Contents

1 Module Index ......................................................... 1
  1.1 Modules ......................................................... 1

2 Class Index .......................................................... 1
  2.1 Class Hierarchy .................................................. 1

3 Class Index .......................................................... 4
  3.1 Class List ......................................................... 4

4 Module Documentation ............................................... 7
  4.1 GDB ................................................................. 7
    4.1.1 Detailed Description ......................................... 15
    4.1.2 Typedef Documentation ....................................... 15
    4.1.3 Enumeration Type Documentation ............................ 15
    4.1.4 Function Documentation ..................................... 19
  4.2 IO ................................................................. 20
    4.2.1 Detailed Description .......................................... 22
  4.3 Script ............................................................ 23
    4.3.1 Detailed Description .......................................... 23
    4.3.2 Function Documentation ..................................... 23
  4.4 Algorithms ....................................................... 24

5 Class Documentation ............................................... 26
  5.1 AppError Class Reference ....................................... 26
    5.1.1 Detailed Description .......................................... 27
    5.1.2 Constructor & Destructor Documentation .................... 27
    5.1.3 Member Function Documentation .............................. 27
  5.2 Attribute Class Reference ....................................... 28
    5.2.1 Detailed Description .......................................... 29
    5.2.2 Member Function Documentation .............................. 29
  5.3 AttributeList Class Reference .................................. 30
    5.3.1 Detailed Description .......................................... 31
    5.3.2 Constructor & Destructor Documentation .................... 31
    5.3.3 Member Function Documentation .............................. 31
  5.4 AttributeListIterator Class Reference .......................... 32
    5.4.1 Detailed Description .......................................... 32
    5.4.2 Member Function Documentation .............................. 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Class Reference</th>
<th>Detailed Description</th>
<th>Constructor &amp; Destructor Documentation</th>
<th>Member Function Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>AttributeStatistics Class Reference</td>
<td>33</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Detailed Description</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.2</td>
<td>Member Function Documentation</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>BooleanList Class Reference</td>
<td>37</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Detailed Description</td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6.3</td>
<td>Member Function Documentation</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>BooleanListIterator Class Reference</td>
<td>38</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Detailed Description</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.7.2</td>
<td>Member Function Documentation</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>ConnectedComponents Class Reference</td>
<td>39</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Detailed Description</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.8.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.8.3</td>
<td>Member Function Documentation</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Connectivity Class Reference</td>
<td>40</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>5.9.1</td>
<td>Detailed Description</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.9.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.9.3</td>
<td>Member Function Documentation</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>Context Class Reference</td>
<td>42</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>5.10.1</td>
<td>Detailed Description</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.10.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.10.3</td>
<td>Member Function Documentation</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>CSVReader Class Reference</td>
<td>47</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>5.11.1</td>
<td>Detailed Description</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.11.2</td>
<td>Member Function Documentation</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.12</td>
<td>CSVWriter Class Reference</td>
<td>51</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>5.12.1</td>
<td>Detailed Description</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.12.2</td>
<td>Member Function Documentation</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.13</td>
<td>Database Class Reference</td>
<td>55</td>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td>5.13.1</td>
<td>Detailed Description</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.13.2</td>
<td>Member Function Documentation</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.14</td>
<td>DatabaseStatistics Class Reference</td>
<td>58</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>5.14.1</td>
<td>Detailed Description</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.14.2</td>
<td>Member Function Documentation</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.15</td>
<td>DefaultExport Class Reference</td>
<td>60</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>5.15.1</td>
<td>Detailed Description</td>
<td>62</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.15.2 Member Function Documentation</td>
<td>63</td>
</tr>
<tr>
<td>5.16 Dex Class Reference</td>
<td>65</td>
</tr>
<tr>
<td>5.16.1 Detailed Description</td>
<td>66</td>
</tr>
<tr>
<td>5.16.2 Constructor &amp; Destructor Documentation</td>
<td>66</td>
</tr>
<tr>
<td>5.16.3 Member Function Documentation</td>
<td>66</td>
</tr>
<tr>
<td>5.17 DexConfig Class Reference</td>
<td>67</td>
</tr>
<tr>
<td>5.17.1 Detailed Description</td>
<td>70</td>
</tr>
<tr>
<td>5.17.2 Constructor &amp; Destructor Documentation</td>
<td>71</td>
</tr>
<tr>
<td>5.17.3 Member Function Documentation</td>
<td>71</td>
</tr>
<tr>
<td>5.18 DexProperties Class Reference</td>
<td>79</td>
</tr>
<tr>
<td>5.18.1 Detailed Description</td>
<td>79</td>
</tr>
<tr>
<td>5.18.2 Member Function Documentation</td>
<td>80</td>
</tr>
<tr>
<td>5.19 EdgeData Class Reference</td>
<td>81</td>
</tr>
<tr>
<td>5.19.1 Detailed Description</td>
<td>81</td>
</tr>
<tr>
<td>5.19.2 Member Function Documentation</td>
<td>82</td>
</tr>
<tr>
<td>5.20 EdgeExport Class Reference</td>
<td>82</td>
</tr>
<tr>
<td>5.20.1 Detailed Description</td>
<td>83</td>
</tr>
<tr>
<td>5.20.2 Member Function Documentation</td>
<td>84</td>
</tr>
<tr>
<td>5.21 EdgeTypeExporter Class Reference</td>
<td>86</td>
</tr>
<tr>
<td>5.21.1 Detailed Description</td>
<td>87</td>
</tr>
<tr>
<td>5.21.2 Constructor &amp; Destructor Documentation</td>
<td>88</td>
</tr>
<tr>
<td>5.21.3 Member Function Documentation</td>
<td>88</td>
</tr>
<tr>
<td>5.22 EdgeTypeLoader Class Reference</td>
<td>90</td>
</tr>
<tr>
<td>5.22.1 Detailed Description</td>
<td>93</td>
</tr>
<tr>
<td>5.22.2 Member Enumeration Documentation</td>
<td>93</td>
</tr>
<tr>
<td>5.22.3 Constructor &amp; Destructor Documentation</td>
<td>93</td>
</tr>
<tr>
<td>5.22.4 Member Function Documentation</td>
<td>94</td>
</tr>
<tr>
<td>5.23 Error Class Reference</td>
<td>97</td>
</tr>
<tr>
<td>5.23.1 Detailed Description</td>
<td>98</td>
</tr>
<tr>
<td>5.23.2 Constructor &amp; Destructor Documentation</td>
<td>98</td>
</tr>
<tr>
<td>5.23.3 Member Function Documentation</td>
<td>98</td>
</tr>
<tr>
<td>5.24 Exception Class Reference</td>
<td>99</td>
</tr>
<tr>
<td>5.24.1 Detailed Description</td>
<td>100</td>
</tr>
<tr>
<td>5.24.2 Constructor &amp; Destructor Documentation</td>
<td>100</td>
</tr>
<tr>
<td>5.24.3 Member Function Documentation</td>
<td>100</td>
</tr>
<tr>
<td>5.25 ExportManager Class Reference</td>
<td>101</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>5.25.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.25.2</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.26</td>
<td>FileNotFoundException Class Reference</td>
</tr>
<tr>
<td>5.26.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.26.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>5.26.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.27</td>
<td>Graph Class Reference</td>
</tr>
<tr>
<td>5.27.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.27.2</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.28</td>
<td>GraphExport Class Reference</td>
</tr>
<tr>
<td>5.28.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.28.2</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.29</td>
<td>Handler&lt; T &gt; Class Template Reference</td>
</tr>
<tr>
<td>5.29.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.30</td>
<td>Int32List Class Reference</td>
</tr>
<tr>
<td>5.30.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.30.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>5.30.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.31</td>
<td>Int32ListIterator Class Reference</td>
</tr>
<tr>
<td>5.31.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.31.2</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.32</td>
<td>IOError Class Reference</td>
</tr>
<tr>
<td>5.32.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.32.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>5.32.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.33</td>
<td>IOException Class Reference</td>
</tr>
<tr>
<td>5.33.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.33.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>5.33.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.34</td>
<td>LicenseError Class Reference</td>
</tr>
<tr>
<td>5.34.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.34.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>5.34.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>5.35</td>
<td>NodeExport Class Reference</td>
</tr>
<tr>
<td>5.35.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>5.35.2</td>
<td>Member Function Documentation</td>
</tr>
</tbody>
</table>
5.36 NodeTypeExporter Class Reference .................................................. 140
  5.36.1 Detailed Description ............................................................... 142
  5.36.2 Constructor & Destructor Documentation ................................... 142
  5.36.3 Member Function Documentation .............................................. 143
5.37 NodeTypeLoader Class Reference ................................................... 145
  5.37.1 Detailed Description ............................................................... 147
  5.37.2 Member Enumeration Documentation ........................................ 148
  5.37.3 Constructor & Destructor Documentation ................................... 148
  5.37.4 Member Function Documentation .............................................. 148
5.38 NoSuchElementException Class Reference ......................................... 151
  5.38.1 Detailed Description ............................................................... 152
  5.38.2 Constructor & Destructor Documentation ................................... 152
  5.38.3 Member Function Documentation .............................................. 153
5.39 Objects Class Reference .............................................................. 153
  5.39.1 Detailed Description ............................................................... 156
  5.39.2 Member Function Documentation .............................................. 156
5.40 ObjectsIterator Class Reference ................................................... 161
  5.40.1 Detailed Description ............................................................... 162
  5.40.2 Member Function Documentation .............................................. 162
5.41 OIDList Class Reference .............................................................. 162
  5.41.1 Detailed Description ............................................................... 163
  5.41.2 Constructor & Destructor Documentation ................................... 163
  5.41.3 Member Function Documentation .............................................. 163
5.42 OIDListIterator Class Reference ................................................... 164
  5.42.1 Detailed Description ............................................................... 165
  5.42.2 Member Function Documentation .............................................. 165
5.43 Platform Class Reference ............................................................ 165
  5.43.1 Detailed Description ............................................................... 165
  5.43.2 Member Function Documentation .............................................. 166
5.44 PlatformStatistics Class Reference ................................................. 166
  5.44.1 Detailed Description ............................................................... 166
  5.44.2 Member Function Documentation .............................................. 167
5.45 RowReader Class Reference .......................................................... 168
  5.45.1 Detailed Description ............................................................... 168
  5.45.2 Constructor & Destructor Documentation ................................... 169
  5.45.3 Member Function Documentation .............................................. 169
<table>
<thead>
<tr>
<th>Section</th>
<th>Class Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.46</td>
<td>RowWriter Class Reference</td>
<td>170</td>
</tr>
<tr>
<td>5.46.1</td>
<td>Detailed Description</td>
<td>170</td>
</tr>
<tr>
<td>5.46.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>171</td>
</tr>
<tr>
<td>5.46.3</td>
<td>Member Function Documentation</td>
<td>171</td>
</tr>
<tr>
<td>5.47</td>
<td>ScriptParser Class Reference</td>
<td>171</td>
</tr>
<tr>
<td>5.47.1</td>
<td>Detailed Description</td>
<td>172</td>
</tr>
<tr>
<td>5.47.2</td>
<td>Member Function Documentation</td>
<td>172</td>
</tr>
<tr>
<td>5.48</td>
<td>Session Class Reference</td>
<td>174</td>
</tr>
<tr>
<td>5.48.1</td>
<td>Detailed Description</td>
<td>176</td>
</tr>
<tr>
<td>5.48.2</td>
<td>Member Function Documentation</td>
<td>176</td>
</tr>
<tr>
<td>5.49</td>
<td>ShortestPath Class Reference</td>
<td>177</td>
</tr>
<tr>
<td>5.49.1</td>
<td>Detailed Description</td>
<td>179</td>
</tr>
<tr>
<td>5.49.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>179</td>
</tr>
<tr>
<td>5.49.3</td>
<td>Member Function Documentation</td>
<td>179</td>
</tr>
<tr>
<td>5.49.4</td>
<td>Member Data Documentation</td>
<td>180</td>
</tr>
<tr>
<td>5.50</td>
<td>SinglePairShortestPath Class Reference</td>
<td>180</td>
</tr>
<tr>
<td>5.50.1</td>
<td>Detailed Description</td>
<td>183</td>
</tr>
<tr>
<td>5.50.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>184</td>
</tr>
<tr>
<td>5.50.3</td>
<td>Member Function Documentation</td>
<td>184</td>
</tr>
<tr>
<td>5.50.4</td>
<td>Member Data Documentation</td>
<td>186</td>
</tr>
<tr>
<td>5.51</td>
<td>SinglePairShortestPathBFS Class Reference</td>
<td>186</td>
</tr>
<tr>
<td>5.51.1</td>
<td>Detailed Description</td>
<td>189</td>
</tr>
<tr>
<td>5.51.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>189</td>
</tr>
<tr>
<td>5.51.3</td>
<td>Member Function Documentation</td>
<td>189</td>
</tr>
<tr>
<td>5.51.4</td>
<td>Member Data Documentation</td>
<td>191</td>
</tr>
<tr>
<td>5.52</td>
<td>SinglePairShortestPathDijkstra Class Reference</td>
<td>191</td>
</tr>
<tr>
<td>5.52.1</td>
<td>Detailed Description</td>
<td>195</td>
</tr>
<tr>
<td>5.52.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>195</td>
</tr>
<tr>
<td>5.52.3</td>
<td>Member Function Documentation</td>
<td>195</td>
</tr>
<tr>
<td>5.52.4</td>
<td>Member Data Documentation</td>
<td>197</td>
</tr>
<tr>
<td>5.53</td>
<td>SQLException Class Reference</td>
<td>198</td>
</tr>
<tr>
<td>5.53.1</td>
<td>Detailed Description</td>
<td>199</td>
</tr>
<tr>
<td>5.53.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>199</td>
</tr>
<tr>
<td>5.53.3</td>
<td>Member Function Documentation</td>
<td>199</td>
</tr>
<tr>
<td>5.54</td>
<td>StringList Class Reference</td>
<td>200</td>
</tr>
<tr>
<td>5.54.1</td>
<td>Detailed Description</td>
<td>200</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.54.2 Constructor &amp; Destructor Documentation</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>5.54.3 Member Function Documentation</td>
<td>201</td>
<td></td>
</tr>
<tr>
<td>5.55 StringListIterator Class Reference</td>
<td>201</td>
<td></td>
</tr>
<tr>
<td>5.55.1 Detailed Description</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>5.55.2 Member Function Documentation</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>5.56 StrongConnectivity Class Reference</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>5.56.1 Detailed Description</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>5.56.2 Constructor &amp; Destructor Documentation</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>5.56.3 Member Function Documentation</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>5.57 StrongConnectivityGabow Class Reference</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>5.57.1 Detailed Description</td>
<td>211</td>
<td></td>
</tr>
<tr>
<td>5.57.2 Constructor &amp; Destructor Documentation</td>
<td>212</td>
<td></td>
</tr>
<tr>
<td>5.57.3 Member Function Documentation</td>
<td>212</td>
<td></td>
</tr>
<tr>
<td>5.58 SystemError Class Reference</td>
<td>213</td>
<td></td>
</tr>
<tr>
<td>5.58.1 Detailed Description</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>5.58.2 Constructor &amp; Destructor Documentation</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>5.58.3 Member Function Documentation</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>5.59 TextStream Class Reference</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>5.59.1 Detailed Description</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>5.59.2 Constructor &amp; Destructor Documentation</td>
<td>218</td>
<td></td>
</tr>
<tr>
<td>5.59.3 Member Function Documentation</td>
<td>218</td>
<td></td>
</tr>
<tr>
<td>5.60 Traversal Class Reference</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>5.60.1 Detailed Description</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>5.60.2 Constructor &amp; Destructor Documentation</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>5.60.3 Member Function Documentation</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>5.61 TraversalBFS Class Reference</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>5.61.1 Detailed Description</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>5.61.2 Constructor &amp; Destructor Documentation</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>5.61.3 Member Function Documentation</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>5.62 TraversalDFS Class Reference</td>
<td>227</td>
<td></td>
</tr>
<tr>
<td>5.62.1 Detailed Description</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>5.62.2 Constructor &amp; Destructor Documentation</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>5.62.3 Member Function Documentation</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>5.63 Type Class Reference</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>5.63.1 Detailed Description</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>5.63.2 Member Function Documentation</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.64</td>
<td>TypeExporter Class Reference</td>
<td>235</td>
</tr>
<tr>
<td>5.64.1</td>
<td>Detailed Description</td>
<td>236</td>
</tr>
<tr>
<td>5.64.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>236</td>
</tr>
<tr>
<td>5.64.3</td>
<td>Member Function Documentation</td>
<td>237</td>
</tr>
<tr>
<td>5.65</td>
<td>TypeExporterEvent Class Reference</td>
<td>239</td>
</tr>
<tr>
<td>5.65.1</td>
<td>Detailed Description</td>
<td>240</td>
</tr>
<tr>
<td>5.65.2</td>
<td>Member Function Documentation</td>
<td>240</td>
</tr>
<tr>
<td>5.66</td>
<td>TypeExporterListener Class Reference</td>
<td>241</td>
</tr>
<tr>
<td>5.66.1</td>
<td>Detailed Description</td>
<td>241</td>
</tr>
<tr>
<td>5.66.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>241</td>
</tr>
<tr>
<td>5.66.3</td>
<td>Member Function Documentation</td>
<td>241</td>
</tr>
<tr>
<td>5.67</td>
<td>TypeList Class Reference</td>
<td>242</td>
</tr>
<tr>
<td>5.67.1</td>
<td>Detailed Description</td>
<td>242</td>
</tr>
<tr>
<td>5.67.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>242</td>
</tr>
<tr>
<td>5.67.3</td>
<td>Member Function Documentation</td>
<td>242</td>
</tr>
<tr>
<td>5.68</td>
<td>TypeListIterator Class Reference</td>
<td>243</td>
</tr>
<tr>
<td>5.68.1</td>
<td>Detailed Description</td>
<td>244</td>
</tr>
<tr>
<td>5.68.2</td>
<td>Member Function Documentation</td>
<td>244</td>
</tr>
<tr>
<td>5.69</td>
<td>TypeLoader Class Reference</td>
<td>244</td>
</tr>
<tr>
<td>5.69.1</td>
<td>Detailed Description</td>
<td>246</td>
</tr>
<tr>
<td>5.69.2</td>
<td>Member Enumeration Documentation</td>
<td>247</td>
</tr>
<tr>
<td>5.69.3</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>247</td>
</tr>
<tr>
<td>5.69.4</td>
<td>Member Function Documentation</td>
<td>247</td>
</tr>
<tr>
<td>5.70</td>
<td>TypeLoaderEvent Class Reference</td>
<td>251</td>
</tr>
<tr>
<td>5.70.1</td>
<td>Detailed Description</td>
<td>252</td>
</tr>
<tr>
<td>5.70.2</td>
<td>Member Function Documentation</td>
<td>252</td>
</tr>
<tr>
<td>5.71</td>
<td>TypeLoaderListener Class Reference</td>
<td>253</td>
</tr>
<tr>
<td>5.71.1</td>
<td>Detailed Description</td>
<td>254</td>
</tr>
<tr>
<td>5.71.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>254</td>
</tr>
<tr>
<td>5.71.3</td>
<td>Member Function Documentation</td>
<td>254</td>
</tr>
<tr>
<td>5.72</td>
<td>UnsupportedOperationError Class Reference</td>
<td>254</td>
</tr>
<tr>
<td>5.72.1</td>
<td>Detailed Description</td>
<td>255</td>
</tr>
<tr>
<td>5.72.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>256</td>
</tr>
<tr>
<td>5.72.3</td>
<td>Member Function Documentation</td>
<td>256</td>
</tr>
<tr>
<td>5.73</td>
<td>Value Class Reference</td>
<td>256</td>
</tr>
<tr>
<td>5.73.1</td>
<td>Detailed Description</td>
<td>260</td>
</tr>
</tbody>
</table>
1 Module Index

1.1 Modules

Here is a list of all modules:

GDB 7
IO 20
Script 23
Algorithms 24

2 Class Index

2.1 Class Hierarchy

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

Attribute 28
2.1 Class Hierarchy

AttributeList 30
AttributeListIterator 32
AttributeStatistics 33
BooleanList 37
BooleanListIterator 38
ConnectedComponents 39
Connectivity 42
  StrongConnectivity 202
    StrongConnectivityGabow 208
  WeakConnectivity 271
    WeakConnectivityDFS 277
Context 47
DatabaseStatistics 60
DexConfig 67
DexProperties 79
EdgeData 81
EdgeExport 82
Exception 99
  Error 97
    AppError 26
      LicenseError 134
    UnsupportedOperationError 254
    WrongArgumentError 283
SystemError 213
  IOException 130
IOException 132
  FileNotFoundException 104
NoSuchElementException 151
SQLException 198
2.1 Class Hierarchy

- RowReader 168
- CSVReader 51
- RowWriter 170
- CSVWriter 55
- ScriptParser 171
- ShortestPath 177
  - SinglePairShortestPath 180
    - SinglePairShortestPathBFS 186
    - SinglePairShortestPathDijkstra 191
- StringList 200
- StringListIterator 201
- Traversal 219
  - TraversalBFS 223
  - TraversalDFS 227
- Type 232
- TypeExporter 235
  - EdgeTypeExporter 86
  - NodeTypeExporter 140
- TypeExporterEvent 239
- TypeExporterListener 241
- TypeList 242
- TypeListIterator 243
- TypeLoader 244
  - EdgeTypeLoader 90
  - NodeTypeLoader 145
- TypeLoaderEvent 251
- TypeLoaderListener 253
3 Class Index

3.1 Class List

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

- **AppError** (Application error class) at page 26
- **Attribute** (Attribute data class) at page 28
- **AttributeList** (Dex attribute identifier list) at page 30
- **AttributeListIterator** (AttributeList iterator class) at page 32
- **AttributeStatistics** (Attribute statistics class) at page 33
- **BooleanList** (Boolean list) at page 37
- **BooleanListIterator** (BooleanList iterator class) at page 38
- **ConnectedComponents** (ConnectedComponents class) at page 39
- **Connectivity** (Connectivity class) at page 42
- **Context** (Context class) at page 47
- **CSVReader** (CSVReader interface) at page 51
- **CSVWriter** (CSVWriter interface) at page 55
- **Database** (Database class) at page 58
- **DatabaseStatistics** (Database statistics) at page 60
- **DefaultExport** (Default implementation for ExportManager class) at page 62
- **Dex** (Dex class) at page 65
- **DexConfig** (Dex configuration class) at page 67
- **DexProperties** (Dex properties file) at page 79
- **EdgeData** (Edge data class) at page 81
- **EdgeExport** (Stores edge exporting values) at page 82
- **EdgeTypeExporter** (EdgeTypeExporter class) at page 86
- **EdgeTypeLoader** (EdgeTypeLoader class) at page 90
- **Error** (Error class) at page 97
- **Exception** (Exception class) at page 99
- **ExportManager** (Defines how to export a graph to an external format) at page 101
- **FileNotFoundException** (File not found exception class) at page 104
3.1 Class List

Graph (Graph class) 106
GraphExport (Stores the graph exporting values) 125
Handler< T > (Handles a reference) 126
Int32List (Dex 32-bit signed integer list) 127
Int32ListIterator (Int32List iterator class) 129
IOException (IO error class) 130
IOException (IO exception class) 132
LicenseError (License error class) 134
NodeExport (Stores the node exporting values) 136
NodeTypeExporter (NodeTypeExporter class) 140
NodeTypeLoader (NodeTypeLoader class) 145
NoSuchElementException (No such element exception class) 151
Objects (Object identifier set class) 153
ObjectsIterator (ObjectsIterator class) 161
OIDList (Dex object identifier list) 162
OIDListIterator (OIDList iterator class) 164
Platform (Platform class) 165
PlatformStatistics (Platform data and statistics) 166
RowReader (RowReader interface) 168
RowWriter (RowWriter interface) 170
ScriptParser (ScriptParser) 171
Session (Session class) 174
ShortestPath (ShortestPath class) 177
SinglePairShortestPath (SinglePairShortestPath class) 180
SinglePairShortestPathBFS (SinglePairShortestPathBFS class) 186
SinglePairShortestPathDijkstra (SinglePairShortestPathDijkstra class) 191
SQLException (SQL exception class) 198
StringList (String list) 200
StringListIterator (StringList iterator class) 201
3.1 Class List

**StrongConnectivity** (StrongConnectivity class) 202

**StrongConnectivityGabow** (This class can be used to solve the problem of finding strongly connected components in a directed graph) 208

**SystemError** (System error class) 213

**TextStream** (TextStream class) 216

**Traversal** (Traversal class) 219

**TraversalBFS** (Breadth-First Search implementation of Traversal) 223

**TraversalDFS** (Depth-First Search (DFS) implementation of Traversal) 227

**Type** (Type data class) 232

**TypeExporter** (Base TypeExporter class) 235

**TypeExporterEvent** (Provides information about the progress of a TypeExporter instance) 239

**TypeExporterListener** (Interface to be implemented to receive TypeExporterEvent events from a TypeExporter) 241

**TypeList** (Dex type identifier list) 242

**TypeListIterator** (TypeList iterator class) 243

**TypeLoader** (Base TypeLoader class) 244

**TypeLoaderEvent** (Provides information about the progress of a TypeLoader instance) 251

**TypeLoaderListener** (Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader) 253

**UnsupportedOperationError** (Unsupported operation error class) 254

**Value** (Value class) 256

**Values** (Value set class) 266

**ValuesIterator** (Values iterator class) 268

**WeakConnectivity** (WeakConnectivity class) 271

**WeakConnectivityDFS** (WeakConnectivityDFS class) 277

**WrongArgumentError** (Wrong argument error class) 283
4 Module Documentation

4.1 GDB

Collaboration diagram for GDB:

This is the API for the management of a Dex graph-database.

Files

- file common.h
  It contains common includes and definitions as well as set basic data types and enumerations.

- file Database.h
  It contains the definition of Database class as well as some other related classes.

- file Dex.h
  It contains the declaration of Dex and DexConfig classes.

- file Exception.h
  It contains a hierarchy of exceptions.

- file Export.h
  It contains the declaration of ExportManager interface and GraphExport, NodeExport, etc classes.

- file Graph.h
  It contains the definition of Graph class.

- file Graph_data.h
  It contains the definition of some Graph-related classes.

- file Handler.h
  It contains the definition of Handler class.

- file Objects.h
  It contains the definition of Object class.

- file ObjectsIterator.h
  It contains the definition of ObjectsIterator class.

- file Session.h
  It contains the definition of Session class.

- file Stream.h
4.1 GDB

It contains the definition of stream classes.

- file Value.h
  It contains the definition of Value class.

- file Values.h
  It contains the definition of Values class.

- file ValuesIterator.h
  It contains the definition of ValuesIterator class.

Classes

- class PlatformStatistics
  Platform data and statistics.

- class Platform
  Platform class.

- class DatabaseStatistics
  Database statistics.

- class Database
  Database class.

- class DexProperties
  Dex properties file.

- class DexConfig
  Dex configuration class.

- class Dex
  Dex class.

- class Exception
  Exception class.

- class IOException
  IO exception class.

- class FileNotFoundException
  File not found exception class.

- class NoSuchElementException
  No such element exception class.

- class SQLException
  SQL exception class.
• class Error
  Error class.

• class SystemError
  System error class.

• class AppError
  Application error class.

• class WrongArgumentError
  Wrong argument error class.

• class IOError
  IO error class.

• class LicenseError
  License error class.

• class UnsupportedOperationError
  Unsupported operation error class.

• class GraphExport
  Stores the graph exporting values.

• class NodeExport
  Stores the node exporting values.

• class EdgeExport
  Stores edge exporting values.

• class ExportManager
  Defines how to export a graph to an external format.

• class DefaultExport
  Default implementation for ExportManager class.

• class Graph
  Graph class.

• class Type
  Type data class.

• class TypeList
  Dex type identifier list.

• class TypeListIterator
  TypeList iterator class.

• class Attribute
  Attribute data class.
• class AttributeList
  
  *Dex* attribute identifier list.

• class AttributeListIterator
  
  *AttributeList* iterator class.

• class OIDList
  
  *Dex* object identifier list.

• class OIDListIterator
  
  *OIDList* iterator class.

• class AttributeStatistics
  
  *Attribute* statistics class.

• class EdgeData
  
  *Edge* data class.

• class StringList
  
  *String* list.

• class StringListIterator
  
  *StringList* iterator class.

• class BooleanList
  
  *Boolean* list.

