection will be excluded from the resulting one.
   samples [in] Number of elements into the resulting collection.

Returns:
   Sample collection.

3.38.11  def sparksee.Objects.close (self)

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

3.38.12  def sparksee.Objects.combine_intersection (self, objs1, objs2)

Creates a new Objects instance which is the intersection of the two given.
Two given Objects belong to the same Session.

Parameters:
   objs1 [in] Objects instance.
3.38 sparksee.Objects Class Reference

objs2 [in] Objects instance.

Returns:

New Objects instance.

3.38.2.13 def sparksee.Objects.contains ( self, objs )

Check if this objects contains the other one.

Parameters:

objs Objects collection.

Returns:

True if it contains the given object.

3.38.2.14 def sparksee.Objects.count ( self )

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

3.38.2.15 def sparksee.Objects.__iter__ ( self )

Gets a new ObjectsIterator.

Returns:

ObjectsIterator instance

3.38.2.16 def sparksee.Objects.iterator_from_index ( self, index )

Gets an ObjectsIterator skipping index elements.

Objects collection has no order, so this method is implementation-dependent.

Parameters:

index [in] The number of elements to skip from the beginning. It must be in the range [0..Size).

Returns:

ObjectsIterator instance.
3.38.2.17  def sparksee.Objects.remove (self, e)
Removes an element from the collection.

Parameters:
 e  [in] Element to be removed.

Returns:
 TRUE if the element is removed, FALSE if the element was not into the collection.

3.38.2.18  def sparksee.Objects.intersection (self, objs)
Performs the intersection operation.
Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

Parameters:
 objs  [in] Objects instance.

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

3.38.2.19  def sparksee.Objects.iterator (self)
Gets an ObjectsIterator.

Returns:
 ObjectsIterator instance.

3.38.2.20  def sparksee.Objects.union (self, objs)
Performs the union operation.
This adds all existing elements of the given Objects instance to the Objects calling instance.

Parameters:
 objs  [in] Objects instance.

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

3.38.2.21  def sparksee.Objects.add (self, e)
Adds an element into the collection.

Parameters:
 e  [in] Element to be added.

Returns:
 TRUE if the element is added, FALSE if the element was already into the collection.
3.39 sparksee.ObjectsIterator Class Reference

ObjectsIterator class.

Public Member Functions

• def has_next
  Gets if there are more elements to traverse.

• def next
  Gets the next element to traverse.

• def is_closed
  Gets if ObjectsIterator instance has been closed or not.

• def __next__
  Used in next().

• def close
  Closes the ObjectsIterator instance.

3.39.1 Detailed Description

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

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

3.39.2 Member Function Documentation

3.39.2.1 def sparksee.ObjectsIterator.has_next ( self)
Gets if there are more elements to traverse.

Returns:
TRUE if there are more elements to traverse, FALSE otherwise.
3.39.2.2  def sparksee.ObjectsIterator.next (self)
Gets the next element to traverse.

Returns:

The next element.

3.39.2.3  def sparksee.ObjectsIterator.is_closed (self)
Gets if ObjectsIterator instance has been closed or not.

See also:

close()

Returns:

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

3.39.2.4  def sparksee.ObjectsIterator.__next__ (self)
Used in next().

Returns:

The next element

3.39.2.5  def sparksee.ObjectsIterator.close (self)
Closes the ObjectsIterator instance.
It must be called to ensure the integrity of all data.

3.40  sparksee.ObjectType Class Reference

Object type enumeration.

Static Public Attributes

- int NODE = 0
  Node object type.

- int EDGE = 1
  Edge object type.

3.40.1  Detailed Description

Object type enumeration.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
3.41 sparksee.OIDList Class Reference

Sparksee object identifier list.

Public Member Functions

• def add
  Adds a Sparksee object identifier at the end of the list.

• def clear
  Clears the list.

• def __iter__
  Gets a new TypeListIterator.

• def set
  Sets a Sparksee object identifier at the specified position of the list.

• def iterator
  Gets a new OIDListIterator.

• def count
  Number of elements in the list.

• def __init__
  Constructor.

• def __init__
  Constructor.

3.41.1 Detailed Description

Sparksee object identifier list.
It stores a Sparksee object identifier list.
Use OIDListIterator to access all elements into this collection.

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

3.41.2 Member Function Documentation

3.41.2.1 def sparksee.OIDList.add (self, attr)
Adds a Sparksee object identifier at the end of the list.

Parameters:
  
  attr [in] Sparksee object identifier.
3.41.2.2  def sparksee.OIDList.__iter__ (self)

Gets a new TypeListIterator.

Returns:

TypeListIterator instance

3.41.2.3  def sparksee.OIDList.set (self, pos, oid)

Sets a Sparksee object identifier at the specified position of the list.

Parameters:

    pos [in] List position [0..Count()-1].
    oid [in] Sparksee object identifier.

3.41.2.4  def sparksee.OIDList.iterator (self)

Gets a new OIDListIterator.

Returns:

OIDListIterator instance.

3.41.2.5  def sparksee.OIDList.count (self)

Number of elements in the list.

Returns:

Number of elements in the list.

3.41.2.6  def sparksee.OIDList.__init__ (self, numInvalidOIDs)

Constructor.

This creates a list with N invalid oids.

Parameters:

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

3.41.2.7  def sparksee.OIDList.__init__ (self)

Constructor.

This creates an empty list.
Public Member Functions

- `def has_next`
  
  *Gets if there are more elements.*

- `def next`
  
  *Moves to the next element.*

- `def __next__`
  
  *Used in next().*

### 3.42.1 Detailed Description

**OIDList** iterator class.

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

### 3.42.2 Member Function Documentation

#### 3.42.2.1 `def sparksee.OIDListIterator.has_next ( self )`

**Gets if there are more elements.**

**Returns:**

TRUE if there are more elements, FALSE otherwise.

#### 3.42.2.2 `def sparksee.OIDListIterator.next ( self )`

**Moves to the next element.**

**Returns:**

The next element.

#### 3.42.2.3 `def sparksee.OIDListIterator.__next__ ( self )`

**Used in next().**

**Returns:**

The next element

### 3.43 sparksee.Order Class Reference

**Order** enumeration.
Static Public Attributes

- `int ASCENDENT = 0
  From lower to higher.
- `int DESCENDENT = 1
  From higher to lower.

3.43.1 Detailed Description

Order enumeration.

Author:
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.44 sparksee.Platform Class Reference

Platform class.

Public Member Functions

- `def get_statistics
  Gets platform data and statistics.

3.44.1 Detailed Description

Platform class.

Author:
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.44.2 Member Function Documentation

3.44.2.1 `def sparksee.Platform.get_statistics (self, stats)

Gets platform data and statistics.

Parameters:

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

3.45 sparksee.PlatformStatistics Class Reference

Platform data and statistics.
Public Member Functions

- def get_real_time
  
  Gets time in microseconds (since epoch).

- def get_available_mem
  
  Gets available (free) memory size in Bytes.

- def get_total_mem
  
  Gets physical memory size in Bytes.

- def get_system_time
  
  Gets CPU system time.

- def get_user_time
  
  Gets CPU user time.

- def get_num_c_p_us
  
  Gets the number of CPUs.

- def __init__
  
  Creates a new instance setting all values to 0.

3.45.1 Detailed Description

Platform data and statistics.

Author:

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

3.45.2 Member Function Documentation

3.45.2.1 def sparksee.PlatformStatistics.get_real_time (self)

Gets time in microseconds (since epoch).

Returns:

Time in microseconds (since epoch).

3.45.2.2 def sparksee.PlatformStatistics.get_available_mem (self)

Gets available (free) memory size in Bytes.

Returns:

Available (free) memory size in Bytes.
3.45.2.3  def sparksee.PlatformStatistics.get_total_mem (self)
Gets physical memory size in Bytes.

Returns:

Physical memory size in Bytes.

3.45.2.4  def sparksee.PlatformStatistics.get_system_time (self)
Gets CPU system time.

Returns:

CPU system time.

3.45.2.5  def sparksee.PlatformStatistics.get_user_time (self)
Gets CPU user time.

Returns:

CPU user time.

3.45.2.6  def sparksee.PlatformStatistics.get_num_c_p_us (self)
Gets the number of CPUs.

Returns:

The number of CPUs.

3.46  sparksee.Query Class Reference

Query class.

Public Member Functions

• def set_dynamic
  
  Sets the value for a dynamic parameter.

• def set_stream
  
  Sets a query stream handler.

• def execute
  
  Executes the given statement.

• def is_closed
  
  Gets if the Query instance has been closed or not.

• def close
  
  Closes the Query instance.
3.46.1 Detailed Description

Query class.

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

3.46.2 Member Function Documentation

3.46.2.1 def sparksee.Query.set_dynamic (self, name, value)
Sets the value for a dynamic parameter.

Parameters:

    name  [in] Parameter name
    value [in] Parameter value

3.46.2.2 def sparksee.Query.set_stream (self, stream, handler)
Sets a query stream handler.
Query streams handlers are created and destroyed by the caller.

Parameters:

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

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

3.46.2.3 def sparksee.Query.execute (self, stmt)
Executes the given statement.

Parameters:


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

3.46.2.4 def sparksee.Query.is_closed (self)
Gets if the Query instance has been closed or not.

See also:

    close()

Returns:
TRUE if the Query instance has been closed, FALSE otherwise.
3.46.2.5  def sparksee.Query.close ( self)
Closes the Query instance.
It must be called to ensure the integrity of all data.

3.47  sparksee.QueryContext Class Reference

Query context interface.

Public Member Functions

• def __init__
  Default constructor.

• def new_query
  Creates a new Query.

3.47.1  Detailed Description

Query context interface.

A QueryContext contains and manages the resources required to run a Query. A Session is one example of
a QueryContext connected to a Sparksee database. The applications can implement their own contexts to
run queries out of Sparksee.

Author:

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

3.48  sparksee.QueryStream Class Reference

Query stream interface.

Public Member Functions

• def start
  Starts the stream.

• def fetch
  Gets the next row and moves the iterator forward.

• def prepare
  Prepares the stream before it is started.
3.48.1 Detailed Description

Query stream interface.

A QueryStream is the interface between the application and the STREAM operator. When the operator starts inside a Query, the method is prepared with query-defined arguments. Then, if there are input operations, the STREAM operator builds the ResultSets and starts the iteration. Finally, the operator fetches rows until no more are available.

Application exceptions must be cached by the subclass that implements the interface.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.48.2 Member Function Documentation

3.48.2.1 def sparksee.QueryStream.start (self, list)

Starts the stream.

Parameters:

list [in] Optional list of input ResultSets

Returns:

FALSE on error

3.48.2.2 def sparksee.QueryStream.fetch (self, list)

Gets the next row and moves the iterator forward.

The end of sequence is denoted by returning TRUE with an empty row. A valid row must contain as many values (even NULL) as expected by the query.

Parameters:

list [out] Storage for the new rows

Returns:

TRUE if there is a row or end of sequence, FALSE on error

3.48.2.3 def sparksee.QueryStream.prepare (self, list)

Prepares the stream before it is started.

Parameters:

list [in] Optional list of arguments

Returns:

FALSE on error


3.49 sparksee.ResultSet Class Reference

ResultSet class.

Public Member Functions

- def **rewind**
  Positions the cursor before the first row.

- def **get_column**
  Gets the value for the given column.

- def **get_column**
  Gets the value for the given column.

- def **get_column_index**
  Gets the column index for the given column name.

- def **next**
  Fetches the next row.

- def **is_closed**
  Gets if the ResultSet instance has been closed or not.

- def **get_json**
  Returns rows in JSON format.

- def **get_column_name**
  Gets the name for the given column.

- def **get_column_data_type**
  Gets the datatype for the given column.

- def **get_num_columns**
  Gets the number of columns.

3.49.1 Detailed Description

ResultSet class.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.49.2 Member Function Documentation

3.49.2.1 def sparksee.ResultSet.get_column (self, index, value)

Gets the value for the given column.

QueryException If a database access error occurs.
Parameters:

\[\text{index} \ [\text{[in]} \text{ Column index.}]\]
\[\text{value} \ [\text{[in/out]} \text{ Value.}]\]

3.49.2.2 def sparksee.ResultSet.get_column (self, index)

Gets the value for the given column.
QueryExceptionIf a database access error occurs.

Parameters:

\[\text{index} \ [\text{[in]} \text{ Column index.}]\]

Returns:

The Value of the given column.

3.49.2.3 def sparksee.ResultSet.get_column_index (self, name)

Gets the column index for the given column name.

Parameters:

\[\text{name} \ [\text{[in]} \text{ Column name.}]\]

Returns:

Column index.

3.49.2.4 def sparksee.ResultSet.next (self)

Fetches the next row.
A ResultSet cursor is initially positioned before the first row; the first call to the method "Next" makes the first row the current row; the second call makes the second row the current row, and so on.
QueryExceptionIf a database access error occurs.

Returns:

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

3.49.2.5 def sparksee.ResultSet.is_closed (self)

Gets if the ResultSet instance has been closed or not.

See also:

close()

Returns:

TRUE if the ResultSet instance has been closed, FALSE otherwise.
3.49.2.6  def sparksee.ResultSet.get_j_s_o_n ( self, rows)
Returns rows in JSON format.
Rows are returned from the current position.

Parameters:
rows  [in] Maximum number of rows

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

3.49.2.7  def sparksee.ResultSet.get_column_name ( self, index)
Gets the name for the given column.

Parameters:
index  [in] Column index.

Returns:
Column name.

3.49.2.8  def sparksee.ResultSet.get_column_data_type ( self, index)
Gets the datatype for the given column.

Parameters:
index  [in] Column index.

Returns:
DataType for the given column.

3.49.2.9  def sparksee.ResultSet.get_num_columns ( self)
Gets the number of columns.
Columns are in the range [0...COLUMNS).

Returns:
The number of columns.

3.50  sparksee.ResultSetList Class Reference

ResultSet list.
Public Member Functions

- def __init__
  Constructor.

- def get
  Returns the ResultSet at the specified position in the list.

- def clear
  Clears the list.

- def __iter__
  Gets a new ResultSetListIterator.

- def iterator
  Gets a new ResultSetListIterator.

- def count
  Number of elements in the list.

3.50.1 Detailed Description

ResultSet list.
It stores a ResultSet list.

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

3.50.2 Member Function Documentation

3.50.2.1 def sparksee.ResultSetList.__init__ (self)
Constructor.
This creates an empty list.

3.50.2.2 def sparksee.ResultSetList.get (self, index)
Returns the ResultSet at the specified position in the list.

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

3.50.2.3 def sparksee.ResultSetList.__iter__ (self)
Gets a new ResultSetListIterator.

Returns:
  ResultSetListIterator instance
3.50.2.4  `def sparksee.ResultSetList.iterator (self)`

Gets a new `ResultSetListIterator`.

Returns:
- `ResultSetListIterator` instance.

3.50.2.5  `def sparksee.ResultSetList.count (self)`

Number of elements in the list.

Returns:
- Number of elements in the list.

3.51  `sparksee.ResultSetListIterator Class Reference`

`ResultSetList` iterator class.

Public Member Functions

- `def has_next`
  Gets if there are more elements.

- `def next`
  Moves to the next element.

- `def __next__`
  Used in `next()`.

3.51.1  Detailed Description

`ResultSetList` iterator class.

Iterator to traverse all the values into a `ResultSetList` instance.

Author:
  Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

3.51.2  Member Function Documentation

3.51.2.1  `def sparksee.ResultSetListIterator.has_next (self)`

Gets if there are more elements.

Returns:
- TRUE if there are more elements, FALSE otherwise.
3.51.2.2  def sparksee.ResultSetListIterator.next (self)
Moves to the next element.

Returns:
   The next element.

3.51.2.3  def sparksee.ResultSetListIterator.__next__ (self)
Used in next().

Returns:
   The next element

3.52  sparksee.RowReader Class Reference
RowReader interface.
Inheritance diagram for sparksee.RowReader:

Public Member Functions

  • def close
    Closes the reader.

  • def get_row
    The row number for the current row.

  • def reset
    Moves the reader to the beginning.

  • def read
    Reads the next row as a string array.

3.52.1  Detailed Description
RowReader interface.
Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once
close is called no more read operations can be executed.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.
3.52 sparksee.RowReader Class Reference

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

3.52.2 Member Function Documentation

3.52.2.1 def sparksee.RowReader.close (self)
Closes the reader.

Exceptions:
    IOError If the close fails.

Reimplemented in sparksee.CSVReader.

3.52.2.2 def sparksee.RowReader.get_row (self)
The row number for the current row.

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

Exceptions:
    IOError If it fails.

Reimplemented in sparksee.CSVReader.

3.52.2.3 def sparksee.RowReader.reset (self)
Moves the reader to the beginning.
Restarts the reader.

Returns:
    true if the reader can be restarted, false otherwise.

Exceptions:
    IOError If bad things happen during the restart.

Reimplemented in sparksee.CSVReader.

3.52.2.4 def sparksee.RowReader.read (self, row)
Reads the next row as a string array.