• class BooleanListIterator
  
  *BooleanList* iterator class.

• class Int32List
  
  *Dex* 32-bit signed integer list.

• class Int32ListIterator
  
  *Int32List* iterator class.

• class Handler&lt; T &gt;
  
  Handles a reference.

• class Objects
  
  *Object* identifier set class.

• class ObjectsIterator
  
  *ObjectsIterator* class.

• class Session
  
  *Session* class.

• class TextStream
TextStream class.

- class Value
  Value class.

- class Values
  Value set class.

- class ValuesIterator
  Values iterator class.

Defines

- #define BEGIN_DEX_NAMESPACE namespace dex {
  Begining macro for the dex namespace.

- #define END_DEX_NAMESPACE }
  Ending macro for the dex namespace.

- #define BEGIN_DEX_GDB_NAMESPACE BEGIN_DEX_NAMESPACE namespace gdb {
  Begining macro for the dex::gdb namespace.

- #define END_DEX_GDB_NAMESPACE END_DEX_NAMESPACE }
  Ending macro for the dex::gdb namespace.

- #define BEGIN_DEX_IO_NAMESPACE BEGIN_DEX_NAMESPACE namespace io {
  Begining macro for the dex::io namespace.

- #define END_DEX_IO_NAMESPACE END_DEX_NAMESPACE }
  Ending macro for the dex::io namespace.

- #define BEGIN_DEX_SCRIPT_NAMESPACE BEGIN_DEX_NAMESPACE namespace script {
  Begining macro for the dex::script namespace.

- #define END_DEX_SCRIPT_NAMESPACE END_DEX_NAMESPACE }
  Ending macro for the dex::script namespace.

- #define BEGIN_DEX_ALGORITHMS_NAMESPACE BEGIN_DEX_NAMESPACE namespace algorithms {
  Begining macro for the dex::algorithms namespace.

- #define END_DEX_ALGORITHMS_NAMESPACE END_DEX_NAMESPACE }
  Ending macro for the dex::algorithms namespace.
4.1 GDB

Typedefs

- typedef bool bool_t
  
  Boolean type.

- typedef char char_t

  Character type.

- typedef wchar_t uchar_t

  Unicode character type.

- typedef signed int int32_t

  32-bit signed integer type.

- typedef signed long long int64_t

  64-bit signed integer type.

- typedef double double64_t

  64-bit double type.

- typedef int32_t type_t

  Graph node or edge type type.

- typedef int32_t attr_t

  Graph attribute type.

- typedef int64_t oid_t

  Graph OID type.

- typedef int32_t ColorRGB

  Color codified as RGB 32-bit int.

Enumerations

- enum ObjectType {
    Node,
    Edge
  }

  Object type enumeration.

- enum DataType {
    Boolean,
    Integer,
    Long,
    Double,
    Timestamp,
    String,
    Text,
    OID
  }

  Datatypes enumeration.
Data type (domain) enumeration.

- enum **Condition**
  - Equal,
  - GreaterEqual,
  - GreaterThan,
  - LessEqual,
  - LessThan,
  - NotEqual,
  - Like,
  - LikeNoCase,
  - Between,
  - RegExp

  *Condition operators enumeration.*

- enum **Order**
  - Ascendent,
  - Descendent

  *Order enumeration.*

- enum **EdgesDirection**
  - Ingoing,
  - Outgoing,
  - Any

  *Edges direction enumeration.*

- enum **AttributeKind**
  - Basic,
  - Indexed,
  - Unique

  *Attribute kind enumeration.*

- enum **LogLevel**
  - Off,
  - Severe,
  - Warning,
  - Info,
  - Config,
  - Fine,
  - Debug

  *Log level enumeration.*
• enum ExportType {
  Graphviz,
  GraphML,
  YGraphML }
  Export type.

• enum NodeShape {
  Box,
  Round }
  Node shape.

Functions

• std::wostream & operator<< (std::wostream &wostrm, const enum DataType &dt)
  Easy STL printing operator redefinition.

• std::wostream & operator<< (std::wostream &wostrm, const enum AttributeKind &ak)
  Easy STL printing operator redefinition.

• Handler::Handler (T *h)
  Creates a new instance with the given reference.

• virtual Handler::~Handler ()
  Destructor.

• T * Handler::GetHandler ()
  Gets the handled reference.

• const T * Handler::GetHandler () const
  Gets the handled reference.

• void Handler::SetHandler (T *h)
  Sets the handled reference.

• void Handler::FreeHandler ()
  Frees (deletes) the handled reference.

• bool_t Handler::IsNull () const
  Gets if the handler is NULL.

Variables

• BEGIN_DEX_GDB_NAMESPACE typedef unsigned char byte_t
  Byte type.
4.1 GDB

4.1.1 Detailed Description

This is the API for the management of a Dex graph-database.
All classes belonging to this group can be found into the dex::gdb namespace.

4.1.2 Typedef Documentation

4.1.2.1 typedef int32_t ColorRGB

Color codified as RGB 32-bit int.
Bits 24-31 are alpha, 16-23 are red, 8-15 are green, 0-7 are blue.

4.1.3 Enumeration Type Documentation

4.1.3.1 enum AttributeKind

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

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

Enumerator:

- **Basic** Basic attribute (non indexed attribute).
- **Indexed** Indexed attribute.
- **Unique** Unique attribute (indexed + unique restriction).
  Unique restriction sets two objects cannot have the same value for an attribute but NULL.

4.1.3.2 enum Condition

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

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

Enumerator:

- **Equal** Equal condition (==).
  Null values can be used together with this condition to retrieve all objects having a null value for an attribute.
- **GreaterEqual** Greater or equal condition (>=).
  Null values cannot be used together with this condition.
- **GreaterThan** Greater than condition (>).
  Null values cannot be used together with this condition.
- **LessEqual** Less or equal condition (<=).
  Null values cannot be used together with this condition.
4.1 GDB

**LessThan**  Less than condition (<).

Null values cannot be used together with this condition.

**NotEqual**  Not equal condition (!=).

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

**Like**  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:

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

**LikeNoCase**  Substring (no case sensitive) condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:

```plaintext
'AAABBBCCCD' LikeNoCase 'BBB' returns TRUE
'AAABBBCCCD' LikeNoCase 'bbb' returns TRUE
'AAABBBCCCD' LikeNoCase 'E' returns FALSE
```

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

Null values cannot be used together with this condition.

**RegExp**  Regular expression condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values.

Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.

See the 'Regular expressions' section in the 'DEX User Manual' for details.

4.1.3.3 enum DataType

Data type (domain) enumeration.

**Author:**

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

**Enumerator:**

- **Boolean**  Boolean data type.
- **Integer**  32-bit signed integer data type.
- **Long**  64-bit signed integer data type.
- **Double**  64-bit signed double data type.
- **Timestamp**  Distance from Epoch (UTC) time in milliseconds precision.

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

**String**  Unicode string data type.
2048 characters maximum length.

**Text**  Large unicode character object (CLOB) data type.

See also:

**TextStream**

**OID**  Object identifier (OID) data type.

4.1.3.4 **enum EdgesDirection**

Edges direction enumeration.

Author:

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

Enumerator:

**Ingoing**  In-going edges.

**Outgoing**  Out-going edges.

**Any**  In-going or out-going edges.

4.1.3.5 **enum ExportType**

Export type.

Author:

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

Enumerator:

**Graphviz**  Export to Graphviz format.
Graphviz home page: [http://www.graphviz.org](http://www.graphviz.org)

**GraphML**  Export to GraphML format.
GraphML home page: [http://graphml.graphdrawing.org/](http://graphml.graphdrawing.org/)

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

4.1.3.6 **enum LogLevel**

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](http://www.sparsity-technologies.com)
4.1 GDB Enumerator:

**Off** Disable log.

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

**Warning** Warning log level.
   Errors and warnings are shown.

**Info** Info log level.
   Errors, warnings and information messages are shown.

**Config** Config log level.
   Errors, warnings, information messages and configuration details of the different components are shown.

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

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

4.1.3.7 enum NodeShape

Node shape.

Author:

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

Enumerator:

**Box** Box shape.

**Round** Round shape.

4.1.3.8 enum ObjectType

Object type enumeration.

Author:

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

Enumerator:

**Node** Node object type.

**Edge** Edge object type.

4.1.3.9 enum Order

Order enumeration.

Author:

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

Enumerator:

**Ascendent** From lower to higher.

**Descendent** From higher to lower.
4.1.4 Function Documentation

4.1.4.1 template<typename T> const T* Handler< T >::GetHandler () const [inline, inherited]

Gets the handled reference.

Returns:

The handled reference.

4.1.4.2 template<typename T> T* Handler< T >::GetHandler () [inline, inherited]

Gets the handled reference.

Returns:

The handled reference.

4.1.4.3 template<typename T> Handler< T >::Handler (T* h) [inline, inherited]

Creates a new instance with the given reference.

Parameters:

h [in] Reference to be handled.

4.1.4.4 template<typename T> bool_t Handler< T >::IsNull () const [inline, protected, inherited]

Gets if the handler is NULL.

Returns:

TRUE if the handler is NULL, FALSE otherwise.

Reimplemented in TextStream, and Value.

4.1.4.5 std::wostream& operator<< (std::wostream & wostream, const enum AttributeKind & ak)

Easy STL printing operator redefinition.
It allows to do: ... << dex::gdb::Basic << ...

4.1.4.6 std::wostream& operator<< (std::wostream & wostream, const enum DataType & dt)

Easy STL printing operator redefinition.
It allows to do: ... << dex::gdb::String << ...
4.2 IO

4.1.4.7 template<typename T> void Handler<T>::SetHandler (T * h) [inline, protected, inherited]
Sets the handled reference.

Parameters:


4.1.4.8 template<typename T> virtual Handler<T>::~Handler () [inline, virtual, inherited]
Destructor.
Frees the handled reference.

4.2 IO

Collaboration diagram for IO:

This is the API for loading and exporting node and edge types from or to external data sources.

Files

- file CSVReader.h
  It contains the definition of CSVReader class.

- file CSVWriter.h
  It contains the definition of CSVWriter class.

- file EdgeTypeExporter.h
  It contains the definition of EdgeTypeExporter class.

- file EdgeTypeLoader.h
  It contains the definition of EdgeTypeLoader class.

- file NodeTypeExporter.h
  It contains the definition of NodeTypeExporter class.

- file NodeTypeLoader.h
  It contains the definition of NodeTypeLoader class.

- file RowReader.h
  It contains the definition of RowReader interface.
• file RowWriter.h
  It contains the definition of RowWriter interface.

• file TypeExporter.h
  It contains the definition of TypeExporter classes.

• file TypeLoader.h
  It contains the definition of TypeLoader classes.

Classes

• class CSVReader
  CSVReader interface.

• class CSVWriter
  CSVWriter interface.

• class EdgeTypeExporter
  EdgeTypeExporter class.

• class EdgeTypeLoader
  EdgeTypeLoader class.

• class NodeExporter
  NodeExporter class.

• class NodeTypeLoader
  NodeTypeLoader class.

• class RowReader
  RowReader interface.

• class RowWriter
  RowWriter interface.

• class TypeExporterEvent
  Provides information about the progress of a TypeExporter instance.

• class TypeExporterListener
  Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

• class TypeExporter
  Base TypeExporter class.

• class TypeLoaderEvent
  Provides information about the progress of a TypeLoader instance.

• class TypeLoaderListener
4.3 Script

*Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.*

- **class TypeLoader**
  Base TypeLoader class.

**Functions**

- std::wostream & operator<< (std::wostream &wostrm, const TypeExporterEvent &ev)
  Easy STL printing operator redefinition.

- std::wostream & operator<< (std::wostream &wostrm, const TypeLoaderEvent &ev)
  Easy STL printing operator redefinition.

4.2.1 Detailed Description

This is the API for loading and exporting node and edge types from or to external data sources. All classes belonging to this group can be found into the dex::io namespace.

4.3 Script

Collaboration diagram for Script:

```
Script
  
  Algorithms
  
  dex
  
  IO
  
  dexcore
  
  GDB
```

This is the API for creating and loading a Dex graph-database from a script file.

**Files**

- file ScriptParser.h
  It contains the declaration of ScriptParser class.

**Classes**

- **class ScriptParser**
  ScriptParser.
Functions

• std::wostream & operator<< (std::wostream &wostrm, const enum ScriptParserState &state)
  
  *Easy STL printing operator redefinition.*

4.3.1 Detailed Description

This is the API for creating and loading a Dex graph-database from a script file. All classes belonging to this group can be found into the dex::script namespace.

4.3.2 Function Documentation

4.3.2.1 std::wostream & operator<< (std::wostream & wostrm, const enum ScriptParserState & state)

*Easy STL printing operator redefinition.*

It allows to do: ... << dex::script::SyntaxError << ...

4.4 Algorithms

Collaboration diagram for Algorithms:

```
Algorithms #> Dex #> script
```

Files

• file ConnectedComponents.h
  
  *It contains the definition of ConnectedComponents class.*

• file Connectivity.h
  
  *It contains the definition of Connectivity class.*

• file Context.h
  
  *It contains the definition of Context class.*

• file ShortestPath.h
  
  *It contains the definition of ShortestPath class.*

• file SinglePairShortestPath.h
  
  *It contains the definition of SinglePairShortestPath class.*

• file SinglePairShortestPathBFS.h
  
  *It contains the definition of SinglePairShortestPathBFS class.*

• file SinglePairShortestPathDijkstra.h
  
  *It contains the definition of SinglePairShortestPathDijkstra class.*
• file StrongConnectivity.h
  It contains the definition of StrongConnectivity class.

• file StrongConnectivityGabow.h
  It contains the definition of StrongConnectivityGabow class.

• file Traversal.h
  It contains the definition of Traversal class.

• file TraversalBFS.h
  It contains the definition of TraversalBFS class.

• file TraversalDFS.h
  It contains the definition of TraversalDFS class.

• file WeakConnectivity.h
  It contains the definition of WeakConnectivity class.

• file WeakConnectivityDFS.h
  It contains the definition of WeakConnectivityDFS class.

Classes

• class ConnectedComponents
  ConnectedComponents class.

• class Connectivity
  Connectivity class.

• class Context
  Context class.

• class ShortestPath
  ShortestPath class.

• class SinglePairShortestPath
  SinglePairShortestPath class.

• class SinglePairShortestPathBFS
  SinglePairShortestPathBFS class.

• class SinglePairShortestPathDijkstra
  SinglePairShortestPathDijkstra class.

• class StrongConnectivity
  StrongConnectivity class.

• class StrongConnectivityGabow
This class can be used to solve the problem of finding strongly connected components in a directed graph.

- class **Traversal**
  *Traversal class.*

- class **TraversalBFS**
  *Breadth-First Search implementation of Traversal.*

- class **TraversalDFS**
  *Depth-First Search (DFS) implementation of Traversal.*

- class **WeakConnectivity**
  *WeakConnectivity class.*

- class **WeakConnectivityDFS**
  *WeakConnectivityDFS class.*

## 5 Class Documentation

### 5.1 AppError Class Reference

Application error class.

Inheritance diagram for AppError:

```plaintext
Exception

Error

AppError

LicenseError

UnsupportedOperationError

WrongArgumentError
```
Collaboration diagram for AppError:

```
Exception
↓
Error
↓
AppError
```

**Public Member Functions**

- `AppError ()`
  *Creates a new instance.*

- `AppError (const std::string &mess)`
  *Creates a new instance.*

- `virtual ~AppError ()`
  *Destructor.*

- `const std::string & Message () const`
  *Gets the message of the exception.*

- `void SetMessage (const std::string &mess)`
  *Sets the message of the exception.*

**Static Public Member Functions**

- `static Error NewError (int32_t coreErrorCode)`
  *Creates a new Error instance from a dexcore error code.*

**Protected Attributes**

- `std::string message`
  *Message of the exception.*

### 5.1.1 Detailed Description

Application error class.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.1.2 Constructor & Destructor Documentation

5.1.2.1 AppError::AppError (const std::string & mess)
Creates a new instance.

Parameters:

 mess  [in] Message of the exception.

5.1.3 Member Function Documentation

5.1.3.1 static Error Error::NewError (int32_t coreErrorCode)  [static, inherited]
Creates a new Error instance from a dexcore error code.

Parameters:


Returns:

 Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.

5.1.3.2 const std::string& Exception::Message () const  [inherited]
Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

 The message of the exception.

5.1.3.3 void Exception::SetMessage (const std::string & mess)  [inherited]
Sets the message of the exception.

Parameters:


The documentation for this class was generated from the following file:

• Exception.h

5.2 Attribute Class Reference

Attribute data class.
Public Member Functions

• attr_t GetId () const
  
  Gets the Dex attribute identifier.

• type_t GetTypeld () const
  
  Gets the Dex type identifier.

• const std::wstring & GetName () const
  
  Gets the unique attribute name.

• DataType GetDataType () const
  
  Gets the data type.

• int64_t GetSize () const
  
  Gets the number of different values.

• int64_t GetCount () const
  
  Gets the number of non-NULL values.

• AttributeKind GetKind () const
  
  Gets the attribute kind.

• bool_t IsSessionAttribute () const
  
  Check if it’s a session attribute or a persistent one.

Static Public Attributes

• static const attr_t InvalidAttribute
  
  Invalid attribute identifier constant.

Friends

• class Graph

5.2.1 Detailed Description

Attribute data class.
It contains information about an attribute.

Author:

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

5.2.2.1 attr_t Attribute::GetId () const [inline]
Gets the Dex attribute identifier.

Returns:
   The Dex attribute identifier.

5.2.2.2 type_t Attribute::GetTypeId () const [inline]
Gets the Dex type identifier.

Returns:
   The Dex type identifier.

5.2.2.3 const std::wstring& Attribute::GetName () const [inline]
Gets the unique attribute name.

Returns:
   The unique attribute name.

5.2.2.4 DataType Attribute::GetDataType () const [inline]
Gets the data type.

Returns:
   The DataType.

5.2.2.5 int64_t Attribute::GetSize () const [inline]
Gets the number of different values.

Returns:
   The number of different values.

5.2.2.6 int64_t Attribute::GetCount () const [inline]
Gets the number of non-NULL values.

Returns:
   The number of non-NULL values.
5.3 AttributeList Class Reference

5.2.2.7 AttributeKind Attribute::GetKind () const [inline]
Gets the attribute kind.

Returns:
The AttributeKind.

5.2.2.8 bool_t Attribute::IsSessionAttribute () const [inline]
Check if it’s a session attribute or a persistent one.

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

The documentation for this class was generated from the following file:

- Graph_data.h

5.3 AttributeList Class Reference

Dex attribute identifier list.

Public Member Functions

- int32_t Count () const
  Number of elements in the list.

- AttributeListIterator * Iterator ()
  Gets a new AttributeListIterator.

- AttributeList ()
  Constructor.

- AttributeList (const std::vector< attr_t >&v)
  Constructor.

- void Add (attr_t attr)
  Adds a Dex attribute identifier at the end of the list.

- void Clear ()
  Clears the list.

5.3.1 Detailed Description

Dex attribute identifier list.
It stores a Dex attribute identifier list.
Use AttributeListIterator to access all elements into this collection.
5.4 AttributeListIterator Class Reference

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

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AttributeList::AttributeList ()
Constructor.
This creates an empty list.

5.3.2.2 AttributeList::AttributeList (const std::vector<attr_t>& v)
Constructor.

Parameters:

v [in] Vector.

5.3.3 Member Function Documentation

5.3.3.1 int32_t AttributeList::Count () const [inline]
Number of elements in the list.

Returns:

Number of elements in the list.

5.3.3.2 AttributeListIterator* AttributeList::Iterator ()
Gets a new AttributeListIterator.

Returns:

AttributeListIterator instance.

5.3.3.3 void AttributeList::Add (attr_t attr) [inline]
Adds a Dex attribute identifier at the end of the list.

Parameters:

attr [in] Dex attribute identifier.

The documentation for this class was generated from the following file:

- Graph_data.h

5.4 AttributeListIterator Class Reference

AttributeList iterator class.
5.5 AttributeStatistics Class Reference

Public Member Functions

• attr_t Next ()
   
   Moves to the next element.

• bool_t HasNext ()
   
   Gets if there are more elements.

Friends

• class AttributeList

5.4.1 Detailed Description

AttributeList iterator class.
Iterator to traverse all the Dex attribute identifier into a AttributeList instance.

Author:

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

5.4.2 Member Function Documentation

5.4.2.1 attr_t AttributeListIterator::Next () [inline]

Moves to the next element.

Returns:

   The next element.

5.4.2.2 bool_t AttributeListIterator::HasNext () [inline]

Gets if there are more elements.

Returns:

   TRUE if there are more elements, FALSE otherwise.

The documentation for this class was generated from the following file:

• Graph_data.h

5.5 AttributeStatistics Class Reference

Attribute statistics class.
Collaboration diagram for `AttributeStatistics`:

```
  T
  | handler
  | Handler< T >
  |   | | Value
  |   | | maxV
  |   | | minV
  |   | | modeV
  | AttributeStatistics
```

**Public Member Functions**

- `int64_t GetTotal () const`
  
  *Gets the number of objects with a non-NULL Value (BASIC statistic).*

- `int64_t GetNull () const`
  
  *Gets the number of objects NULL a Value (BASIC statistics).*

- `int64_t GetDistinct () const`
  
  *Gets the number of distinct values (BASIC statistics).*

- `const Value & GetMin () const`
  
  *Gets the minimum existing value (BASIC statistics).*

- `const Value & GetMax () const`
  
  *Gets the maximum existing value (BASIC statistics).*

- `int32_t GetMaxLengthString () const`
  
  *Gets the maximum length.*

- `int32_t GetMinLengthString () const`
  
  *Gets the minimum length.*

- `double64_t GetAvgLengthString () const`
  
  *Gets the average length.*

- `const Value & GetMode () const`
  
  *Gets the mode.*

- `int64_t GetModeCount () const`
5.5 AttributeStatistics Class Reference

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

- **double64_t GetMean () const**
  
  Gets the mean or average.

- **double64_t GetVariance () const**
  
  Gets the variance.

- **double64_t GetMedian () const**
  
  Gets the median.

Friends

- class Graph

5.5.1 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 method Graph::getAttributeStatistics or check out the DEX User Manual for more details on this.

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

5.5.2 Member Function Documentation

5.5.2.1 **int64_t AttributeStatistics::GetTotal () const** [inline]

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

**Returns:**

The number of objects with a non-NULL Value.

5.5.2.2 **int64_t AttributeStatistics::GetNull () const** [inline]

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

**Returns:**

The number of objects NULL a Value.
5.5.2.3 int64_t AttributeStatistics::GetDistinct () const [inline]

Gets the number of distinct values (BASIC statistics).

**Returns:**

The number of distinct values.

5.5.2.4 const Value& AttributeStatistics::GetMin () const [inline]

Gets the minimum existing value (BASIC statistics).

**Returns:**

The minimum existing value.

5.5.2.5 const Value& AttributeStatistics::GetMax () const [inline]

Gets the maximum existing value (BASIC statistics).

**Returns:**

The maximum existing value.

5.5.2.6 int32_t AttributeStatistics::GetMaxLengthString () const [inline]

Gets the maximum length.

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

**Returns:**

The maximum length.

5.5.2.7 int32_t AttributeStatistics::GetMinLengthString () const [inline]

Gets the minimum length.

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

**Returns:**

The minimum length.

5.5.2.8 double64_t AttributeStatistics::GetAvgLengthString () const [inline]

Gets the average length.

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

**Returns:**

The average length.
5.5.2.9 const Value& AttributeStatistics::GetMode () const [inline]
Gets the mode.
Mode: Most frequent Value.

Returns:
    The mode.

5.5.2.10 int64_t AttributeStatistics::GetModeCount () const [inline]
Gets the number of objects with a Value equal to the mode.

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

5.5.2.11 double64_t AttributeStatistics::GetMean () const [inline]
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.

5.5.2.12 double64_t AttributeStatistics::GetVariance () const [inline]
Gets the variance.
It is computed just for numerical attributes.

Returns:
    The variance.

5.5.2.13 double64_t AttributeStatistics::GetMedian () const [inline]
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.

The documentation for this class was generated from the following file:

- Graph_data.h
5.6 BooleanList Class Reference

Boolean list.

Public Member Functions

- **int32_t Count () const**
  
  *Number of elements in the list.*

- **BooleanListIterator * Iterator ()**
  
  *Gets a new BooleanListIterator.*

- **BooleanList ()**
  
  *Constructor.*

- **BooleanList (const std::vector<bool_t> &v)**
  
  *Constructor.*

- **void Add (dex::gdb::bool_t b)**
  
  *Adds a Boolean at the end of the list.*

- **void Clear ()**
  
  *Clears the list.*

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

5.6.2 Constructor & Destructor Documentation

5.6.2.1 BooleanList::BooleanList ()

Constructor.

This creates an empty list.

5.6.2.2 BooleanList::BooleanList (const std::vector<bool_t> & v)

Constructor.

**Parameters:**

\( v \) [in] Vector.
5.6.3 Member Function Documentation

5.6.3.1 int32_t BooleanList::Count () const  [inline]
Number of elements in the list.

Returns:
Number of elements in the list.

5.6.3.2 BooleanListIterator* BooleanList::Iterator ()
Gets a new BooleanListIterator.

Returns:
BooleanListIterator instance.

5.6.3.3 void BooleanList::Add (dex::gdb::bool_t b)  [inline]
Adds a Boolean at the end of the list.

Parameters:
b  [in] Boolean.

The documentation for this class was generated from the following file:

• Graph_data.h

5.7 BooleanListIterator Class Reference

BooleanList iterator class.

Public Member Functions

• dex::gdb::bool_t Next ()
  Moves to the next element.

• bool_t HasNext ()
  Gets if there are more elements.

Friends

• class BooleanList
5.7.1 Detailed Description

BooleanList iterator class.
Iterator to traverse all the strings into a BooleanList instance.

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

5.7.2 Member Function Documentation

5.7.2.1 dex::gdb::bool_t BooleanListIterator::Next () [inline]
Moves to the next element.

Returns:
The next element.

5.7.2.2 bool_t BooleanListIterator::HasNext () [inline]
Gets if there are more elements.

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

The documentation for this class was generated from the following file:

• Graph_data.h

5.8 ConnectedComponents Class Reference

ConnectedComponents class.

Public Member Functions

• ConnectedComponents (dex::gdb::Session &s, const std::wstring &materializedattribute)
  Creates a new instance of ConnectedComponents.

• virtual ~ConnectedComponents ()
  Destructor.

• dex::gdb::int64_t GetConnectedComponent (dex::gdb::oid_t idNode)
  Returns the connected component where the given node belongs to.

• dex::gdb::int64_t GetCount ()
  Returns the number of connected components found in the graph.

• dex::gdb::Objects * GetNodes (dex::gdb::int64_t idConnectedComponent)
  Returns the collection of nodes contained in the given connected component.
5.8 ConnectedComponents Class Reference

• `dex::gdb::int64_t GetSize (dex::gdb::int64_t idConnectedComponent)`
  Returns the number of nodes contained in the given connected component.

5.8.1 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 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 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 DEX User Manual for more details on this.

Author:

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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 `ConnectedComponents::ConnectedComponents (dex::gdb::Session & s, const std::wstring & 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.

5.8.3 Member Function Documentation

5.8.3.1 `dex::gdb::int64_t ConnectedComponents::GetConnectedComponent (dex::gdb::oid_t idNode)`

Returns the connected component where the given node belongs to.

Parameters:

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

5.8.3.2  dex::gdb::int64_t ConnectedComponents::GetCount ()
Returns the number of connected components found in the graph.

Returns:
The number of connected components found in the graph.

5.8.3.3  dex::gdb::Objects* ConnectedComponents::GetNodes (dex::gdb::int64_t idConnectedComponent)
Returns the collection of nodes contained in the given connected component.

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

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

5.8.3.4  dex::gdb::int64_t ConnectedComponents::GetSize (dex::gdb::int64_t idConnectedComponent)
Returns the number of nodes contained in the given connected component.

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

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

The documentation for this class was generated from the following file:
• ConnectedComponents.h

5.9  Connectivity Class Reference
Connectivity class.
Inheritance diagram for Connectivity:

```
Inheritance diagram for Connectivity:

Connectivity

StrongConnectivity WeakConnectivity

StrongConnectivityGabow WeakConnectivityDFS
```

Collaboration diagram for Connectivity:

```
Collaboration diagram for Connectivity:

ConnectedComponents

ccs
```

**Public Member Functions**

- virtual ~Connectivity ()
  
  Destructor.