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

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

* **IOError** If bad things happen during the read.

Reimplemented in sparksee.CSVReader.

### 3.53 sparksee.RowWriter Class Reference

**RowWriter** interface.

Inheritance diagram for sparksee.RowWriter:

```
sparksee.RowWriter
  sparksee.CSVWriter
```

#### Public Member Functions

- **def write**
  
  *Writes the next row.*

- **def close**
  
  *Closes the writer.*

#### 3.53.1 Detailed Description

**RowWriter** interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more write operations can be executed.

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

#### 3.53.2 Member Function Documentation

#### 3.53.2.1 def sparksee.RowWriter.write (self, row)

Writes the next row.

**Parameters:**

- **row** [in] Row of data.

**Exceptions:**

* **IOError** If bad things happen during the write.*
RuntimeError null

Reimplemented in sparksee.CSVWriter.

3.53.2.2  def sparksee.RowWriter.close ( self)
Closes the writer.

Exceptions:

   IOError  If the close fails.
   RuntimeError null

Reimplemented in sparksee.CSVWriter.

3.54  sparksee.ScriptParser Class Reference

ScriptParser.

Public Member Functions

•  def parse
   Parses the given input file.

•  def set_error_log
   Sets the error log.

•  def __init__
   Constructor.

•  def generate_schema_script
   Writes an script with the schema definition for the given database.

•  def set_output_log
   Sets the output log.

3.54.1  Detailed Description

ScriptParser.

The ScriptParser can create schemas and load data from a set of commands in a sparksee script.

A SPARKSEE script contains an ordered list of commands. ScriptParser will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined SPARKSEE schema.

Check out the ‘Scripting’ chapter in the SPARKSEE User Manual for a comprehensive explanation on the grammar of the SPARKSEE commands and how they work.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
3.54.2 Member Function Documentation

3.54.2.1 def sparksee.ScriptParser.parse (self, path, execute, localeStr)

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.

Exceptions:

IOError If bad things happen opening the file.

3.54.2.2 def sparksee.ScriptParser.set_error_log (self, path)

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters:

- path [in] Path of the error log.

Exceptions:

IOError If bad things happen opening the file.

3.54.2.3 def sparksee.ScriptParser.generate_schema_script (self, path, db)

Writes an script with the schema definition for the given database.

Parameters:

- path [in] Filename of the script to be writen.
- db [in] Database.

Exceptions:

IOError If bad things happen opening or writing the file.

3.54.2.4 def sparksee.ScriptParser.set_output_log (self, path)

Sets the output log.

If not set, output log corresponds to standard output.

Parameters:

- path [in] Path of the output log.
3.55 sparksee.Session Class Reference

Session class.

Public Member Functions

• def rollback
  Rollbacks a transaction.

• def begin_update
  Begins an update transaction.

• def is_closed
  Gets if Session instance has been closed or not.

• def new_query
  Creates a new Query.

• def commit
  Commits a transaction.

• def get_graph
  Gets the Graph instance.

• def close
  Closes the Session instance.

• def begin
  Begins a transaction.

• def new_objects
  Creates a new Objects instance.

3.55.1 Detailed Description

Session class.

A Session is a stateful period of activity of a user with the Database.

All the manipulation of a Database must be enclosed into a Session. A Session can be initiated from a Database instance and allows for getting a Graph instance which represents the persistent graph (the graph database).

Also, temporary data is associated to the Session, thus when a Session is closed, all the temporary data associated to the Session is removed too. Objects or Values instances or even session attributes are an example of temporary data.
Moreover, a `Session` is exclusive for a thread, thus if it is shared among threads results may be fatal or unexpected.

Check out the `Processing` and `Transactions` sections in the SPARKSEE User Manual for details about how Sessions work and the use of transactions.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

---

### 3.55.2 Member Function Documentation

#### 3.55.2.1 `def sparksee.Session.is_closed (self)`

Gets if `Session` instance has been closed or not.

**See also:**

- `close()`

**Returns:**

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

#### 3.55.2.2 `def sparksee.Session.get_graph (self)`

Gets the `Graph` instance.

**Returns:**

The `Graph` instance.

#### 3.55.2.3 `def sparksee.Session.close (self)`

Closes the `Session` instance.

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

#### 3.55.2.4 `def sparksee.Session.new_objects (self)`

Creates a new `Objects` instance.

**Returns:**

The new `Objects` instance.

---

### 3.56 sparksee.ShortestPath Class Reference

`ShortestPath` class.
Inheritance diagram for sparksee.ShortestPath:

- **sparksee.ShortestPath**
- **sparksee.SinglePairShortestPath**
  - **sparksee.SinglePairShortestPathBFS**
  - **sparksee.SinglePairShortestPathDijkstra**

**Public Member Functions**

- **def add_node_type**
  
  Allows for traversing nodes of the given type.

- **def add_all_node_types**
  
  Allows for traversing all node types of the graph.

- **def run**
  
  Runs the algorithm.

- **def exclude_nodes**
  
  Set which nodes can’t be used.

- **def add_edge_type**
  
  Allows for traversing edges of the given type.

- **def is_closed**
  
  Gets if ShortestPath has been closed or not.

- **def exclude_edges**
  
  Set which edges can’t be used.

- **def set_maximum_hops**
  
  Sets the maximum hops restriction.

- **def add_all_edge_types**
  
  Allows for traversing all edge types of the graph.

- **def close**
  
  Closes the ShortestPath instance.
3.56 sparksee.ShortestPath Class Reference

3.56.1 Detailed Description

ShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph.

The user must set which node and edge types can be used for the traversal.

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

Author:

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

3.56.2 Member Function Documentation

3.56.2.1 def sparksee.ShortestPath.add_node_type (self, type)

Allows for traversing nodes of the given type.

Parameters:

   type null


3.56.2.2 def sparksee.ShortestPath.run (self)

Runs the algorithm.

This method can only be called once.


3.56.2.3 def sparksee.ShortestPath.exclude_nodes (self, nodes)

Set which nodes can’t be used.

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

Parameters:

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


3.56.2.4 def sparksee.ShortestPath.add_edge_type (self, type, dir)

Allows for traversing edges of the given type.

Parameters:

   type [in] Edge type.
   dir [in] Edge direction.
3.56.2.5 **def** sparksee.ShortestPath.is_closed (**self**)  
 Gets if ShortestPath has been closed or not.

**See also:**  
 close()

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

3.56.2.6 **def** sparksee.ShortestPath.exclude_edges (**self**, **edges**)  
 Set which edges can’t be used.

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

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


3.56.2.7 **def** sparksee.ShortestPath.set_maximum_hops (**self**, **maxhops**)  
 Sets the maximum hops restriction.  
 All paths longer than the maximum hops restriction will be ignored.

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


3.56.2.8 **def** sparksee.ShortestPath.add_all_edge_types (**self**, **dir**)  
 Allows for traversing all edge types of the graph.

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

3.56.2.9  def sparksee.ShortestPath.close ( self)
Closes the ShortestPath instance.
It must be called to ensure the integrity of all data.

3.57  sparksee.SinglePairShortestPath Class Reference

SinglePairShortestPath class.
Inheritance diagram for sparksee.SinglePairShortestPath:

Collaboration diagram for sparksee.SinglePairShortestPath:

Public Member Functions

- def exists
  Returns TRUE If a path exists or FALSE otherwise.

- def add_node_type
  Allows for traversing nodes of the given type.

- def exclude_nodes
  Set which nodes can’t be used.

- def run
  Runs the algorithm.

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

- def set_maximum_hops
3.57 sparksee.SinglePairShortestPath Class Reference

Sets the maximum hops restriction.

• def add_all_edge_types
  Allows for traversing all edge types of the graph.

• def add_all_node_types
  Allows for traversing all node types of the graph.

• def add_edge_type
  Allows for traversing edges of the given type.

• def get_cost
  Gets the cost of the shortest path.

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

• def exclude_edges
  Set which edges can’t be used.

• def is_closed
  Gets if ShortestPath has been closed or not.

• def close
  Closes the ShortestPath instance.

3.57.1 Detailed Description

SinglePairShortestPath class.

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

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

Author:

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

3.57.2 Member Function Documentation

3.57.2.1 def sparksee.SinglePairShortestPath.add_node_type ( self, type)

Allows for traversing nodes of the given type.

Parameters:

  type  null

Reimplemented from sparksee.ShortestPath.

3.57.2.2  def sparksee.SinglePairShortestPath.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage
of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented from sparksee.ShortestPath.

3.57.2.3  def sparksee.SinglePairShortestPath.run (self)
Runs the algorithm.
This method can only be called once.
Reimplemented from sparksee.ShortestPath.

3.57.2.4  def sparksee.SinglePairShortestPath.get_path_as_edges (self)
Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:
   Ordered set of edge identifiers.


3.57.2.5  def sparksee.SinglePairShortestPath.set_maximum_hops (self, maxhops)
Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

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

Reimplemented from sparksee.ShortestPath.

3.57.2.6  def sparksee.SinglePairShortestPath.add_all_edge_types (self, dir)
Allows for traversing all edge types of the graph.

Parameters:
   dir [in] Edge direction.

Reimplemented from sparksee.ShortestPath.
### sparksee.SinglePairShortestPath Class Reference

#### 3.57.2.7 def sparksee.SinglePairShortestPath.add_edge_type (self, type, dir)

Allows for traversing edges of the given type.

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

Reimplemented from [sparksee.ShortestPath](#).
Reimplemented in [sparksee.SinglePairShortestPathBFS](#), and [sparksee.SinglePairShortestPathDijkstra](#).

#### 3.57.2.8 def sparksee.SinglePairShortestPath.get_cost (self)

Gets the cost of the shortest path.

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

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

Reimplemented in [sparksee.SinglePairShortestPathBFS](#), and [sparksee.SinglePairShortestPathDijkstra](#).

#### 3.57.2.9 def sparksee.SinglePairShortestPath.get_path_as_nodes (self)

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

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

Reimplemented in [sparksee.SinglePairShortestPathBFS](#), and [sparksee.SinglePairShortestPathDijkstra](#).

#### 3.57.2.10 def sparksee.SinglePairShortestPath.exclude_edges (self, edges)

Set which edges can’t be used.

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

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

Reimplemented from [sparksee.ShortestPath](#).
Reimplemented in [sparksee.SinglePairShortestPathBFS](#), and [sparksee.SinglePairShortestPathDijkstra](#).

#### 3.57.2.11 def sparksee.ShortestPath.is_closed (self) [inherited]

Gets if [ShortestPath](#) has been closed or not.

**See also:**
- close()
3.58 sparksee.SinglePairShortestPathBFS Class Reference

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

3.57.2.12 def sparksee.ShortestPath.close (self) [inherited]
Closes the ShortestPath instance.
It must be called to ensure the integrity of all data.

3.58 sparksee.SinglePairShortestPathBFS Class Reference

SinglePairShortestPathBFS class.
Inheritance diagram for sparksee.SinglePairShortestPathBFS:

Collaboration diagram for sparksee.SinglePairShortestPathBFS:

Public Member Functions

• def exists
Returns TRUE If a path exists or FALSE otherwise.

• def add_node_type
Allows for traversing nodes of the given type.

• def exclude_nodes
Set which nodes can’t be used.

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

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

- def __init__
  Creates a new instance.

- def set_maximum_hops
  Sets the maximum hops restriction.

- def add_all_edge_types
  Allows for traversing all edge types of the graph.

- def get_cost
  Gets the cost of the shortest path.

- def add_all_node_types
  Allows for traversing all node types of the graph.

- def add_edge_type
  Allows for traversing edges of the given type.

- def run
  Executes the algorithm.

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

- def exclude_edges
  Set which edges can’t be used.

- def is_closed
  Gets if ShortestPath has been closed or not.

- def close
  Closes the ShortestPath instance.

3.58.1 Detailed Description

SinglePairShortestPathBFS class.
It solves the single-pair shortest path problem using a BFS-based implementation.
It is a unweighted algorithm, that is it assumes all edges have the same cost.
Check out the ‘Algorithms’ section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
3.58.2 Member Function Documentation

3.58.2.1 def sparksee.SinglePairShortestPathBFS.add_node_type (self, type)
Allows for traversing nodes of the given type.

Parameters:

  type null

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.2 def sparksee.SinglePairShortestPathBFS.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.3 def sparksee.SinglePairShortestPathBFS.get_path_as_edges (self)
Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

  Ordered set of edge identifiers.

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.4 def sparksee.SinglePairShortestPathBFS.get_path_as_nodes (self)
Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

  Ordered set of node identifiers.

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.5 def sparksee.SinglePairShortestPathBFS.__init__ (self, session, src, dst)
Creates a new instance.

Parameters:

  session [in] Session to get the graph from and perform traversal.
  src [in] Source node.
  dst [dst] Destination node.
3.58.2.6  
def sparksee.SinglePairShortestPathBFS.set_maximum_hops ( self, maxhops)
Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.7  
def sparksee.SinglePairShortestPathBFS.add_all_edge_types ( self, dir)
Allows for traversing all edge types of the graph.

Parameters:

dir [in] Edge direction.

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.8  
def sparksee.SinglePairShortestPathBFS.get_cost ( self)
Gets the cost of the shortest path.
The cost is the number of hops of the shortest path.

Returns:

The cost of the shortest path.

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.9  
def sparksee.SinglePairShortestPathBFS.add_edge_type ( self, type, dir)
Allows for traversing edges of the given type.

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.58.2.10  
def sparksee.SinglePairShortestPathBFS.check_only_existence ( self)
Set that only the path existence must be calculated and not the path itself.
That method should improve the performance of the algorithm, but a call to GetPathAsNodes or GetPathAsEdges will generate an exception even if the path exists.
def sparksee.SinglePairShortestPathBFS.exclude_edges ( self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

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

Reimplemented from sparksee.SinglePairShortestPath.

def sparksee.ShortestPath.is_closed ( self)  [inherited]
Gets if ShortestPath has been closed or not.

See also:
close()

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

def sparksee.ShortestPath.close ( self)  [inherited]
Closes the ShortestPath instance.
It must be called to ensure the integrity of all data.

sparksee.SinglePairShortestPathDijkstra Class Reference

SinglePairShortestPathDijkstra class.
Inheritance diagram for sparksee.SinglePairShortestPathDijkstra:

Collaboration diagram for sparksee.SinglePairShortestPathDijkstra:
Public Member Functions

- `def exists`: Returns `TRUE` if a path exists or `FALSE` otherwise.

- `def add_node_type`: Allows for traversing nodes of the given type.

- `def exclude_nodes`: Set which nodes can't be used.

- `def get_path_as_edges`: Gets the shortest path between the source node and the destination node as an ordered set of edges.

- `def get_path_as_nodes`: Gets the shortest path between the source node and the destination node as an ordered set of nodes.

- `def set_unweighted_edge_cost`: Sets the weight assigned to the unweighted edges.

- `def set_maximum_hops`: Sets the maximum hops restriction.

- `def add_all_edge_types`: Allows for traversing all edge types of the graph.

- `def get_cost`: Gets the cost of the shortest path.

- `def __init__`: Creates a new instance.

- `def add_weighted_edge_type`: Allows for traversing edges of the given type using the given attribute as the weight.

- `def add_all_node_types`: Allows for traversing all node types of the graph.

- `def add_edge_type`: Allows for traversing edges of the given type.

- `def run`: Executes the algorithm.

- `def exclude_edges`: Set which edges can't be used.

- `def is_closed`: Gets if `ShortestPath` has been closed or not.
• def close
  
  Closes the ShortestPath instance.

3.59.1 Detailed Description

SinglePairShortestPathDijkstra class.

It solves the single-pair shortest path problem using a Dijkstra-based implementation.

It is a weighted algorithm, so it takes into account the cost of the edges to compute a minimum-cost shortest path. That is, the user may set for each edge type which attribute should be used to retrieve the cost of the edge. If no attribute is given for an edge type, this will assume the edge has a fixed cost (the default is 1). Only numerical attribute can be set as weight attributes (that is Long, Integer or Double attributes are allowed).

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

Author:

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

3.59.2 Member Function Documentation

3.59.2.1 def sparksee.SinglePairShortestPathDijkstra.add_node_type (self, type)

Allows for traversing nodes of the given type.

Parameters:

type null

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.2 def sparksee.SinglePairShortestPathDijkstra.exclude_nodes (self, nodes)

Set which nodes can’t be used.

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

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.3 def sparksee.SinglePairShortestPathDijkstra.get_path_as_edges (self)

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

Returns:

Ordered set of edge identifiers.

Reimplemented from sparksee.SinglePairShortestPath.
3.59.2.4  def sparksee.SinglePairShortestPathDijkstra.get_path_as_nodes ( self)

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

**Returns:**

Ordered set of node identifiers.

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.5  def sparksee.SinglePairShortestPathDijkstra.set_unweighted_edge_cost ( self, weight)

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

**Parameters:**

*weight* [in] The weight value for unweighted edges.