- virtual void AddNodeType (dex::gdb::type_t t)
  
  Allows connectivity through nodes of the given type.

- virtual void AddAllNodeTypes ()
  
  Allows connectivity through all node types of the graph.

- virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  
  Set which nodes can’t be used.

- virtual void ExcludeEdges (dex::gdb::Objects &edges)
  
  Set which edges can’t be used.

- ConnectedComponents * GetConnectedComponents ()
  
  Returns the results generated by the execution of the algorithm.

- virtual void Run ()=0
  
  Runs the algorithm in order to find the connected components.

- void SetMaterializedAttribute (const std::wstring &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.
5.9 Connectivity Class Reference

Protected Types

- `typedef std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > EdgeTypes_t`
  A type definition to store allowed edge types.

- `typedef std::vector< dex::gdb::type_t > NodeTypes_t`
  A type definition to store allowed node types.

Protected Member Functions

- `Connectivity (dex::gdb::Session &s)`
  Creates a new instance of `Connectivity`.

- `void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)`
  Allows connectivity through edges of the given type.

- `void AddAllEdgeTypes (dex::gdb::EdgesDirection d)`
  Allows connectivity through all edge types of the graph.

- `void AssertAddedEdges ()`
  Check that edges had been added.

- `void AssertAddedNodes ()`
  Check that nodes had been added.

- `void AssertNotComputed ()`
  Check that the connectivity had not been calculated yet.

- `void SetConnectedComponent (dex::gdb::oid_t idNode)`
  Assigns the current component to the given node.

- `void SetNodesNotVisited ()`
  Set all the selected nodes in `nodesNotVisited`.

- `void AssertNotComponentAttribute (const std::wstring &attributeName)`
  Check that the given attribute name is not already in use.

- `void AssertComputed ()`
  Check that the connectivity had been calculated.

- `void AssertEdgeType (dex::gdb::type_t edgetype)`
  Check that the given edge type is valid.

- `void AssertNodeType (dex::gdb::type_t nodetype)`
  Check that the given node type is valid.

- `void CreateGlobalPersistentAttribute (const std::wstring &attributeName)`
  Set a new persistent global attribute to store the connectivity information.
• void `CreateGlobalTransientAttribute`()  
  Set a new temporary global attribute to store the connectivity information.

• void `RemoveGlobalAttribute`()  
  Removes the global attribute where the connectivity information is stored.

• `dex::gdb::bool_t IsNodeTypeAllowed(dex::gdb::oid_t nodeId)`  
  Check if the given node has an allowed type.

• `dex::gdb::bool_t IsNodeExcluded(dex::gdb::oid_t node)`  
  Check if the given node is forbidden.

• `dex::gdb::bool_t IsEdgeExcluded(dex::gdb::oid_t edge)`  
  Check if the given edge is forbidden.

**Protected Attributes**

• `dex::gdb::Session * sess`  
  `Session`.

• `dex::gdb::Graph * graph`  
  `Graph`.

• `EdgeTypes_t edgeTypes`  
  Allowed edge types.

• `std::vector< dex::gdb::type_t > nodeTypes`  
  Allowed node types.

• `dex::gdb::attr_t attrComponent`  
  Common attribute where the connected component information is stored.

• `std::wstring attrComponentName`  
  Name of the common attribute where the connected component information is stored.

• `dex::gdb::int64_t actualComponent`  
  Current component identifier.

• `dex::gdb::Objects * nodesNotVisited`  
  Identifiers of the nodes not visited.

• `dex::gdb::bool_t matResults`  
  Materialized results.

• `dex::gdb::bool_t computed`  
  True if the connectivity has been calculated.

• `dex::gdb::Objects * excludedNodes`
The set of excluded nodes.

- dex::gdb::Objects * excludedEdges
  The set of excluded edges.

- ConnectedComponents * ccs
  The calculated connectivity information.

5.9.1 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 or finding the weakly connected components.

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

Author:

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

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Connectivity::Connectivity (dex::gdb::Session & s) [protected]

Creates a new instance of Connectivity.

Parameters:

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

5.9.3 Member Function Documentation

5.9.3.1 virtual void Connectivity::ExcludeNodes (dex::gdb::Objects & nodes) [virtual]

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.

5.9.3.2 virtual void Connectivity::ExcludeEdges (dex::gdb::Objects & edges) [virtual]

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.
5.9.3.3 **ConnectedComponents** - Connectivity::GetConnectedComponents()

Returns the results generated by the execution of the algorithm.

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

**Returns:**

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

---

5.9.3.4 **virtual void Connectivity::Run()** [pure virtual]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in `StrongConnectivityGabow`, and `WeakConnectivityDFS`.

---

5.9.3.5 **void Connectivity::SetMaterializedAttribute (const std::wstring & 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.

---

5.9.3.6 **void Connectivity::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [protected]**

Allows connectivity through edges of the given type.

**Parameters:**

`t` [in] Edge type.

`d` [in] Edge direction.

Reimplemented in `StrongConnectivity`.

---

5.9.3.7 **void Connectivity::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [protected]**

Allows connectivity through all edge types of the graph.
5.10 Context Class Reference

Parameters:

\[ d \] [in] Edge direction.

Reimplemented in StrongConnectivity.

5.9.3.8 **void Connectivity::SetNodesNotVisited ()** [protected]

Set all the selected nodes in nodesNotVisited.

That’s all the nodes of the allowed node types but not the excluded ones.

The documentation for this class was generated from the following file:

- Connectivity.h

5.10 Context Class Reference

Context class.

Public Member Functions

- **void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)**
  
  Allows for traversing edges of the given type.

- **void AddAllEdgeTypes (dex::gdb::EdgesDirection d)**
  
  Allows for traversing all edge types of the graph.

- **void AddNodeType (dex::gdb::type_t t)**
  
  Allows for traversing nodes of the given type.

- **void AddAllNodeTypes ()**
  
  Allows for traversing all node types of the graph.

- **void ExcludeNodes (dex::gdb::Objects &nodes)**
  
  Set which nodes can’t be used.

- **void ExcludeEdges (dex::gdb::Objects &edges)**
  
  Set which edges can’t be used.

- **dex::gdb::Objects * Compute ()**

  Gets the resulting collection of nodes.

- **void SetMaximumHops (dex::gdb::int32_t maxhops, dex::gdb::bool_t include)**

  Sets the maximum hops restriction.

- **virtual ~Context ()**

  Destructor.

- **Context (dex::gdb::Session &s, dex::gdb::oid_t node)**

  Creates a new instance.
Static Public Member Functions

- static dex::gdb::Objects * Compute (dex::gdb::Session &, dex::gdb::oid_t node, dex::gdb::TypeList *nodeTypes, dex::gdb::TypeList *edgeTypes, dex::gdb::EdgesDirection dir, dex::gdb::int32_t max-hops, dex::gdb::bool_t include)
  
  Helper method to easily compute a context from a node.

Protected Attributes

- dex::gdb::Session * sess
  
  Session.

- dex::gdb::Graph * graph
  
  Graph.

- dex::gdb::oid_t src
  
  Source node of the traversal.

- std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes
  
  Allowed edge types.

- std::vector< dex::gdb::type_t > nodeTypes
  
  Allowed node types.

- dex::gdb::int32_t maxHops
  
  Maximum number of hops allowed.

- dex::gdb::bool_t inclusive
  
  Include those nodes at distance <= maxhops or just those nodes at distance == maxhops.

- dex::gdb::Objects * excludedNodes
  
  The set of excluded nodes.

- dex::gdb::Objects * excludedEdges
  
  The set of excluded edges.

5.10.1 Detailed Description

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 DEX User Manual for more details on this.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
5.10.2 Constructor & Destructor Documentation

5.10.2.1 Context::Context (dex::gdb::Session & s, dex::gdb::oid_t node)

Creates a new instance.

Parameters:

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

5.10.3 Member Function Documentation

5.10.3.1 void Context::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)

Allows for traversing edges of the given type.

Parameters:

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

5.10.3.2 void Context::AddAllEdgeTypes (dex::gdb::EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.10.3.3 void Context::ExcludeNodes (dex::gdb::Objects & nodes)

Set which nodes can’t be used.

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

Parameters:

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

5.10.3.4 void Context::ExcludeEdges (dex::gdb::Objects & edges)

Set which edges can’t be used.

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

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.
5.10.3.5  

**dex::gdb::Objects* Context::Compute ()**

Gets the resulting collection of nodes.

**Returns:**

The resulting collection of nodes.

5.10.3.6  

**void Context::SetMaximumHops (dex::gdb::int32_t maxhops, dex::gdb::bool_t 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.

5.10.3.7  

**static dex::gdb::Objects* Context::Compute (dex::gdb::Session & s, dex::gdb::oid_t node, dex::gdb::TypeList * nodeTypes, dex::gdb::TypeList * edgeTypes, dex::gdb::EdgesDirection dir, dex::gdb::int32_t maxhops, dex::gdb::bool_t include)** [static]

Helper method to easily compute a context from a node.

**Parameters:**

- **s** [in] Session to get the graph from and perform operation.

- **node** [in] Node to start traversal from.

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

- **edgeTypes** [in] Allowed edge type list. NULL means all edge types are allowed.

- **dir** [in] Allowed direction for the allowed edge types.

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

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

The documentation for this class was generated from the following file:

- **Context.h**
5.11 CSVReader Class Reference

CSVReader interface.

Inheritance diagram for CSVReader:

Collaboration diagram for CSVReader:

Public Member Functions

- CSVReader()
  Constructs CSVReader.

- void SetSeparator(const std::wstring &sep) throw (dex::gdb::Error)
  Sets the character used to separate fields in the file.

- void SetQuotes(const std::wstring &quotes) throw (dex::gdb::Error)
  Sets the character used to quote fields.

- void SetMultilines(dex::gdb::int32_t numExtralines)
  Allows the use of fields with more than one line.

- void SetSingleLine()
  Only allows single line fields.

- void SetStartLine(dex::gdb::int32_t startLine)
  Sets the number of lines to be skipped from the beginning.

- void SetNumLines(dex::gdb::int32_t numLines)
  Used to limit the number of lines that will be read.

- void SetLocale(const std::wstring &localeStr)
  Sets the locale that will be used to read the file.

- void Open(const std::wstring &f) throw (dex::gdb::IOException)
  Opens the source file path.
• **dex::gdb::bool_t Reset () throw (dex::gdb::IOException)**
  Moves the reader to the beginning.

• **dex::gdb::bool_t Read (dex::gdb::StringList &row) throw (dex::gdb::IOException)**
  Reads the next row as a string array.

• **dex::gdb::int32_t GetRow () throw (dex::gdb::IOException)**
  The row number for the current row.

• **void Close () throw (dex::gdb::IOException)**
  Closes the reader.

• **virtual ~CSVReader ()**
  Destructor.

### 5.11.1 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.


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 DEX User Manual for more details on this.

**Author:**

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

### 5.11.2 Member Function Documentation

#### 5.11.2.1 void CSVReader::SetSeparator (const std::wstring & sep) throw (dex::gdb::Error)

Sets the character used to separate fields in the file.
5.11 CSVReader Class Reference

Parameters:

sep [in] Separator character.

5.11.2.2 void CSVReader::SetQuotes (const std::wstring & quotes) throw (dex::gdb::Error)
Sets the character used to quote fields.

Parameters:

quotes [in] Quote character.

5.11.2.3 void CSVReader::SetMultilines (dex::gdb::int32_t numExtralines)
Allows the use of fields with more than one line.

Parameters:

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

5.11.2.4 void CSVReader::SetStartLine (dex::gdb::int32_t startLine)
Sets the number of lines to be skipped from the beginning.

Parameters:

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

5.11.2.5 void CSVReader::SetNumLines (dex::gdb::int32_t numLines)
Used to limit the number of lines that will be read.

Parameters:

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

5.11.2.6 void CSVReader::SetLocale (const std::wstring & localeStr)
Sets the locale that will be used to read the file.

Parameters:

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

5.11.2.7 void CSVReader::Open (const std::wstring & f) throw (dex::gdb::IOException)
Opens the source file path.

Parameters:

f [in] CSV file path.

Exceptions:

IOException If bad things happen opening the file.
5.11 CSVReader Class Reference

5.11.2.8 \texttt{dex::gdb::bool\_t CSVReader::Reset (\texttt{dex::gdb::IOException})} [virtual]

Moves the reader to the beginning.

Restarts the reader.

\textbf{Returns:}

true if the reader can be restarted, false otherwise.

\textbf{Exceptions:}

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

Implements \texttt{RowReader}.

5.11.2.9 \texttt{dex::gdb::bool\_t CSVReader::Read (dex::gdb::StringList & row) throw (dex::gdb::IOException)} [virtual]

Reads the next row as a string array.

\textbf{Parameters:}

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

\textbf{Returns:}

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

\textbf{Exceptions:}

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

Implements \texttt{RowReader}.

5.11.2.10 \texttt{dex::gdb::int32\_t CSVReader::GetRow (\texttt{dex::gdb::IOException})} [virtual]

The row number for the current row.

\textbf{Returns:}

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

\textbf{Exceptions:}

\textit{IOException} If it fails.

Implements \texttt{RowReader}.

5.11.2.11 \texttt{void CSVReader::Close (\texttt{dex::gdb::IOException})} [virtual]

Closes the reader.

\textbf{Exceptions:}

\textit{IOException} If the close fails.
Impplements **RowReader**.

The documentation for this class was generated from the following file:

- CSVReader.h

## 5.12 CSVWriter Class Reference

**CSVWriter** interface.

Inheritance diagram for CSVWriter:

```
    RowWriter
     
      CSVWriter
```

Collaboration diagram for CSVWriter:

```
    RowWriter
     
      CSVWriter
```

### Public Member Functions

- **CSVWriter ()**
  
  *Creates a new instance.*

- **void SetSeparator (const std::wstring &sep) throw (dex::gdb::Error)**
  
  *Sets the character used to separate fields in the file.*

- **void SetQuotes (const std::wstring &quotes) throw (dex::gdb::Error)**
  
  *Sets the character used to quote fields.*

- **void SetAutoQuotes (dex::gdb::bool_t autoquotes)**
  
  *Sets on/off the automatic quote mode.*

- **void SetForcedQuotes (dex::gdb::BooleanList &forcequotes)**
  
  *Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.*

- **void SetLocale (const std::wstring &localeStr)**
  
  *Sets the locale that will be used to write the file.*

- **void Open (const std::wstring &f) throw (dex::gdb::IOException)**
  
  *Opens the output file path.*
5.12 CSVWriter Class Reference

- void **Write** (dex::gdb::StringList &row) throw (dex::gdb::IOException, dex::gdb::Error)
  
  *W*rites the next row.

- void **Close** () throw (dex::gdb::IOException, dex::gdb::Error)
  
  *C*loses the writer.

- virtual ~**CSVWriter** ()
  
  *D*estructor.

5.12.1 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.


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 DEX User Manual.

Check out the ‘Data export’ section in the DEX User Manual for more details on this.

**Author:**

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

5.12.2 Member Function Documentation

5.12.2.1 void **CSVWriter::SetSeparator** (const std::wstring & *sep*) throw (dex::gdb::Error)

Sets the character used to separate fields in the file.

**Parameters:**

*sep* [in] Separator character.

5.12.2.2 void **CSVWriter::SetQuotes** (const std::wstring & *quotes*) throw (dex::gdb::Error)

Sets the character used to quote fields.

**Parameters:**

*quotes* [in] Quote character.

5.12.2.3 void **CSVWriter::SetAutoQuotes** (dex::gdb::bool_t *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.

5.12.2.4 void CSVWriter::SetForcedQuotes (dex::gdb::BooleanList & forcequotes)
Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

Parameters:

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

5.12.2.5 void CSVWriter::SetLocale (const std::wstring & localeStr)
Sets the locale that will be used to write the file.

Parameters:

- **localeStr** [in] The locale string for the file encoding.

5.12.2.6 void CSVWriter::Open (const std::wstring & f) throw (dex::gdb::IOException)
Opens the output file path.

Parameters:

- **f** [in] Output file path.

Exceptions:

- **IOException** If bad things happen opening the file.

5.12.2.7 void CSVWriter::Write (dex::gdb::StringList & row) throw (dex::gdb::IOException, dex::gdb::Error) [virtual]
Writes the next row.

Parameters:

- **row** [in] Row of data.

Exceptions:

- **IOException** If bad things happen during the write.

Implements **RowWriter**.

5.12.2.8 void CSVWriter::Close () throw (dex::gdb::IOException, dex::gdb::Error) [virtual]
Closes the writer.
Exceptions:

**IOException** If the close fails.

Implements *RowWriter*.

The documentation for this class was generated from the following file:

- CSVWriter.h

### 5.13 Database Class Reference

**Database** class.

Inheritance diagram for Database:

Collaboration diagram for Database:

**Public Member Functions**

- **virtual ~Database ()**
  *Destructor.*

- **const std::wstring & GetAlias () const**
  *Gets the alias of the Database.*
5.13 Database Class Reference

- const std::wstring & GetPath () const
  Gets the path of the Database.

- Session * NewSession ()
  Creates a new Session.

- void GetStatistics (DatabaseStatistics &stats)
  Gets Database statistics.

Friends
- class Dex
- class Graph

5.13.1 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 Dex instance.

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

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

Author:

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

5.13.2 Member Function Documentation

5.13.2.1 const std::wstring& Database::GetAlias () const [inline]

Gets the alias of the Database.

Returns:

The alias of the Database.

5.13.2.2 const std::wstring& Database::GetPath () const [inline]

Gets the path of the Database.

Returns:

The path of the Database.
5.13.2.3  void Database::GetStatistics (DatabaseStatistics & stats)

Gets Database statistics.

Parameters:


The documentation for this class was generated from the following file:

- Database.h

5.14  DatabaseStatistics Class Reference

Database statistics.

Public Member Functions

- int64_t GetRead () const
  Gets total read data in KBytes.

- int64_t GetWrite () const
  Gets total written data in KBytes.

- int64_t GetData () const
  Gets database size in KBytes.

- int64_t GetCache () const
  Gets cache size in KBytes.

- int64_t GetTemp () const
  Gets temporary storage file size in KBytes.

- int64_t GetSessions () const
  Gets the number of sessions.

Friends

- class Database

5.14.1  Detailed Description

Database statistics.

Author:

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

5.14.2.1 int64_t DatabaseStatistics::GetRead () const [inline]
 Gets total read data in KBytes.

 Returns:
 Total read data in KBytes.

5.14.2.2 int64_t DatabaseStatistics::GetWrite () const [inline]
 Gets total written data in KBytes.

 Returns:
 Total read written in KBytes.

5.14.2.3 int64_t DatabaseStatistics::GetData () const [inline]
 Gets database size in KBytes.

 Returns:
 Database size in KBytes.

5.14.2.4 int64_t DatabaseStatistics::GetCache () const [inline]
 Gets cache size in KBytes.

 Returns:
 Cache size in KBytes.

5.14.2.5 int64_t DatabaseStatistics::GetTemp () const [inline]
 Gets temporary storage file size in KBytes.

 Returns:
 Temporary storage file size in KBytes.

5.14.2.6 int64_t DatabaseStatistics::GetSessions () const [inline]
 Gets the number of sessions.

 Returns:
 The number of sessions.

The documentation for this class was generated from the following file:

- Database.h

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
5.15 DefaultExport Class Reference

Default implementation for ExportManager class.

Inheritance diagram for DefaultExport:

Collaboration diagram for DefaultExport:
5.15 DefaultExport Class Reference

Public Member Functions

• DefaultExport ()
  Creates a new instance.

• virtual ~DefaultExport ()
  Destructor.

• void Prepare (Graph *graph)
  ExportManager::Prepare.

• void Release ()
  ExportManager::Release.

• bool_t GetGraph (GraphExport &graphExport)
  ExportManager::GetGraph.

• bool_t GetNodeType (type_t type, NodeExport &nodeExport)
  ExportManager::GetNodeType.

• bool_t GetEdgeType (type_t type, EdgeExport &edgeExport)
  ExportManager::GetEdgeType.

• bool_t GetNode (oid_t node, NodeExport &nodeExport)
  ExportManager::GetNode.

• bool_t GetEdge (oid_t edge, EdgeExport &edgeExport)
  ExportManager::GetEdge.

• bool_t EnableType (type_t type)
  ExportManager::EnableType.

5.15.1 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

5.15.2 Member Function Documentation

5.15.2.1 bool_t DefaultExport::GetGraph (GraphExport & graphExport) [virtual]
ExportManager::GetGraph.
This sets the default GraphExport values and "Graph" as the label.
Implements ExportManager.
5.15.2.2 bool_t DefaultExport::GetNodeType (type_t type, NodeExport & nodeExport) [virtual]
ExportManager::GetNodeType.
This sets the default NodeExport values.
Implements ExportManager.

5.15.2.3 bool_t DefaultExport::GetEdgeType (type_t type, EdgeExport & edgeExport) [virtual]
ExportManager::GetEdgeType.
This sets the default EdgeExport values.
Implements ExportManager.

5.15.2.4 bool_t DefaultExport::GetNode (oid_t node, NodeExport & nodeExport) [virtual]
ExportManager::GetNode.
This sets the default NodeExport values and sets the OID as the label.
Returns:
TRUE.
Implements ExportManager.

5.15.2.5 bool_t DefaultExport::GetEdge (oid_t edge, EdgeExport & edgeExport) [virtual]
ExportManager::GetEdge.
This sets the default EdgeExport values and sets the OID as the label. Also, it exports the edge as directed just if the edge is directed.
Returns:
TRUE.
Implements ExportManager.

5.15.2.6 bool_t DefaultExport::EnableType (type_t type) [virtual]
ExportManager::EnableType.
This enables all node and edge types to be exported.
Returns:
TRUE.
Implements ExportManager.

The documentation for this class was generated from the following file:

- Export.h
5.16  Dex Class Reference

Dex class.

Inheritance diagram for Dex:

```
Dex
|    
|    Handler< T >
|    |    < dexcore::DEX >
|    Handler< dexcore::DEX >
|    Dex
```

Collaboration diagram for Dex:

```
T
handler
|    
|    Handler< T >
|    |    < dexcore::DEX >
|    Handler< dexcore::DEX >
|    Dex
```

Public Member Functions

- **Dex** (const DexConfig &config)
  
  *Creates a new instance.*

- **virtual ~Dex ()**

  *Destructor.*

- **Database * Create** (const std::wstring &path, const std::wstring &alias) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)

  *Creates a new Database instance.*

- **Database * Open** (const std::wstring &path, bool_t read) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)

  *Opens an existing Database instance.*

- **Database * Restore** (const std::wstring &path, const std::wstring &backupFile) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)

  *Restores a Database from a backup file.*
Static Public Attributes

• static const std::wstring Version
  Dex version.

5.16.1 Detailed Description

Dex class.

All Dex programs must have one single Dex instance to manage one or more Database instances.
This class allows for the creation of new Databases or open an existing one.

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

5.16.2 Constructor & Destructor Documentation

5.16.2.1 Dex::Dex (const DexConfig & config)

Creates a new instance.

Parameters:
  config [in] Dex configuration.

5.16.3 Member Function Documentation

5.16.3.1 Database∗ Dex::Create (const std::wstring & path, const std::wstring & alias) throw
  (dex::gdb::FileNotFoundException, dex::gdb::Error)

Creates a new Database instance.

Parameters:
  alias [in] Database alias name.

Returns:
  A Database instance.

Exceptions:
  FileNot FoundException If the given file cannot be created.

5.16.3.2 Database∗ Dex::Open (const std::wstring & path, bool_t read) throw
  (dex::gdb::FileNotFoundException, dex::gdb::Error)

Opens an existing Database instance.

Parameters:
read [in] If TRUE, open Database in read-only mode.

Returns:
A Database instance.

Exceptions:
FileNotFoundException If the given file does not exist.

5.16.3.3 Database* Dex::Restore (const std::wstring & path, const std::wstring & backupFile)
throw (dex::gdb::FileNotFoundException, dex::gdb::Error)
Restores a Database from a backup file.
See Graph::Backup.

Parameters:
backupFile [in] The Backup file to be restored.

Returns:
A Database instance.

Exceptions:
FileNotFoundException If the given file cannot be created, or the exported data file does not exists.

The documentation for this class was generated from the following file:

- Dex.h

5.17 DexConfig Class Reference

Dex configuration class.

Public Member Functions

- DexConfig ()
  Creates a new instance.

- int32_t GetExtentSize () const
  Gets the size of a extent.

- void SetExtentSize (int32_t v)
  Sets the size of a pool frame in number of extents.

- int32_t GetExtentPages () const
  Gets the number of pages per extent.
void SetExtentPages (int32_t v)
Sets the number of pages per extent.

int32_t GetPoolFrameSize () const
Gets the size of a pool frame in number of extents.

void SetPoolFrameSize (int32_t v)
Sets the size of a pool frame in number of extents.

int32_t GetPoolPersistentMinSize () const
Gets the minimum size for the persistent pool in number of frames.

void SetPoolPersistentMinSize (int32_t v)
Sets the minimum size for the persistent pool in number of frames.

int32_t GetPoolPersistentMaxSize () const
Gets the maximum size for the persistent pool in number of frames.

void SetPoolPersistentMaxSize (int32_t v)
Sets the maximum size for the persistent pool in number of frames.

int32_t GetPoolTemporaryMinSize () const
Gets the minimum size for the temporary pool in number of frames.

void SetPoolTemporaryMinSize (int32_t v)
Sets the minimum size for the temporary pool in number of frames.

int32_t GetPoolTemporaryMaxSize () const
Gets the maximum size for the temporary pool in number of frames.

void SetPoolTemporaryMaxSize (int32_t v)
Sets the maximum size for the temporary pool in number of frames.

int32_t GetCacheMaxSize () const
Gets the maximum size for the cache (all pools) in MB.

void SetCacheMaxSize (int32_t v)
Sets the maximum size for the cache (all pools) in MB.

const std::wstring & GetLicense () const
Gets the license code.

void SetLicense (const std::wstring &v)
Sets the license code.

const std::wstring & GetLogFile () const
Gets the log file.

void SetLogFile (const std::wstring &v)
Sets the log file.
- `LogLevel GetLogLevel () const`  
  Gets the log level.

- `void SetLogLevel (LogLevel v)`  
  Sets the log level.

- `bool_t GetCacheStatisticsEnabled () const`  
  Gets whether cache statistics are enabled or disabled.

- `void SetCacheStatisticsEnabled (bool_t v)`  
  Enables or disables cache statistics.

- `const std::wstring & GetCacheStatisticsFile () const`  
  Gets the cache statistics log file.

- `void SetCacheStatisticsFile (const std::wstring &v)`  
  Sets the cache statistics log file.

- `int64_t GetCacheStatisticsSnapshotTime () const`  
  Gets the cache statistics snapshot time in microseconds.

- `void SetCacheStatisticsSnapshotTime (int64_t v)`  
  Sets the cache statistics snapshot time.

- `bool_t GetRecoveryEnabled () const`  
  Gets whether the recovery is enabled or disabled.

- `void SetRecoveryEnabled (bool_t v)`  
  Enables or disables the recovery.

- `const std::wstring & GetRecoveryLogFile () const`  
  Gets the recovery log file.

- `void SetRecoveryLogFile (const std::wstring &v)`  
  Sets the recovery log file.

- `int32_t GetRecoveryCacheMaxSize () const`  
  Gets the maximum size for the recovery log cache in extents.

- `void SetRecoveryCacheMaxSize (int32_t v)`  
  Sets the maximum size for the recovery log cache in extents.

- `int64_t GetRecoveryCheckpointTime () const`  
  Gets the delay time (in microseconds) between automatic checkpoints.

- `void SetRecoveryCheckpointTime (int64_t v)`  
  Sets the delay time (in microseconds) between automatic checkpoints.

- `bool_t GetHighAvailabilityEnabled () const`  
  Gets high availability enabled status.
5.17 DexConfig Class Reference

Gets whether high availability mode is enabled or disabled.

- void SetHighAvailabilityEnabled (bool_t v)
  Enables or disables high availability mode.

- const std::wstring & GetHighAvailabilityIP () const
  Gets the IP address and port of the instance.

- void SetHighAvailabilityIP (const std::wstring &v)
  Sets the IP address and port of the instance.

- const std::wstring & GetHighAvailabilityCoordinators () const
  Gets the coordinators address and port list.

- void SetHighAvailabilityCoordinators (const std::wstring &v)
  Sets the coordinators address and port list.

- int64_t GetHighAvailabilitySynchronization () const
  Gets the synchronization polling time.

- void SetHighAvailabilitySynchronization (int64_t v)
  Sets the synchronization polling time.

- int64_t GetHighAvailabilityMasterHistory () const
  Gets the master's history log.

- void SetHighAvailabilityMasterHistory (int64_t v)
  Sets the master's history log.

5.17.1 Detailed Description

Dex configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overrided with values from a properties file (see DexProperties class). Also, this settings can be overrided calling a specific setter.

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

- Extent size: 4KB (‘dex.storage.extentsize’ at DexProperties).
- Pages per extent: 1 page (‘dex.storage.extentpages’ at DexProperties).
- Pool frame size: 1 extent (‘dex.io.pool.frame.size’ at DexProperties).
- Minimum size for the persistent pool: 64 frames (‘dex.io.pool.persistent.minsize’ at DexProperties).
- Maximum size for the persistent pool: 0 frames (‘dex.io.pool.persistent.maxsize’ at DexProperties).
- Maximum size for the temporary pool: 0 frames (‘dex.io.pool.temporal.maxsize’ at DexProperties).
- Maximum size for the cache (all pools): 0 MB (‘dex.io.cache.maxsize’ at DexProperties).
License code: "" ('dex.license' at DexProperties). No license code means evaluation license.

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

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


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


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

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

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

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

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

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

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

Use of TimeUnit:

Those variables using TimeUnit allow for:

<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

5.17.2 Constructor & Destructor Documentation

5.17.2.1 DexConfig::DexConfig ()

Creates a new instance.

Values are set with default values.

5.17.3 Member Function Documentation

5.17.3.1 int32_t DexConfig::GetExtentSize () const [inline]

Gets the size of a extent.

Returns:

The size of a extent in KB.
5.17.3.2 void DexConfig::SetExtentSize (int32_t v) [inline]
Sets the size of a pool frame in number of extents.

Parameters:

v [in] The size of a extent size in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

5.17.3.3 int32_t DexConfig::GetExtentPages () const [inline]
Gets the number of pages per extent.

Returns:

The number of pages per extent.

5.17.3.4 void DexConfig::SetExtentPages (int32_t v) [inline]
Sets the number of pages per extent.

Parameters:

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

5.17.3.5 int32_t DexConfig::GetPoolFrameSize () const [inline]
Gets the size of a pool frame in number of extents.

Returns:

The size of a pool frame in number of extents.

5.17.3.6 void DexConfig::SetPoolFrameSize (int32_t v) [inline]
Sets the size of a pool frame in number of extents.

Parameters:

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

5.17.3.7 int32_t DexConfig::GetPoolPersistentMinSize () const [inline]
Gets the minimum size for the persistent pool in number of frames.

Returns:

The minimum size for the persistent pool in number of frames.
5.17.3.8  void DexConfig::SetPoolPersistentMinSize (int32_t v)  [inline]
Sets the minimum size for the persistent pool in number of frames.

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

5.17.3.9  int32_t DexConfig::GetPoolPersistentMaxSize () const  [inline]
Gets the maximum size for the persistent pool in number of frames.

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

5.17.3.10 void DexConfig::SetPoolPersistentMaxSize (int32_t v)  [inline]
Sets the maximum size for the persistent pool in number of frames.

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

5.17.3.11 int32_t DexConfig::GetPoolTemporaryMinSize () const  [inline]
Gets the minimum size for the temporary pool in number of frames.

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

5.17.3.12 void DexConfig::SetPoolTemporaryMinSize (int32_t v)  [inline]
Sets the minimum size for the temporary pool in number of frames.

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

5.17.3.13 int32_t DexConfig::GetPoolTemporaryMaxSize () const  [inline]
Gets the maximum size for the temporary pool in number of frames.

Returns:
  
  The maximum size for the temporary pool in number of frames.
5.17.3.14 **void DexConfig::SetPoolTemporaryMaxSize (int32_t v) [inline]**
Sets the maximum size for the temporary pool in number of frames.

**Parameters:**

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

5.17.3.15 **int32_t DexConfig::GetCacheMaxSize () const [inline]**
Gets the maximum size for the cache (all pools) in MB.

**Returns:**

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

5.17.3.16 **void DexConfig::SetCacheMaxSize (int32_t v) [inline]**
Sets the maximum size for the cache (all pools) in MB.

**Parameters:**

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

5.17.3.17 **const std::wstring& DexConfig::GetLicense () const [inline]**
Gets the license code.

**Returns:**

The license code.

5.17.3.18 **void DexConfig::SetLicense (const std::wstring & v) [inline]**
Sets the license code.

**Parameters:**

v [in] The license code.

5.17.3.19 **const std::wstring& DexConfig::GetLogFile () const [inline]**
Gets the log file.

**Returns:**

The log file.

5.17.3.20 **void DexConfig::SetLogFile (const std::wstring & v) [inline]**
Sets the log file.

**Parameters:**

5.17.3.21 **LogLevel DexConfig::GetLogLevel () const [inline]**

Gets the log level.

**Returns:**

The LogLevel.

5.17.3.22 **void DexConfig::SetLogLevel (LogLevel \(v\)) [inline]**

Sets the log level.

**Parameters:**

\(v\) [in] The LogLevel.

5.17.3.23 **bool_t DexConfig::GetCacheStatisticsEnabled () const [inline]**

Gets whether cache statistics are enabled or disabled.

**Returns:**

TRUE if cache statistics are enabled, FALSE otherwise.

5.17.3.24 **void DexConfig::SetCacheStatisticsEnabled (bool_t \(v\)) [inline]**

Enables or disables cache statistics.

**Parameters:**

\(v\) [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

5.17.3.25 **const std::wstring& DexConfig::GetCacheStatisticsFile () const [inline]**

Gets the cache statistics log file.

Useless if cache statistics are disabled.

**Returns:**

The cache statistics log file.

5.17.3.26 **void DexConfig::SetCacheStatisticsFile (const std::wstring & \(v\)) [inline]**

Sets the cache statistics log file.

Useless if cache statistics are disabled.

**Parameters:**

5.17 DexConfig Class Reference

5.17.3.27 int64_t DexConfig::GetCacheStatisticsSnapshotTime () const [inline]
Gets the cache statistics snapshot time in microseconds.
Useless if cache statistics are disabled.

Returns:
The cache statistics snapshot time in microseconds.

5.17.3.28 void DexConfig::SetCacheStatisticsSnapshotTime (int64_t v) [inline]
Sets the cache statistics snapshot time.
Useless if cache statistics are disabled.

Parameters:

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

5.17.3.29 bool_t DexConfig::GetRecoveryEnabled () const [inline]
Gets whether the recovery is enabled or disabled.

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

5.17.3.30 void DexConfig::SetRecoveryEnabled (bool_t v) [inline]
Enables or disables the recovery.

Parameters:

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

5.17.3.31 const std::wstring& DexConfig::GetRecoveryLogFile () const [inline]
Gets the recovery log file.

Returns:
The recovery log file.

5.17.3.32 void DexConfig::SetRecoveryLogFile (const std::wstring & v) [inline]
Sets the recovery log file.

Parameters:

v [in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log)
5.17.3.33 int32_t DexConfig::GetRecoveryCacheMaxSize () const [inline]

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

**Returns:**

The maximum size for the recovery log cache in extents.

5.17.3.34 void DexConfig::SetRecoveryCacheMaxSize (int32_t v) [inline]

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

**Parameters:**

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

5.17.3.35 int64_t DexConfig::GetRecoveryCheckpointTime () const [inline]

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

**Returns:**

The delay time (in microseconds) between automatic checkpoints.

5.17.3.36 void DexConfig::SetRecoveryCheckpointTime (int64_t v) [inline]

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

**Parameters:**

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

5.17.3.37 bool_t DexConfig::GetHighAvailabilityEnabled () const [inline]

Gets whether high availability mode is enabled or disabled.

**Returns:**

TRUE if high availability mode is enabled, FALSE otherwise.

5.17.3.38 void DexConfig::SetHighAvailabilityEnabled (bool_t v) [inline]

Enables or disables high availability mode.

**Parameters:**

v [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.
5.17.3.39  const std::wstring& DexConfig::GetHighAvailabilityIP () const  [inline]
Gets the IP address and port of the instance.

Returns:

The IP address and port of the instance.

5.17.3.40  void DexConfig::SetHighAvailabilityIP (const std::wstring & v)  [inline]
Sets the IP address and port of the instance.

Parameters:

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

5.17.3.41  const std::wstring & DexConfig::GetHighAvailabilityCoordinators () const  [inline]
Gets the coordinators address and port list.

Returns:

The coordinators address and port list.

5.17.3.42  void DexConfig::SetHighAvailabilityCoordinators (const std::wstring & v)  [inline]
Sets the coordinators address and port list.

Parameters:

v  [in] The coordinators address and port list.

5.17.3.43  int64_t DexConfig::GetHighAvailabilitySynchronization () const  [inline]
Gets the synchronization polling time.

Returns:

The Synchronization polling time.

5.17.3.44  void DexConfig::SetHighAvailabilitySynchronization (int64_t v)  [inline]
Sets the synchronization polling time.

Parameters:

v  [in] The synchronization polling time.
5.17.3.45 int64_t DexConfig::GetHighAvailabilityMasterHistory () const [inline]

Gets the master’s history log.

**Returns:**

The master’s history log.

5.17.3.46 void DexConfig::SetHighAvailabilityMasterHistory (int64_t v) [inline]

Sets the master’s history log.

**Parameters:**

v [in] The master’s history log.

The documentation for this class was generated from the following file:

- Dex.h

5.18 DexProperties Class Reference

Dex properties file.

**Static Public Member Functions**

- static void Load (const std::wstring &path)
  
  Loads properties from the given file path.

- static const std::wstring & Get (const std::wstring &key, const std::wstring &def)
  
  Gets a property.

- static int32_t GetInteger (const std::wstring &key, int32_t def)
  
  Gets a property as an integer.

- static bool_t GetBoolean (const std::wstring &key, bool_t def)
  
  Gets a property as a boolean.

- static int64_t GetTimeUnit (const std::wstring &key, int64_t def)
  
  Gets a property as a time unit.

5.18.1 Detailed Description

Dex properties file.

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: key=value
By default, this loads properties from the file './dex.cfg'. The user may choose to load a different file by calling the method `Load()`.

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

### 5.18.2 Member Function Documentation

#### 5.18.2.1 `static void DexProperties::Load (const std::wstring & path)` [static]

Loads properties from the given file path.

**Parameters:**

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

#### 5.18.2.2 `static const std::wstring& DexProperties::Get (const std::wstring & key, const std::wstring & def)` [static]

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.

#### 5.18.2.3 `static int32_t DexProperties::GetInteger (const std::wstring & key, int32_t def)` [static]

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.

#### 5.18.2.4 `static bool_t DexProperties::GetBoolean (const std::wstring & key, bool_t def)` [static]

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.
5.18.2.5 static int64_t DexProperties::GetTimeUnit (const std::wstring & key, int64_t def) [static]

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Valid format for the string representation: Blanks at the begining or at the end are ignored. No blanks are allowed between the time duration and the unit time.

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

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

Parameters:

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

Returns:

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

The documentation for this class was generated from the following file:

  • Dex.h

5.19 EdgeData Class Reference

Edge data class.

Public Member Functions

• oid_t GetEdge () const
  Gets the edge identifier.

• oid_t GetTail () const
  Gets the tail of the edge.

• oid_t GetHead () const
  Gets the head of the edge.

Friends

• class Graph

5.19.1 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.
5.20 EdgeExport Class Reference

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

5.19.2 Member Function Documentation

5.19.2.1 oid_t EdgeData::GetEdge () const [inline]
Gets the edge identifier.

Returns:
The Dex edge identifier.

5.19.2.2 oid_t EdgeData::GetTail () const [inline]
Gets the tail of the edge.

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

5.19.2.3 oid_t EdgeData::GetHead () const [inline]
Gets the head of the edge.

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

The documentation for this class was generated from the following file:

• Graph_data.h

5.20 EdgeExport Class Reference

Stores edge exporting values.

Public Member Functions

• EdgeExport ()
  Creates a new instance.

• virtual ~EdgeExport ()
  Destructor.

• void SetDefaults ()
  Sets to default values.

• const std::wstring & GetLabel () const
  Gets the edge label.
• void SetLabel (const std::wstring &l)
  
  Sets the edge label.

• bool_t AsDirected () const
  
  Gets if the edge should be managed as directed.

• void SetAsDirected (bool_t b)
  
  Sets if the edge should be managed as directed.

• ColorRGB GetColorRGB () const
  
  Gets the edge color.

• void SetColorRGB (ColorRGB c)
  
  Sets the edge color.

• ColorRGB GetLabelColorRGB () const
  
  Gets the edge label color.

• void SetLabelColorRGB (ColorRGB c)
  
  Sets the edge label color.

• int32_t GetWidth () const
  
  Gets the edge width.

• void SetWidth (int32_t w)
  
  Sets the edge width.

• int32_t GetFontSize () const
  
  Gets the edge label font size.

• void SetFontSize (int32_t s)
  
  Sets the edge label font size.

### 5.20.1 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.
5.20 EdgeExport Class Reference

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

5.20.2 Member Function Documentation

5.20.2.1 const std::wstring& EdgeExport::GetLabel () const [inline]
Gets the edge label.

Returns:
The edge label.

5.20.2.2 void EdgeExport::SetLabel (const std::wstring & l) [inline]
Sets the edge label.

Parameters:
  l [in] The edge label.

5.20.2.3 bool_t EdgeExport::AsDirected () const [inline]
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.

5.20.2.4 void EdgeExport::SetAsDirected (bool_t b) [inline]
Sets if the edge should be managed as directed.

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

5.20.2.5 ColorRGB EdgeExport::getColorRGB () const [inline]
Gets the edge color.

Returns:
The edge color.

5.20.2.6 void EdgeExport::setColorRGB (ColorRGB c) [inline]
Sets the edge color.

Parameters:
  c [in] The edge color.
5.20.2.7 ColorRGB EdgeExport::GetLabelColorRGB () const [inline]
Gets the edge label color.

Returns:
   The edge label color.

5.20.2.8 void EdgeExport::SetLabelColorRGB (ColorRGB c) [inline]
Sets the edge label color.

Parameters:
   c [in] The edge label color.

5.20.2.9 int32_t EdgeExport::GetWidth () const [inline]
Gets the edge width.

Returns:
   The edge width.

5.20.2.10 void EdgeExport::SetWidth (int32_t w) [inline]
Sets the edge width.

Parameters:
   w [in] The edge width.

5.20.2.11 int32_t EdgeExport::GetFontSize () const [inline]
Gets the edge label font size.

Returns:
   The edge label font size.

5.20.2.12 void EdgeExport::SetFontSize (int32_t s) [inline]
Sets the edge label font size.

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

The documentation for this class was generated from the following file:

   • Export.h
5.21 EdgeTypeExporter Class Reference

EdgeTypeExporter class.

Inheritance diagram for EdgeTypeExporter:

![Inheritance Diagram]

Collaboration diagram for EdgeTypeExporter:

![Collaboration Diagram]

Public Member Functions

- **EdgeTypeExporter ()**
  
  *Creates a new instance.*

- **EdgeTypeExporter (RowWriter &rw, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs, dex::gdb::int32_t hPos, dex::gdb::int32_t tPos, dex::gdb::attr_t hAttr, dex::gdb::attr_t tAttr)**
  
  *Creates a new instance.*

- **virtual ~EdgeTypeExporter ()**
  
  *Destructor.*

- **void Run () throw (dex::gdb::IOException, dex::gdb::Error)**
  
  *See TypeExporter::Run.*

- **void SetHeadAttribute (dex::gdb::attr_t attr)**
  
  *Sets the attribute that will be used to get the value to be dumped for the head of the edge.*

- **void SetHeadPosition (dex::gdb::int32_t pos)**
  
  *Sets the position (index column) of the head attribute in the exported data.*

- **void SetTailAttribute (dex::gdb::attr_t attr)**
  
  *Sets the attribute that will be used to get the value to be dumped for the tail of the edge.*

- **void SetTailPosition (dex::gdb::int32_t pos)**
Sets the position (index column) of the tail attribute in the exported data.

- void Register (TypeExporterListener &tel)
  Registers a new listener.
- void SetRowWriter (RowWriter &rw)
  Sets the output data destination.
- void SetGraph (dex::gdb::Graph &g)
  Sets the graph that will be exported.
- void SetType (dex::gdb::type_t t)
  Sets the type to be exported.
- void SetAttributes (dex::gdb::AttributeList &attrs)
  Sets the list of Attributes.
- void SetFrequency (dex::gdb::int32_t freq)
  Sets the frequency of listener notification.
- void SetHeader (dex::gdb::bool_t header)
  Sets the presence of a header row.

Protected Member Functions

- dex::gdb::bool_t CanRun ()
  Checks that all the required settings are ready to run.
- void NotifyListeners (const TypeExporterEvent &ev)
  Notifies progress to all registered listeners.
- void RunProcess () throw (dex::gdb::IOException, dex::gdb::Error)
  Runs export process.

5.21.1 Detailed Description

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

Author:

Sparsity Technologies http://www.sparsity-technologies.com
5.21.2 Constructor & Destructor Documentation

5.21.2.1 EdgeTypeExporter::EdgeTypeExporter (RowWriter & rw, dex::gdb::Graph & g, dex::gdb::type_t t, dex::gdb::AttributeList & attrs, dex::gdb::int32_t hPos, dex::gdb::int32_t tPos, dex::gdb::attr_t hAttr, dex::gdb::attr_t tAttr) [inline]

Creates a new instance.

Parameters:

- g [in] Graph.
- t [in] Type identifier.
- attrs [in] Attribute identifiers to be exported.
- hPos [in] The position (index column) for the head value.
- tPos [in] The position (index column) for the tail value.
- hAttr [in] The attribute identifier to get the value to be dumped for the head.
- tAttr [in] The attribute identifier to get the value to be dumped for the tail.

5.21.3 Member Function Documentation

5.21.3.1 void EdgeTypeExporter::SetHeadAttribute (dex::gdb::attr_t attr) [inline]

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

Parameters:

- attr [in] Head Attribute

Reimplemented from TypeExporter.

5.21.3.2 void EdgeTypeExporter::SetHeadPosition (dex::gdb::int32_t pos) [inline]

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

Parameters:

- pos [in] Head position

Reimplemented from TypeExporter.

5.21.3.3 void EdgeTypeExporter::SetTailAttribute (dex::gdb::attr_t attr) [inline]

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

Reimplemented from TypeExporter.
5.21.3.4 void EdgeTypeExporter::SetTailPosition (dex::gdb::int32_t pos) [inline]
Sets the position (index column) of the tail attribute in the exported data.

Parameters:
   pos [in] Tail position

Reimplemented from TypeExporter.

5.21.3.5 dex::gdb::bool_t TypeExporter::CanRun () [protected, inherited]
Checks that all the required settings are ready to run.

Returns:
   Returns true if all the settings are ready.

5.21.3.6 void TypeExporter::NotifyListeners (const TypeExporterEvent & ev) [protected, inherited]
Notifies progress to all registered listeners.

Parameters:
   ev [in] Progress event to be notified.

5.21.3.7 void TypeExporter::RunProcess () throw (dex::gdb::IOException, dex::gdb::Error) [protected, inherited]
Runs export process.

Exceptions:
   IOException If bad things happen writing to the RowWriter.

5.21.3.8 void TypeExporter::Register (TypeExporterListener & tel) [inherited]
Registers a new listener.

Parameters:
   tel [in] TypeExporterListener to be registered.

5.21.3.9 void TypeExporter::SetRowWriter (RowWriter & rw) [inherited]
Sets the output data destination.

Parameters:
   rw [in] Input RowWriter.
5.21.3.10 void TypeExporter::SetGraph (dex::gdb::Graph & g) [inherited]
Sets the graph that will be exported.

Parameters:
  g [in] Graph.

5.21.3.11 void TypeExporter::SetType (dex::gdb::type_t t) [inherited]
Sets the type to be exported.

Parameters:
  t [in] Type identifier.

5.21.3.12 void TypeExporter::SetAttributes (dex::gdb::AttributeList & attrs) [inherited]
Sets the list of Attributes.

Parameters:
  attrs [in] Attribute identifiers to be exported

5.21.3.13 void TypeExporter::SetFrequency (dex::gdb::int32_t freq) [inherited]
Sets the frequency of listener notification.
freq [in] Frequency in number of rows managed to notify progress to all listeners

5.21.3.14 void TypeExporter::SetHeader (dex::gdb::bool_t header) [inherited]
Sets the presence of a header row.

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

The documentation for this class was generated from the following file:

• EdgeTypeExporter.h

5.22 EdgeTypeLoader Class Reference

EdgeTypeLoader class.
Inheritance diagram for EdgeTypeLoader:

```
  TypeLoader
     |
     V
EdgeTypeLoader
```

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
Collaboration diagram for EdgeTypeLoader:

![Collaboration diagram for EdgeTypeLoader](image)

**Public Member Functions**

- **EdgeTypeLoader ()**
  
  Creates a new instance.

- **EdgeTypeLoader (RowReader &rr, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs, dex::gdb::Int32List &attrsPos, dex::gdb::int32_t hPos, dex::gdb::int32_t tPos, dex::gdb::attr_t hAttr, dex::gdb::attr_t tAttr)**

  Creates a new instance.

- **virtual ~EdgeTypeLoader ()**

  Destructor.

- **void Run () throw (dex::gdb::IOException, dex::gdb::Error)**

  See TypeLoader::Run.

- **void RunTwoPhases () throw (dex::gdb::IOException, dex::gdb::Error)**

  See TypeLoader::RunTwoPhases.

- **void RunNPhases (dex::gdb::int32_t partitions) throw (dex::gdb::IOException, dex::gdb::Error)**

  See TypeLoader::RunNPhases.

- **void SetHeadAttribute (dex::gdb::attr_t attr)**

  Sets the attribute that will be used to find the head of the edge.

- **void SetHeadPosition (dex::gdb::int32_t pos)**

  Sets the position of the head attribute in the source data.

- **void SetTailAttribute (dex::gdb::attr_t attr)**

  Sets the attribute that will be used to find the tail of the edge.

- **void SetTailPosition (dex::gdb::int32_t pos)**

  Sets the position of the tail attribute in the source data.

- **void SetLogError (const std::wstring &path) throw (dex::gdb::IOException)**

  Sets a log error file.

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
• void `SetLogOff()` 
  Trans off all the error reporting.

• void `Register (TypeLoaderListener &tel)` 
  Registers a new listener.

• void `SetRowReader (RowReader &rr)` 
  Sets the input data source.

• void `SetGraph (dex::gdb::Graph &g)` 
  Sets the graph where the data will be loaded.

• void `SetLocale (const std::wstring &localeStr)` 
  Sets the locale that will be used to read the data.

• void `SetType (dex::gdb::type_t t)` 
  Sets the type to be loaded.

• void `SetAttributes (dex::gdb::AttributeList &attrs)` 
  Sets the list of Attributes.

• void `SetAttributePositions (dex::gdb::Int32List &attrsPos)` 
  Sets the list of attribute positions.

• void `SetTimestampFormat (const std::wstring &timestampFormat)` 
  Sets a specific timestamp format.

• void `SetFrequency (dex::gdb::int32_t freq)` 
  Sets the frequency of listener notification.

**Protected Types**

• enum `Mode` { 
  ONE_PHASE, 
  TWO_PHASES, 
  N_PHASES } 
  Load can work in different ways.

**Protected Member Functions**

• `dex::gdb::bool_t CanRun ()` 
  Checks that all the required settings are ready to run.

• void `Run (Mode ph, dex::gdb::int32_t par)` throw (dex::gdb::IOException, dex::gdb::Error) 
  Runs load process.

• void `NotifyListeners (const TypeLoaderEvent &ev)` 
  Notifies progress to all registered listeners.
5.22.1 Detailed Description

**EdgeTypeLoader** class.

Specific **TypeLoader** implementation for edge types.

Check out the ‘Data import’ section in the DEX User Manual for more details on this.

**Author:**

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

5.22.2 Member Enumeration Documentation

5.22.2.1 **enum TypeLoader::Mode** [protected, inherited]

Load can work in different ways.

**Enumerator:**

- **ONE_PHASE** Performs the load in one phase.
  
  Load all objects and attributes at the same time.

- **TWO_PHASES** Performs the load in two phases.
  
  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.

- **N_PHASES** Performs the load in N phases.
  
  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 to the **RowReader** are necessary.
  
  Working on this mode it is necessary to build a temporary file.

5.22.3 Constructor & Destructor Documentation

5.22.3.1 **EdgeTypeLoader::EdgeTypeLoader** (RowReader & rr, dex::gdb::Graph & g, dex::gdb::type_t t, dex::gdb::AttributeList & attrs, dex::gdb::Int32List & attrsPos, dex::gdb::int32_t hPos, dex::gdb::int32_t tPos, dex::gdb::attr_t hAttr, dex::gdb::attr_t tAttr) [inline]

Creates a new instance.

**Parameters:**

- **rr** [in] Input **RowReader**.
- **g** [in] **Graph**.
- **t** [in] **Type** identifier.
- **attrs** [in] **Attribute** identifiers to be loaded.
- **attrsPos** [in] **Attribute** positions (column index >=0) to all listeners.
- **hPos** [in] The position (index column) for the head value.
- **tPos** [in] The position (index column) for the tail value.
- **hAttr** [in] The attribute identifier for the head.
- **tAttr** [in] The attribute identifier for the tail.
5.22.4 Member Function Documentation

5.22.4.1 void EdgeTypeLoader::SetHeadAttribute (dex::gdb::attr_t attr) [inline]
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

Reimplemented from TypeLoader.

5.22.4.2 void EdgeTypeLoader::SetHeadPosition (dex::gdb::int32_t pos) [inline]
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

Reimplemented from TypeLoader.

5.22.4.3 void EdgeTypeLoader::SetTailAttribute (dex::gdb::attr_t attr) [inline]
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

Reimplemented from TypeLoader.

5.22.4.4 void EdgeTypeLoader::SetTailPosition (dex::gdb::int32_t pos) [inline]
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

Reimplemented from TypeLoader.

5.22.4.5 dex::gdb::bool_t TypeLoader::CanRun () [protected, inherited]
Checks that all the required settings are ready to run.

Returns:
    Returns true if all the settings are ready.
5.22.4.6 void TypeLoader::Run (Mode *ph, dex::gdb::int32_t *par) throw (dex::gdb::IOException, dex::gdb::Error) [protected, inherited]
Runs load process.

Exceptions:

IOException If bad things happen reading from the RowReader.

Parameters:

*ph [in] The load mode.
*par [in] Number of horizontal partitions to perform the load.

5.22.4.7 void TypeLoader::NotifyListeners (const TypeLoaderEvent & ev) [protected, inherited]
Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.22.4.8 void TypeLoader::SetLogError (const std::wstring & path) throw (dex::gdb::IOException) [inherited]
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.

Exceptions:

IOException If bad things happen opening the file.

5.22.4.9 void TypeLoader::SetLogOff () [inherited]

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.

5.22.4.10 void TypeLoader::Register (TypeLoaderListener & tel) [inherited]

Registers a new listener.

Parameters:

← *tel TypeLoaderListener to be registered.
5.22.4.11 void TypeLoader::SetRowReader (RowReader & rr)  [inherited]
Sets the input data source.
Parameters:
   rr [in] Input RowReader.

5.22.4.12 void TypeLoader::SetGraph (dex::gdb::Graph & g)  [inherited]
Sets the graph where the data will be loaded.
Parameters:
   g [in] Graph.

5.22.4.13 void TypeLoader::SetLocale (const std::wstring & localeStr)  [inherited]
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.

5.22.4.14 void TypeLoader::SetType (dex::gdb::type_t t)  [inherited]
Sets the type to be loaded.
Parameters:
   t [in] Type identifier.

5.22.4.15 void TypeLoader::SetAttributes (dex::gdb::AttributeList & attrs)  [inherited]
Sets the list of Attributes.
Parameters:
   attrs [in] Attribute identifiers to be loaded

5.22.4.16 void TypeLoader::SetAttributePositions (dex::gdb::Int32List & attrsPos)  [inherited]
Sets the list of attribute positions.
attrsPos [in] Attribute positions (column index >=0).