3.59.2.6  def sparksee.SinglePairShortestPathDijkstra.set_maximum_hops ( self, maxhops)

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

**Parameters:**

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

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.7  def sparksee.SinglePairShortestPathDijkstra.add_all_edge_types ( self, dir)

Allows for traversing all edge types of the graph.

**Parameters:**

*dir* [in] Edge direction.

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.8  def sparksee.SinglePairShortestPathDijkstra.get_cost ( self)

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

**Returns:**

The cost of the shortest path.

Reimplemented from sparksee.SinglePairShortestPath.
3.59.2.9  def sparksee.SinglePairShortestPathDijkstra.__init__ (self, session, src, dst)
Creates a new instance.

Parameters:

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

3.59.2.10 def sparksee.SinglePairShortestPathDijkstra.add_weighted_edge_type (self, type, dir, attr)
Allows for traversing edges of the given type using the given attribute as the weight.

Parameters:

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

3.59.2.11 def sparksee.SinglePairShortestPathDijkstra.add_edge_type (self, type, dir)
Allows for traversing edges of the given type.

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.12 def sparksee.SinglePairShortestPathDijkstra.exclude_edges (self, edges)
Set which edges can’t be used.

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

Parameters:

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

Reimplemented from sparksee.SinglePairShortestPath.

3.59.2.13 def sparksee.ShortestPath.is_closed (self) [inherited]
Gets if ShortestPath has been closed or not.

See also:

  close()

Returns:

  TRUE if the ShortestPath instance has been closed, FALSE otherwise.
3.59.2.14  def sparksee.ShortestPath.close (self)  [inherited]
Closes the ShortestPath instance.
It must be called to ensure the integrity of all data.

3.60  sparksee.Sparksee Class Reference

Sparksee class.

Public Member Functions

•  def create
   Creates a new Database instance.

•  def restore
   Restores a Database from a backup file.

•  def open
   Opens an existing Database instance.

•  def __init__
   Creates a new instance.

•  def close
   Closes the Sparksee instance.

Static Public Attributes

•  float VERSION = 5.0
   Sparksee version.

3.60.1  Detailed Description

Sparksee class.
All Sparksee programs must have one single Sparksee instance to manage one or more Database instances.
This class allows for the creation of new Databases or open an existing one.

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

3.60.2  Member Function Documentation

3.60.2.1  def sparksee.Sparksee.create (self, path, alias)
Creates a new Database instance.
3.60  sparksee.Sparksee Class Reference

Parameters:

alias  [in] Database alias name.

Returns:

A Database instance.

Exceptions:

  IOError  If the given file cannot be created.
  RuntimeError  null

3.60.2.2  def sparksee.Sparksee.restore (self, path, backupFile)

Restores a Database from a backup file.
See the Graph class Backup method.

Parameters:

backupFile  [in] The Backup file to be restored.

Returns:

A Database instance.

Exceptions:

  IOError  If the given file cannot be created, or the exported data file does not exists.
  RuntimeError  null

3.60.2.3  def sparksee.Sparksee.open (self, path, readOnly)

Opens an existing Database instance.

Parameters:

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

Returns:

A Database instance.

Exceptions:

  IOError  If the given file does not exist.
  RuntimeError  null
3.60.2.4 def sparksee.Sparksee.__init__ (self, config)
Creates a new instance.

Parameters:

config [in] Sparksee configuration.

3.60.2.5 def sparksee.Sparksee.close (self)
Closes the Sparksee instance.
It must be called to ensure the integrity of all data.

3.61 sparksee.SparkseeConfig Class Reference
Sparksee configuration class.

Public Member Functions

• def set_license
  Sets the license code.

• def get_high_availability_synchronization
  Gets the synchronization polling time.

• def set_extent_pages
  Sets the number of pages per extent.

• def set_high_availability_COORDINATORS
  Sets the coordinators address and port list.

• def get_extent_size
  Gets the size of a extent.

• def set_log_file
  Sets the log file.

• def set_log_level
  Sets the log level.

• def set_cache_statistics_enabled
  Enables or disables cache statistics.

• def get_log_file
  Gets the log file.

• def pool_cluster_size
  Sets the number of pools in each PoolCluster.

• def cache_statistics_snapshot_time
Sets the cache statistics snapshot time.

- `def get_recovery_checkpoint_time`
  Gets the delay time (in microseconds) between automatic checkpoints.

- `def get_cache_statistics_enabled`
  Gets whether cache statistics are enabled or disabled.

- `def get_pool_persistent_max_size`
  Gets the maximum size for the persistent pool in number of frames.

- `def set_pool_persistent_max_size`
  Sets the maximum size for the persistent pool in number of frames.

- `def get_recovery_log_file`
  Gets the recovery log file.

- `def set_high_availability_master_history`
  Sets the master’s history log.

- `def __init__`
  Creates a new instance.

- `def set_cache_statistics_file`
  Sets the cache statistics log file.

- `def get_high_availability_coordinators`
  Gets the coordinators address and port list.

- `def set_pool_frame_size`
  Sets the size of a pool frame in number of extents.

- `def get_cache_statistics_file`
  Gets the cache statistics log file.

- `def get_cache_statistics_snapshot_time`
  Gets the cache statistics snapshot time in microseconds.

- `def get_pool_temporary_max_size`
  Gets the maximum size for the temporary pool in number of frames.

- `def set_recovery_enabled`
  Enables or disables the recovery.

- `def get_license`
  Gets the license code.

- `def set_pool_temporary_min_size`
  Sets the minimum size for the temporary pool in number of frames.
• def get_high_availability_i_p
  Gets the IP address and port of the instance.

• def get_log_level
  Gets the log level.

• def set_high_availability_synchronization
  Sets the synchronization polling time.

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

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

• def set_high_availability_enabled
  Enables or disables high availability mode.

• def get_pool_cluster_size
  Gets the number of pools in each PoolCluster.

• def set_recovery_cache_max_size
  Sets the maximum size for the recovery log cache in extents.

• def get_high_availability_i_p
  Sets the IP address and port of the instance.

• def set_extent_size
  Sets the size of the extents in KB.

• def set_rollback_enabled
  Enables or disables the rollback.

• def get_extent_pages
  Gets the number of pages per extent.

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

• def get_high_availability_enabled
  Gets whether high availability mode is enabled or disabled.

• def get_recovery_enabled
  Gets whether the recovery is enabled or disabled.

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

• def get_cache_max_size
  Gets the maximum size for the cache (all pools) in MB.
3.61 sparksee.SparkseeConfig Class Reference

- **def get_pool_frame_size**
  
  *Gets the size of a pool frame in number of extents.*

- **def get_rollback_enabled**
  
  *Gets whether the rollback is enabled or disabled.*

- **def get_pool_temporary_min_size**
  
  *Gets the minimum size for the temporary pool in number of frames.*

- **def set_recovery_checkpoint_time**
  
  *Sets the delay time (in microseconds) between automatic checkpoints.*

- **def get_recovery_cache_max_size**
  
  *Gets the maximum size for the recovery log cache in extents.*

- **def get_high_availability_master_history**
  
  *Gets the master’s history log.*

- **def set_recovery_log_file**
  
  *Sets the recovery log file.*

### 3.61.1 Detailed Description

**Sparksee** configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that’s the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overridden with values from a properties file (see SparkseeProperties class). Also, this settings can be overridden calling a specific setter.

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

- Extent size: 4KB (`sparksee.storage.extentsize` at SparkseeProperties).
- Pages per extent: 1 page (`sparksee.storage.extentpages` at SparkseeProperties).
- Pool frame size: 1 extent (`sparksee.io.pool.frame.size` at SparkseeProperties).
- Minimum size for the persistent pool: 64 frames (`sparksee.io.pool.persistent.minsize` at SparkseeProperties).
- Maximum size for the persistent pool: 0 frames (`sparksee.io.pool.persistent.maxsize` at SparkseeProperties).
- Maximum size for the temporary pool: 0 frames (`sparksee.io.pool.temporal.maxsize` at SparkseeProperties).
- Number of pools in the pool cluster: 0 pools (`sparksee.io.pool.clustersize` at SparkseeProperties). 0 or 1 means the clustering is disabled.

- Maximum size for the cache (all pools): 0 MB (`sparksee.io.cache.maxsize` at SparkseeProperties).
License code: "" (‘sparksee.license’ at SparkseeProperties). No license code means evaluation license.
Log level: Info (‘sparksee.log.level’ at SparkseeProperties).
Log file: "sparksee.log" (‘sparksee.log.file’ at SparkseeProperties).
Recovery enabled: false (‘sparksee.io.recovery’ at SparkseeProperties).
Recovery cache max size: 1MB (‘sparksee.io.recovery.cachesize’ at SparkseeProperties).
Recovery checkpoint time: 60 seconds [TimeUnit] (‘sparksee.io.recovery.checkpointTime’ at SparkseeProperties).
High-availability: false (disabled) (‘sparksee.ha’ at SparkseeProperties).
High-availability coordinators: "" (‘sparksee.ha.coordinators’ at SparkseeProperties).
High-availability IP: "" (‘sparksee.ha.ip’ at SparkseeProperties).
High-availability sync polling: 0 (disabled) [TimeUnit] (‘sparksee.ha.sync’ at SparkseeProperties).
High-availability master history: 1D (1 day) [TimeUnit] (‘sparksee.ha.master.history’ at SparkseeProperties).

Use of TimeUnit:
Those variables using TimeUnit allow for:

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

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

Author:

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

3.61.2 Member Function Documentation

3.61.2.1 def sparksee.SparkseeConfig.set_license (self, key)
Sets the license code.

Parameters:

key [in] The license code.

3.61.2.2 def sparksee.SparkseeConfig.get_high_availability_synchronization (self)
Gets the synchronization polling time.

Returns:

The Synchronization polling time.
3.61.2.3  def sparksee.SparkseeConfig.set_extent_pages (self, pages)
Sets the number of pages per extent.

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

3.61.2.4  def sparksee.SparkseeConfig.set_high_availability_coordinators (self, ip)
Sets the coordinators address and port list.

Parameters:
    ip     [in] The coordinators address and port list.

3.61.2.5  def sparksee.SparkseeConfig.get_extent_size (self)
Gets the size of a extent.

Returns:
    The size of a extent in KB.

3.61.2.6  def sparksee.SparkseeConfig.set_log_file (self, filePath)
Sets the log file.

Parameters:

3.61.2.7  def sparksee.SparkseeConfig.set_log_level (self, level)
Sets the log level.

Parameters:

3.61.2.8  def sparksee.SparkseeConfig.set_cache_statistics_enabled (self, status)
Enables or disables cache statistics.

Parameters:
    status  [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.
3.61.2.9  def sparksee.SparkseeConfig.get_log_file (self)

Gets the log file.

Returns:

The log file.

3.61.2.10  def sparksee.SparkseeConfig.set_pool_cluster_size (self, pools)

Sets the number of pools in each PoolCluster.

Parameters:

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

3.61.2.11  def sparksee.SparkseeConfig.set_cache_statistics_snapshot_time (self, microSeconds)

Sets the cache statistics snapshot time.
Useless if cache statistics are disabled.

Parameters:


3.61.2.12  def sparksee.SparkseeConfig.get_recovery_checkpoint_time (self)

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

Returns:

The delay time (in microseconds) between automatic checkpoints.

3.61.2.13  def sparksee.SparkseeConfig.get_cache_statistics_enabled (self)

Gets whether cache statistics are enabled or disabled.

Returns:

TRUE if cache statistics are enabled, FALSE otherwise.

3.61.2.14  def sparksee.SparkseeConfig.get_pool_persistent_max_size (self)

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

Returns:

The maximum size for the persistent pool in number of frames.
3.61.2.15  def sparksee.SparkseeConfig.set_pool_persistent_max_size (self, frames)
Sets the maximum size for the persistent pool in number of frames.

Parameters:

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

3.61.2.16  def sparksee.SparkseeConfig.get_recovery_log_file (self)
Gets the recovery log file.

Returns:

The recovery log file.

3.61.2.17  def sparksee.SparkseeConfig.set_high_availability_master_history (self, filePath)
Sets the master's history log.

Parameters:

filePath [in] The master’s history log.

3.61.2.18  def sparksee.SparkseeConfig.__init__ (self)
Creates a new instance.

Values are set with default values.

3.61.2.19  def sparksee.SparkseeConfig.set_cache_statistics_file (self, filePath)
Sets the cache statistics log file.
Useless if cache statistics are disabled.

Parameters:


3.61.2.20  def sparksee.SparkseeConfig.get_high_availability_coordinators (self)
Gets the coordinators address and port list.

Returns:

The coordinators address and port list.

3.61.2.21  def sparksee.SparkseeConfig.set_pool_frame_size (self, extents)
Sets the size of a pool frame in number of extents.

Parameters:

extents [in] The size of a pool frame in number of extents. It must be non-negative.
def sparksee.SparkseeConfig.get_cache_statistics_file (self)

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

Returns:
    The cache statistics log file.

def sparksee.SparkseeConfig.get_cache_statistics_snapshot_time (self)

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

Returns:
    The cache statistics snapshot time in microseconds.

def sparksee.SparkseeConfig.get_pool_temporary_max_size (self)

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

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

def sparksee.SparkseeConfig.set_recovery_enabled (self, status)

Enables or disables the recovery.

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

def sparksee.SparkseeConfig.get_license (self)

Gets the license code.

Returns:
    The license code.

def sparksee.SparkseeConfig.set_pool_temporary_min_size (self, frames)

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

Parameters:
    frames [in] The minimum size for the temporary pool in number of frames. It must be non-negative.
3.61.2.28 def sparksee.SparkseeConfig.get_high_availability_i_p (self)

Gets the IP address and port of the instance.

**Returns:**

The IP address and port of the instance.

3.61.2.29 def sparksee.SparkseeConfig.get_log_level (self)

Gets the log level.

**Returns:**

The LogLevel.

3.61.2.30 def sparksee.SparkseeConfig.set_high_availability_synchronization (self, microSeconds)

Sets the synchronization polling time.

**Parameters:**

* microSeconds [in] The synchronization polling time.

3.61.2.31 def sparksee.SparkseeConfig.set_cache_max_size (self, megaBytes)

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

**Parameters:**

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

3.61.2.32 def sparksee.SparkseeConfig.get_pool_persistent_min_size (self)

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

**Returns:**

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

3.61.2.33 def sparksee.SparkseeConfig.set_high_availability_enabled (self, status)

Enables or disables high availability mode.

**Parameters:**

* status [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.
3.61.2.34  def sparksee.SparkseeConfig.get_pool_cluster_size (self)

Gets the number of pools in each PoolCluster.

Returns:

The number of pools in each PoolCluster.

3.61.2.35  def sparksee.SparkseeConfig.set_recovery_cache_max_size (self, extents)

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

Parameters:

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

3.61.2.36  def sparksee.SparkseeConfig.set_high_availability_i_p (self, ip)

Sets the IP address and port of the instance.

Parameters:

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

3.61.2.37  def sparksee.SparkseeConfig.set_extent_size (self, kBytes)

Sets the size of the extents in KB.

Parameters:

- kBytes [in] The size of an extent in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

3.61.2.38  def sparksee.SparkseeConfig.set_rollback_enabled (self, status)

Enables or disables the rollback.

Parameters:

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

3.61.2.39  def sparksee.SparkseeConfig.get_extent_pages (self)

Gets the number of pages per extent.

Returns:

The number of pages per extent.
3.61.2.40 def sparksee.SparkseeConfig.set_pool_temporary_max_size (self, frames)
Sets the maximum size for the temporary pool in number of frames.

Parameters:

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

3.61.2.41 def sparksee.SparkseeConfig.get_high_availability_enabled (self)
Gets whether high availability mode is enabled or disabled.

Returns:

TRUE if high availability mode is enabled, FALSE otherwise.

3.61.2.42 def sparksee.SparkseeConfig.get_recovery_enabled (self)
Gets whether the recovery is enabled or disabled.

Returns:

TRUE if the recovery is enabled, FALSE otherwise.

3.61.2.43 def sparksee.SparkseeConfig.set_pool_persistent_min_size (self, frames)
Sets the minimum size for the persistent pool in number of frames.

Parameters:

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

3.61.2.44 def sparksee.SparkseeConfig.get_cache_max_size (self)
Gets the maximum size for the cache (all pools) in MB.

Returns:

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

3.61.2.45 def sparksee.SparkseeConfig.get_pool_frame_size (self)
Gets the size of a pool frame in number of extents.

Returns:

The size of a pool frame in number of extents.
3.61.2.46  def sparksee.SparkseeConfig.get_rollback_enabled (self)
 Gets whether the rollback is enabled or disabled.

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

3.61.2.47  def sparksee.SparkseeConfig.get_pool_temporary_min_size (self)
 Gets the minimum size for the temporary pool in number of frames.

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

3.61.2.48  def sparksee.SparkseeConfig.set_recovery_checkpoint_time (self, microSeconds)
 Sets the delay time (in microseconds) between automatic checkpoints.

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

3.61.2.49  def sparksee.SparkseeConfig.get_recovery_cache_max_size (self)
 Gets the maximum size for the recovery log cache in extents.

Returns:
 The maximum size for the recovery log cache in extents.

3.61.2.50  def sparksee.SparkseeConfig.get_high_availability_master_history (self)
 Gets the master’s history log.

Returns:
 The master’s history log.

3.61.2.51  def sparksee.SparkseeConfig.set_recovery_log_file (self, filePath)
 Sets the recovery log file.

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

3.62  sparksee.SparkseeProperties Class Reference

Sparksee properties file.
Public Member Functions

- **def get**  
  Gets a property.

- **def get_time_unit**  
  Gets a property as a time unit.

- **def get_boolean**  
  Gets a property as a boolean.

- **def load**  
  Loads properties from the given file path.

- **def get_integer**  
  Gets a property as an integer.

### 3.62.1 Detailed Description

**Sparksee properties file.**

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: `key=value`

By default, this loads properties from the file `./sparksee.cfg`. The user may choose to load a different file by calling the method `load()`.

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

### 3.62.2 Member Function Documentation

#### 3.62.2.1 def sparksee.SparkseeProperties.get (self, key, def)

Gets a property.

**Parameters:**

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

**Returns:**

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

#### 3.62.2.2 def sparksee.SparkseeProperties.get_time_unit (self, key, def)

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
Valid format for the string representation: Blanks at the beginning or at the end are ignored. No blanks are allowed between the time duration and the unit time.

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

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

**Parameters:**

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

**Returns:**

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

---

**3.62.2.3** `def sparksee.SparkseeProperties.get_boolean (self, key, def)`

Gets a property as a boolean.

**Parameters:**

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

**Returns:**

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

---

**3.62.2.4** `def sparksee.SparkseeProperties.load (self, path)`

Loads properties from the given file path.

**Parameters:**

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

---

**3.62.2.5** `def sparksee.SparkseeProperties.get_integer (self, key, def)`

Gets a property as an integer.

**Parameters:**

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

**Returns:**

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

---

**3.63** `sparksee.StringList Class Reference`

String list.
Public Member Functions

- def clear
  Clears the list.

- def __init__
  Constructor.

- def __iter__
  Gets a new TypeListIterator.

- def iterator
  Gets a new StringListIterator.

- def count
  Number of elements in the list.

- def add
  Adds a String at the end of the list.

### 3.63.1 Detailed Description

String list.
It stores a String (unicode) list.
Use StringListIterator to access all elements into this collection.

**Author:**
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

### 3.63.2 Member Function Documentation

#### 3.63.2.1 def sparksee.StringList.__init__ ( self)

Constructor.
This creates an empty list.

#### 3.63.2.2 def sparksee.StringList.__iter__ ( self)

Gets a new TypeListIterator.

**Returns:**
TypeListIterator instance
3.63.2.3  def sparksee.StringList.iterator ( self)  
Gets a new StringListIterator.

Returns:  
StringListIterator instance.

3.63.2.4  def sparksee.StringList.count ( self)  
Number of elements in the list.

Returns:  
Number of elements in the list.

3.63.2.5  def sparksee.StringList.add ( self, str)  
Adds a String at the end of the list.

Parameters:  

3.64  sparksee.StringListIterator Class Reference  
StringList iterator class.

Public Member Functions

•  def has_next  
  Gets if there are more elements.

•  def next  
  Moves to the next element.

•  def __next__  
  Used in next().

3.64.1  Detailed Description  
StringList iterator class.
Iterator to traverse all the strings into a StringList instance.

Author:  
Sparsity Technologies http://www.sparsity-technologies.com
3.64.2 Member Function Documentation

3.64.2.1 def sparksee.StringListIterator.has_next (self)
Gets if there are more elements.

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

3.64.2.2 def sparksee.StringListIterator.next (self)
Moves to the next element.

Returns:
The next element.

3.64.2.3 def sparksee.StringListIterator.__next__ (self)
Used in next().

Returns:
The next element

3.65 sparksee.StrongConnectivity Class Reference

StrongConnectivity class.
Inheritance diagram for sparksee.StrongConnectivity:

Collaboration diagram for sparksee.StrongConnectivity:

Public Member Functions

• def add_all_node_types
Allows connectivity through all node types of the graph.

- `def run`
  Runs the algorithm in order to find the connected components.

- `def exclude_nodes`
  Set which nodes can’t be used.

- `def add_edge_type`
  Allows connectivity through edges of the given type.

- `def add_node_type`
  Allows connectivity through nodes of the given type.

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

- `def exclude_edges`
  Set which edges can’t be used.

- `def add_all_edge_types`
  Allows connectivity through all edge types of the graph.

- `def get_connected_components`
  Returns the results generated by the execution of the algorithm.

- `def is_closed`
  Gets if Connectivity has been closed or not.

- `def close`
  Closes the Connectivity instance.

### 3.65.1 Detailed Description

**StrongConnectivity** class.

Any class implementing this abstract class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type.

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

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

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
3.65.2 Member Function Documentation

3.65.2.1 def sparksee.StrongConnectivity.run (self)
Runs the algorithm in order to find the connected components.
This method can be called only once.
Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.2 def sparksee.StrongConnectivity.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage
of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

Parameters:

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.3 def sparksee.StrongConnectivity.add_edge_type (self, type, dir)
Allows connectivity through edges of the given type.

Parameters:

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

Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.4 def sparksee.StrongConnectivity.add_node_type (self, t)
Allows connectivity through nodes of the given type.

Parameters:

   t null

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.5 def sparksee.StrongConnectivity.set_materialized_attribute (self, 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.

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.6 def sparksee.StrongConnectivity.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

**Parameters:**

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.7 def sparksee.StrongConnectivity.add_all_edge_types (self, dir)
Allows connectivity through all edge types of the graph.

**Parameters:**

dir [in] Edge direction.

Reimplemented in sparksee.StrongConnectivityGabow.

3.65.2.8 def sparksee.StrongConnectivity.get_connected_components (self)
Returns the results generated by the execution of the algorithm.
These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

**Returns:**

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.StrongConnectivityGabow.
3.65.2.9  def sparksee.Connectivity.is_closed ( self) [inherited]  
Gets if Connectivity has been closed or not.

See also:
   close()

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

3.65.2.10  def sparksee.Connectivity.close ( self) [inherited]  
Closes the Connectivity instance.
It must be called to ensure the integrity of all data.

3.66  sparksee.StrongConnectivityGabow Class Reference  
This class can be used to solve the problem of finding strongly connected components in a directed graph.

Inheritance diagram for sparksee.StrongConnectivityGabow:

```
  sparksee.StrongConnectivityGabow
  |                  |
  |                  |
sparksee.StrongConnectivity
  |                  |
  |                  |
sparksee.Connectivity
```

Collaboration diagram for sparksee.StrongConnectivityGabow:

```
  sparksee.StrongConnectivityGabow
  |                  |
  |                  |
sparksee.StrongConnectivity
  |                  |
  |                  |
sparksee.Connectivity
```

Public Member Functions

-  def add_all_node_types
   Allows connectivity through all node types of the graph.

-  def __init__
   Creates a new instance of StrongConnectivityGabow.
3.66 sparksee.StrongConnectivityGabow Class Reference

- def exclude_nodes
  Set which nodes can’t be used.

- def add_edge_type
  Allows connectivity through edges of the given type.

- def run
  Executes the algorithm.

- def add_node_type
  Allows connectivity through nodes of the given type.

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

- def exclude_edges
  Set which edges can’t be used.

- def add_all_edge_types
  Allows connectivity through all edge types of the graph.

- def get_connected_components
  Returns the results generated by the execution of the algorithm.

- def is_closed
  Gets if Connectivity has been closed or not.

- def close
  Closes the Connectivity instance.

3.66.1 Detailed Description

This class can be used to solve the problem of finding strongly connected components in a directed graph. It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type. This implementation is based on the Gabow algorithm.

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

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

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

Author:

Sparsity Technologies http://www.sparsity-technologies.com
3.66.2 Member Function Documentation

3.66.2.1 def sparksee.StrongConnectivityGabow.__init__ (self, session)

Creates a new instance of StrongConnectivityGabow.

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

Parameters:

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

3.66.2.2 def sparksee.StrongConnectivityGabow.exclude_nodes (self, nodes)

Set which nodes can’t be used.

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

Parameters:

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

Reimplemented from sparksee.StrongConnectivity.

3.66.2.3 def sparksee.StrongConnectivityGabow.add_edge_type (self, type, dir)

Allows connectivity through edges of the given type.

Parameters:

type [in] Edge type.

dir [in] Edge direction.

Reimplemented from sparksee.StrongConnectivity.

3.66.2.4 def sparksee.StrongConnectivityGabow.add_node_type (self, t)

Allows connectivity through nodes of the given type.

Parameters:

null

Reimplemented from sparksee.StrongConnectivity.

3.66.2.5 def sparksee.StrongConnectivityGabow.set_materialized_attribute (self, 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.

Reimplemented from `sparksee.StrongConnectivity`.

### 3.66.2.6 def sparksee.StrongConnectivityGabow.exclude_edges (self, edges)

Set which edges can’t be used.

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

**Parameters:**

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

Reimplemented from `sparksee.StrongConnectivity`.

### 3.66.2.7 def sparksee.StrongConnectivityGabow.add_all_edge_types (self, dir)

Allows connectivity through all edge types of the graph.

**Parameters:**

- `dir` [in] Edge direction.

Reimplemented from `sparksee.StrongConnectivity`.

### 3.66.2.8 def sparksee.StrongConnectivityGabow.get_connected_components (self)

Returns the results generated by the execution of the algorithm.

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

**Returns:**

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

Reimplemented from `sparksee.StrongConnectivity`.

### 3.66.2.9 def sparksee.Connectivity.is_closed (self) [inherited]

Gets if `Connectivity` has been closed or not.

**See also:**

- `close()`
Returns:

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

3.66.2.10 def sparksee.Connectivity.close (self) [inherited]
Closes the Connectivity instance.
It must be called to ensure the integrity of all data.

3.67 sparksee.TextStream Class Reference

TextStream class.

Public Member Functions

• def read
  Read data from the stream.

• def is_null
  Returns TRUE if the stream is not available.

• def write
  Write data to the stream.

• def __init__
  Creates a new instance.

• def close
  Closes the stream.

3.67.1 Detailed Description

TextStream class.
It allows for reading and writing Text attribute values.

It is very important to close the stream once no more reading or writing operations will be performed to
ensure data is successfully stored.

Whereas string attributes are set and got using the Value class, text attributes are operated using a stream
pattern.

Use of TextStream for writing: (i) Create a TextStream instance and (ii) set the stream for a text attribute of
a node or edge instance with the graph SetAttributeText method. Once the set attribute text has been done,
(iii) perform as many write operations as you need to the TextStream instance. Lastly, (iv) execute Close
to flush and close the stream.

Use of TextStream for reading: (i) Get the stream of a text attribute of a node or edge instance with
the GetAttributeText graph method. Once you have the TextStream instance, (ii) you can execute Read
operations to read from the stream. (iii) The end of the stream is reached when Read returns 0. Finally, (iv)
execute Close to close stream resources.

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.
3.67 sparksee.TextStream Class Reference

Author:
Sparsity Technologies\url{http://www.sparsity-technologies.com}

3.67.2 Member Function Documentation

3.67.2.1 def sparksee.TextStream.read (self, length)
Read data from the stream.

Parameters:

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

Returns:
The read data.

3.67.2.2 def sparksee.TextStream.is_null (self)
Returns TRUE if the stream is not available.
It returns for reading or writing data.

Returns:
FALSE if the stream is ready

3.67.2.3 def sparksee.TextStream.write (self, dataIN, length)
Write data to the stream.

Parameters:

dataIN The string to write data from.
length Number of characters to write. It must be > 0.

3.67.2.4 def sparksee.TextStream.__init__ (self, append)
Creates a new instance.
A TextStream only can be created by the user to write data.

Parameters:

append [in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the begining of the stream.

3.67.2.5 def sparksee.TextStream.close (self)
Closes the stream.
Once the Stream is closed, it cannot be used again.
Closing the stream is mandatory when the stream is not null and strongly recommended when it’s null to avoid deallocation problems in some platforms.
**Public Member Functions**

- `def add_node_type`
  
  *Allows for traversing nodes of the given type.*

- `def has_next`
  
  *Gets if there are more objects to be traversed.*

- `def exclude_nodes`
  
  *Set which nodes can’t be used.*

- `def is_closed`
  
  *Gets if Traversal has been closed or not.*

- `def set_maximum_hops`
  
  *Sets the maximum hops restriction.*

- `def add_all_edge_types`
  
  *Allows for traversing all edge types of the graph.*

- `def __next__`
  
  *Used in next().*

- `def get_current_depth`
  
  *Returns the depth of the current node.*

- `def close`
  
  *Closes the Traversal instance.*

- `def add_all_node_types`
  
  *Allows for traversing all node types of the graph.*

- `def add_edge_type`
  
  *Allows for traversing edges of the given type.*

- `def next`
  
  *Gets the next object of the traversal.*

- `def __iter__`
3.68 sparksee.Traversal Class Reference

*Gets a new TraversalIterator.*

- def exclude_edges
  
  Set which edges can’t be used.

### 3.68.1 Detailed Description

**Traversal** class.

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

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

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

### 3.68.2 Member Function Documentation

#### 3.68.2.1 def sparksee.Traversal.add_node_type (self, type)

Allows for traversing nodes of the given type.

**Parameters:**

- **type** null

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

#### 3.68.2.2 def sparksee.Traversal.has_next (self)

Gets if there are more objects to be traversed.

**Returns:**

TRUE if there are more objects, FALSE otherwise.

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

#### 3.68.2.3 def sparksee.Traversal.exclude_nodes (self, nodes)

Set which nodes can’t be used.

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

**Parameters:**

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

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.
3.68.2.4 def sparksee.Traversal.is_closed (self)
 Gets if Traversal has been closed or not.

See also:
  close()

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

3.68.2.5 def sparksee.Traversal.set_maximum_hops (self, maxhops)
 Sets the maximum hops restriction.
 All paths longer than the maximum hops restriction will be ignored.

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

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

3.68.2.6 def sparksee.Traversal.add_all_edge_types (self, dir)
 Allows for traversing all edge types of the graph.

Parameters:
  dir [in] Edge direction.

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

3.68.2.7 def sparksee.Traversal.__next__ (self)
 Used in next().

Returns:
  The next element

3.68.2.8 def sparksee.Traversal.get_current_depth (self)
 Returns the depth of the current node.
 That is, it returns the depth of the node returned in the last call to Next().

Returns:
  The depth of the current node.

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.
3.68.2.9  def sparksee.Traversal.close (self)
Closes the Traversal instance.
It must be called to ensure the integrity of all data.

3.68.2.10  def sparksee.Traversal.add_edge_type (self, type, dir)
Allows for traversing edges of the given type.

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

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

3.68.2.11  def sparksee.Traversal.next (self)
Gets the next object of the traversal.

Returns:
   A node or edge identifier.

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

3.68.2.12  def sparksee.Traversal.__iter__ (self)
Gets a new TraversalIterator.

Returns:
   TraversalIterator instance

3.68.2.13  def sparksee.Traversal.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

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

Reimplemented in sparksee.TraversalDFS, and sparksee.TraversalBFS.

3.69  sparksee.TraversalBFS Class Reference
Breadth-First Search implementation of Traversal.
Inheritance diagram for sparksee.TraversalBFS:

Collaboration diagram for sparksee.TraversalBFS:

Public Member Functions

- `__init__`
  Creates a new instance.

- `add_node_type`
  Allows for traversing nodes of the given type.

- `add_all_node_types`
  Allows for traversing all node types of the graph.

- `has_next`
  Gets if there are more objects to be traversed.

- `exclude_nodes`
  Set which nodes can’t be used.

- `add_edge_type`
  Allows for traversing edges of the given type.

- `next`
  Gets the next object of the traversal.

- `get_current_depth`
  Returns the depth of the current node.

- `set_maximum_hops`
  Sets the maximum hops restriction.

- `exclude_edges`
  Set which edges can’t be used.

- `add_all_edge_types`
 Allows for traversing all edge types of the graph.

- **def is_closed**
  
  Gets if Traversal has been closed or not.

- **def __next__**
  
  Used in next().

- **def close**
  
  Closes the Traversal instance.

- **def __iter__**
  
  Gets a new TraversalIterator.

### 3.69.1 Detailed Description

Breadth-First Search implementation of Traversal.

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

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

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

### 3.69.2 Member Function Documentation

#### 3.69.2.1 def sparksee.TraversalBFS.__init__(self, session, node)

Creates a new instance.

**Parameters:**

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

#### 3.69.2.2 def sparksee.TraversalBFS.add_node_type(self, type)

Allows for traversing nodes of the given type.

**Parameters:**

- `type` null

Reimplemented from sparksee.Traversal.
3.69.2.3 `def sparksee.TraversalBFS.has_next (self)`
Gets if there are more objects to be traversed.

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

Reimplemented from `sparksee.Traversal`.