5.22.4.17 void TypeLoader::SetFrequency (dex::gdb::int32_t freq)  [inherited]
Sets the frequency of listener notification.
freq [in] Frequency in number of rows managed to notify progress to all listeners
The documentation for this class was generated from the following file:
   • EdgeTypeLoader.h
5.23  Error Class Reference

Error class.

Inheritance diagram for Error:

[Inheritance diagram]

Collaboration diagram for Error:

[Collaboration diagram]

Public Member Functions

- **Error ()**
  
  Creates a new instance.

- **Error (const std::string &mess)**
  
  Creates a new instance.

- **virtual ~Error ()**
  
  Destructor.

- **const std::string & Message () const**
  
  Gets the message of the exception.

- **void SetMessage (const std::string &mess)**
  
  Sets the message of the exception.
Static Public Member Functions

- static Error NewError (int32_t coreErrorCode)
  
  Creates a new Error instance from a dexcore error code.

Protected Attributes

- std::string message
  
  Message of the exception.

5.23.1 Detailed Description

Error class.

An Error corresponds to an unexpected and unpredictable exception.
As all methods can throw an Error at any moment, it is not expected they declare the Error (or subclasses) they may throw.

Author:

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

5.23.2 Constructor & Destructor Documentation

5.23.2.1 Error::Error (const std::string & mess)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.23.3 Member Function Documentation

5.23.3.1 static Error Error::NewError (int32_t coreErrorCode) [static]

Creates a new Error instance from a dexcore error code.

Parameters:


Returns:

Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.
5.23.3.2 const std::string& Exception::Message () const [inherited]
Gets the message of the exception.
It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:
The message of the exception.

5.23.3.3 void Exception::SetMessage (const std::string &mess) [inherited]
Sets the message of the exception.

Parameters:

The documentation for this class was generated from the following file:

• Exception.h

5.24 Exception Class Reference

Exception class.

Inheritance diagram for Exception:

Public Member Functions

• Exception ()
  Creates a new instance.

• Exception (const std::string &mess)
  Creates a new instance.

• const std::string & Message () const
  Gets the message of the exception.

• void SetMessage (const std::string &mess)
  Sets the message of the exception.
• virtual ~Exception() 
  Destructor.

Protected Attributes

• std::string message
  Message of the exception.

5.24.1 Detailed Description

Exception class.
This is the superclass of those exceptions that can be thrown during the normal execution of a program.
It is expected all methods declare all Exception (or subclasses) they throw.

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

5.24.2 Constructor & Destructor Documentation

5.24.2.1 Exception::Exception (const std::string & mess)
Creates a new instance.

Parameters:
  mess [in] Message of the exception.

5.24.3 Member Function Documentation

5.24.3.1 const std::string& Exception::Message () const
Gets the message of the exception.
It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:
  The message of the exception.

5.24.3.2 void Exception::SetMessage (const std::string & mess)
Sets the message of the exception.

Parameters:

The documentation for this class was generated from the following file:

• Exception.h
5.25 ExportManager Class Reference

Defines how to export a graph to an external format.

Inheritance diagram for ExportManager:

```
ExportManager
    ^
   / \
  /   \
 /     \
DefaultExport
```

Public Member Functions

- virtual `~ExportManager` ()
  Destructor.

- virtual void `Prepare (Graph *graph)=0`
  Prepares the graph for the export process.

- virtual void `Release ()=0`
  Ends the export process.

- virtual `bool_t GetGraph (GraphExport &graphExport)=0`
  Gets the graph export definition.

- virtual `bool_t GetNodeType (type_t type, NodeExport &nodeExport)=0`
  Gets the default node export definition for the given node type.

- virtual `bool_t GetEdgeType (type_t type, EdgeExport &edgeExport)=0`
  Gets the default node export definition for the given edge type.

- virtual `bool_t GetNode (oid_t node, NodeExport &nodeExport)=0`
  Gets the node export definition for the given node.

- virtual `bool_t GetEdge (oid_t edge, EdgeExport &edgeExport)=0`
  Gets the edge export definition for the given edge.

- virtual `bool_t EnableType (type_t type)=0`
  Gets whether a node or edge type must be exported or not.

5.25.1 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 formats. Nowadays, available formats are defined in the ExportType enum.
Author:
Sparsity Technologies http://www.sparsity-technologies.com

5.25.2 Member Function Documentation

5.25.2.1 virtual void ExportManager::Prepare (Graph * graph) [pure virtual]
Prepares the graph for the export process.
It is called once before the export process.

Parameters:

graph Graph to be exported.

Implemented in DefaultExport.

5.25.2.2 virtual void ExportManager::Release () [pure virtual]
Ends the export process.
It is called once after the export process.
Implemented in DefaultExport.

5.25.2.3 virtual bool_t ExportManager::GetGraph (GraphExport & graphExport) [pure virtual]
Gets the graph export definition.

Parameters:

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

Returns:
TRUE.

Implemented in DefaultExport.

5.25.2.4 virtual bool_t ExportManager::GetNodeType (type_t type, NodeExport & nodeExport) [pure virtual]
Gets the default node export definition for the given node type.
GetNode has a higher priority than this function. That is, only if GetNode returns FALSE, the NodeExport
of this function will be used.

Parameters:

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

Returns:
TRUE.

Implemented in DefaultExport.
5.25.2.5 virtual bool_t ExportManager::GetEdgeType (type_t type, EdgeExport & edgeExport) [pure virtual]

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.

Implemented in DefaultExport.

5.25.2.6 virtual bool_t ExportManager::GetNode (oid_t node, NodeExport & nodeExport) [pure virtual]

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.

Implemented in DefaultExport.

5.25.2.7 virtual bool_t ExportManager::GetEdge (oid_t edge, EdgeExport & edgeExport) [pure virtual]

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.

Implemented in DefaultExport.
5.26 FileNotFoundException Class Reference

File not found exception class.

Inheritance diagram for FileNotFoundException:

```
Exception
   ↓
IOException
   ↓
FileNotFoundException
```

Collaboration diagram for FileNotFoundException:

```
Exception
   ↓
IOException
   ↓
FileNotFoundException
```

Public Member Functions

- `FileNotFoundException ()`
  *Creates a new instance.*

- `FileNotFoundException (const std::string &mess)`
  *Creates a new instance.*
virtual ~FileNotFoundException ()
  Destructor.

const std::string & Message () const
  Gets the message of the exception.

void SetMessage (const std::string &mess)
  Sets the message of the exception.

Protected Attributes

std::string message
  Message of the exception.

5.26.1 Detailed Description

File not found exception class.

Author:

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

5.26.2 Constructor & Destructor Documentation

5.26.2.1 FileNotFoundException::FileNotFoundException (const std::string & mess)
  Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.26.3 Member Function Documentation

5.26.3.1 const std::string& Exception::Message () const [inherited]
  Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.26.3.2 void Exception::SetMessage (const std::string & mess) [inherited]
  Sets the message of the exception.

Parameters:

The documentation for this class was generated from the following file:

- Exception.h

5.27 Graph Class Reference

Graph class.

Inheritance diagram for Graph:

Collaboration diagram for Graph:
Public Member Functions

- virtual ~Graph ()
  Destructor.

- type_t NewNodeType (const std::wstring &name)
  Creates a new node type.

- oid_t NewNode (type_t type)
  Creates a new node instance.

- type_t NewEdgeType (const std::wstring &name, bool_t directed, bool_t neighbors)
  Creates a new edge type.

- type_t NewRestrictedEdgeType (const std::wstring &name, type_t tail, type_t head, bool_t neighbors)
  Creates a new restricted edge type.

- oid_t NewEdge (type_t type, oid_t tail, oid_t head)
  Creates a new edge instance.

- oid_t NewEdge (type_t type, attr_t tailAttr, Value &tailV, attr_t headAttr, Value &headV)
  Creates a new edge instance.

- int64_t CountNodes ()
  Gets the number of nodes.

- int64_t CountEdges ()
  Gets the number of edges.

- EdgeData * GetEdgeData (oid_t edge)
  Gets information about an edge.

- oid_t GetEdgePeer (oid_t edge, oid_t node)
  Gets the other end for the given edge.

- void Drop (oid_t oid)
  Drops the given OID.

- void Drop (Objects *objs)
  Drops all the OIDs from the given collection.

- type_t GetObjectType (oid_t oid)
  Gets the Dex node or edge type identifier for the given OID.

- attr_t NewAttribute (type_t type, const std::wstring &name, DataType dt, AttributeKind kind)
  Creates a new attribute.

- attr_t NewAttribute (type_t type, const std::wstring &name, DataType dt, AttributeKind kind, Value &defaultValue)
Creates a new attribute with a default value.

- attr_t NewSessionAttribute (type_t type, DataType dt, AttributeKind kind)
  Creates a new Session attribute.

- attr_t NewSessionAttribute (type_t type, DataType dt, AttributeKind kind, Value &defaultValue)
  Creates a new Session attribute with a default value.

- void SetAttributeDefaultValue (attr_t attr, Value &v)
  Sets a default value for an attribute.

- void IndexAttribute (attr_t attr, AttributeKind kind)
  Updates the index of the given attribute.

- void GetAttribute (oid_t oid, attr_t attr, Value &v)
  Gets the Value for the given attribute and OID.

- Value * GetAttribute (oid_t oid, attr_t attr)
  Gets the Value for the given attribute and OID.

- TextStream * GetAttributeText (oid_t oid, attr_t attr)
  Gets the read-only TextStream for the given text attribute and OID.

- void SetAttributeText (oid_t oid, attr_t attr, TextStream *tstream)
  Sets the writable TextStream for the given text attribute and OID.

- void SetAttribute (oid_t oid, attr_t attr, Value &v)
  Sets the Value for the given attribute and OID.

- AttributeStatistics * GetAttributeStatistics (attr_t attr, bool_t basic)
  Gets statistics from the given attribute.

- int64_t GetAttributeIntervalCount (attr_t attr, Value &lower, bool_t includeLower, Value &higher, bool_t includeHigher)
  Gets how many objects have a value into the given range for the given attribute.

- type_t FindType (const std::wstring &name)
  Gets the Dex type identifier for the given type name.

- Type * GetType (type_t type)
  Gets information about the given type.

- void RemoveType (type_t type)
  Removes the given type.

- attr_t FindAttribute (type_t type, const std::wstring &name)
  Gets the Dex attribute identifier for the given type identifier and attribute name.

- Attribute * GetAttribute (attr_t attr)
  Gets information about the given attribute.
• void RemoveAttribute (attr_t attr)
  Removes the given attribute.

• oid_t FindObject (attr_t attr, Value &v)
  Finds one object having the given Value for the given attribute.

• Objects * Select (type_t type)
  Selects all OIDs belonging to the given type.

• Objects * Select (attr_t attr, Condition cond, const Value &v)
  Selects all OIDs satisfying the given condition for the given attribute.

• Objects * Select (attr_t attr, Condition cond, const Value &lower, const Value &higher)
  Selects all OIDs satisfying the given condition for the given attribute.

• Objects * Explode (oid_t oid, type_t etype, EdgesDirection dir)
  Selects all edges from or to the given node OID and for the given edge type.

• Objects * Explode (Objects *objs, type_t etype, EdgesDirection dir)
  Selects all edges from or to each of the node OID in the given collection and for the given edge type.

• int64_t Degree (oid_t oid, type_t etype, EdgesDirection dir)
  Gets the number of edges from or to the given node OID and for the given edge type.

• Objects * Neighbors (oid_t oid, type_t etype, EdgesDirection dir)
  Selects all neighbor nodes from or to the given node OID and for the given edge type.

• Objects * Neighbors (Objects *objs, type_t etype, EdgesDirection dir)
  Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

• Objects * Edges (type_t etype, oid_t tail, oid_t head)
  Gets all the edges of the given type between two given nodes (tail and head).

• oid_t FindEdge (type_t etype, oid_t tail, oid_t head)
  Gets any of the edges of the given type between two given nodes (tail and head).

• Objects * Tails (Objects *edges)
  Gets all the tails from the given edges collection.

• Objects * Heads (Objects *edges)
  Gets all the heads from the given edges collection.

• void TailsAndHeads (Objects *edges, Objects *tails, Objects *heads)
  Gets all the tails and heads from the given edges collection.

• TypeList * FindNodeTypes ()
  Gets all existing Dex node type identifiers.
• **TypeList * FindEdgeTypes ()**
  
  *Gets all existing *Dex* edge type identifiers.*

• **TypeList * FindTypes ()**
  
  *Gets all existing *Dex* node and edge type identifiers.*

• **AttributeList * FindAttributes (type_t type)**
  
  *Gets all existing *Dex* attribute identifiers for the given type identifier.*

• **AttributeList * GetAttributes (oid_t oid)**
  
  *Gets all *Dex* attribute identifiers with a non-NULL value for the given *Dex* OID.*

• **Values * GetValues (attr_t attr)**
  
  *Gets the *Value* collection for the given attribute.*

• **void DumpData (const std::wstring &file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)**
  
  *Dumps logical data to a file.*

• **void DumpStorage (const std::wstring &file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)**
  
  *Dumps internal storage data to a file.*

• **void Export (const std::wstring &file, ExportType type, ExportManager *e)**
  
  *Exports the *Graph.*

• **void Backup (const std::wstring &file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)**
  
  *Dumps all the data to a backup file.*

**Friends**

• class *Session*

• class *Values*

• class *ValuesIterator*

5.27.1 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 'DEX graph database' sections in the DEX User Manual for more details on the use of this class.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.27.2 Member Function Documentation

5.27.2.1 type_t Graph::NewNodeType (const std::wstring & name)
Creates a new node type.

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

Returns:
   Unique Dex type identifier.

5.27.2.2 oid_t Graph::NewNode (type_t type)
Creates a new node instance.

Parameters:
   type [in] Dex type identifier.

Returns:
   Unique OID of the new node instance.

5.27.2.3 type_t Graph::NewEdgeType (const std::wstring & name, bool_t directed, bool_t neighbors)
Creates a new edge type.

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

Returns:
   Unique Dex type identifier.

5.27.2.4 type_t Graph::NewRestrictedEdgeType (const std::wstring & name, type_t tail, type_t head, bool_t neighbors)
Creates a new restricted edge type.

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

Returns:
   Unique Dex type identifier.
5.27 Graph Class Reference

5.27.2.5 oid_t Graph::NewEdge (type_t type, oid_t tail, oid_t head)

Creates a new edge instance.

Parameters:
- \textit{type} [in] Dex type identifier.
- \textit{tail} [in] Source Dex OID.
- \textit{head} [in] Target Dex OID.

Returns:
Unique OID of the new edge instance.

5.27.2.6 oid_t Graph::NewEdge (type_t type, attr_t tailAttr, Value & tailV, attr_t headAttr, Value & headV)

Creates a new edge instance.

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

Parameters:
- \textit{type} [in] Dex type identifier.
- \textit{headAttr} [in] Dex attribute identifier.
- \textit{headV} [in] Value.

Returns:
Unique OID of the new edge instance.

5.27.2.7 int64_t Graph::CountNodes ()

Gets the number of nodes.

Returns:
The number of nodes.

5.27.2.8 int64_t Graph::CountEdges ()

Gets the number of edges.

Returns:
The number of edges.
5.27 Graph Class Reference

5.27.2.9 EdgeData* Graph::GetEdgeData (oid_t edge)

Gets information about an edge.

Parameters:

  edge [in] Dex edge identifier.

Returns:

  An EdgeData instance.

5.27.2.10 oid_t Graph::GetEdgePeer (oid_t edge, oid_t node)

Gets the other end for the given edge.

Parameters:

  edge [in] Dex edge identifier.
  node [in] Dex node identifier. It must be one of the ends of the edge.

Returns:

  The other end of the edge.

5.27.2.11 void Graph::Drop (oid_t oid)

Drops the given OID.

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

Parameters:

  oid [in] Dex OID to be removed.

5.27.2.12 void Graph::Drop (Objects * objs)

Drops all the OIDs from the given collection.

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

Parameters:

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

5.27.2.13 type_t Graph::GetObjectType (oid_t oid)

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

Parameters:

  oid [in] Dex OID.

Returns:

  Dex node or edge type identifier.
5.27.2.14 attr_t Graph::NewAttribute (type_t type, const std::wstring & name, DataType dt, AttributeKind kind)

Creates a new attribute.

Parameters:

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

Returns:

Unique Dex attribute identifier.

5.27.2.15 attr_t Graph::NewAttribute (type_t type, const std::wstring & name, DataType dt, AttributeKind kind, Value & defaultValue)

Creates a new attribute with a default value.

Parameters:

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

Returns:

Unique Dex attribute identifier.

5.27.2.16 attr_t Graph::NewSessionAttribute (type_t type, DataType dt, AttributeKind kind)

Creates a new Session attribute.

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

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

Parameters:

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

Returns:

Unique Dex attribute identifier.
5.27.2.17  
***attr_t Graph::NewSessionAttribute (type_t type, DataType dt, AttributeKind kind, Value & defaultValue)***

Creates a new Session attribute with a default value.

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

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

**Parameters:**

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

**Returns:**

Unique Dex attribute identifier.

5.27.2.18  
***void Graph::SetAttributeDefaultValue (attr_t attr, Value & v)***

Sets a default value for an attribute.

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

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

**Parameters:**

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

5.27.2.19  
***void Graph::IndexAttribute (attr_t attr, AttributeKind kind)***

Updates the index of the given attribute.

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

**Parameters:**

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

5.27.2.20  
***void Graph::GetAttribute (oid_t oid, attr_t attr, Value & v)***

Gets the Value for the given attribute and OID.

**Parameters:**

- *oid*  [in] Dex OID.
- *attr*  [in] Dex attribute identifier.
- *v*  [in/out] Value for the given attribute and for the given OID.
5.27.2.21 Value* Graph::GetAttribute (oid_t oid, attr_t attr)

Gets the Value for the given attribute and OID.

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

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

Parameters:

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

Returns:

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

5.27.2.22 TextStream* Graph::GetAttributeText (oid_t oid, attr_t attr)

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

Parameters:

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

Returns:

A TextStream. This returns a TextStream to read.

5.27.2.23 void Graph::SetAttributeText (oid_t oid, attr_t attr, TextStream *tstream)

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

Parameters:

oid [in] Dex OID.
attr [in] Dex attribute identifier.
tstream [in] New Text value. This corresponds to a TextStream to write.

5.27.2.24 void Graph::SetAttribute (oid_t oid, attr_t attr, Value &v)

Sets the Value for the given attribute and OID.

Parameters:

oid [in] Dex OID.
attr [in] Dex attribute identifier.
v [in] Value for the given attribute and for the given OID.
5.27.2.25  AttributeStatistics∗ Graph::GetAttributeStatistics (attr_t attr, bool_t basic)

Gets statistics from the given attribute.

Parameters:

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

Returns:

An AttributeStatistics instance.

5.27.2.26  int64_t Graph::GetAttributeIntervalCount (attr_t attr, Value & lower, bool_t includeLower, Value & higher, bool_t includeHigher)

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

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

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

Parameters:

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

Returns:

Number of objects having a value into the given range.

5.27.2.27  type_t Graph::FindType (const std::wstring & name)

Gets the Dex type identifier for the given type name.

Parameters:

- *name* [in] Unique type name.

Returns:

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

5.27.2.28  Type∗ Graph::GetType (type_t type)

Gets information about the given type.
Parameters:

`type` [in] Dex type identifier.

Returns:

The Type for the given Dex type identifier.

### 5.27.2.29 `void Graph::RemoveType (type_t type)`

Removes the given type.

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

Parameters:

`type` [in] Dex type identifier.

### 5.27.2.30 `attr_t Graph::FindAttribute (type_t type, const std::wstring & name)`

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

Parameters:

`type` [in] Dex type identifier.

`name` [in] Unique attribute name.

Returns:

The Dex attribute identifier for the given type and attribute name or Attribute::InvalidAttribute if there is no attribute with the given name for the given type.

### 5.27.2.31 `Attribute* Graph::GetAttribute (attr_t attr)`

Gets information about the given attribute.

Parameters:

`attr` [in] Dex attribute identifier.

Returns:

The Attribute for the given Dex attribute identifier.

### 5.27.2.32 `void Graph::RemoveAttribute (attr_t attr)`

Removes the given attribute.

Parameters:

`attr` [in] Dex attribute identifier.
5.27.2.33  oid_t Graph::FindObject (attr_t attr, Value & v)
Finds one object having the given Value for the given attribute.
If there are more than one, then any of them will be returned. And in case there are no object having this Value, Objects::InvalidOID will be returned.

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

Returns:
  Dex OID or Objects::InvalidOID.

5.27.2.34  Objects* Graph::Select (type_t type)
Selects all OIDs belonging to the given type.

Parameters:
  type [in] Dex type identifier.

Returns:
  Objects instance.

5.27.2.35  Objects* Graph::Select (attr_t attr, Condition cond, const Value & v)
Selects all OIDs satisfying the given condition for the given attribute.

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

Returns:
  Objects instance.

5.27.2.36  Objects* Graph::Select (attr_t attr, Condition cond, const Value & lower, const Value & higher)
Selects all OIDs satisfying the given condition for the given attribute.
This allows to perform the Condition::Between operation, thus it has two Value arguments.

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

**Returns:**

Objects instance.

5.27.2.37  **Objects** Graph::Explode (oid_t *oid, type_t etype, EdgesDirection dir)

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

**Parameters:**

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

**Returns:**

Objects instance.

5.27.2.38  **Objects** Graph::Explode (Objects *objs, type_t etype, EdgesDirection dir)

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

**Parameters:**

- *etype* [in] Dex edge type identifier.
- *dir* [in] Direction.

**Returns:**

Objects instance.

5.27.2.39  **int64_t** Graph::Degree (oid_t *oid, type_t etype, EdgesDirection dir)

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

**Parameters:**

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

**Returns:**

The number of edges.
5.27.2.40 Objects* Graph::Neighbors (oid_t oid, type_t etype, EdgesDirection dir)
Selects all neighbor nodes from or to the given node OID and for the given edge type.

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

Returns:
   Objects instance.

5.27.2.41 Objects* Graph::Neighbors (Objects* objs, type_t etype, EdgesDirection dir)
Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

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

Returns:
   Objects instance.

5.27.2.42 Objects* Graph::Edges (type_t etype, oid_t tail, oid_t head)
Gets all the edges of the given type between two given nodes (tail and head).

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

Returns:
   Objects instance.

5.27.2.43 oid_t Graph::FindEdge (type_t etype, oid_t tail, oid_t head)
Gets any of the edges of the given type between two given nodes (tail and head).
If there are more than one, then any of them will be returned. And in case there are no edge between the
given tail and head, Objects::InvalidOID will be returned.

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

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

5.27.2.44 Objects Graph::Tails (Objects * edges)
Gets all the tails from the given edges collection.

Parameters:

Returns:
The tails collection.

5.27.2.45 Objects Graph::Heads (Objects * edges)
Gets all the heads from the given edges collection.

Parameters:

Returns:
The heads collection.

5.27.2.46 void Graph::TailsAndHeads (Objects * edges, Objects * tails, Objects * heads)
Gets all the tails and heads from the given edges collection.

Parameters:
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.

5.27.2.47 TypeList* Graph::FindNodeTypes ()
Gets all existing Dex node type identifiers.

Returns:
Dex node type identifier list.
5.27.2.48 TypeList* Graph::FindEdgeTypes ()
Gets all existing Dex edge type identifiers.

Returns:
   Dex edge type identifier list.

5.27.2.49 TypeList* Graph::FindTypes ()
Gets all existing Dex node and edge type identifiers.

Returns:
   Dex node and edge type identifier list.

5.27.2.50 AttributeList* Graph::FindAttributes (type_t type)
Gets all existing Dex attribute identifiers for the given type identifier.

Parameters:
   type [in] Dex type identifier.

Returns:
   Dex attribute identifier list.

5.27.2.51 AttributeList* Graph::GetAttributes (oid_t oid)
Gets all Dex attribute identifiers with a non-NULL value for the given Dex OID.

Parameters:
   oid [in] Dex OID.

Returns:
   Dex attribute identifier list.

5.27.2.52 Values* Graph::GetValues (attr_t attr)
Gets the Value collection for the given attribute.

Parameters:
   attr [in] Dex attribute identifier.
5.27.2.53 void Graph::DumpData (const std::wstring & file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)
Dumps logical data to a file.

Parameters:
    file [in] Output file path.

Exceptions:
    
FileNotFoundException If the given file cannot be created.

5.27.2.54 void Graph::DumpStorage (const std::wstring & file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)
Dumps internal storage data to a file.

Parameters:
    file [in] Output file path.

Exceptions:
    
FileNotFoundException If the given file cannot be created.

5.27.2.55 void Graph::Export (const std::wstring & file, ExportType type, ExportManager * e)
Exports the Graph.

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

5.27.2.56 void Graph::Backup (const std::wstring & file) throw (dex::gdb::FileNotFoundException, dex::gdb::Error)
Dumps all the data to a backup file.
See Dex::Restore.

Parameters:
    file [in] Output backup file path.

Exceptions:
    
FileNotFoundException If the given file cannot be created.

The documentation for this class was generated from the following file:

* Graph.h
5.28 GraphExport Class Reference

Stores the graph exporting values.

Public Member Functions

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

- **virtual ~GraphExport ()**
  
  Destructor.

- **void SetDefaults ()**
  
  Sets to default values.

- **const std::wstring & GetLabel () const**
  
  Gets the graph label.

- **void SetLabel (const std::wstring &l)**
  
  Sets the graph label.

5.28.1 Detailed Description

Stores the graph exporting values.

Author:

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

5.28.2 Member Function Documentation

5.28.2.1 **const std::wstring& GraphExport::GetLabel () const** [inline]

Gets the graph label.

Returns:

The graph label.

5.28.2.2 **void GraphExport::SetLabel (const std::wstring &l)** [inline]

Sets the graph label.

Parameters:

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

The documentation for this class was generated from the following file:

- **Export.h**
5.29 Handler< T > Class Template Reference

Handles a reference.

Inheritance diagram for Handler< T >:

Collaboration diagram for Handler< T >:

Public Member Functions

- **Handler ()**
  
  *Creates a new instance.*

- **Handler (T *h)**
  
  *Creates a new instance with the given reference.*

- **virtual ~Handler ()**
  
  *Destructor.*

- **T = GetHandler ()**
  
  *Gets the handled reference.*
5.30 Int32List Class Reference

- const T * GetHandler () const
  Gets the handled reference.

Protected Member Functions

- void SetHandler (T *h)
  Sets the handled reference.

- void FreeHandler ()
  Frees (deletes) the handled reference.

- bool_t IsNull () const
  Gets if the handler is NULL.

5.29.1 Detailed Description

template<typename T> class Handler<T>

 Handles a reference.
The handled reference is automatically destroyed (deleted) when the instance is destroyed.

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

The documentation for this class was generated from the following file:

- Handler.h

5.30 Int32List Class Reference

Dex 32-bit signed integer list.

Public Member Functions

- int32_t Count () const
  Number of elements in the list.

- Int32ListIterator * Iterator ()
  Gets a new Int32ListIterator.

- Int32List ()
  Constructor.

- Int32List (const std::vector<int32_t> &v)
  Constructor.
• void Add (int32_t value)
  Adds an 32-bit signed integer at the end of the list.

• void Clear ()
  Clears the list.

5.30.1 Detailed Description

Dex 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

5.30.2 Constructor & Destructor Documentation

5.30.2.1 Int32List::Int32List ()
Constructor.
This creates an empty list.

5.30.2.2 Int32List::Int32List (const std::vector<int32_t> & v)
Constructor.

Parameters:
  v [in] Vector.

5.30.3 Member Function Documentation

5.30.3.1 int32_t Int32List::Count () const [inline]
Number of elements in the list.

Returns:
  Number of elements in the list.

5.30.3.2 Int32ListIterator* Int32List::Iterator ()
Gets a new Int32ListIterator.

Returns:
  Int32ListIterator instance.
5.30.3.3  void Int32List::Add (int32_t value) [inline]
Adds an 32-bit signed integer at the end of the list.

Parameters:
  type  [in] The integer.

The documentation for this class was generated from the following file:

  • Graph_data.h

5.31  Int32ListIterator Class Reference

Int32List iterator class.

Public Member Functions

  • int32_t Next ()
    Moves to the next element.

  • bool_t HasNext ()
    Gets if there are more elements.