3.69.2.4 `def sparksee.TraversalBFS.exclude_nodes (self, nodes)`
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented from `sparksee.Traversal`.

3.69.2.5 `def sparksee.TraversalBFS.add_edge_type (self, type, dir)`
Allows for traversing edges of the given type.

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

Reimplemented from `sparksee.Traversal`.

3.69.2.6 `def sparksee.TraversalBFS.next (self)`
Gets the next object of the traversal.

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

Reimplemented from `sparksee.Traversal`.

3.69.2.7 `def sparksee.TraversalBFS.get_current_depth (self)`
Returns the depth of the current node.
That is, it returns the depth of the node returned in the last call to Next().

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

Reimplemented from `sparksee.Traversal`. 
3.69.2.8  def sparksee.TraversalBFS.set_maximum_hops (self, maxhops)
Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

Parameters:

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

Reimplemented from sparksee.Traversal.

3.69.2.9  def sparksee.TraversalBFS.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

Parameters:

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

Reimplemented from sparksee.Traversal.

3.69.2.10 def sparksee.TraversalBFS.add_all_edge_types (self, dir)
Allows for traversing all edge types of the graph.

Parameters:

dir [in] Edge direction.

Reimplemented from sparksee.Traversal.

3.69.2.11 def sparksee.Traversal.is_closed (self) [inherited]
Gets if Traversal has been closed or not.

See also:

close()

Returns:

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

3.69.2.12 def sparksee.Traversal.__next__ (self) [inherited]
Used in next().

Returns:

The next element
3.69.2.13 `def sparksee.Traversal.close (self) [inherited]`
Closes the `Traversal` instance.
It must be called to ensure the integrity of all data.

3.69.2.14  
`def sparksee.Traversal.__iter__ (self) [inherited]`
Gets a new TraversalIterator.

**Returns:**
TraversalIterator instance

### 3.70 `sparksee.TraversalDFS Class Reference`

Depth-First Search (DFS) implementation of `Traversal`.

Inheritance diagram for `sparksee.TraversalDFS`:

```
sparksee.TraversalDFS
    `sparksee.Traversal`
```

Collaboration diagram for `sparksee.TraversalDFS`:

```
sparksee.TraversalDFS
    `sparksee.Traversal`
```

#### Public Member Functions

- **def add_node_type**
  
  *` Allows for traversing nodes of the given type.*

- **def add_all_node_types**
  
  *` Allows for traversing all node types of the graph.*

- **def has_next**
  
  *` Gets if there are more objects to be traversed.*

- **def exclude_nodes**
  
  *` Set which nodes can't be used.*

- **def add_edge_type**
  
  *` Allows for traversing edges of the given type.*
3.70.1 Detailed Description

Depth-First Search (DFS) implementation of Traversal.
Starting from a source or root node, it visits as far as possible along each branch before backtracking.
Check out the ’Algorithms’ section in the SPARKSEE User Manual for more details on this.

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

3.70.2 Member Function Documentation

3.70.2.1 def sparksee.TraversalDFS.add_node_type ( self, type )
Allows for traversing nodes of the given type.

Parameters:

- type  null

Reimplemented from sparksee.Traversal.
3.70.2.2  def sparksee.TraversalDFS.has_next (self)
Gets if there are more objects to be traversed.

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

Reimplemented from sparksee.Traversal.

3.70.2.3  def sparksee.TraversalDFS.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage
of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

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

Reimplemented from sparksee.Traversal.

3.70.2.4  def sparksee.TraversalDFS.add_edge_type (self, type, dir)
Allows for traversing edges of the given type.

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

Reimplemented from sparksee.Traversal.

3.70.2.5  def sparksee.TraversalDFS.__init__ (self, session, node)
Creates a new instance.

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

3.70.2.6  def sparksee.TraversalDFS.next (self)
Gets the next object of the traversal.

Returns:
A node or edge identifier.

Reimplemented from sparksee.Traversal.
3.70.2.7  def sparksee.TraversalDFS.get_current_depth ( self)
Returns the depth of the current node.
That is, it returns the depth of the node returned in the last call to Next().

Returns:

The depth of the current node.

Reimplemented from sparksee.Traversal.

3.70.2.8  def sparksee.TraversalDFS.set_maximum_hops ( self, maxhops)
Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

Parameters:

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

Reimplemented from sparksee.Traversal.

3.70.2.9  def sparksee.TraversalDFS.exclude_edges ( self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage
of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

Parameters:

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

Reimplemented from sparksee.Traversal.

3.70.2.10  def sparksee.TraversalDFS.add_all_edge_types ( self, dir)
Allows for traversing all edge types of the graph.

Parameters:

dir [in] Edge direction.

Reimplemented from sparksee.Traversal.

3.70.2.11  def sparksee.Traversal.is_closed ( self) [inherited]
Gets if Traversal has been closed or not.

See also:

close()

Returns:

TRUE if the Traversal instance has been closed, FALSE otherwise.
3.70.2.12 def sparksee.Traversal.__next__ (self) [inherited]
Used in next().

Returns:
The next element

3.70.2.13 def sparksee.Traversal.close (self) [inherited]
Closes the Traversal instance.
It must be called to ensure the integrity of all data.

3.70.2.14 def sparksee.Traversal.__iter__ (self) [inherited]
Gets a new TraversalIterator.

Returns:
TraversalIterator instance

3.71 sparksee.Type Class Reference

Type data class.

Public Member Functions

• def get_restricted_from
  Gets the tail or source type identifier for restricted edge types.

• def get_are_neighbors_indexed
  Gets if this is an edge type with neighbors index.

• def get_object_type
  Gets the object type.

• def get_restricted_to
  Gets the head or target type identifier for restricted edge types.

• def get_is_restricted
  Gets if this is a restricted edge type.

• def get_num_objects
  Gets the number of objects belonging to the type.

• def get_id
  Gets the Sparksee type identifier.

• def get_is_directed
  Gets if this is a directed edge type.
• def get_name
  
  Gets the unique type name.

Static Public Attributes

• int EDGES_TYPE = 3
  
  Identifier for all edgeType attributes.

• int GLOBAL_TYPE = 1
  
  Global type identifier.

• int INVALID_TYPE = 0
  
  Invalid type identifier.

• int NODES_TYPE = 2
  
  Identifier for all nodeType attributes.

3.71.1 Detailed Description

Type data class.

It contains information about a node or edge type.

Author:

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

3.71.2 Member Function Documentation

3.71.2.1 def sparksee.Type.get_restricted_from ( self)

Gets the tail or source type identifier for restricted edge types.

Returns:

For restricted edge types, the tail or source type identifier, the Type InvalidType otherwise.

3.71.2.2 def sparksee.Type.get_are_neighbors_indexed ( self)

Gets if this is an edge type with neighbors index.

Returns:

TRUE for edges types with neighbors index, FALSE otherwise.
3.71.2.3  def sparksee.Type.get_object_type (self)
Gets the object type.

Returns:
   The object type.

3.71.2.4  def sparksee.Type.get_restricted_to (self)
Gets the head or target type identifier for restricted edge types.

Returns:
   For restricted edge types, the head or target type identifier, the Type InvalidType otherwise.

3.71.2.5  def sparksee.Type.get_is_restricted (self)
Gets if this is a restricted edge type.

Returns:
   TRUE for restricted edge types, FALSE otherwise.

3.71.2.6  def sparksee.Type.get_num_objects (self)
Gets the number of objects belonging to the type.

Returns:
   The number of objects belonging to the type.

3.71.2.7  def sparksee.Type.get_id (self)
Gets the Sparksee type identifier.

Returns:
   The Sparksee type identifier.

3.71.2.8  def sparksee.Type.get_is_directed (self)
Gets if this is a directed edge type.

Returns:
   TRUE for directed edge types, FALSE otherwise.

3.71.2.9  def sparksee.Type.get_name (self)
Gets the unique type name.

Returns:
   The unique type name.
3.72 sparksee.TypeExporter Class Reference

Base TypeExporter class.
Inheritance diagram for sparksee.TypeExporter:

```
sparksee.TypeExporter
  ↓
sparksee.EdgeTypeExporter
  ↓
sparksee.NodeTypeExporter
```

Public Member Functions

- def set_row_writer
  Sets the output data destination.

- def set_frequency
  Sets the frequency of listener notification.

- def run
  Runs export process.

- def register
  Registers a new listener.

- def set_graph
  Sets the graph that will be exported.

- def set_header
  Sets the presence of a header row.

- def set_type
  Sets the type to be exported.

- def set_attributes
  Sets the list of Attributes.

3.72.1 Detailed Description

Base TypeExporter class.
Base class to export a node or edge type from a graph using a RowWriter.

TypeExporterListener can be registered to receive information about the progress of the export process by means of TypeExporterEvent. The default frequency of notification to listeners is 100000.

By default no header row is created.
Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
3.72.2 Member Function Documentation

3.72.2.1 `def sparksee.TypeExporter.set_row_writer (self, rw)`
Sets the output data destination.

Parameters:
- `rw` [in] Input RowWriter.

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.2 `def sparksee.TypeExporter.set_frequency (self, freq)`
Sets the frequency of listener notification.

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

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.3 `def sparksee.TypeExporter.run (self)`
Runs export process.

Exceptions:
- `IOError` If bad things happen writing to the RowWriter.
- `RuntimeError` null

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.4 `def sparksee.TypeExporter.register (self, tel)`
Registers a new listener.

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

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.5 `def sparksee.TypeExporter.set_graph (self, graph)`
Sets the graph that will be exported.

Parameters:
- `graph` [in] Graph.

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.
3.72.2.6   def sparksee.TypeExporter.set_header ( self, header)
Sets the presence of a header row.

Parameters:

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

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.7   def sparksee.TypeExporter.set_type ( self, type)
Sets the type to be exported.

Parameters:

    type [in] Type identifier.

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.72.2.8   def sparksee.TypeExporter.set_attributes ( self, attrs)
Sets the list of Attributes.

Parameters:

    attrs [in] Attribute identifiers to be exported

Reimplemented in sparksee.NodeTypeExporter, and sparksee.EdgeTypeExporter.

3.73   sparksee.TypeExporterEvent Class Reference
Provides information about the progress of an TypeExproter instance.

Public Member Functions

• def get_count
  Gets the current number of objects exported.

• def is_last
  Gets if this is the last event or not.

• def get_type_id
  Gets the type identifier.

• def get_total
  Gets the total number of objects exported.
3.73 Detailed Description

Provides information about the progress of an TypeExproter instance. Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

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

3.73.2 Member Function Documentation

3.73.2.1 def sparksee.TypeExporterEvent.get_count ( self)

Gets the current number of objects exported.

Returns:

The current number of objects exported.

3.73.2.2 def sparksee.TypeExporterEvent.is_last ( self)

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

3.73.2.3 def sparksee.TypeExporterEvent.get_type_id ( self)

Gets the type identifier.

Returns:

The type identifier.

3.73.2.4 def sparksee.TypeExporterEvent.get_total ( self)

Gets the total number of objects exported.

Returns:

The total number of objects exported.

3.74 sparksee.TypeExporterListener Class Reference

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Public Member Functions

- def notify_event

  Method to be notified from a TypeExporter.
3.74.1 Detailed Description

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.
Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

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

3.74.2 Member Function Documentation

3.74.2.1 def sparksee.TypeExporterListener.notify_event ( self, tee)
Method to be notified from a TypeExporter.

Parameters:

tee [in] Notified event.

3.75 sparksee.TypeList Class Reference

Sparksee type identifier list.

Public Member Functions

• def add
  Adds a Sparksee type identifier at the end of the list.

• def clear
  Clears the list.

• def __iter__
  Gets a new TypeListIterator.

• def iterator
  Gets a new TypeListIterator.

• def __init__
  Constructor.

• def count
  Number of elements in the list.

3.75.1 Detailed Description

Sparksee type identifier list.
It stores a Sparksee node or edge type identifier list.
Use TypeListIterator to access all elements into this collection.
3.75.2 Member Function Documentation

3.75.2.1 def sparksee.TypeList.add (self, type)
Adds a Sparksee type identifier at the end of the list.

Parameters:
    type [in] Sparksee type identifier.

3.75.2.2 def sparksee.TypeList.__iter__ (self)
Gets a new TypeListIterator.

Returns:
    TypeListIterator instance

3.75.2.3 def sparksee.TypeList.iterator (self)
Gets a new TypeListIterator.

Returns:
    TypeListIterator instance.

3.75.2.4 def sparksee.TypeList.__init__ (self)
Constructor.
This creates an empty list.

3.75.2.5 def sparksee.TypeList.count (self)
Number of elements in the list.

Returns:
    Number of elements in the list.

3.76 sparksee.TypeListIterator Class Reference

TypeList iterator class.
Public Member Functions

- def has_next
  
  Gets if there are more elements.

- def next
  
  Moves to the next element.

- def __next__
  
  Used in next().

3.76.1 Detailed Description

TypeList iterator class.

Iterator to traverse all the Sparksee node or edge type identifiers into a TypeList instance.

Author:

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

3.76.2 Member Function Documentation

3.76.2.1 def sparksee.TypeListIterator.has_next ( self )

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

3.76.2.2 def sparksee.TypeListIterator.next ( self )

Moves to the next element.

Returns:

The next element.

3.76.2.3 def sparksee.TypeListIterator.__next__ ( self )

Used in next().

Returns:

The next element
3.77  sparksee.TypeLoader Class Reference

Base Sparksee.TypeLoader class.

Inheritance diagram for sparksee.TypeLoader:

```
sparksee.TypeLoader
  sparksee.EdgeTypeLoader
  sparksee.NodeTypeLoader
```

Public Member Functions

- def run_two_phases
  
  Run the loader for two phases loading.

- def run_n_phases
  
  Run the loader for N phases loading.

- def set_frequency
  
  Sets the frequency of listener notification.

- def set_log_off
  
  Trans off all the error reporting.

- def run
  
  Run the loader.

- def set_log_error
  
  Sets a log error file.

- def set_type
  
  Sets the type to be loaded.

- def set_row_reader
  
  Sets the input data source.

- def register
  
  Registers a new listener.

- def set_attribute_positions
  
  Sets the list of attribute positions.

- def set_locale
  
  Sets the locale that will be used to read the data.

- def set_graph
  
  Sets the graph where the data will be loaded.

- def set_timestamp_format

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
Sets a specific timestamp format.

- def set_attributes
  Sets the list of Attributes.

3.77.1 Detailed Description

Base TypeLoader class.

Base class to load a node or edge type from a graph using a RowReader. TypeLoaderListener can be registered to receive information about the progress of the load process by means of TypeLoaderEvent. The default frequency of notification to listeners is 100000.

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

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

3.77.2 Member Function Documentation

3.77.2.1 def sparksee.TypeLoader.run_two_phases (self)

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

Exceptions:

- IOError null
  RuntimeError null

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.2 def sparksee.TypeLoader.run_n_phases (self, partitions)

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

Parameters:

- partitions [in] Number of horizontal partitions to perform the load.

Exceptions:

- IOError null
  RuntimeError null

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.
3.77.2.3  def sparksee.TypeLoader.set_frequency ( self, freq)
Sets the frequency of listener notification.

Parameters:

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

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.4  def sparksee.TypeLoader.set_log_off ( self)
Truns off all the error reporting.
The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn
off the logs, but abort at the first error what you should do is not call this method and not set a logError file.
Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.5  def sparksee.TypeLoader.run ( self)
Run the loader.

Exceptions:

    IOError   null
    RuntimeError   null

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.6  def sparksee.TypeLoader.set_log_error ( self, path)
Sets a log error file.
By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged
there and the load process does not stop.

Parameters:

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

Exceptions:

    IOError   If bad things happen opening the file.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.7  def sparksee.TypeLoader.set_type ( self, type)
Sets the type to be loaded.

Parameters:

type  [in] Type identifier.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.
3.77.2.8  def sparksee.TypeLoader.set_row_reader (self, rr)
Sets the input data source.

Parameters:
    rr [in] Input RowReader.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.9  def sparksee.TypeLoader.register (self, tel)
Registers a new listener.

Parameters:
    tel TypeLoaderListener to be registered.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.10  def sparksee.TypeLoader.set_attribute_positions (self, attrsPos)
Sets the list of attribute positions.

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

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.11  def sparksee.TypeLoader.set_locale (self, 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.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.12  def sparksee.TypeLoader.set_graph (self, graph)
Sets the graph where the data will be loaded.

Parameters:
    graph [in] Graph.

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.
3.77.2.13 def sparksee.TypeLoader.set_timestamp_format (self, timestampFormat)
Sets a specific timestamp format.

Parameters:

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

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.77.2.14 def sparksee.TypeLoader.set_attributes (self, attrs)
Sets the list of Attributes.

Parameters:

attrs [in] Attribute identifiers to be loaded

Reimplemented in sparksee.NodeTypeLoader, and sparksee.EdgeTypeLoader.

3.78 sparksee.TypeLoaderEvent Class Reference

Provides information about the progress of a TypeLoader instance.

Public Member Functions

• def get_total_phases
  
  Gets the total number of phases.