Friends

  • class Int32List

5.31.1  Detailed Description

Int32List iterator class.
Iterator to traverse all the integer into a Int32List instance.

Author:

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

5.31.2  Member Function Documentation

5.31.2.1  int32_t Int32ListIterator::Next () [inline]
Moves to the next element.

Returns:

  The next element.
5.31.2.2 bool_t Int32ListIterator::HasNext () [inline]

Gets if there are more elements.

**Returns:**

TRUE if there are more elements, FALSE otherwise.

The documentation for this class was generated from the following file:

- Graph_data.h

5.32 IOError Class Reference

IO error class.

Inheritance diagram for IOError:

```
IOError
   
SystemError
   
Error
   
Exception
```

Collaboration diagram for IOError:

```
IOError
   
SystemError
   
Error
   
Exception
```

**Public Member Functions**

- IOError ()
  
  *Creates a new instance.*
• IOError (const std::string &mess)
  Creates a new instance.

• virtual ~IOError ()
  Destructor.

• const std::string & Message () const
  Gets the message of the exception.

• void SetMessage (const std::string &mess)
  Sets the message of the exception.

Static Public Member Functions

• static Error NewError (int32_t coreErrorCode)
  Creates a new Error instance from a dexcore error code.

Protected Attributes

• std::string message
  Message of the exception.

5.32.1 Detailed Description

IO error class.

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

5.32.2 Constructor & Destructor Documentation

5.32.2.1 IOError::IOError (const std::string & mess)
Creates a new instance.

Parameters:

  mess [in] Message of the exception.

5.32.3 Member Function Documentation

5.32.3.1 static Error Error::NewError (int32_t coreErrorCode) [static, inherited]
Creates a new Error instance from a dexcore error code.

Parameters:

Returns:

Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.

5.32.3.2 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.32.3.3 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:


The documentation for this class was generated from the following file:

• Exception.h

5.33 IOException Class Reference

IO exception class.

Inheritance diagram for IOException:

Collaboration diagram for IOException:
5.33 IOException Class Reference

Public Member Functions

• IOException ()
  Creates a new instance.

• IOException (const std::string &mess)
  Creates a new instance.

• virtual ~IOException ()
  Destructor.

• const std::string & Message () const
  Gets the message of the exception.

• void SetMessage (const std::string &mess)
  Sets the message of the exception.

Protected Attributes

• std::string message
  Message of the exception.

5.33.1 Detailed Description

IO exception class.

Author:

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

5.33.2 Constructor & Destructor Documentation

5.33.2.1 IOException::IOException (const std::string & mess)

Creates a new instance.

Parameters:

  mess  [in] Message of the exception.

5.33.3 Member Function Documentation

5.33.3.1 const std::string & Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

  The message of the exception.
5.33.3.2 void Exception::setMessage (const std::string & mess) [inherited]
Sets the message of the exception.

**Parameters:**


The documentation for this class was generated from the following file:

- Exception.h

5.34 LicenseError Class Reference

License error class.

Inheritance diagram for LicenseError:

```
+-----------------+  
| LicenseError    |
+-----------------+  
|                 v
| AppError        |
+-----------------+  
|                 v
| Error           |
+-----------------+  
|                 v
| Exception       |
```

Collaboration diagram for LicenseError:

```
+-----------------+  
| LicenseError    |
+-----------------+  
|                 v
| AppError        |
+-----------------+  
|                 v
| Error           |
+-----------------+  
|                 v
| Exception       |
```

**Public Member Functions**

- **LicenseError ()**

  * Creates a new instance.*
LicenseError (const std::string &mess)

Creates a new instance.

virtual ~LicenseError ()

Destructor.

const std::string & Message () const

 Gets the message of the exception.

void SetMessage (const std::string &mess)

Sets the message of the exception.

Static Public Member Functions

static Error NewError (int32_t coreErrorCode)

Creates a new Error instance from a dexcore error code.

Protected Attributes

std::string message

Message of the exception.

5.34.1 Detailed Description

License error class.

Author:

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

5.34.2 Constructor & Destructor Documentation

5.34.2.1 LicenseError::LicenseError (const std::string & mess)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.34.3 Member Function Documentation

5.34.3.1 static Error Error::NewError (int32_t coreErrorCode) [static, inherited]

Creates a new Error instance from a dexcore error code.

Parameters:

Returns:
Depending on the given dexcore error, this may return an Error instance or a specific Error subclass instance.

5.34.3.2  const std::string& Exception::Message () const  [inherited]
Gets the message of the exception.
It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:
The message of the exception.

5.34.3.3  void Exception::SetMessage (const std::string & mess)  [inherited]
Sets the message of the exception.

Parameters:

The documentation for this class was generated from the following file:

- Exception.h

5.35  NodeExport Class Reference
Stores the node exporting values.

Public Member Functions

- NodeExport ()
  *Creates a new instance.*

- virtual ~NodeExport ()
  *Destructor.*

- void SetDefaults ()
  *Sets to default values.*

- const std::wstring & GetLabel () const
  *Gets the node label.*

- void SetLabel (const std::wstring &l)
  *Sets the node label.*

- NodeShape GetShape () const
  *Gets the node shape.*
• **void SetShape (NodeShape s)**
  
  Sets the node shape.

• **ColorRGB GetColorRGB () const**
  
  Gets the node color.

• **void SetColorRGB (ColorRGB c)**
  
  Sets the node color.

• **ColorRGB GetLabelColorRGB () const**
  
  Gets the node label color.

• **void SetLabelColorRGB (ColorRGB c)**
  
  Sets the node label color.

• **int32_t GetHeight () const**
  
  Gets the node height.

• **void SetHeight (int32_t h)**
  
  Sets the node height.

• **int32_t GetWidth () const**
  
  Gets the node width.

• **void SetWidth (int32_t w)**
  
  Gets the node width.

• **bool_t IsFit () const**
  
  Gets whether the node size is fitted to the label or not.

• **void SetFit (bool_t f)**
  
  Sets the node fit property.

• **int32_t GetFontSize () const**
  
  Gets the node label font size.

• **void SetFontSize (int32_t s)**
  
  Sets the node label font size.

### 5.35.1 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 (Ox000000, Black).
Height: 25px.
Width: 25px.
Fit: TRUE.
Font size: 10.

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

5.35.2 Member Function Documentation

5.35.2.1 const std::wstring& NodeExport::GetLabel () const [inline]
Gets the node label.

Returns:
The node label.

5.35.2.2 void NodeExport::SetLabel (const std::wstring & l) [inline]
Sets the node label.

Parameters:
   l [in] The node label.

5.35.2.3 NodeShape NodeExport::GetShape () const [inline]
Gets the node shape.

Returns:
The node shape.

5.35.2.4 void NodeExport::SetShape (NodeShape s) [inline]
Sets the node shape.

Parameters:
   s [in] The node shape.

5.35.2.5 ColorRGB NodeExport::GetColorRGB () const [inline]
Gets the node color.

Returns:
The node color.
5.35.2.6 void NodeExport::SetColorRGB (ColorRGB c)  \[inline\]
Sets the node color.

Parameters:
  
  c The node color.

5.35.2.7 ColorRGB NodeExport::GetLabelColorRGB () const  \[inline\]
Gets the node label color.

Returns:
  
  The node label color.

5.35.2.8 void NodeExport::SetLabelColorRGB (ColorRGB c)  \[inline\]
Sets the node label color.

Parameters:
  
  c [in] The node label color.

5.35.2.9 int32_t NodeExport::GetHeight () const  \[inline\]
Gets the node height.

Returns:
  
  The node height in pixels.

5.35.2.10 void NodeExport::SetHeight (int32_t h)  \[inline\]
Sets the node height.

Parameters:
  
  h [in] The node height in pixels.

5.35.2.11 int32_t NodeExport::GetWidth () const  \[inline\]
Gets the node width.

Returns:
  
  The node width in pixels.

5.35.2.12 void NodeExport::SetWidth (int32_t w)  \[inline\]
Gets the node width.

Parameters:
  
  w The node width in pixels.
5.35.2.13  bool_t NodeExport::IsFit () const [inline]

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'.

5.35.2.14  void NodeExport::SetFit (bool_t f) [inline]

Sets the node fit property.

**Parameters:**

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

5.35.2.15  int32_t NodeExport::GetFontSize () const [inline]

Gets the node label font size.

**Returns:**

The node label font size.

5.35.2.16  void NodeExport::SetFontSize (int32_t s) [inline]

Sets the node label font size.

**Parameters:**

s [in] The node label font size.

The documentation for this class was generated from the following file:

- Export.h

5.36  NodeTypeExporter Class Reference

**NodeTypeExporter** class.

Inheritance diagram for NodeTypeExporter:
Collaboration diagram for NodeTypeExporter:

![Collaboration Diagram](image)

Public Member Functions

- **NodeTypeExporter ()**
  
  Creates a new instance.

- **NodeTypeExporter (RowWriter &rw, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs)**
  
  Creates a new instance.

- **virtual ~NodeTypeExporter ()**
  
  Destructor.

- **void Run () throw (dex::gdb::IOException, dex::gdb::Error)**
  
  See TypeExporter::Run.

- **void Register (TypeExporterListener &tel)**
  
  Registers a new listener.

- **void SetRowWriter (RowWriter &rw)**
  
  Sets the output data destination.

- **void SetGraph (dex::gdb::Graph &g)**
  
  Sets the graph that will be exported.

- **void SetType (dex::gdb::type_t t)**
  
  Sets the type to be exported.

- **void SetAttributes (dex::gdb::AttributeList &attrs)**
  
  Sets the list of Attributes.

- **void SetFrequency (dex::gdb::int32_t freq)**
  
  Sets the frequency of listener notification.

- **void SetHeader (dex::gdb::bool_t header)**
  
  Sets the presence of a header row.
Protected Member Functions

- `dex::gdb::bool_t CanRun ()`
  Checks that all the required settings are ready to run.

- `void NotifyListeners (const TypeExporterEvent &ev)`
  Notifies progress to all registered listeners.

- `void RunProcess () throw (dex::gdb::IOException, dex::gdb::Error)`
  Runs export process.

- `void SetHeadAttribute (dex::gdb::attr_t attr)`
  Sets the attribute that will be used to get the value to be dumped for the head of the edge.

- `void SetHeadPosition (dex::gdb::int32_t pos)`
  Sets the position (index column) of the head attribute in the exported data.

- `void SetTailAttribute (dex::gdb::attr_t attr)`
  Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

- `void SetTailPosition (dex::gdb::int32_t pos)`
  Sets the position (index column) of the tail attribute in the exported data.

5.36.1 Detailed Description

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

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

5.36.2 Constructor & Destructor Documentation

5.36.2.1 `NodeTypeExporter::NodeTypeExporter (RowWriter & rw, dex::gdb::Graph & g, dex::gdb::type_t t, dex::gdb::AttributeList & attrs) [inline]`
Creates a new instance.

Parameters:
- `g` [in] `Graph`.
- `t` [in] `Type` identifier.
- `attrs` [in] `Attribute` identifiers to be exported.
5.36.3  Member Function Documentation

5.36.3.1  
\texttt{dex::gdb::bool\_t TypeExporter::CanRun ()} \ [protected, inherited]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.36.3.2  void TypeExporter::NotifyListeners (const TypeExporterEvent & \textit{ev}) \ [protected, inherited]

Notifies progress to all registered listeners.

Parameters:

\textit{ev} \ [in] Progress event to be notified.

5.36.3.3  void TypeExporter::RunProcess () throw (dex::gdb::IOException, dex::gdb::Error) \ [protected, inherited]

Runs export process.

Exceptions:

\textit{IOException} If bad things happen writting to the RowWriter.

5.36.3.4  void TypeExporter::SetHeadAttribute (dex::gdb::attr\_t \textit{attr}) \ [protected, inherited]

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

This method is protected because only the Edge exporters should have it.

Parameters:

\textit{attr} \ [in] Head Attribute

Reimplemented in EdgeTypeExporter.

5.36.3.5  void TypeExporter::SetHeadPosition (dex::gdb::int32\_t \textit{pos}) \ [protected, inherited]

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

This method is protected because only the Edge exporters should have it.

Parameters:

\textit{pos} \ [in] Head position

Reimplemented in EdgeTypeExporter.
5.36.3.6 void TypeExporter::SetTailAttribute (dex::gdb::attr_t attr) [protected, inherited]
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Tail Attribute

Reimplemented in EdgeTypeExporter.

5.36.3.7 void TypeExporter::SetTailPosition (dex::gdb::int32_t pos) [protected, inherited]
Sets the position (index column) of the tail attribute in the exported data.
This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Tail position

Reimplemented in EdgeTypeExporter.

5.36.3.8 void TypeExporter::Register (TypeExporterListener & tel) [inherited]
Registers a new listener.

Parameters:

tel [in] TypeExporterListener to be registered.

5.36.3.9 void TypeExporter::SetRowWriter (RowWriter & rw) [inherited]
Sets the output data destination.

Parameters:

rw [in] Input RowWriter.

5.36.3.10 void TypeExporter::SetGraph (dex::gdb::Graph & g) [inherited]
Sets the graph that will be exported.

Parameters:

g [in] Graph.

5.36.3.11 void TypeExporter::SetType (dex::gdb::type_t t) [inherited]
Sets the type to be exported.

Parameters:

t [in] Type identifier.
5.36.3.12  void TypeExporter::SetAttributes (dex::gdb::AttributeList & attrs)  [inherited]
Sets the list of Attributes.

Parameters:
   * attrs [in] Attribute identifiers to be exported

5.36.3.13  void TypeExporter::SetFrequency (dex::gdb::int32_t freq)  [inherited]
Sets the frequency of listener notification.
freq [in] Frequency in number of rows managed to notify progress to all listeners

5.36.3.14  void TypeExporter::SetHeader (dex::gdb::bool_t header)  [inherited]
Sets the presence of a header row.

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

The documentation for this class was generated from the following file:

* NodeTypeExporter.h

5.37  NodeTypeLoader Class Reference

NodeTypeLoader class.

Inheritance diagram for NodeTypeLoader:

```
TypeLoader

NodeTypeLoader
```

Collaboration diagram for NodeTypeLoader:

```
RowReader

rowReader

TypeLoader

NodeTypeLoader
```
Public Member Functions

- **NodeTypeLoader ()**
  
  Creates a new instance.

- **NodeTypeLoader (RowReader &rr, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs, dex::gdb::Int32List &attrsPos)**

  Creates a new instance.

- **virtual ~NodeTypeLoader ()**
  
  Destructor.

- **void Run () throw (dex::gdb::IOException, dex::gdb::Error)**
  
  See TypeLoader::Run.

- **void RunTwoPhases () throw (dex::gdb::IOException, dex::gdb::Error)**
  
  See TypeLoader::RunTwoPhases.

- **void RunNPhases (dex::gdb::int32_t partitions) throw (dex::gdb::IOException, dex::gdb::Error)**
  
  See TypeLoader::RunNPhases.

- **void SetLogError (const std::wstring &path) throw (dex::gdb::IOException)**
  
  Sets a log error file.

- **void SetLogOff ()**

  Turns off all the error reporting.

- **void Register (TypeLoaderListener &tel)**

  Registers a new listener.

- **void SetRowReader (RowReader &rr)**

  Sets the input data source.

- **void SetGraph (dex::gdb::Graph &g)**

  Sets the graph where the data will be loaded.

- **void SetLocale (const std::wstring &localeStr)**

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

- **void SetType (dex::gdb::type_t t)**

  Sets the type to be loaded.

- **void SetAttributes (dex::gdb::AttributeList &attrs)**

  Sets the list of Attributes.

- **void SetAttributePositions (dex::gdb::Int32List &attrsPos)**

  Sets the list of attribute positions.

- **void SetTimestampFormat (const std::wstring &timestampFormat)**

  Sets a specific timestamp format.
• void SetFrequency (dex::gdb::int32_t freq)
  
  Sets the frequency of listener notification.

Protected Types

• enum Mode {
  ONE_PHASE,
  TWO_PHASES,
  N_PHASES }
  
  Load can work in different ways.

Protected Member Functions

• dex::gdb::bool_t CanRun ()
  
  Checks that all the required settings are ready to run.

• void Run (Mode ph, dex::gdb::int32_t par) throw (dex::gdb::IOException, dex::gdb::Error)
  
  Runs load process.

• void NotifyListeners (const TypeLoaderEvent &ev)
  
  Notifies progress to all registered listeners.

• void SetHeadAttribute (dex::gdb::attr_t attr)
  
  Sets the attribute that will be used to find the head of the edge.

• void SetHeadPosition (dex::gdb::int32_t pos)
  
  Sets the position of the head attribute in the source data.

• void SetTailAttribute (dex::gdb::attr_t attr)
  
  Sets the attribute that will be used to find the tail of the edge.

• void SetTailPosition (dex::gdb::int32_t pos)
  
  Sets the position of the tail attribute in the source data.

5.37.1 Detailed Description

NodeTypeLoader class.

Specific TypeLoader implementation for node types.

Check out the ‘Data import’ section in the DEX User Manual for more details on this.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
5.37 Member Enumeration Documentation

5.37.2.1 enum NodeTypeLoader::Mode [protected, inherited]

Load can work in different ways.

**Enumerator:**

- **ONE_PHASE**  Performs the load in a phases.
  Load all objects an attributes at the same time.

- **TWO_PHASES**  Performs the load in two phases.
  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.

- **N_PHASES**  Performs the load in N phases.
  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 to the RowReader are necessary.
  Working on this mode it is necessary to build a temporary file.

5.37.3 Constructor & Destructor Documentation

5.37.3.1 NodeTypeLoader::NodeTypeLoader (RowReader & rr, dex::gdb::Graph & g, dex::gdb::type_t t, dex::gdb::AttributeList & attrs, dex::gdb::Int32List & attrsPos) [inline]

Creates a new instance.

**Parameters:**

- **rr**  [in] Input RowReader.
- **g**  [in] Graph.
- **t**  [in] Type identifier.
- **attrs**  [in] Attribute identifiers to be loaded.
- **attrsPos**  [in] Attribute positions (column index >=0).

5.37.4 Member Function Documentation

5.37.4.1 dex::gdb::bool_t TypeLoader::CanRun () [protected, inherited]

Checks that all the required settings are ready to run.

**Returns:**

Returns true if all the settings are ready.

5.37.4.2 void TypeLoader::Run (Mode ph, dex::gdb::int32_t par) throw (dex::gdb::IOException, dex::gdb::Error) [protected, inherited]

Runs load process.

**Exceptions:**

- **IOException**  If bad things happen reading from the RowReader.
Parameters:

- **ph** [in] The load mode.
- **par** [in] Number of horizontal partitions to perform the load.

5.37.4.3 void TypeLoader::NotifyListeners (const TypeLoaderEvent & ev) [protected, inherited]
Notifies progress to all registered listeners.

Parameters:

- **ev** [in] Progress event to be notified.

5.37.4.4 void TypeLoader::SetHeadAttribute (dex::gdb::attr_t attr) [protected, inherited]
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

Reimplemented in EdgeTypeLoader.

5.37.4.5 void TypeLoader::SetHeadPosition (dex::gdb::int32_t pos) [protected, inherited]
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

Reimplemented in EdgeTypeLoader.

5.37.4.6 void TypeLoader::SetTailAttribute (dex::gdb::attr_t attr) [protected, inherited]
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

Reimplemented in EdgeTypeLoader.
5.37.4.7  void TypeLoader::SetTailPosition (dex::gdb::int32_t pos) [protected, inherited]

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

Reimplemented in EdgeTypeLoader.

5.37.4.8  void TypeLoader::SetLogError (const std::wstring & path) throw (dex::gdb::IOException) [inherited]

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.

Exceptions:

IOException  If bad things happen opening the file.

5.37.4.9  void TypeLoader::SetLogOff () [inherited]

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.

5.37.4.10 void TypeLoader::Register (TypeLoaderListener & tel) [inherited]

Registers a new listener.

Parameters:

← tel  TypeLoaderListener to be registered.

5.37.4.11 void TypeLoader::SetRowReader (RowReader & rr) [inherited]

Sets the input data source.

Parameters:

rr  [in] Input RowReader.

5.37.4.12 void TypeLoader::SetGraph (dex::gdb::Graph & g) [inherited]

Sets the graph where the data will be loaded.

Parameters:

   g  [in] Graph.
5.38 NoSuchElementException Class Reference

NoSuchElementException

Inheritance diagram for NoSuchElementException:

Exception

NoSuchElementException

5.38 NoSuchElementException Class Reference

No such element exception class.

Inheritance diagram for NoSuchElementException:

Exception

NoSuchElementException
Collaboration diagram for NoSuchElementException:

```
Exception
NoSuchElementException
```

**Public Member Functions**

- `NoSuchElementException ()`
  Creates a new instance.

- `NoSuchElementException (const std::string &mess)`
  Creates a new instance.

- virtual `~NoSuchElementException ()`
  Destructor.

- `const std::string & Message () const`
  Gets the message of the exception.

- void `setMessage` (const std::string &mess)
  Sets the message of the exception.

**Protected Attributes**

- `std::string message`
  Message of the exception.

### 5.38.1 Detailed Description

No such element exception class.

**Author:**

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

### 5.38.2 Constructor & Destructor Documentation

#### 5.38.2.1 `NoSuchElementException::NoSuchElementException (const std::string & mess)`

Creates a new instance.

**Parameters:**

5.38.3 Member Function Documentation

5.38.3.1 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.
It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.38.3.2 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:


The documentation for this class was generated from the following file:

• Exception.h

5.39 Objects Class Reference

Object identifier set class.

Inheritance diagram for Objects:

```
Objects
Handler< dexcore::Objects >
Handler< T >
< dexcore::Objects >
```

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
Collaboration diagram for Objects:

Public Member Functions

- `virtual ~Objects ()`
  
  Destructor.

- `Objects * Copy ()`
  
  Creates a new `Objects` instance as a copy of the given one.

- `int64_t Count ()`
  
  Gets the number of elements into the collection.

- `bool_t Add ( oid_t e )`
  
  Adds an element into the collection.

- `bool_t Exists ( oid_t e )`
  
  Gets if the given element exists into the collection.

- `oid_t Any () throw (dex::gdb::NoSuchElementException, dex::gdb::Error)`
5.39 Objects Class Reference

- **bool_t Remove (oid_t e)**
  - Gets an element from the collection.
  - Removes an element from the collection.

- **void Clear ()**
  - Clears the collection removing all its elements.

- **int64_t Union (Objects *objs)**
  - Performs the union operation.

- **int64_t Intersection (Objects *objs)**
  - Performs the intersection operation.

- **int64_t Difference (Objects *objs)**
  - Performs the difference operation.

- **bool_t Equals (Objects *objs)**
  - Checks if the given Objects contains the same information.

- **bool_t Contains (Objects *objs)**
  - Check if this objects contains the other one.

- **int64_t Copy (Objects *objs)**
  - Performs the copy operation.

- **Objects * Sample (Objects *exclude, int64_t samples)**
  - Creates a new Objects instance which is a sample of the calling one.

- **ObjectsIterator * Iterator ()**
  - Gets an ObjectsIterator.

- **ObjectsIterator * IteratorFromIndex (int64_t index)**
  - Gets an ObjectsIterator skipping index elements.

- **ObjectsIterator * IteratorFromElement (oid_t e)**
  - Gets an ObjectsIterator starting from the given element.

**Static Public Member Functions**

- **static Objects * CombineUnion (Objects *objs1, Objects *objs2)**
  - Creates a new Objects instance which is the union of the two given.

- **static Objects * CombineIntersection (Objects *objs1, Objects *objs2)**
  - Creates a new Objects instance which is the intersection of the two given.

- **static Objects * CombineDifference (Objects *objs1, Objects *objs2)**
  - Creates a new Objects instance which is the difference of the two given.
Static Public Attributes

- static const oid_t InvalidOID
  
  Invalid OID constant.

Friends

- class Session
- class Graph
- class ObjectsIterator

5.39.1 Detailed Description

Object identifier set class.

It stores a collection of Dex 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

5.39.2 Member Function Documentation

5.39.2.1 Objects* Objects::Copy ()

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

Returns:

The new Objects instance.

5.39.2.2 int64_t Objects::Count ()

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

5.39.2.3 bool_t Objects::Add (oid_t 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.

5.39.2.4 bool_t Objects::Exists (oid_t 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.

5.39.2.5 oid_t Objects::Any () throw (dex::gdb::NoSuchElementException, dex::gdb::Error)
Gets an element from the collection.

Returns:
Any element from the collection.

Exceptions:
   NoSuchElementException whether the collection is empty.

5.39.2.6 bool_t Objects::Remove (oid_t 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.

5.39.2.7 int64_t Objects::Union (Objects * objs)
Performs the union operation.
This adds all existing elements of the given Objects instance to the Objects calling instance.

Parameters:
   objs [in] Objects instance.

Returns:
Number of elements into the collection once the operation has been executed.
5.39.2.8 int64_t Objects::Intersection (Objects * objs)
Performs the intersection operation.
Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

Parameters:
   
   objs [in] Objects instance.

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

5.39.2.9 int64_t Objects::Difference (Objects * objs)
Performs the difference operation.
This updates the Objects calling instance removing those existing elements at the given Objects instance.

Parameters:
   
   objs [in] Objects instance.

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

5.39.2.10 bool_t Objects::Equals (Objects * objs)
Checks if the given Objects contains the same information.

Parameters:
   
   objs [in] Objects instance.

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

5.39.2.11 bool_t Objects::Contains (Objects * objs)
Check if this objects contains the other one.

Parameters:
   
   objs Objects collection.

Returns:
   
   True if it contains the given object.
5.39.2.12  static Objects* Objects::CombineUnion (Objects * objs1, Objects * objs2) [static]

Creates a new Objects instance which is the union of the two given.
Two given Objects belong to the same Session.

Parameters:

Returns:
   New Objects instance.

5.39.2.13  static Objects* Objects::CombineIntersection (Objects * objs1, Objects * objs2) [static]

Creates a new Objects instance which is the intersection of the two given.
Two given Objects belong to the same Session.

Parameters:

Returns:
   New Objects instance.

5.39.2.14  static Objects* Objects::CombineDifference (Objects * objs1, Objects * objs2) [static]

Creates a new Objects instance which is the difference of the two given.
Two given Objects belong to the same Session.

Parameters:

Returns:
   New Objects instance.

5.39.2.15   int64_t Objects::Copy (Objects * objs)

Performs the copy operation.
This updates the Objects calling instance and copies the given Objects instance.

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

5.39.2.16 Objects::Sample (Objects * _exclude_, int64_t _samples_)
Creates a new Objects instance which is a sample of the calling one.

Parameters:
clude [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.

5.39.2.17 ObjectsIterator* Objects::Iterator ()
Gets an ObjectsIterator.

Returns:
ObjectsIterator instance.

5.39.2.18 ObjectsIterator* Objects::IteratorFromIndex (int64_t _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.

5.39.2.19 ObjectsIterator* Objects::IteratorFromElement (oid_t _e_)
Gets an ObjectsIterator starting from the given element.
Objects collection has no order, so this method is implementation-dependent.

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

Returns:
ObjectsIterator instance.

The documentation for this class was generated from the following file:

- Objects.h
5.40 ObjectsIterator Class Reference

ObjectsIterator class.

Collaboration diagram for ObjectsIterator:

---

Public Member Functions

- **virtual ~ObjectsIterator ()**
  
  Destructor.

- **bool_t HasNext ()**
  
  Gets if there are more elements to traverse.

- **oid_t Next ()**
  
  Gets the next element to traverse.
5.41 OIDList Class Reference

Dex object identifier list.

Public Member Functions

- `int32_t Count () const`
  
  Number of elements in the list.

- `OIDListIterator * Iterator ()`
  
  Gets a new `OIDListIterator`.

- `OIDList ()`
  
  Constructor.
• **OIDList (int32_t N)**
  Constructor.

• **OIDList (const std::vector< oid_t > &v)**
  Constructor.

• void **Add (oid_t attr)**
  Adds a Dex object identifier at the end of the list.

• void **Set (int32_t pos, oid_t oid)**
  Sets a Dex object identifier at the specified position of the list.

• void **Clear ()**
  Clears the list.

### 5.41.1 Detailed Description

Dex object identifier list.

It stores a Dex object identifier list.

Use OIDListIterator to access all elements into this collection.

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

### 5.41.2 Constructor & Destructor Documentation

#### 5.41.2.1 OIDList::OIDList ()

Constructor.

This creates an empty list.