• def get_count
  
  Gets the current number of objects created.

• def get_total_partition_steps
  
  Gets the total number of steps in the current partition.

• def is_last
  
  Gets if this is the last event or not.

• def get_partition
  
  Gets the current partition.

• def get_type_id
  
  Gets the type identifier.

• def get_total_partitions
  
  Gets the total number of partitions.

• def get_phase
  
  Gets the current phase.
3.78 sparksee.TypeLoaderEvent Class Reference

3.78.1 Detailed Description

Provides information about the progress of a TypeLoader instance.
Check out the ‘Data import’ section in the SPARKSEE User Manual for more details on this.

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

3.78.2 Member Function Documentation

3.78.2.1 def sparksee.TypeLoaderEvent.get_total_phases ( self)
Gets the total number of phases.

Returns:
   The total number of phases.

3.78.2.2 def sparksee.TypeLoaderEvent.get_count ( self)
Gets the current number of objects created.

Returns:
   The current number of objects created.

3.78.2.3 def sparksee.TypeLoaderEvent.get_total_partition_steps ( self)
Gets the total number of steps in the current partition.

Returns:
   The total number steps in the current partition.

3.78.2.4 def sparksee.TypeLoaderEvent.is_last ( self)
Gets if this is the last event or not.

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

3.78.2.5 def sparksee.TypeLoaderEvent.get_partition ( self)
Gets the current partition.

Returns:
   The current partition.
3.78.2.6 def sparksee.TypeLoaderEvent.get_type_id (self)
Gets the type identifier.

Returns:
The type identifier.

3.78.2.7 def sparksee.TypeLoaderEvent.get_total_partitions (self)
Gets the total number of partitions.

Returns:
The total number of partitions.

3.78.2.8 def sparksee.TypeLoaderEvent.get_phase (self)
Gets the current phase.

Returns:
The current phase.

3.79 sparksee.TypeLoaderListener Class Reference

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

Public Member Functions

- def notify_event
  Method to receive events from a Loader.

3.79.1 Detailed Description

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.
Check out the ‘Data import’ section in the SPARKSEE User Manual for more details on this.

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

3.79.2 Member Function Documentation

3.79.2.1 def sparksee.TypeLoaderListener.notify_event (self, ev)
Method to receive events from a Loader.

Parameters:

- ev Loader.LoaderEvent with information from a running Loader.
3.80 sparksee.Value Class Reference

Value class.

Public Member Functions

- def `set_long_void`
  - Sets the Value.

- def `get_oid`
  - Gets OID Value.

- def `set_boolean`
  - Sets the Value.

- def `__init__`
  - Copy constructor.

- def `get_boolean`
  - Gets Boolean Value.

- def `set_null_void`
  - Sets the Value to NULL.

- def `set_timestamp_void`
  - Sets the Value.

- def `set_string_void`
  - Sets the Value.

- def `__init__`
  - Creates a new instance.

- def `to_string`
  - Returns a String representation of the Value, used in `__unicode__` and `__str__`.

- def `set_integer_void`
  - Sets the Value.

- def `get_data_type`
  - Gets the DataType.

- def `get_string`
  - Gets String Value.

- def `set_double_void`
  - Sets the Value.

- def `equals`
  - Compares with the given Value.
• def is_null
  
  Gets if this is a NULL Value.

• def set_void
  
  Sets the Value.

• def set_integer
  
  Sets the Value.

• def set_timestamp_void
  
  Sets the Value.

• def set_string
  
  Sets the Value.

• def set_timestamp
  
  Sets the Value.

• def set_oid_void
  
  Sets the OID Value.

• def get_double
  
  Gets Double Value.

• def operator
  
  Assignment operator.

• def set_timestamp
  
  Sets the Value.

• def compare
  
  Compares with the given Value.

• def get_long
  
  Gets Long Value.

• def set_boolean_void
  
  Sets the Value.

• def set_long
  
  Sets the Value.

• def get_timestamp
  
  Gets Timestamp Value.

• def set_null
  
  Sets the Value to NULL.

• def get_integer
3.80 sparksee.Value Class Reference

Gets Integer Value.

- def set_double
  Sets the Value.

- def set_oid
  Sets the Value.

Static Public Attributes

- int MAX_LENGTH_STRING = 2047
  Maximum number of characters allowed for a String.

3.80.1 Detailed Description

Value class.

It is a container which stores a value and its data type (domain). A Value can be NULL.

Author:

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

3.80.2 Member Function Documentation

3.80.2.1 def sparksee.Value.set_long_void (self, value)
Sets the Value.

Parameters:


3.80.2.2 def sparksee.Value.get_oid (self)
Gets OID Value.

This must be an non-NULL OID Value.

Returns:

  The OID Value.

3.80.2.3 def sparksee.Value.set_boolean (self, value)
Sets the Value.

Parameters:

  value [in] New Boolean value.

Returns:

  The calling instance.
3.80.2.4  def sparksee.Value.__init__ ( self, value)
Copy constructor.

Parameters:

   value  [in] Value to be copied.

3.80.2.5  def sparksee.Value.get_boolean ( self)
Gets Boolean Value.
This must be a non-NULL Boolean Value.

Returns:

   The Boolean Value.

3.80.2.6  def sparksee.Value.set_timestamp_void ( self, value)
Sets the Value.

Parameters:


3.80.2.7  def sparksee.Value.set_string_void ( self, value)
Sets the Value.

Parameters:

   value  [in] New String value.

3.80.2.8  def sparksee.Value.__init__ ( self)
Creates a new instance.
It creates a NULL Value.

3.80.2.9  def sparksee.Value.to_string ( self)
Returns a String representation of the Value, used in __unicode__ and __str__.

3.80.2.10 def sparksee.Value.set_integer_void ( self, value)
Sets the Value.

Parameters:

   value  [in] New Integer value.
3.80.2.11  def sparksee.Value.get_data_type (self)
Gets the DataType.
Returns:
   The DataType.

3.80.2.12  def sparksee.Value.get_string (self)
Gets String Value.
This must be a non-NULL String Value.
Returns:
   The String Value.

3.80.2.13  def sparksee.Value.set_double_void (self, value)
Sets the Value.
Parameters:
   value [in] New Double value.

3.80.2.14  def sparksee.Value.equals (self, value)
Compares with the given Value.
It does not work if the given Value objects does not have the same DataType.
Parameters:
   value Given value to compare to.
Returns:
   TRUE if this Value is equal to the given one; FALSE otherwise.

3.80.2.15  def sparksee.Value.is_null (self)
Gets if this is a NULL Value.
Returns:
   TRUE if this is a NULL Value, FALSE otherwise.

3.80.2.16  def sparksee.Value.set_void (self, value)
Sets the Value.
Parameters:
   value [in] New value.
3.80.2.17  def sparksee.Value.set_integer (self, value)
Sets the Value.

Parameters:

  value [in] New Integer value.

Returns:

  The calling instance.

3.80.2.18  def sparksee.Value.set_timestamp_void (self, year, month, day, hour, minutes, seconds, millisecs)
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]).
  millisecs [in] The milliseconds ([0..999]).

3.80.2.19  def sparksee.Value.set_string (self, value)
Sets the Value.

Parameters:

  value [in] New String value.

Returns:

  The calling instance.

3.80.2.20  def sparksee.Value.set_timestamp (self, value)
Sets the Value.

Parameters:


Returns:

  The calling instance.
3.80.2.21  def sparksee.Value.set_oid_void ( self, value)
Sets the OID Value.

Parameters:

    value [in] New OID value.

3.80.2.22  def sparksee.Value.get_double ( self)
Gets Double Value.
This must be a non-NULL Double Value.

Returns:

    The Double Value.

3.80.2.23  def sparksee.Value.operator ( self, value)
Assignment operator.

Parameters:

    value [in] Value to be copied.

Returns:

    Returns the Value reference.

3.80.2.24  def sparksee.Value.set_timestamp ( self, year, month, day, hour, minutes, seconds, milliseecs)
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]).
    milliseecs [in] The milliseconds ([0..999]).

Returns:

    The calling instance.
3.80.2.25  def sparksee.Value.compare ( self, value)
Compares with the given Value.
It does not work if the given Value objects does not have the same DataType.

Parameters:
  value  Given value to compare to.

Returns:
  0 if this Value is equal to the given one; a value less than 0 if this Value is less than the given one; and a value greater than 0 if this Value is greater than the given one.

3.80.2.26  def sparksee.Value.get_long ( self)
Gets Long Value.
This must be a non-NULL Long Value.

Returns:
  The Long Value.

3.80.2.27  def sparksee.Value.set_boolean_void ( self, value)
Sets the Value.

Parameters:
  value  [in] New Boolean value.

3.80.2.28  def sparksee.Value.set_long ( self, value)
Sets the Value.

Parameters:

Returns:
  The calling instance.

3.80.2.29  def sparksee.Value.get_timestamp ( self)
Gets Timestamp Value.
This must be a non-NULL Timestamp Value.

Returns:
  The Timestamp Value.
3.80.2.30  def sparksee.Value.set_null (self)
Sets the Value to NULL.

Returns:
The calling instance.

3.80.2.31  def sparksee.Value.get_integer (self)
Gets Integer Value.
This must be a non-NULL Integer Value.

Returns:
The Integer Value.

3.80.2.32  def sparksee.Value.set_double (self, value)
Sets the Value.

Parameters:
value [in] New Double value.

Returns:
The calling instance.

3.80.2.33  def sparksee.Value.set_oid (self, value)
Sets the Value.

Parameters:
value [in] New OID Value.

Returns:
The calling instance.

3.81  sparksee.ValueList Class Reference

Value list.

Public Member Functions

• def __init__
  Constructor:

• def get
  Returns the Value at the specified position in the list.
• def clear
  Clears the list.

• def __iter__
  Gets a new ValueListIterator.

• def add
  Adds a value to the end of the list.

• def iterator
  Gets a new ValueListIterator.

• def count
  Number of elements in the list.

3.81.1 Detailed Description

Value list.
It stores a Value list.
Use ValueListIterator to access all elements into this collection.

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

3.81.2 Member Function Documentation

3.81.2.1 def sparksee.ValueList.__init__ (self)
Constructor.
This creates an empty list.

3.81.2.2 def sparksee.ValueList.get (self, index)
Returns the Value at the specified position in the list.

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

3.81.2.3 def sparksee.ValueList.__iter__ (self)
Gets a new ValueListIterator.

Returns:
  ValueListIterator instance
3.81.2.4 def sparksee.ValueList.add (self, value)
Adds a value to the end of the list.

Parameters:
  value [in] The value to add

3.81.2.5 def sparksee.ValueList.iterator (self)
Gets a new ValueListIterator.

Returns:
  ValueListIterator instance.

3.81.2.6 def sparksee.ValueList.count (self)
Number of elements in the list.

Returns:
  Number of elements in the list.

3.82 sparksee.ValueListIterator Class Reference

ValueList iterator class.

Public Member Functions

• def has_next
  Gets if there are more elements.

• def next
  Moves to the next element.

• def __next__
  Used in next().

3.82.1 Detailed Description

ValueList iterator class.
Iterator to traverse all the values into a ValueList instance.

Author:
  Sparsity Technologies http://www.sparsity-technologies.com
3.82.2 Member Function Documentation

3.82.2.1 def sparksee.ValueListIterator.has_next (self)
Gets if there are more elements.

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

3.82.2.2 def sparksee.ValueListIterator.next (self)
Moves to the next element.

Returns:
The next element.

3.82.2.3 def sparksee.ValueListIterator.__next__ (self)
Used in next().

Returns:
The next element

3.83 sparksee.Values Class Reference

Value set class.

Public Member Functions

• def is_closed
  Gets if Values instance has been closed or not.

• def count
  Gets the number of elements into the collection.

• def __iter__
  Gets a new ValuesIterator.

• def iterator
  Gets a ValuesIterator.

• def close
  Closes the Values instance.
3.83 sparksee.Values Class Reference

3.83.1 Detailed Description

Value set class.
This is a set of Value instances, that is there is no duplicated elements.
Use a ValuesIterator to traverse all the elements into the set.
When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

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

3.83.2 Member Function Documentation

3.83.2.1 def sparksee.Values.is_closed (self)
Gets if Values instance has been closed or not.

See also:
close()

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

3.83.2.2 def sparksee.Values.count (self)
Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

3.83.2.3 def sparksee.Values.__iter__ (self)
Gets a new ValuesIterator.

Returns:
ValuesIterator instance

3.83.2.4 def sparksee.Values.iterator (self, order)
Gets a ValuesIterator.

Parameters:
order [in] Ascending or descending order.

Returns:
ValuesIterator instance.
3.83.2.5  def sparksee.Values.close ( self)
Closes the Values instance.
It must be called to ensure the integrity of all data.

3.84  sparksee.ValuesIterator Class Reference

Values iterator class.

Public Member Functions

• def has_next
  
  Gets if there are more elements to traverse.

• def next
  
  Gets the next element to traverse.

• def is_closed
  
  Gets if ValuesIterator instance has been closed or not.

• def __next__
  
  Used in next().

• def close
  
  Closes the ValuesIterator instance.

3.84.1  Detailed Description

Values iterator class.
It allows for traversing all the elements into a Values instance.

Author:

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

3.84.2  Member Function Documentation

3.84.2.1  def sparksee.ValuesIterator.has_next ( self)
Gets if there are more elements to traverse.

Returns:

  TRUE if there are more elements to traverse, FALSE otherwise.
3.84.2.2  def sparksee.ValuesIterator.next ( self)
Gets the next element to traverse.

Returns:
   The next element.

3.84.2.3  def sparksee.ValuesIterator.is_closed ( self)
Gets if ValuesIterator instance has been closed or not.

See also:
   close()

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

3.84.2.4  def sparksee.ValuesIterator.__next__ ( self)
Used in next().

Returns:
   The next element

3.84.2.5  def sparksee.ValuesIterator.close ( self)
Closes the ValuesIterator instance.
It must be called to ensure the integrity of all data.

3.85  sparksee.WeakConnectivity Class Reference

WeakConnectivity class.
Inheritance diagram for sparksee.WeakConnectivity:

```
sparksee.WeakConnectivity

sparksee.WeakConnectivityDFS

sparksee.Connectivity
```
Collaboration diagram for sparksee.WeakConnectivity:

Public Member Functions

- `def add_edge_type`
  Allows connectivity through edges of the given type.

- `def add_all_node_types`
  Allows connectivity through all node types of the graph.

- `def add_all_edge_types`
  Allows connectivity through all edge types of the graph.

- `def run`
  Runs the algorithm in order to find the connected components.

- `def exclude_nodes`
  Set which nodes can’t be used.

- `def add_node_type`
  Allows connectivity through nodes of the given type.

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

- `def exclude_edges`
  Set which edges can’t be used.

- `def get_connected_components`
  Returns the results generated by the execution of the algorithm.

- `def is_closed`
  Gets if Connectivity has been closed or not.

- `def close`
  Closes the Connectivity instance.

3.85.1 Detailed Description

WeakConnectivity class.
Any class implementing this abstract class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one. It consists in finding components where every pair \((u,v)\) of nodes contained in it has a path from \(u\) to \(v\) and from \(v\) to \(u\).

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results. After the execution, we can retrieve the results stored in an instance of the \texttt{ConnectedComponents} class using the \texttt{getConnectedComponents} method.

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

\textbf{Author:}

Sparsity Technologies \url{http://www.sparsity-technologies.com}

\subsection{Member Function Documentation}

\subsubsection{def \texttt{add\_edge\_type} (self, type)}
Allows connectivity through edges of the given type.

\begin{verbatim}
Parameters:
  type [in] Edge type.
\end{verbatim}

Reimplemented in \texttt{WeakConnectivityDFS}.

\subsubsection{def \texttt{add\_all\_edge\_types} (self)}
Allows connectivity through all edge types of the graph.

\begin{verbatim}
Parameters:
  nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.
\end{verbatim}

Reimplemented in \texttt{WeakConnectivityDFS}.

\subsubsection{def \texttt{run} (self)}
Runs the algorithm in order to find the connected components.

\begin{verbatim}
Parameters:
  nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.
\end{verbatim}

This method can be called only once.

Reimplemented from \texttt{Connectivity}.

Reimplemented in \texttt{WeakConnectivityDFS}.

\subsubsection{def \texttt{exclude\_nodes} (self, nodes)}
Set which nodes can’t be used.

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.WeakConnectivityDFS.

3.85.2.5 def sparksee.WeakConnectivity.add_node_type (self, t)
Allows connectivity through nodes of the given type.

Parameters:

  \textit{t} null

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.WeakConnectivityDFS.

3.85.2.6 def sparksee.WeakConnectivity.set_materialized_attribute (self, 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:

  \textit{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.

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.WeakConnectivityDFS.

3.85.2.7 def sparksee.WeakConnectivity.exclude_edges (self, edges)
Set which edges can’t be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage
of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

Parameters:

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.WeakConnectivityDFS.

3.85.2.8 def sparksee.WeakConnectivity.get_connected_components (self)
Returns the results generated by the execution of the algorithm.
These results contain information related to the connected components found as the number of different
components, the set of nodes contained in each component or many other data.
Returns:

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

Reimplemented from sparksee.Connectivity.
Reimplemented in sparksee.WeakConnectivityDFS.

3.85.2.9 def sparksee.Connectivity.is_closed (self) [inherited]

Gets if Connectivity has been closed or not.

See also:

close()

Returns:

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

3.85.2.10 def sparksee.Connectivity.close (self) [inherited]

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

3.86 sparksee.WeakConnectivityDFS Class Reference

WeakConnectivityDFS class.

Inheritance diagram for sparksee.WeakConnectivityDFS:

Collaboration diagram for sparksee.WeakConnectivityDFS:
Public Member Functions

- def add_edge_type
  
  Allows connectivity through edges of the given type.

- def add_all_node_types
  
  Allows connectivity through all node types of the graph.

- def add_all_edge_types
  
  Allows connectivity through all edge types of the graph.

- def exclude_nodes
  
  Set which nodes can’t be used.

- def run
  
  Executes the algorithm.

- def add_node_type
  
  Allows connectivity through nodes of the given type.

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

- def exclude_edges
  
  Set which edges can’t be used.

- def __init__
  
  Creates a new instance of WeakConnectivityDFS.

- def get_connected_components
  
  Returns the results generated by the execution of the algorithm.

- def is_closed
  
  Gets if Connectivity has been closed or not.

- def close
  
  Closes the Connectivity instance.

3.86.1 Detailed Description

WeakConnectivityDFS class.

This class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair \((u, v)\) of nodes contained in it has a path from \(u\) to \(v\) and from \(v\) to \(u\). This implementation is based on the Depth-First Search (DFS) algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.
After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

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

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

3.86.2 Member Function Documentation

3.86.2.1 def sparksee.WeakConnectivityDFS.add_edge_type (self, type)
Allows connectivity through edges of the given type.
In a weak connectivity the edges can be used in Any direction.

Parameters:

  type [in] Edge type.

Reimplemented from sparksee.WeakConnectivity.

3.86.2.2 def sparksee.WeakConnectivityDFS.add_all_edge_types (self)
Allows connectivity through all edge types of the graph.
In a weak connectivity the edges can be used in Any direction.
Reimplemented from sparksee.WeakConnectivity.

3.86.2.3 def sparksee.WeakConnectivityDFS.exclude_nodes (self, nodes)
Set which nodes can’t be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it’s less efficient than not allowing a node type.

Parameters:

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

Reimplemented from sparksee.WeakConnectivity.

3.86.2.4 def sparksee.WeakConnectivityDFS.add_node_type (self, t)
Allows connectivity through nodes of the given type.

Parameters:

  t null

Reimplemented from sparksee.WeakConnectivity.
3.86.2.5 def sparksee.WeakConnectivityDFS.set_materialized_attribute ( self, 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.

Reimplemented from sparksee.WeakConnectivity.

3.86.2.6 def sparksee.WeakConnectivityDFS.exclude_edges ( self, edges)

Set which edges can’t be used.

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

Parameters:

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

Reimplemented from sparksee.WeakConnectivity.

3.86.2.7 def sparksee.WeakConnectivityDFS.__init__ ( self, session)

Creates a new instance of WeakConnectivityDFS.

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

Parameters:

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

3.86.2.8 def sparksee.WeakConnectivityDFS.get_connected_components ( self)

Returns the results generated by the execution of the algorithm.

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

Returns:

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

Reimplemented from sparksee.WeakConnectivity.
3.86.2.9  def sparksee.Connectivity.is_closed (self) [inherited]
Gets if Connectivity has been closed or not.

See also:
   close()

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

3.86.2.10  def sparksee.Connectivity.close (self) [inherited]
Closes the Connectivity instance.
It must be called to ensure the integrity of all data.
Index

__init__
sparksee::AttributeList, 11
sparksee::BooleanList, 17
sparksee::CommunitiesSCD, 20
sparksee::ConnectedComponents, 27
sparksee::Context, 35
sparksee::DisjointCommunities, 52
sparksee::EdgeTypeExporter, 64
sparksee::EdgeTypeLoader, 70
sparksee::Int32List, 97
sparksee::NodeTypeExporter, 106
sparksee::NodeTypeLoader, 110
sparksee::OIDList, 123
sparksee::ResultSetList, 134
sparksee::Sparksee, 160
sparksee::SparkseeConfig, 168
sparksee::StringList, 176
sparksee::Traversal, 190
sparksee::TraversalBFS, 195
sparksee::TraversalDFS, 199
sparksee::TypeListIterator, 208
sparksee::ValueListIterator, 227
sparksee::ValuesIterator, 230

__iter__
sparksee::AttributeList, 11
sparksee::BooleanList, 16
sparksee::Int32List, 97
sparksee::Objects, 118
sparksee::OIDList, 122
sparksee::ResultadoList, 134
sparksee::Traversal, 190
sparksee::TraversalBFS, 195
sparksee::TraversalDFS, 199
sparksee::TypeList, 207
sparksee::ValueList, 219
sparksee::ValueListIterator, 225
sparksee::ValuesIterator, 228

__next__
sparksee::AttributeList, 12
sparksee::BooleanList, 18
sparksee::Int32List, 98
sparksee::ObjectsIterator, 120
sparksee::OIDList, 124
sparksee::ResultSetListIterator, 135
sparksee::StringListIterator, 177
sparksee::Traversal, 190

add
sparksee::AttributeList, 11
sparksee::BooleanList, 16
sparksee::Int32List, 97
sparksee::Objects, 119
sparksee::OIDList, 122
sparksee::StringList, 176
sparksee::TypeList, 207
sparksee::ValueList, 226

add_all_edge_types
sparksee::CommunitiesSCD, 20
sparksee::Connection, 34
sparksee::DisjointCommunityDetection, 55
sparksee::ShortestPath, 145
sparksee::SinglePairShortestPath, 148
sparksee::SinglePairShortestPathBFS, 152
sparksee::SinglePairShortestPathDFS, 157
sparksee::StrongConnectivity, 181
sparksee::StrongConnectivityGabow, 184
sparksee::Traversal, 189
sparksee::TraversalBFS, 194
sparksee::TraversalDFS, 198
sparksee::WeakConnectivity, 232
sparksee::WeakConnectivityDFS, 236

add_edge_type
sparksee::CommunitiesSCD, 20
sparksee::Connection, 33
sparksee::DisjointCommunityDetection, 55
sparksee::ShortestPath, 144
sparksee::SinglePairShortestPath, 148
sparksee::SinglePairShortestPathBFS, 153
sparksee::SinglePairShortestPathDFS, 157
sparksee::StrongConnectivity, 180
sparksee::StrongConnectivityGabow, 184
sparksee::Traversal, 190
sparksee::TraversalBFS, 193
sparksee::TraversalDFS, 197
sparksee::WeakConnectivity, 232
sparksee::WeakConnectivityDFS, 236

add_node_type
sparksee::CommunitiesSCD, 20
sparksee::Connection, 23
sparksee::Connectivity, 31
sparksee::Context, 35
sparksee::DisjointCommunityDetection, 55
sparksee::ExportManager, 74
EQUAL
sparksee::Condition, 25
equals
sparksee::Objects, 116
sparksee::Value, 220
exclude_edges
sparksee::CommunitiesSCD, 21
sparksee::CommunityDetection, 24
sparksee::Connectivity, 31
sparksee::Context, 35
sparksee::DisjointCommunityDetection, 56
sparksee::ShortestPath, 144
sparksee::SinglePairShortestPath, 149
sparksee::SinglePairShortestPathBFS, 153
sparksee::SinglePairShortestPathDijkstra, 158
sparksee::StrongConnectivity, 180
sparksee::StrongConnectivityGabow, 184
sparksee::Traversal, 191
sparksee::TraversalBFS, 194
sparksee::TraversalDFS, 198
sparksee::WeakConnectivity, 233
sparksee::WeakConnectivityDFS, 237
exclude_nodes
sparksee::CommunitiesSCD, 20
sparksee::CommunityDetection, 23
sparksee::Connectivity, 30
sparksee::Context, 34
sparksee::DisjointCommunityDetection, 55
sparksee::ShortestPath, 144
sparksee::SinglePairShortestPath, 147
sparksee::SinglePairShortestPathBFS, 151
sparksee::SinglePairShortestPathDijkstra, 156
sparksee::StrongConnectivity, 179
sparksee::StrongConnectivityGabow, 183
sparksee::Traversal, 189
sparksee::TraversalBFS, 193
sparksee::TraversalDFS, 197
sparksee::WeakConnectivity, 232
sparksee::WeakConnectivityDFS, 236
execute
sparksee::Query, 128
exists
sparksee::Objects, 116
explode
sparksee::Graph, 89, 91
export
sparksee::Graph, 93
fetch
sparksee::QueryStream, 130
find_attribute
sparksee::Graph, 84
find_attributes
sparksee::Graph, 88
find_edge
sparksee::Graph, 91
find_edge_types
sparksee::Graph, 84
find_node_types
sparksee::Graph, 86
find_object
sparksee::Graph, 92
find_or_create_edge
sparksee::Graph, 90
find_or_create_object
sparksee::Graph, 86
find_type
sparksee::Graph, 94
find_types
sparksee::Graph, 80
FINE
sparksee::LogLevel, 100
fix_current_cache_max_size
sparksee::Database, 44
generate_schema_script
sparksee::ScriptParser, 140
get
sparksee::ResultSetList, 134
sparksee::SparkseeProperties, 174
sparksee::ValueList, 225
get_alias
sparksee::Database, 44
get_are_neighbors_indexed
sparksee::Type, 201
get_attribute
sparksee::Graph, 86, 88, 93
get_attribute_interval_count
sparksee::Graph, 81
get_attribute_statistics
sparksee::Graph, 83
get_attribute_text
sparksee::Graph, 83
get_attributes
sparksee::Graph, 82
get_available_mem
sparksee::PlatformStatistics, 126
get_avg_length_string
sparksee::AttributeStatistics, 15
get_boolean
sparksee::SparkseeProperties, 174
sparksee::Value, 219
get_cache
sparksee::DatabaseStatistics, 46
get_cache_max_size
sparksee::Database, 44
sparksee::SparkseeConfig, 172
get_cache_statistics_enabled
   sparksee::SparkseeConfig, 167
get_cache_statistics_file
   sparksee::SparkseeConfig, 168
get_cache_statistics_snapshot_time
   sparksee::SparkseeConfig, 168
get_color_rgb
   sparksee::EdgeExport, 60
   sparksee::NodeExport, 102
get_column
   sparksee::ResultSet, 131
get_column_data_type
   sparksee::ResultSet, 133
get_column_index
   sparksee::ResultSet, 132
get_column_name
   sparksee::ResultSet, 132
get_communities
   sparksee::CommunitiesSCD, 21
   sparksee::DisjointCommunityDetection, 56
get_community
   sparksee::DisjointCommunities, 53
get_connected_component
   sparksee::ConnectedComponents, 28
get_connected_components
   sparksee::Connectivity, 32
   sparksee::StrongConnectivity, 181
   sparksee::StrongConnectivityGabow, 185
   sparksee::WeakConnectivity, 234
   sparksee::WeakConnectivityDFS, 237
get_cost
   sparksee::SinglePairShortestPath, 148
   sparksee::SinglePairShortestPathBFS, 152
   sparksee::SinglePairShortestPathDijkstra, 157
get_count
   sparksee::Attribute, 8
   sparksee::ConnectedComponents, 28
   sparksee::DisjointCommunities, 52
   sparksee::ExporterEvent, 205
   sparksee::LoaderEvent, 214
get_current_depth
   sparksee::Traversal, 190
   sparksee::TraversalBFS, 194
   sparksee::TraversalDFS, 198
get_data
   sparksee::DatabaseStatistics, 46
get_data_type
   sparksee::Attribute, 9
   sparksee::Value, 220
get_distinct
   sparksee::AttributeStatistics, 14
get_double
   sparksee::Value, 222
get_edge
INDEX

sparksee::EdgeExport, 60

get_license
sparksee::SparkseeConfig, 169

get_log_file
sparksee::SparkseeConfig, 166

get_log_level
sparksee::SparkseeConfig, 169

get_long
sparksee::Value, 223

generate_max
sparksee::AttributeStatistics, 14

generate_max_length_string
sparksee::AttributeStatistics, 13

generate_median
sparksee::AttributeStatistics, 14

generate_min
sparksee::AttributeStatistics, 13

generate_min_length_string
sparksee::AttributeStatistics, 13

generate_mode
sparksee::AttributeStatistics, 14

generate_mode_count
sparksee::AttributeStatistics, 15

generate_name
sparksee::Attribute, 9
sparksee::Type, 202

generate_node
sparksee::DefaultExport, 50
sparksee::ExportManager, 74

generate_node_type
sparksee::DefaultExport, 50
sparksee::ExportManager, 73

generate_nodes
sparksee::ConnectedComponents, 29
sparksee::DisjointCommunities, 53

generate_null
sparksee::AttributeStatistics, 14

generate_num_c_p_us
sparksee::PlatformStatistics, 127

generate_num_columns
sparksee::ResultSet, 133

generate_num_objects
sparksee::Type, 201

generate_object_type
sparksee::Graph, 92
sparksee::Type, 201

generate_oid
sparksee::Value, 218

generate_partition
sparksee::TypeLoaderEvent, 214

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
is_session_attribute
sparksee::Attribute, 8

iterator
sparksee::AttributeList, 11
sparksee::BooleanList, 16
sparksee::Int32List, 97
sparksee::Objects, 118
sparksee::OIDList, 122
sparksee::ResultSetList, 134
sparksee::StringList, 176
sparksee::TypeList, 207
sparksee::ValueList, 226
sparksee::Values, 228

iterator_from_element
sparksee::Objects, 115

iterator_from_index
sparksee::Objects, 118

LESS_EQUAL
sparksee::Condition, 25

LESS_THAN
sparksee::Condition, 25

LIKE
sparksee::Condition, 26

LIKE_NO_CASE
sparksee::Condition, 26

load
sparksee::SparkseeProperties, 175

neighbors
sparksee::Graph, 82, 89

new_attribute
sparksee::Graph, 85, 92

new_edge
sparksee::Graph, 93, 94

new_edge_type
sparksee::Graph, 91

new_node
sparksee::Graph, 83

new_node_type
sparksee::Graph, 89

new_objects
sparksee::Session, 142

new_restricted_edge_type
sparksee::Graph, 95

new_session_attribute
sparksee::Graph, 87, 90

next
sparksee::AttributeListIterator, 12
sparksee::BooleanListIterator, 18
sparksee::Int32ListIterator, 98
sparksee::ObjectsIterator, 120
sparksee::OIDListIterator, 124
sparksee::ResultSet, 132