#### 5.41.2.2 OIDList::OIDList (int32_t N)

Constructor.

This creates a list with N invalid oids.

#### 5.41.2.3 OIDList::OIDList (const std::vector< oid_t > & v)

Constructor.

Parameters:

  • v [in] Vector.

### 5.41.3 Member Function Documentation

#### 5.41.3.1 int32_t OIDList::Count () const [inline]

Number of elements in the list.
5.42 OIDListIterator Class Reference

Returns:
Number of elements in the list.

5.41.3.2 OIDListIterator* OIDList::Iterator ()

Gets a new OIDListIterator.

Returns:
OIDListIterator instance.

5.41.3.3 void OIDList::Add (oid_t attr) [inline]

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

Parameters:
  attr [in] Dex object identifier.

5.41.3.4 void OIDList::Set (int32_t pos, oid_t oid) [inline]

Sets a Dex object identifier at the specified position of the list.

Parameters:
  pos [in] List position [0..Count()-1].
  oid [in] Dex object identifier.

The documentation for this class was generated from the following file:

• Graph_data.h

5.42 OIDListIterator Class Reference

OIDList iterator class.

Public Member Functions

• oid_t Next ()
  Moves to the next element.

• bool_t HasNext ()
  Gets if there are more elements.

Friends

• class OIDList
5.42.1 Detailed Description

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

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

5.42.2 Member Function Documentation

5.42.2.1 oid_t OIDListIterator::Next () [inline]
Moves to the next element.

Returns:
The next element.

5.42.2.2 bool_t OIDListIterator::HasNext () [inline]
Gets if there are more elements.

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

The documentation for this class was generated from the following file:

• Graph_data.h

5.43 Platform Class Reference

Platform class.

Static Public Member Functions

• static void GetStatistics (PlatformStatistics &stats)

  Gets platform data and statistics.

5.43.1 Detailed Description

Platform class.

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

5.43.2.1 static void Platform::GetStatistics (PlatformStatistics & stats) [static]

Gets platform data and statistics.

Parameters:

stats [in|out] This updates the given PlatformStatistics.

The documentation for this class was generated from the following file:

- common.h

5.44 PlatformStatistics Class Reference

Platform data and statistics.

Public Member Functions

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

- int32_t GetNumCPUs () const
  *Gets the number of CPUs.*

- int64_t GetRealTime () const
  *Gets time in microseconds (since epoch).*

- int64_t GetUserTime () const
  *Gets CPU user time.*

- int64_t GetSystemTime () const
  *Gets CPU system time.*

- int64_t GetTotalMem () const
  *Gets physical memory size in Bytes.*

- int64_t GetAvailableMem () const
  *Gets available (free) memory size in Bytes.*

Friends

- class Platform

5.44.1 Detailed Description

Platform data and statistics.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.44.2 Member Function Documentation

5.44.2.1 int32_t PlatformStatistics::GetNumCPUs () const [inline]
Gets the number of CPUs.

Returns:
The number of CPUs.

5.44.2.2 int64_t PlatformStatistics::GetRealTime () const [inline]
Gets time in microseconds (since epoch).

Returns:
Time in microseconds (since epoch).

5.44.2.3 int64_t PlatformStatistics::GetUserTime () const [inline]
Gets CPU user time.

Returns:
CPU user time.

5.44.2.4 int64_t PlatformStatistics::GetSystemTime () const [inline]
Gets CPU system time.

Returns:
CPU system time.

5.44.2.5 int64_t PlatformStatistics::GetTotalMem () const [inline]
Gets physical memory size in Bytes.

Returns:
Physical memory size in Bytes.

5.44.2.6 int64_t PlatformStatistics::GetAvailableMem () const [inline]
Gets avialable (free) memory size in Bytes.

Returns:
Avialable (free) memory size in Bytes.

The documentation for this class was generated from the following file:

• common.h

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
5.45 RowReader Class Reference

RowReader interface.

Inheritance diagram for RowReader:

```
RowReader
    CSVReader
```

Public Member Functions

- virtual `dex::gdb::bool_t Reset ()=0 throw (dex::gdb::IOException)`
  
  Moves the reader to the beginning.

- virtual `dex::gdb::bool_t Read (dex::gdb::StringList &row)=0 throw (dex::gdb::IOException)`
  
  Reads the next row as a string array.

- virtual `dex::gdb::int32_t GetRow ()=0 throw (dex::gdb::IOException)`
  
  The row number for the current row.

- virtual `~RowReader ()`
  
  Destructor.

Protected Member Functions

- `RowReader ()`
  
  Empty constructor.

5.45.1 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 DEX User Manual for more details on this.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.45.2 Constructor & Destructor Documentation

5.45.2.1 RowReader::RowReader () [inline, protected]
Empty constructor.
Protected because no one should instantiate a RowReader. Just inherited classes can use this empty constructor.

5.45.3 Member Function Documentation

5.45.3.1 virtual dex::gdb::bool_t RowReader::Reset () throw (dex::gdb::IOException) [pure virtual]
Moves the reader to the beginning.
Restarts the reader.
Returns:
true if the reader can be restarted, false otherwise.
Exceptions:

IOException If bad things happen during the restart.
Implemented in CSVReader.

5.45.3.2 virtual dex::gdb::bool_t RowReader::Read (dex::gdb::StringList & row) throw (dex::gdb::IOException) [pure virtual]
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:

IOException If bad things happen during the read.
Implemented in CSVReader.

5.45.3.3 virtual dex::gdb::int32_t RowReader::GetRow () throw (dex::gdb::IOException) [pure virtual]
The row number for the current row.
Returns:
The current row number; 0 if there is no current row.
Exceptions:

IOException If it fails.
Implemented in CSVReader.
5.46 RowWriter Class Reference

RowWriter interface.

Inheritance diagram for RowWriter:

```
RowWriter
    CSVWriter
```

Public Member Functions

- virtual void Write (dex::gdb::StringList &row)=0 throw (dex::gdb::IOException, dex::gdb::Error)
  
  Writes the next row.

- virtual void Close ()=0 throw (dex::gdb::IOException, dex::gdb::Error)
  
  Closes the writer.

- virtual ~RowWriter ()
  
  Destructor.

Protected Member Functions

- RowWriter ()
  
  Empty constructor.

5.46.1 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.

Implemented in CSVReader.

The documentation for this class was generated from the following file:

- RowReader.h
Check out the 'Data export' section in the DEX User Manual for more details on this.

**Author:**

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

### 5.46.2 Constructor & Destructor Documentation

#### 5.46.2.1 RowWriter::RowWriter ()  [inline, protected]

Empty constructor.

Protected because no one should instantiate a **RowWriter**. Just inherited classes can use this empty constructor.

### 5.46.3 Member Function Documentation

#### 5.46.3.1 virtual void RowWriter::Write (dex::gdb::StringList & row) throw (dex::gdb::IOException, dex::gdb::Error)  [pure virtual]

Writes the next row.

**Parameters:**

- **row**  [in] Row of data.

**Exceptions:**

- **IOException** If bad things happen during the write.

Implemented in CSVWriter.

#### 5.46.3.2 virtual void RowWriter::Close () throw (dex::gdb::IOException, dex::gdb::Error)  [pure virtual]

Closes the writer.

**Exceptions:**

- **IOException** If the close fails.

Implemented in CSVWriter.

The documentation for this class was generated from the following file:

- RowWriter.h

### 5.47 ScriptParser Class Reference

**ScriptParser.**
Public Member Functions

- **ScriptParser ()**
  
  Constructor.

- **virtual ~ScriptParser ()**
  
  Destructor.

- **void SetOutputLog (const std::wstring &path) throw (dex::gdb::IOException)**
  
  Sets the output log.

- **void setErrorLog (const std::wstring &path) throw (dex::gdb::IOException)**
  
  Sets the error log.

- **dex::gdb::bool_t Parse (dexcore::FileReader *fileReader, dex::gdb::bool_t execute)**
  
  Parses the given input stream.

- **dex::gdb::bool_t Parse (const std::wstring &path, dex::gdb::bool_t execute, const std::wstring &localeStr) throw (dex::gdb::IOException)**
  
  Parses the given input file.

Static Public Member Functions

- **static void GenerateSchemaScript (const std::wstring &path, dex::gdb::Database &db) throw (dex::gdb::IOException)**
  
  Writes a script with the schema definition for the given database.

5.47.1 Detailed Description

**ScriptParser.**

The **ScriptParser** can create schemas and load data from a set of commands in a dex script.

A DEX script contains an ordered list of commands. **ScriptParser** will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined DEX schema.

Check out the 'Scripting' chapter in the DEX User Manual for a comprehensive explanation on the grammar of the DEX commands and how they work.

**Author:**

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

5.47.2 Member Function Documentation

5.47.2.1 void ScriptParser::SetOutputLog (const std::wstring & path) throw (dex::gdb::IOException)

Sets the output log. If not set, output log corresponds to standard output.
Parameters:

\textit{path} [in] Path of the output log.

Exceptions:

\textit{IOException} If bad things happen opening the file.

5.47.2.2 void ScriptParser::SetErrorLog (const std::wstring & \textit{path}) throw (dex::gdb::IOException)

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters:

\textit{path} [in] Path of the error log.

Exceptions:

\textit{IOException} If bad things happen opening the file.

5.47.2.3 dex::gdb::bool_t ScriptParser::Parse (dexcore::FileReader * \textit{fileReader}, dex::gdb::bool_t \textit{execute})

Parser the given input stream.

Parameters:

\textit{fileReader} [in] Input file reader.

\textit{execute} [in] If TRUE the script is executed, if FALSE it is just parsed.

Returns:

TRUE if ok, FALSE in case of error.

5.47.2.4 dex::gdb::bool_t ScriptParser::Parse (const std::wstring & \textit{path}, dex::gdb::bool_t \textit{execute}, const std::wstring & \textit{localeStr}) throw (dex::gdb::IOException)

Parses the given input file.

Parameters:

\textit{path} [in] Input file path.

\textit{execute} [in] If TRUE the script is executed, if FALSE it is just parsed.

\textit{localeStr} [in] The locale string for reading the input file. See CSVReader.

Returns:

TRUE if ok, FALSE in case of error.

Exceptions:

\textit{IOException} If bad things happen opening the file.
5.47.2.5 static void ScriptParser::GenerateSchemaScript (const std::wstring & path, dex::gdb::Database & db) throw (dex::gdb::IOException) [static]

Writes an script with the schema definition for the given database.

Parameters:

  path [in] Filename of the script to be written.
  db [in] Database.

Exceptions:

  IOException If bad things happen opening or writing the file.

The documentation for this class was generated from the following file:

  • ScriptParser.h

5.48 Session Class Reference

Session class.

Inheritance diagram for Session:
Collaboration diagram for Session:

Public Member Functions

- virtual ~Session ()
  Destructor.

- Graph * GetGraph ()
  Gets the Graph instance.

- Objects * NewObjects ()
  Creates a new Objects instance.

- void Begin ()
  Begins a transaction.

- void Commit ()
  Commits a transaction.
5.48 Session Class Reference

Friends

• class Database
• class Graph
• class Objects
• class ObjectsIterator
• class Values
• class ValuesIterator
• class TextStream

5.48.1 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 DEX User Manual for details about how Sessions work and the use of transactions.

Author:

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

5.48.2 Member Function Documentation

5.48.2.1 Graph* Session::GetGraph ()

Gets the Graph instance.

Returns:

The Graph instance.

5.48.2.2 Objects* Session::NewObjects ()

Creates a new Objects instance.

Returns:

The new Objects instance.

The documentation for this class was generated from the following file:

• Session.h
5.49 ShortestPath Class Reference

ShortestPath class.

Inheritance diagram for ShortestPath:

Public Member Functions

- void SetMaximumHops (dex::gdb::int32_t maxhops)
  
  Sets the maximum hops restriction.

- virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  
  Allows for traversing edges of the given type.

- virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  
  Allows for traversing all edge types of the graph.

- virtual void AddNodeType (dex::gdb::type_t t)
  
  Allows for traversing nodes of the given type.

- virtual void AddAllNodeTypes ()
  
  Allows for traversing all node types of the graph.

- virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  
  Set which nodes can’t be used.

- virtual void ExcludeEdges (dex::gdb::Objects &edges)
  
  Set which edges can’t be used.

- virtual void Run ()=0

  Runs the algorithm.

- virtual ~ShortestPath ()

  Destructor.

Protected Member Functions

- ShortestPath (dex::gdb::Session &s)

  Creates a new instance.

- void AssertEdgeType (dex::gdb::type_t edgetype)
Check that the given edge type is valid.

- void AssertAddedEdges ()
  Check that edges had been added.

- void AssertAddedNodes ()
  Check that nodes had been added.

- void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

- void AssertNotComputed ()
  Check that the shortest path had not been calculated yet.

- void AssertComputed ()
  Check that the shortest path had been calculated.

- dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node is forbidden.

- dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.

Protected Attributes

- dex::gdb::Session * sess
  Session.

- dex::gdb::Graph * graph
  Graph.

- std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes
  Allowed edge types.

- std::vector< dex::gdb::type_t > nodeTypes
  Allowed node types.

- dex::gdb::int32_t maxHops
  Maximum hops restriction.

- dex::gdb::bool_t computed
  True if the shortest path has been calculated.

- dex::gdb::Objects * excludedNodes
  The set of excluded nodes.
5.49 ShortestPath Class Reference

- `dex::gdb::Objects * excludedEdges`
  The set of excluded edges.
- `dex::gdb::bool_t areAllNodeTypesAllowed`
  True if all the node types are allowed.

5.49.1 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 DEX User Manual for more details on this.

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

5.49.2 Constructor & Destructor Documentation

5.49.2.1 `ShortestPath::ShortestPath (dex::gdb::Session & s)` [protected]

Creates a new instance.

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

5.49.3 Member Function Documentation

5.49.3.1 `void ShortestPath::SetMaximumHops (dex::gdb::int32_t maxhops)`

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

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

5.49.3.2 `virtual void ShortestPath::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)` [virtual]

Allows for traversing edges of the given type.

Parameters:
- `t` [in] Edge type.
- `d` [in] Edge direction.
5.49.3.3 virtual void ShortestPath::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [virtual]

Allows for traversing all edge types of the graph.

Parameters:

    d [in] Edge direction.

5.49.3.4 virtual void ShortestPath::ExcludeNodes (dex::gdb::Objects & nodes) [virtual]

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.

5.49.3.5 virtual void ShortestPath::ExcludeEdges (dex::gdb::Objects & edges) [virtual]

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.

5.49.3.6 virtual void ShortestPath::Run () [pure virtual]

Runs the algorithm.

This method can only be called once.


5.49.4 Member Data Documentation

5.49.4.1 dex::gdb::int32_t ShortestPath::maxHops [protected]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

The documentation for this class was generated from the following file:

    • ShortestPath.h

5.50 SinglePairShortestPath Class Reference

SinglePairShortestPath class.
Inheritance diagram for SinglePairShortestPath:

Collaboration diagram for SinglePairShortestPath:

Public Member Functions

- virtual dex::gdb::OIDList * GetPathAsNodes ()
  Gets the shortest path between the source node and the destination node as an ordered set of nodes.

- virtual dex::gdb::OIDList * GetPathAsEdges ()
  Gets the shortest path between the source node and the destination node as an ordered set of edges.

- virtual dex::gdb::double64_t GetCost ()
  Gets the cost of the shortest path.

- virtual dex::gdb::bool_t Exists ()
  Returns TRUE if a path exists or FALSE otherwise.

- virtual ~SinglePairShortestPath ()
  Destructor.

- void SetMaximumHops (dex::gdb::int32_t maxhops)
  Sets the maximum hops restriction.

- virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  Allows for traversing edges of the given type.

- virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  Allows for traversing all edge types of the graph.

- virtual void AddNodeType (dex::gdb::type_t t)
  Allows for traversing nodes of the given type.

- virtual void AddAllNodeTypes ()
  Allows for traversing all node types of the graph.
• virtual void **ExcludeNodes** (dex::gdb::Objects &nodes)
  Set which nodes can’t be used.

• virtual void **ExcludeEdges** (dex::gdb::Objects &edges)
  Set which edges can’t be used.

• virtual void **Run** ()=0
  Runs the algorithm.

**Protected Member Functions**

• **SinglePairShortestPath** (dex::gdb::Session &s, dex::gdb::oid_t src, dex::gdb::oid_t dst)
  Creates a new instance.

• void **AssertEdgeType** (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

• void **AssertAddedEdges** ()
  Check that edges had been added.

• void **AssertAddedNodes** ()
  Check that nodes had been added.

• void **AssertNodeType** (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

• dex::gdb::bool_t **IsNodeTypeAllowed** (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

• void **AssertNotComputed** ()
  Check that the shortest path had not been calculated yet.

• void **AssertComputed** ()
  Check that the shortest path had been calculated.

• dex::gdb::bool_t **IsNodeExcluded** (dex::gdb::oid_t node)
  Check if the given node is forbidden.

• dex::gdb::bool_t **IsEdgeExcluded** (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.

**Protected Attributes**

• dex::gdb::oid_t **source**
  Source node.
• `dex::gdb::oid_t destination`
  Destination node.

• `dex::gdb::OIDList * pathAsNodes`
  Ordered set of node identifiers representing the shortest path.

• `dex::gdb::OIDList * pathAsEdges`
  Ordered set of edge identifiers representing the shortest path.

• `dex::gdb::Session * sess`
  Session.

• `dex::gdb::Graph * graph`
  Graph.

• `std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes`
  Allowed edge types.

• `std::vector< dex::gdb::type_t > nodeTypes`
  Allowed node types.

• `dex::gdb::int32_t maxHops`
  Maximum hops restriction.

• `dex::gdb::bool_t computed`
  True if the shortest path has been calculated.

• `dex::gdb::Objects * excludedNodes`
  The set of excluded nodes.

• `dex::gdb::Objects * excludedEdges`
  The set of excluded edges.

• `dex::gdb::bool_t areAllNodeTypesAllowed`
  True if all the node types are allowed.

### 5.50.1 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 DEX User Manual for more details on this.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.50.2 Constructor & Destructor Documentation

5.50.2.1 SinglePairShortestPath::SinglePairShortestPath (dex::gdb::Session & s, dex::gdb::oid_t src, dex::gdb::oid_t dst) [protected]

Creates a new instance.

Parameters:

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

5.50.3 Member Function Documentation

5.50.3.1 virtual dex::gdb::OIDList* SinglePairShortestPath::GetPathAsNodes () [pure virtual]

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

Returns:

Ordered set of node identifiers.


5.50.3.2 virtual dex::gdb::OIDList* SinglePairShortestPath::GetPathAsEdges () [pure virtual]

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

Returns:

Ordered set of edge identifiers.


5.50.3.3 virtual dex::gdb::double64_t SinglePairShortestPath::GetCost () [pure virtual]

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.


5.50.3.4 void ShortestPath::SetMaximumHops (dex::gdb::int32_t maxhops) [inherited]

Sets the maximum hops restriction.

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

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

5.50.3.5 virtual void ShortestPath::AddEdgeType (dex::gdb::type_t \( t \), dex::gdb::EdgesDirection \( d \)) [virtual, inherited]
Allows for traversing edges of the given type.

Parameters:

\( t \) [in] Edge type.
\( d \) [in] Edge direction.

5.50.3.6 virtual void ShortestPath::AddAllEdgeTypes (dex::gdb::EdgesDirection \( d \)) [virtual, inherited]
Allows for traversing all edge types of the graph.

Parameters:

\( d \) [in] Edge direction.

5.50.3.7 virtual void ShortestPath::ExcludeNodes (dex::gdb::Objects & \( nodes \)) [virtual, inherited]
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.

5.50.3.8 virtual void ShortestPath::ExcludeEdges (dex::gdb::Objects & \( edges \)) [virtual, inherited]
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.

5.50.3.9 virtual void ShortestPath::Run () [pure virtual, inherited]
Runs the algorithm.
This method can only be called once.
5.50.4 Member Data Documentation

5.50.4.1 `dex::gdb::OIDList* SinglePairShortestPath::pathAsEdges` [protected]
Ordered set of edge identifiers representing the shortest path.
May be computed lazily when requested from the pathAsNodes.

5.50.4.2 `dex::gdb::int32_t ShortestPath::maxHops` [protected, inherited]
Maximum hops restriction.
It must be positive or zero. Zero means unlimited.
The documentation for this class was generated from the following file:

- SinglePairShortestPath.h

5.51 SinglePairShortestPathBFS Class Reference

`SinglePairShortestPathBFS` class.
Inheritance diagram for `SinglePairShortestPathBFS`:

![Inheritance Diagram](image)

Collaboration diagram for `SinglePairShortestPathBFS`:

![Collaboration Diagram](image)

Public Member Functions

- virtual ~`SinglePairShortestPathBFS` ()
  `Destructor.`

- virtual void `Run` ()
  `Executes the algorithm.`

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
• virtual dex::gdb::OIDList * GetPathAsNodes ()
  Gets the shortest path between the source node and the destination node as an ordered set of nodes.

• virtual dex::gdb::OIDList * GetPathAsEdges ()
  Gets the shortest path between the source node and the destination node as an ordered set of edges.

• virtual dex::gdb::double64_t GetCost ()
  Gets the cost of the shortest path.

• SinglePairShortestPathBFS (dex::gdb::Session &s, dex::gdb::oid_t src, dex::gdb::oid_t dst)
  Creates a new instance.

• virtual void CheckOnlyExistence ()
  Set that only the path existence must be calculated and not the path itself.

• virtual dex::gdb::bool_t Exists ()
  Returns TRUE If a path exists or FALSE otherwise.

• void SetMaximumHops (dex::gdb::int32_t maxhops)
  Sets the maximum hops restriction.

• virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  Allows for traversing edges of the given type.

• virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  Allows for traversing all edge types of the graph.

• virtual void AddNodeType (dex::gdb::type_t t)
  Allows for traversing nodes of the given type.

• virtual void AddAllNodeTypes ()
  Allows for traversing all node types of the graph.

• virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  Set which nodes can’t be used.

• virtual void ExcludeEdges (dex::gdb::Objects &edges)
  Set which edges can’t be used.

Protected Member Functions

• void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

• void AssertAddedEdges ()
  Check that edges had been added.

• void AssertAddedNodes ()
Check that nodes had been added.

- void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t node)
  Check if the given node has an allowed type.

- void AssertNotComputed ()
  Check that the shortest path had not been calculated yet.

- void AssertComputed ()
  Check that the shortest path had been calculated.

- dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node is forbidden.

- dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.

Protected Attributes

- dex::gdb::oid_t source
  Source node.

- dex::gdb::oid_t destination
  Destination node.

- dex::gdb::OIDList * pathAsNodes
  Ordered set of node identifiers representing the shortest path.

- dex::gdb::OIDList * pathAsEdges
  Ordered set of edge identifiers representing the shortest path.

- dex::gdb::Session * sess
  Session.

- dex::gdb::Graph * graph
  Graph.

- std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes
  Allowed edge types.

- std::vector< dex::gdb::type_t > nodeTypes
  Allowed node types.

- dex::gdb::int32_t maxHops
  Maximum hops restriction.
5.51 SinglePairShortestPathBFS Class Reference

- **dex::gdb::bool_t computed**
  True if the shortest path has been calculated.

- **dex::gdb::Objects * excludedNodes**
  The set of excluded nodes.

- **dex::gdb::Objects * excludedEdges**
  The set of excluded edges.

- **dex::gdb::bool_t areAllNodeTypesAllowed**
  True if all the node types are allowed.

### 5.51.1 Detailed Description

**SinglePairShortestPathBFS** class.

It solves the single-pair shortest path problem using a BFS-based implementation.

It is an unweighted algorithm, that is, it assumes all edges have the same cost.

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

**Author:**

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

### 5.51.2 Constructor & Destructor Documentation

#### 5.51.2.1 SinglePairShortestPathBFS::SinglePairShortestPathBFS (dex::gdb::Session & s, dex::gdb::oid_t src, dex::gdb::oid_t dst)

Creates a new instance.

**Parameters:**

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

### 5.51.3 Member Function Documentation

#### 5.51.3.1 virtual dex::gdb::OIDList* SinglePairShortestPathBFS::GetPathAsNodes () [virtual]

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

**Returns:**

Ordered set of node identifiers.

Implements **SinglePairShortestPath**.
5.51.3.2 virtual dex::gdb::OIDList* SinglePairShortestPathBFS::GetPathAsEdges () [virtual]

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

**Returns:**

Ordered set of edge identifiers.

Implements SinglePairShortestPath.

5.51.3.3 virtual dex::gdb::double64_t SinglePairShortestPathBFS::GetCost () [virtual]

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.

Implements SinglePairShortestPath.

5.51.3.4 virtual void SinglePairShortestPathBFS::CheckOnlyExistence () [virtual]

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.

5.51.3.5 void ShortestPath::SetMaximumHops (dex::gdb::int32_t maxhops) [inherited]

Sets the maximum hops restriction.

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

**Parameters:**

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

5.51.3.6 virtual void ShortestPath::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [virtual, inherited]

Allows for traversing edges of the given type.

**Parameters:**

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

5.51.3.7 virtual void ShortestPath::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [virtual, inherited]

Allows for traversing all edge types of the graph.
5.52 SinglePairShortestPathDijkstra Class Reference

Parameters:

\[ d \] [in] Edge direction.

5.51.3.8 virtual void ShortestPath::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]

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.

5.51.3.9 virtual void ShortestPath::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]

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.

5.51.4 Member Data Documentation

5.51.4.1 dex::gdb::OIDList* SinglePairShortestPath::pathAsEdges [protected, inherited]

Ordered set of edge identifiers representing the shortest path.
May be computed lazily when requested from the pathAsNodes.

5.51.4.2 dex::gdb::int32_t ShortestPath::maxHops [protected, inherited]

Maximum hops restriction.
It must be positive or zero. Zero means unlimited.
The documentation for this class was generated from the following file:

- SinglePairShortestPathBFS.h

5.52 SinglePairShortestPathDijkstra Class Reference

SinglePairShortestPathDijkstra class.
Inheritance diagram for SinglePairShortestPathDijkstra:

Collaboration diagram for SinglePairShortestPathDijkstra:

Public Member Functions

- virtual ~SinglePairShortestPathDijkstra ()
  Destructor.

- virtual void Run ()
  Executes the algorithm.

- virtual dex::gdb::OIDList * GetPathAsNodes ()
  Gets the shortest path between the source node and the destination node as an ordered set of nodes.

- virtual dex::gdb::OIDList * GetPathAsEdges ()
  Gets the shortest path between the source node and the destination node as an ordered set of edges.

- virtual dex::gdb::double64_t GetCost ()
  Gets the cost of the shortest path.

- SinglePairShortestPathDijkstra (dex::gdb::Session &s, dex::gdb::oid_t src, dex::gdb::oid_t dst)
  Creates a new instance.
5.52 SinglePairShortestPathDijkstra Class Reference

• virtual void AddWeightedEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d, dex::gdb::attr_t attr)
  Allows for traversing edges of the given type using the given attribute as the weight.

• virtual void SetUnweightedEdgeCost (dex::gdb::double64_t weight)
  Sets the weight assigned to the unweighted edges.

• virtual dex::gdb::bool_t Exists ()
  Returns TRUE if a path exists or FALSE otherwise.

• void SetMaximumHops (dex::gdb::int32_t maxhops)
  Sets the maximum hops restriction.

• virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  Allows for traversing edges of the given type.

• virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  Allows for traversing all edge types of the graph.

• virtual void AddNodeType (dex::gdb::type_t t)
  Allows for traversing nodes of the given type.

• virtual void AddAllNodeTypes ()
  Allows for traversing all node types of the graph.

• virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  Set which nodes can’t be used.

• virtual void ExcludeEdges (dex::gdb::Objects &edges)
  Set which edges can’t be used.

Protected Member Functions

• void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

• void AssertAddedEdges ()
  Check that edges had been added.

• void AssertAddedNodes ()
  Check that nodes had been added.

• void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

• dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

• void AssertNotComputed ()
Check that the shortest path had not been calculated yet.

- void `AssertComputed()`
  Check that the shortest path had been calculated.

- `dex::gdb::bool_t IsNodeExcluded(dex::gdb::oid_t node)`
  Check if the given node is forbidden.

- `dex::gdb::bool_t IsEdgeExcluded(dex::gdb::oid_t edge)`
  Check if the given edge is forbidden.