NOT_EQUAL
sparksee::Condition, 26

notify_event
sparksee::TypeExporterListener, 206
sparksee::TypeLoaderListener, 216

open
sparksee::CSVReader, 38
sparksee::CSVWriter, 42
sparksee::Sparksee, 160

operator
sparksee::Value, 222

call
sparksee::ScriptParser, 139

prepare
sparksee::DefaultExport, 51
sparksee::ExportManager, 74
sparksee::QueryStream, 130

read
sparksee::CSVReader, 39
sparksee::RowReader, 137
sparksee::TextStream, 186

REG_EXP
sparksee::Condition, 26

register
sparksee::EdgeTypeExporter, 65
sparksee::EdgeTypeLoader, 70
sparksee::NodeTypeExporter, 106
sparksee::NodeTypeLoader, 111
sparksee::TypeExporter, 204
sparksee::TypeLoader, 212

release
sparksee::ExportManager, 74

remove
sparksee::Objects, 118

remove_attribute
sparksee::Graph, 87

remove_type
sparksee::Graph, 95

rename_attribute
sparksee::Graph, 80

rename_type
sparksee::Graph, 82, 89

reset
INDEX

sparksee::CSVReader, 37
sparksee::RowReader, 137

restore
sparksee::Sparksee, 159

run
sparksee::CommunityDetection, 23
sparksee::Connectivity, 30
sparksee::DisjointCommunityDetection, 56
sparksee::EdgeTypeExporter, 65
sparksee::EdgeTypeLoader, 70
sparksee::NodeTypeExporter, 107
sparksee::NodeTypeLoader, 111
sparksee::ShortestPath, 143
sparksee::SinglePairShortestPath, 147
sparksee::StrongConnectivity, 179
sparksee::TypeExporter, 203
sparksee::TypeLoader, 211
sparksee::WeakConnectivity, 232

run_n_phases
sparksee::EdgeTypeLoader, 71
sparksee::NodeTypeLoader, 112
sparksee::TypeLoader, 210

run_two_phases
sparksee::EdgeTypeLoader, 69
sparksee::NodeTypeLoader, 110
sparksee::TypeLoader, 210

sample
sparksee::Objects, 117

select
sparksee::Graph, 81, 84, 85, 94

set
sparksee::OIDList, 122
set_as_directed
sparksee::EdgeExport, 60
set_attribute
sparksee::Graph, 88
set_attribute_default_value
sparksee::Graph, 87
set_attribute_positions
sparksee::EdgeTypeLoader, 69
sparksee::NodeTypeLoader, 111
sparksee::TypeLoader, 212
set_attribute_text
sparksee::Graph, 80
set_attributes
sparksee::EdgeTypeExporter, 66
sparksee::EdgeTypeLoader, 71
sparksee::NodeTypeExporter, 107
sparksee::NodeTypeLoader, 112
sparksee::TypeExporter, 204
sparksee::TypeLoader, 213
set_auto_quotes
sparksee::CSVWriter, 41

set_boolean
sparksee::Value, 218
set_boolean_void
sparksee::Value, 223
set_cache_max_size
sparksee::Database, 45
sparksee::SparkseeConfig, 170
set_cache_statistics_enabled
sparksee::SparkseeConfig, 166
set_cache_statistics_file
sparksee::SparkseeConfig, 168
set_cache_statistics_snapshot_time
sparksee::SparkseeConfig, 166
set_color_rgb
sparksee::EdgeExport, 60
sparksee::NodeExport, 102
set_double
sparksee::Value, 224
set_double_void
sparksee::Value, 220
set_dynamic
sparksee::Query, 127
set_error_log
sparksee::ScriptParser, 139
set_extent_pages
sparksee::SparkseeConfig, 165
set_extent_size
sparksee::SparkseeConfig, 171
set_fit
sparksee::NodeExport, 104
set_font_size
sparksee::EdgeExport, 61
sparksee::NodeExport, 104
set_forced_quotes
sparksee::CSVWriter, 42
set_frequency
sparksee::EdgeTypeExporter, 64
sparksee::EdgeTypeLoader, 68
sparksee::NodeTypeExporter, 106
sparksee::NodeTypeLoader, 109
sparksee::TypeExporter, 203
sparksee::TypeLoader, 211
set_graph
sparksee::EdgeTypeExporter, 65
sparksee::EdgeTypeLoader, 70
sparksee::NodeTypeExporter, 107
sparksee::NodeTypeLoader, 111
sparksee::TypeExporter, 204
sparksee::TypeLoader, 212
set_head_attribute
sparksee::EdgeTypeExporter, 64
sparksee::EdgeTypeLoader, 68
set_head_position
sparksee::EdgeTypeExporter, 65

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
set_materialized_attribute
  sparksee::CommunitiesSCD, 21
  sparksee::Connectivity, 31
  sparksee::DisjointCommunityDetection, 56
  sparksee::StrongConnectivity, 180
  sparksee::StrongConnectivityGabow, 184
  sparksee::WeakConnectivity, 233
  sparksee::WeakConnectivityDFS, 237

set_maximum_hops
  sparksee::Context, 35
  sparksee::ShortestPath, 144
  sparksee::SinglePairShortestPath, 147
  sparksee::SinglePairShortestPathBFS, 152
  sparksee::SinglePairShortestPathDijkstra, 156
  sparksee::Traversal, 189
  sparksee::TraversalBFS, 194
  sparksee::TraversalDFS, 198

set_multilines
  sparksee::CSVReader, 38

set_null
  sparksee::Value, 224

set_num_lines
  sparksee::CSVReader, 38

set_oid
  sparksee::Value, 224

set_oid_void
  sparksee::Value, 222

set_output_log
  sparksee::ScriptParser, 140

set_pool_cluster_size
  sparksee::SparkseeConfig, 166

set_pool_frame_size
  sparksee::SparkseeConfig, 168

set_pool_persistent_max_size
  sparksee::SparkseeConfig, 167

set_pool_persistent_min_size
  sparksee::SparkseeConfig, 172

set_pool_temporary_max_size
  sparksee::SparkseeConfig, 171

set_pool_temporary_min_size
  sparksee::SparkseeConfig, 169

set_quotes
  sparksee::CSVReader, 38
  sparksee::CSVWriter, 41

set_recovery_cache_max_size
  sparksee::SparkseeConfig, 170

set_recovery_checkpoint_time
  sparksee::SparkseeConfig, 172

set_recovery_enabled
  sparksee::SparkseeConfig, 171

set_recovery_log_file
  sparksee::SparkseeConfig, 173

set_rollback_enabled
  sparksee::SparkseeConfig, 171
set_row_reader
  sparksee::EdgeTypeLoader, 69
  sparksee::NodeTypeLoader, 110
  sparksee::TypeLoader, 212
set_row_writer
  sparksee::EdgeTypeExporter, 64
  sparksee::NodeTypeExporter, 106
  sparksee::TypeExporter, 203
set_separator
  sparksee::CSVReader, 38
  sparksee::CSVWriter, 41
set_shape
  sparksee::NodeExport, 104
set_start_line
  sparksee::CSVReader, 39
set_stream
  sparksee::Query, 128
set_string
  sparksee::Value, 221
set_string_void
  sparksee::Value, 219
set_tail_attribute
  sparksee::EdgeTypeExporter, 63
  sparksee::EdgeTypeLoader, 68
set_tail_position
  sparksee::EdgeTypeExporter, 64
  sparksee::EdgeTypeLoader, 69
set_timestamp
  sparksee::Value, 221, 222
set_timestamp_format
  sparksee::EdgeTypeLoader, 71
  sparksee::NodeTypeLoader, 112
  sparksee::TypeLoader, 213
set_timestamp_void
  sparksee::Value, 219, 221
set_type
  sparksee::EdgeTypeExporter, 64
  sparksee::EdgeTypeLoader, 69
  sparksee::NodeTypeExporter, 107
  sparksee::NodeTypeLoader, 110
  sparksee::TypeExporter, 204
  sparksee::TypeLoader, 211
set_unweighted_edge_cost
  sparksee::SinglePairShortestPathDijkstra, 156
set_void
  sparksee::Value, 220
set_width
  sparksee::EdgeExport, 61
  sparksee::NodeExport, 104
SEVERE
  sparksee::LogLevel, 99
sparksee::Attribute, 7
  get_count, 8
  get_data_type, 9
  get_id, 9
  get_kind, 8
  get_name, 9
  get_size, 8
  get_type_id, 8
  is_session_attribute, 8
sparksee::AttributeKind, 9
  UNIQUE, 10
sparksee::AttributeList, 10
  __init__, 11
  __iter__, 11
  add, 11
  count, 11
  iterator, 11
sparksee::AttributeListIterator, 11
  __next__, 12
  has_next, 12
  next, 12
sparksee::AttributeStatistics, 12
  get_avg_length_string, 15
  get_distinct, 14
  get_max, 14
  get_max_length_string, 15
  get_mean, 14
  get_median, 15
  get_min, 13
  get_min_length_string, 13
  get_mode, 14
  get_mode_count, 15
  get_null, 14
  get_total, 15
  get_variance, 14
sparksee::BooleanList, 16
  __init__, 17
  __iter__, 16
  add, 16
  count, 17
  iterator, 16
sparksee::BooleanListIterator, 17
  __next__, 18
  has_next, 18
  next, 18
sparksee::CommunitiesSCD, 18
  __init__, 20
  add_all_edge_types, 20
  add_edge_type, 20
  add_node_type, 20
  close, 22
  exclude_edges, 21
  exclude_nodes, 20
  get_communities, 21
  is_closed, 22
  set_look_ahead, 21
  set_materialized_attribute, 21

Generated on Tue Jun 2 16:12:16 2015 for SparkseePython by Doxygen
new_node, 83
new_node_type, 89
new_restricted_edge_type, 95
new_session_attribute, 87, 90
remove_attribute, 87
remove_type, 95
rename_attribute, 80
rename_type, 82, 89
select, 81, 84, 85, 94
set_attribute, 88
set_attribute_default_value, 87
set_attribute_text, 80
tails, 90
tails_and_heads, 81
sparksee::GraphExport, 95
get_label, 96
set_label, 96
sparksee::Int32List, 96
__init__, 97
__iter__, 97
add, 97
count, 97
iterator, 97
sparksee::Int32ListIterator, 98
__next__, 98
has_next, 98
next, 98
sparksee::LogLevel, 99
CONFIG, 100
DEBUG, 100
FINE, 100
INFO, 100
SEVERE, 99
WARNING, 99
sparksee::NodeExport, 100
get_color_rgb, 102
get_font_size, 102
get_height, 103
get_label, 103
get_labelcolor_rgb, 103
get_shape, 102
get_width, 103
is_fit, 102
set_color_rgb, 102
set_fit, 104
set_font_size, 104
set_height, 102
set_label, 103
set_labelcolor_rgb, 103
set_shape, 104
set_width, 104
sparksee::NodeShape, 104
sparksee::NodeTypeExporter, 105
__init__, 106
register, 106
run, 107
set_attributes, 107
set_frequency, 106
set_graph, 107
set_header, 107
set_row_writer, 106
set_type, 107
sparksee:: NodeTypeLoader, 108
__init__, 110
register, 111
run, 111
run_n_phases, 112
run_two_phases, 110
set_attribute_positions, 111
set_attributes, 112
set_frequency, 109
set_graph, 111
set_locale, 111
set_log_error, 110
set_log_off, 109
set_row_reader, 110
set_timestamp_format, 112
set_type, 110
sparksee::Objects, 112
__iter__, 118
add, 119
any, 114
close, 117
combine_difference, 116
combine_intersection, 117
combine_union, 115
contains, 117
copy, 116, 119
count, 117
difference, 115
equals, 116
exists, 116
intersection, 118
is_closed, 115
iterator, 118
iterator_from_element, 115
iterator_from_index, 118
remove, 118
sample, 117
union, 119
sparksee::ObjectsIterator, 119
__next__, 120
close, 121
has_next, 120
is_closed, 120
next, 120
sparksee::ObjectType, 121
sparksee::OIDList, 121
__init__, 123
__iter__, 122
add, 122
count, 123
iterator, 122
set, 122
sparksee::OIDListIterator, 123
__next__, 124
has_next, 124
next, 124
sparksee::Order, 124
sparksee::Platform, 125
get_statistics, 125
sparksee::PlatformStatistics, 125
generate_available_mem, 126
generate_num_c_p_us, 127
generate_real_time, 126
generate_system_time, 126
generate_total_mem, 126
generate_user_time, 126
sparksee::Query, 127
close, 128
execute, 128
is_closed, 128
set_dynamic, 127
set_stream, 128
sparksee::QueryContext, 128
sparksee::QueryStream, 129
fetch, 130
prepare, 130
start, 130
sparksee::ResultSet, 130
generate_column, 131
generate_column_data_type, 133
generate_column_index, 132
generate_column_name, 132
generate_j_s_o_n, 132
generate_num_columns, 133
is_closed, 132
next, 132
sparksee::ResultSetList, 133
__init__, 134
_adapter__, 134
_count__, 134
count, 134
generate, 134
iterator, 134
sparksee::ResultSetListIterator, 135
__next__, 135
has_next, 135
next, 135
sparksee::RowReader, 136
close, 136
generate_row, 136
read, 137
reset, 137
sparksee::RowWriter, 137
close, 138
write, 138
sparksee::ScriptParser, 138
generate_schema_script, 140
parse, 139
set_error_log, 139
set_output_log, 140
sparksee::Session, 140
close, 142
get_graph, 142
is_closed, 141
new_objects, 142
sparksee::ShortestPath, 142
generate_all_edge_types, 145
generate_edge_type, 144
generate_node_type, 143
close, 145
exclude_edges, 144
exclude_nodes, 144
is_closed, 144
run, 143
set_maximum_hops, 144
sparksee::SinglePairShortestPath, 145
generate_all_edge_types, 148
generate_edge_type, 148
generate_node_type, 147
close, 149
exclude_edges, 149
exclude_nodes, 147
get_cost, 148
get_path_as_edges, 147
get_path_as_nodes, 148
is_closed, 149
run, 147
set_maximum_hops, 147
sparksee::SinglePairShortestPathBFS, 149
__init__, 152
_adapter__, 152
_add_edge_types, 152
add_edge_type, 153
add_node_type, 151
check_only_existence, 153
close, 153
exclude_edges, 153
exclude_nodes, 151
get_cost, 152
get_path_as_edges, 151
get_path_as_nodes, 152
is_closed, 153
set_maximum_hops, 152
sparksee::SinglePairShortestPathDijkstra, 154
__init__, 157
_adapter__, 157
_add_all_edge_types, 157
sparksee::TextStream, 185
  __init__, 187
close, 187
is_null, 186
read, 186
write, 187
sparksee::Traversal, 187
  __iter__, 190
  __next__, 190
  add_all_edge_types, 189
add_edge_type, 190
add_node_type, 189
close, 190
exclude_edges, 191
exclude_nodes, 189
get_current_depth, 190
has_next, 189
is_closed, 189
next, 190
set_maximum_hops, 189
sparksee::TraversalBFS, 191
  __init__, 193
  __iter__, 195
  __next__, 195
  add_all_edge_types, 194
add_edge_type, 193
add_node_type, 193
close, 195
exclude_edges, 194
exclude_nodes, 193
get_current_depth, 194
has_next, 193
is_closed, 194
next, 193
set_maximum_hops, 194
sparksee::TraversalDFS, 195
  __init__, 197
  __iter__, 199
  __next__, 199
  add_all_edge_types, 198
add_edge_type, 197
add_node_type, 197
close, 199
exclude_edges, 198
exclude_nodes, 197
get_current_depth, 198
has_next, 197
is_closed, 199
next, 198
set_maximum_hops, 198
sparksee::Type, 199
get_are_neighbors_indexed, 201
get_id, 201
get_is_directed, 202
get_is_restricted, 201
get_name, 202
get_num_objects, 201
get_object_type, 201
get_restricted_from, 201
get_restricted_to, 201
sparksee::TypeExporter, 202
  register, 204
run, 203
set_attributes, 204
set_frequency, 203
set_graph, 204
set_header, 204
set_row_writer, 203
set_type, 204
sparksee::TypeExporterEvent, 205
  get_count, 205
get_total, 205
get_type_id, 205
is_last, 205
sparksee::TypeExporterListener, 206
  notify_event, 206
sparksee::TypeList, 206
  __init__, 207
  __iter__, 207
  add, 207
next, 208
set_maximum_hops, 209
sparksee::TypeLoader, 209
  register, 212
run, 211
run_n_phases, 210
run_two_phases, 210
set_attribute_positions, 212
set_attributes, 213
set_frequency, 211
set_graph, 212
set_locale, 212
set_log_error, 211
set_log_off, 211
set_row_reader, 212
set_timestamp_format, 213
set_type, 211
sparksee::TypeLoaderEvent, 213
  get_count, 214
get_partition, 214
get_phase, 215
get_total_partition_steps, 214
get_total_partitions, 215
get_total_phases, 214
get_type_id, 215
is_last, 214
sparksee::TypeLoaderListener, 215
notify_event, 216
sparksee::Value, 216
__init__, 219
compare, 223
equals, 220
get_boolean, 219
get_data_type, 220
get_double, 222
get_integer, 224
get_long, 223
get_oid, 218
get_string, 220
get_timestamp, 223
is_null, 220
operator, 222
set_boolean, 218
set_boolean_void, 223
set_double, 224
set_double_void, 220
set_integer, 221
set_integer_void, 219
set_long, 223
set_long_void, 218
set_null, 224
set_oid, 244
set_oid_void, 222
set_string, 221
set_string_void, 219
set_timestamp, 221, 222
set_timestamp_void, 219, 221
set_void, 220
to_string, 219
sparksee::ValueList, 224
__init__, 225
__iter__, 225
add, 226
count, 226
get, 225
iterator, 226
sparksee::ValueListIterator, 226
__next__, 227
has_next, 227
next, 227
sparksee::Values, 227
__iter__, 228
close, 229
count, 228
is_closed, 228
iterator, 228
sparksee::ValuesIterator, 229
__next__, 230
close, 230
has_next, 230
is_closed, 230
next, 230
sparksee::WeakConnectivity, 230
add_all_edge_types, 232
add_edge_type, 232
add_node_type, 233
close, 234
exclude_edges, 233
exclude_nodes, 232
get_connected_components, 234
is_closed, 234
run, 232
set_materialized_attribute, 233
sparksee::WeakConnectivityDFS, 234
__init__, 237
add_all_edge_types, 236
add_edge_type, 236
add_node_type, 236
close, 238
exclude_edges, 237
exclude_nodes, 236
get_connected_components, 237
is_closed, 238
start
sparksee::QueryStream, 130
STRING
sparksee::DataType, 48
tails
sparksee::Graph, 90
tails_and_heads
sparksee::Graph, 81
TEXT
sparksee::DataType, 48
TIMESTAMP
sparksee::DataType, 48
to_string
to_string
sparksee::Value, 219
union
sparksee::Objects, 119
UNIQUE
sparksee::AttributeKind, 10
WARNING
sparksee::LogLevel, 99
write
sparksee::CSVWriter, 42
sparksee::RowWriter, 138
sparksee::TextStream, 187
YGRAPHML
sparksee::ExportType, 75