Protected Attributes

- `dex::gdb::oid_t source`
  Source node.

- `dex::gdb::oid_t destination`
  Destination node.

- `dex::gdb::OIDList * pathAsNodes`
  Ordered set of node identifiers representing the shortest path.

- `dex::gdb::OIDList * pathAsEdges`
  Ordered set of edge identifiers representing the shortest path.

- `dex::gdb::Session * sess`
  Session.

- `dex::gdb::Graph * graph`
  Graph.

- `std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes`
  Allowed edge types.

- `std::vector< dex::gdb::type_t > nodeTypes`
  Allowed node types.

- `dex::gdb::int32_t maxHops`
  Maximum hops restriction.

- `dex::gdb::bool_t computed`
  True if the shortest path has been calculated.

- `dex::gdb::Objects * excludedNodes`
  The set of excluded nodes.

- `dex::gdb::Objects * excludedEdges`
  The set of excluded edges.
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 DEX User Manual for more details on this.

Author:

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

5.52.3 Member Function Documentation

5.52.3.1 virtual dex::gdb::OIDList* SinglePairShortestPathDijkstra::GetPathAsNodes () [virtual]
Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

Implements SinglePairShortestPath.
5.52.3.2  virtual  dex::gdb::OIDList*  SinglePairShortestPathDijkstra::GetPathAsEdges ()  [virtual]

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

**Returns:**

Ordered set of edge identifiers.

Implements SinglePairShortestPath.

5.52.3.3  virtual  dex::gdb::double64_t  SinglePairShortestPathDijkstra::GetCost ()  [virtual]

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.

Implements SinglePairShortestPath.

5.52.3.4  virtual  void  SinglePairShortestPathDijkstra::AddWeightedEdgeType (dex::gdb::type_t t,
 dex::gdb::EdgesDirection d,  dex::gdb::attr_t attr)  [virtual]

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

**Parameters:**

- `t`  [in]  Edge type.
- `d`  [in]  Edge direction.
- `attr`  [in]  Attribute to be used as the weight. It must be a global attribute or an attribute of the given edge type.

5.52.3.5  virtual  void  SinglePairShortestPathDijkstra::SetUnweightedEdgeCost (dex::gdb::double64_t weight)  [virtual]

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 these edges is 1.

5.52.3.6  void  ShortestPath::SetMaximumHops (dex::gdb::int32_t maxhops)  [inherited]

Sets the maximum hops restriction.

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

**Parameters:**

- `m`  [in]  The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
5.52.3.7 virtual void ShortestPath::AddEdgeType (dex::gdb::type_t \textit{t}, dex::gdb::EdgesDirection \textit{d}) [virtual, inherited]
Allows for traversing edges of the given type.

Parameters:
\begin{itemize}
\item \textit{t} [in] Edge type.
\item \textit{d} [in] Edge direction.
\end{itemize}

5.52.3.8 virtual void ShortestPath::AddAllEdgeTypes (dex::gdb::EdgesDirection \textit{d}) [virtual, inherited]
Allows for traversing all edge types of the graph.

Parameters:
\begin{itemize}
\item \textit{d} [in] Edge direction.
\end{itemize}

5.52.3.9 virtual void ShortestPath::ExcludeNodes (dex::gdb::Objects \textit{& nodes}) [virtual, inherited]
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:
\begin{itemize}
\item \textit{nodes} [in] A set of node identifiers that must be kept intact until the destruction of the class.
\end{itemize}

5.52.3.10 virtual void ShortestPath::ExcludeEdges (dex::gdb::Objects \textit{& edges}) [virtual, inherited]
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:
\begin{itemize}
\item \textit{edges} [in] A set of edge identifiers that must be kept intact until the destruction of the class.
\end{itemize}

5.52.4 Member Data Documentation

5.52.4.1 dex::gdb::OIDList* SinglePairShortestPath::pathAsEdges [protected, inherited]
Ordered set of edge identifiers representing the shortest path.
May be computed lazily when requested from the pathAsNodes.
5.52.4.2  *ShortestPath::maxHops* [protected, inherited]
Maximum hops restriction.
It must be positive or zero. Zero means unlimited.
The documentation for this class was generated from the following file:

- SinglePairShortestPathDijkstra.h

5.53 SQL Exception Class Reference

SQL exception class.

Inheritance diagram for SQL Exception:

![Inheritance Diagram](image)

Collaboration diagram for SQL Exception:

![Collaboration Diagram](image)

**Public Member Functions**

- `SQLException ()`
  *Creates a new instance.*

- `SQLException (const std::string &mess)`
  *Creates a new instance.*

- `SQLException (const dexcore::SQLError *core_error)`
  *Creates a new instance.*

- `virtual ~SQLException ()`
  *Destructor.*

- `const std::string & Message () const`
  *Gets the message of the exception.*

- `void SetMessage (const std::string &mess)`
  *Sets the message of the exception.*
5.53 SQLException Class Reference

Protected Attributes

- std::string message
  
  Message of the exception.

5.53.1 Detailed Description

SQL exception class.

Author:

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

5.53.2 Constructor & Destructor Documentation

5.53.2.1 SQLException::SQLException (const std::string & mess)

Creates a new instance.

Parameters:


5.53.2.2 SQLException::SQLException (const dexcore::SQLError * core_error)

Creates a new instance.

Parameters:


5.53.3 Member Function Documentation

5.53.3.1 const std::string & Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

- The message of the exception.

5.53.3.2 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:


The documentation for this class was generated from the following file:

- Exception.h
5.54 StringList Class Reference

String list.

Public Member Functions

- **int32_t Count () const**
  
  Number of elements in the list.

- **StringListIterator * Iterator ()**
  
  Gets a new StringListIterator.

- **StringList ()**
  
  Constructor.

- **StringList (const std::vector<std::wstring> &v)**
  
  Constructor.

- **void Add (const std::wstring &str)**
  
  Adds a String at the end of the list.

- **void Clear ()**
  
  Clears the list.

5.54.1 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)

5.54.2 Constructor & Destructor Documentation

5.54.2.1 StringList::StringList ()

Constructor.

This creates an empty list.

5.54.2.2 StringList::StringList (const std::vector<std::wstring> &v)

Constructor.

Parameters:

  - *v [in]* Vector.
5.54.3 Member Function Documentation

5.54.3.1 `int32_t StringList::Count () const [inline]`
Number of elements in the list.

Returns:
Number of elements in the list.

5.54.3.2 `StringListIterator* StringList::Iterator ()`
Gets a new `StringListIterator`.

Returns:
`StringListIterator` instance.

5.54.3.3 `void StringList::Add (const std::wstring & str) [inline]`
Adds a String at the end of the list.

Parameters:

The documentation for this class was generated from the following file:

- Graph_data.h

5.55 StringListIterator Class Reference

StringList iterator class.

Public Member Functions

- `const std::wstring & Next ()`
  `Moves to the next element.`

- `bool_t HasNext ()`
  `Gets if there are more elements.`

Friends

- class `StringList`
5.55.1 Detailed Description

StringList iterator class.
Iterator to traverse all the strings into a StringList instance.

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

5.55.2 Member Function Documentation

5.55.2.1 const std::wstring& StringListIterator::Next () [inline]
Moves to the next element.

Returns:
The next element.

5.55.2.2 bool_t StringListIterator::HasNext () [inline]
Gets if there are more elements.

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

The documentation for this class was generated from the following file:

• Graph_data.h

5.56 StrongConnectivity Class Reference

StrongConnectivity class.
Inheritance diagram for StrongConnectivity:

```
  Connectivity
    ▼
     ▼
StrongConnectivity
     ▼
StrongConnectivityGabow
```
Collaboration diagram for StrongConnectivity:

```plaintext
ConnectedComponents
   |
   v
Connectivity
   |
   v
StrongConnectivity
```

Public Member Functions

- virtual `~StrongConnectivity()`  
  Destructor.
- virtual void `AddEdgeType(dex::gdb::type_t t, dex::gdb::EdgesDirection d)`  
  Allows connectivity through edges of the given type.
- virtual void `AddAllEdgeTypes(dex::gdb::EdgesDirection d)`  
  Allows connectivity through all edge types of the graph.
- virtual void `AddNodeType(dex::gdb::type_t t)`  
  Allows connectivity through nodes of the given type.
- virtual void `AddAllNodeTypes()`  
  Allows connectivity through all node types of the graph.
- virtual void `ExcludeNodes(dex::gdb::Objects &nodes)`  
  Set which nodes can’t be used.
- virtual void `ExcludeEdges(dex::gdb::Objects &edges)`  
  Set which edges can’t be used.
- `ConnectedComponents * GetConnectedComponents()`  
  Returns the results generated by the execution of the algorithm.
- virtual void `Run()=0`  
  Runs the algorithm in order to find the connected components.
- void `SetMaterializedAttribute(const std::wstring &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.

Protected Types

- typedef `std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > EdgeTypes_t`  
  A type definition to store allowed edge types.
typedef std::vector< dex::gdb::type_t > NodeTypes_t
A type definition to store allowed node types.

Protected Member Functions

- StrongConnectivity (dex::gdb::Session &s)
  Creates a new instance of StrongConnectivity.

- void AssertAddedEdges ()
  Check that edges had been added.

- void AssertAddedNodes ()
  Check that nodes had been added.

- void AssertNotComputed ()
  Check that the connectivity had not been calculated yet.

- void SetConnectedComponent (dex::gdb::oid_t idNode)
  Assigns the current component to the given node.

- void SetNodesNotVisited ()
  Set all the selected nodes in nodesNotVisited.

- void AssertNotComponentAttribute (const std::wstring &attributeName)
  Check that the given attribute name is not already in use.

- void AssertComputed ()
  Check that the connectivity had been calculated.

- void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

- void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- void CreateGlobalPersistentAttribute (const std::wstring &attributeName)
  Set a new persistent global attribute to store the connectivity information.

- void CreateGlobalTransientAttribute ()
  Set a new temporary global attribute to store the connectivity information.

- void RemoveGlobalAttribute ()
  Removes the global attribute where the connectivity information is stored.

- dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

- dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node has an excluded type.
Check if the given node is forbidden.

- `dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)`
  Check if the given node is forbidden.

 Protected Attributes

- `dex::gdb::Session * sess`
  Session.

- `dex::gdb::Graph * graph`
  Graph.

- `EdgeTypes_t edgeTypes`
  Allowed edge types.

- `std::vector< dex::gdb::type_t > nodeTypes`
  Allowed node types.

- `dex::gdb::attr_t attrComponent`
  common attribute where the connected component information is stored.

- `std::wstring attrComponentName`
  name of the common attribute where the connected component information is stored.

- `dex::gdb::int64_t actualComponent`
  Current component identifier.

- `dex::gdb::Objects * nodesNotVisited`
  Identifiers of the nodes not visited.

- `dex::gdb::bool_t matResults`
  Materialized results.

- `dex::gdb::bool_t computed`
  True if the connectivity has been calculated.

- `dex::gdb::Objects * excludedNodes`
  The set of excluded nodes.

- `dex::gdb::Objects * excludedEdges`
  The set of excluded edges.

- `ConnectedComponents * ccs`
  The calculated connectivity information.
5.56 StrongConnectivity Class Reference

5.56.1 Detailed Description

The `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 DEX User Manual for more details on this.

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

5.56.2 Constructor & Destructor Documentation

5.56.2.1 `StrongConnectivity::StrongConnectivity (dex::gdb::Session & s)` [protected]

Creates a new instance of `StrongConnectivity`.

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

5.56.3 Member Function Documentation

5.56.3.1 `virtual void StrongConnectivity::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)` [virtual]

Allows connectivity through edges of the given type.

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

Reimplemented from `Connectivity`.

5.56.3.2 `virtual void StrongConnectivity::AddAllEdgeTypes (dex::gdb::EdgesDirection d)` [virtual]

Allows connectivity through all edge types of the graph.

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

Reimplemented from `Connectivity`. 
5.56.3.3 virtual void Connectivity::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]

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.

5.56.3.4 virtual void Connectivity::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]

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.

5.56.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

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.

5.56.3.6 virtual void Connectivity::Run () [pure virtual, inherited]

Runs the algorithm in order to find the connected components.
This method can be called only once.
Implemented in StrongConnectivityGabow, and WeakConnectivityDFS.

5.56.3.7 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]

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.

5.56.3.8 void Connectivity::SetNodesNotVisited () [protected, inherited]
Set all the selected nodes in nodesNotVisited.
That’s all the nodes of the allowed node types but not the excluded ones.
The documentation for this class was generated from the following file:

- StrongConnectivity.h

5.57 StrongConnectivityGabow Class Reference

This class can be used to solve the problem of finding strongly connected components in a directed graph.

Inheritance diagram for StrongConnectivityGabow:

```
Connectivity
  StrongConnectivity
  StrongConnectivityGabow
```

Collaboration diagram for StrongConnectivityGabow:

```
ConnectedComponents
  ~StrongConnectivityGabow
  Connectivity
  StrongConnectivity
  StrongConnectivityGabow
```

Public Member Functions

- StrongConnectivityGabow (dex::gdb::Session &s)
  Creates a new instance of StrongConnectivityGabow.

- virtual ~StrongConnectivityGabow ()
Destructor.

- **void Run ()**
  
  Executes the algorithm.

- **virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)**
  
  Allows connectivity through edges of the given type.

- **virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)**
  
  Allows connectivity through all edge types of the graph.

- **virtual void AddNodeType (dex::gdb::type_t t)**
  
  Allows connectivity through nodes of the given type.

- **virtual void AddAllNodeTypes ()**
  
  Allows connectivity through all node types of the graph.

- **virtual void ExcludeNodes (dex::gdb::Objects &nodes)**
  
  Set which nodes can’t be used.

- **virtual void ExcludeEdges (dex::gdb::Objects &edges)**
  
  Set which edges can’t be used.

- **ConnectedComponents * GetConnectedComponents ()**
  
  Returns the results generated by the execution of the algorithm.

- **void SetMaterializedAttribute (const std::wstring &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.

**Protected Types**

- typedef std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > EdgeTypes_t
  
  A type definition to store allowed edge types.

- typedef std::vector< dex::gdb::type_t > NodeTypes_t
  
  A type definition to store allowed node types.

**Protected Member Functions**

- **void AssertAddedEdges ()**
  
  Check that edges had been added.

- **void AssertAddedNodes ()**
  
  Check that nodes had been added.

- **void AssertNotComputed ()**

Generated on Tue Jul 16 12:09:09 2013 for Dex by Doxygen
Check that the connectivity had not been calculated yet.

• void SetConnectedComponent (dex::gdb::oid_t idNode)
  Assigns the current component to the given node.

• void SetNodesNotVisited ()
  Set all the selected nodes in nodesNotVisited.

• void AssertNotComponentAttribute (const std::wstring &attributeName)
  Check that the given attribute name is not already in use.

• void AssertComputed ()
  Check that the connectivity had been calculated.

• void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

• void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

• void CreateGlobalPersistentAttribute (const std::wstring &attributeName)
  Set a new persistent global attribute to store the connectivity information.

• void CreateGlobalTransientAttribute ()
  Set a new temporary global attribute to store the connectivity information.

• void RemoveGlobalAttribute ()
  Removes the global attribute where the connectivity information is stored.

• dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

• dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node is forbidden.

• dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.

Protected Attributes

• dex::gdb::Session * sess
  Session.

• dex::gdb::Graph * graph
  Graph.

• EdgeTypes_t edgeTypes
  Allowed edge types.
5.57 StrongConnectivityGabow Class Reference

- std::vector< dex::gdb::type_t > nodeTypes
  
  Allowed node types.

- dex::gdb::attr_t attrComponent
  
  common attribute where the connected component information is stored.

- std::wstring attrComponentName
  
  name of the common attribute where the connected component information is stored.

- dex::gdb::int64_t actualComponent
  
  Current component identifier.

- dex::gdb::Objects * nodesNotVisited
  
  Identifiers of the nodes not visited.

- dex::gdb::bool_t matResults
  
  Materialized results.

- dex::gdb::bool_t computed
  
  True if the connectivity has been calculated.

- dex::gdb::Objects * excludedNodes
  
  The set of excluded nodes.

- dex::gdb::Objects * excludedEdges
  
  The set of excluded edges.

- ConnectedComponents * ccs
  
  The calculated connectivity information.

Classes

- class InfoNode

5.57.1 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 DEX User Manual for more details on this.

Author:

Sparsity Technologies http://www.sparsity-technologies.com
5.57 Constructor & Destructor Documentation

5.57.2.1 StrongConnectivityGabow::StrongConnectivityGabow (dex::gdb::Session & s)

Creates a new instance of StrongConnectivityGabow.

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

Parameters:

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

5.57.3 Member Function Documentation

5.57.3.1 virtual void StrongConnectivity::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [virtual, inherited]

Allows connectivity through edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

Reimplemented from Connectivity.

5.57.3.2 virtual void StrongConnectivity::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [virtual, inherited]

Allows connectivity through all edge types of the graph.

Parameters:

d [in] Edge direction.

Reimplemented from Connectivity.

5.57.3.3 virtual void Connectivity::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]

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.

5.57.3.4 virtual void Connectivity::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]

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.

5.57.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]
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.

5.57.3.6 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]
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.

5.57.3.7 void Connectivity::SetNodesNotVisited () [protected, inherited]
Set all the selected nodes in nodesNotVisited.
That’s all the nodes of the allowed node types but not the excluded ones.
The documentation for this class was generated from the following file:

- StrongConnectivityGabow.h

5.58 SystemError Class Reference

System error class.
Inheritance diagram for SystemError:

```
Exception
 |
 v
 Error
 |
 v
 SystemError
 |
 v
 IOError
```

Collaboration diagram for SystemError:

```
Exception
 |
 v
 Error
 |
 v
 SystemError
```

Public Member Functions

- **SystemError ()**
  
  Creates a new instance.

- **SystemError (const std::string &mess)**
  
  Creates a new instance.

- **virtual ~SystemError ()**
  
  Destructor.

- **const std::string & Message () const**
  
  Gets the message of the exception.

- **void SetMessage (const std::string &mess)**
  
  Sets the message of the exception.

Static Public Member Functions

- **static Error NewError (int32_t coreErrorCode)**
  
  Creates a new Error instance from a dexcore error code.
Protected Attributes

- std::string message
  
  Message of the exception.

5.58.1 Detailed Description

System error class.

Author:

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

5.58.2 Constructor & Destructor Documentation

5.58.2.1 SystemError::SystemError (const std::string & mess)

Creates a new instance.

Parameters:


5.58.3 Member Function Documentation

5.58.3.1 static Error Error::NewError (int32_t coreErrorCode) [static, inherited]

Creates a new Error instance from a dexcore error code.

Parameters:


Returns:

Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.

5.58.3.2 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.58.3.3 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.
Parameters:


The documentation for this class was generated from the following file:

* Exception.h

## 5.59 TextStream Class Reference

The **TextStream** class.

Inheritance diagram for TextStream:

Collaboration diagram for TextStream:
Public Member Functions

- **TextStream**(bool_t append)
  
  *Creates a new instance.*

- **int32_t Read**(uchar_t *dataOUT, int32_t length) const
  
  *Read data from the stream.*

- **void Write**(const uchar_t *dataIN, int32_t length)
  
  *Write data to the stream.*

- **void Close**()
  
  *Closes the stream.*

- **virtual ~TextStream**()
  
  *Destructor.*

- **bool_t IsNull**( ) const
  
  *Returns TRUE if the stream is not available.*

Friends

- **class Graph**

### 5.59.1 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 DEX User Manual for more details on this.

**Author:**

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.59 TextStream Class Reference

5.59.2 Constructor & Destructor Documentation

5.59.2.1 TextStream::TextStream (bool_t append)

Creates a new instance.

A TextStream only can be created by the user to write data.

Parameters:

append [in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the beginning of the stream.

5.59.3 Member Function Documentation

5.59.3.1 int32_t TextStream::Read (uchar_t * dataOUT, int32_t length) const

Read data from the stream.

Parameters:

dataOUT [out] Buffer to read data to. It must be allocated by the user.
length [in] Length of the given data buffer. It must be > 0.

Returns:

Amount of read data (<= length). If 0, there is no more data to be read from the stream.

5.59.3.2 void TextStream::Write (const uchar_t * dataIN, int32_t length)

Write data to the stream.

Parameters:

dataIN [in] Buffer to write data from.
length [in] Length of the data buffer. It must be > 0.

5.59.3.3 void TextStream::Close ()

Closes the stream.

Once the Stream is closed, it cannot be used again.

5.59.3.4 bool_t TextStream::IsNull () const

Returns TRUE if the stream is not available.

It returns FALSE if the stream is ready for reading or writing data.

Returns:

FALSE if the stream is ready

Reimplemented from Handler< dexcore::GraphTextStream >.

The documentation for this class was generated from the following file:

* Stream.h
5.60  Traversal Class Reference

Traversal class.

Inheritance diagram for Traversal:

- Traversal
- TraversalBFS
- TraversalDFS

Public Member Functions

- virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  
  Allows for traversing edges of the given type.

- virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  
  Allows for traversing all edge types of the graph.

- virtual void AddNodeType (dex::gdb::type_t t)
  
  Allows for traversing nodes of the given type.

- virtual void AddAllNodeTypes ()
  
  Allows for traversing all node types of the graph.

- virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  
  Set which nodes can’t be used.

- virtual void ExcludeEdges (dex::gdb::Objects &edges)
  
  Set which edges can’t be used.

- virtual dex::gdb::oid_t Next ()=0
  
  Gets the next object of the traversal.

- virtual dex::gdb::bool_t HasNext ()=0
  
  Gets if there are more objects to be traversed.

- virtual dex::gdb::int32_t GetCurrentDepth () const =0
  
  Returns the depth of the current node.

- virtual void SetMaximumHops (dex::gdb::int32_t maxhops)
  
  Sets the maximum hops restriction.

- virtual ~Traversal ()
  
  Destructor.
5.60 Traversal Class Reference

Protected Member Functions

- **Traversal** (dex::gdb::Session &s, dex::gdb::oid_t node)
  
  Creates a new instance.

- **void AssertAddedEdges** ()
  
  Check that edges had been added.

- **void AssertAddedNodes** ()
  
  Check that nodes had been added.

- **void AssertEdgeType** (dex::gdb::type_t edgetype)
  
  Check that the given edge type is valid.

- **void AssertNodeType** (dex::gdb::type_t nodetype)
  
  Check that the given node type is valid.

- **dex::gdb::bool_t IsNodeTypeAllowed** (dex::gdb::oid_t nodeId)
  
  Check if the given node has an allowed type.

- **dex::gdb::bool_t IsNodeExcluded** (dex::gdb::oid_t node)
  
  Check if the given node is forbidden.

- **dex::gdb::bool_t IsEdgeExcluded** (dex::gdb::oid_t edge)
  
  Check if the given edge is forbidden.

Protected Attributes

- **dex::gdb::Session * sess**
  
  Session.

- **dex::gdb::Graph * graph**
  
  Graph.

- **dex::gdb::oid_t src**
  
  Source node of the traversal.

- **std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes**
  
  Allowed edge types.

- **std::vector< dex::gdb::type_t > nodeTypes**
  
  Allowed node types.

- **dex::gdb::int32_t maxHops**
  
  Maximum number of hops allowed.

- **dex::gdb::Objects * excludedNodes**
  
  The set of excluded nodes.
5.60 Traversal Class Reference

- dex::gdb::Objects * excludedEdges

  The set of excluded edges.

5.60.1 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 DEX User Manual for more details on this.

Author:

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

5.60.2 Constructor & Destructor Documentation

5.60.2.1 Traversal::Traversal (dex::gdb::Session & s, dex::gdb::oid_t node) [protected]

Creates a new instance.

Parameters:

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

5.60.3 Member Function Documentation

5.60.3.1 virtual void Traversal::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [virtual]

Allows for traversing edges of the given type.

Parameters:

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

5.60.3.2 virtual void Traversal::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [virtual]

Allows for traversing all edge types of the graph.

Parameters:

  d [in] Edge direction.
5.60.3.3 virtual void Traversal::ExcludeNodes (dex::gdb::Objects & nodes) [virtual]
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.

5.60.3.4 virtual void Traversal::ExcludeEdges (dex::gdb::Objects & edges) [virtual]
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.

5.60.3.5 virtual dex::gdb::oid_t Traversal::Next () [pure virtual]
Gets the next object of the traversal.

Returns:

   A node or edge identifier.

Implemented in TraversalBFS, and TraversalDFS.

5.60.3.6 virtual dex::gdb::bool_t Traversal::HasNext () [pure virtual]
Gets if there are more objects to be traversed.

Returns:

   TRUE if there are more objects, FALSE otherwise.

Implemented in TraversalBFS, and TraversalDFS.

5.60.3.7 virtual dex::gdb::int32_t Traversal::GetCurrentDepth () const [pure virtual]
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.

Implemented in TraversalBFS, and TraversalDFS.
5.60.3.8 virtual void Traversal::SetMaximumHops (dex::gdb::int32_t maxhops) [virtual]
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.

The documentation for this class was generated from the following file:

  • Traversal.h

5.61 TraversalBFS Class Reference

Breadth-First Search implementation of Traversal.

Inheritance diagram for TraversalBFS:

Collaboration diagram for TraversalBFS:

Public Member Functions

• virtual dex::gdb::oid_t Next ()
  Gets the next object of the traversal.

• virtual dex::gdb::bool_t HasNext ()
  Gets if there are more objects to be traversed.

• virtual ~TraversalBFS ()
  Destructor.

• virtual dex::gdb::int32_t GetCurrentDepth () const
  Returns the depth of the current node.

• TraversalBFS (dex::gdb::Session &s, dex::gdb::oid_t node)
Creates a new instance.

- virtual void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  Allows for traversing edges of the given type.

- virtual void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  Allows for traversing all edge types of the graph.

- virtual void AddNodeType (dex::gdb::type_t t)
  Allows for traversing nodes of the given type.

- virtual void AddAllNodeTypes ()
  Allows for traversing all node types of the graph.

- virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  Set which nodes can’t be used.

- virtual void ExcludeEdges (dex::gdb::Objects &edges)
  Set which edges can’t be used.

- virtual void SetMaximumHops (dex::gdb::int32_t maxhops)
  Sets the maximum hops restriction.

**Protected Member Functions**

- void AssertAddedEdges ()
  Check that edges had been added.

- void AssertAddedNodes ()
  Check that nodes had been added.

- void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

- void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

- dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node is forbidden.

- dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.
### 5.61 TraversalBFS Class Reference

#### Protected Attributes

- `dex::gdb::Session * sess`
  
  *Session.*

- `dex::gdb::Graph * graph`
  
  *Graph.*

- `dex::gdb::oid_t src`
  
  Source node of the traversal.

- `std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes`
  
  Allowed edge types.

- `std::vector< dex::gdb::type_t > nodeTypes`
  
  Allowed node types.

- `dex::gdb::int32_t maxHops`
  
  Maximum number of hops allowed.

- `dex::gdb::Objects * excludedNodes`
  
  The set of excluded nodes.

- `dex::gdb::Objects * excludedEdges`
  
  The set of excluded edges.

#### 5.61.1 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 DEX User Manual for more details on this.

**Author:**

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

#### 5.61.2 Constructor & Destructor Documentation

##### 5.61.2.1 TraversalBFS::TraversalBFS (dex::gdb::Session & s, dex::gdb::oid_t node)

Creates a new instance.

**Parameters:**

- `s` [in] *Session* to get the graph from and perform traversal.
- `node` [in] Node to start traversal from.
5.61.3 Member Function Documentation

5.61.3.1 virtual dex::gdb::oid_t TraversalBFS::Next () [virtual]
Gets the next object of the traversal.

Returns:
A node or edge identifier.

Implements Traversal.

5.61.3.2 virtual dex::gdb::bool_t TraversalBFS::HasNext () [virtual]
Gets if there are more objects to be traversed.

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

Implements Traversal.

5.61.3.3 virtual dex::gdb::int32_t TraversalBFS::GetCurrentDepth () const [virtual]
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.

Implements Traversal.

5.61.3.4 virtual void Traversal::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [virtual, inherited]
Allows for traversing edges of the given type.

Parameters:

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

5.61.3.5 virtual void Traversal::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [virtual, inherited]
Allows for traversing all edge types of the graph.

Parameters:

  d [in] Edge direction.
5.62 TraversalDFS Class Reference

5.61.3.6 virtual void Traversal::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]

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.

5.61.3.7 virtual void Traversal::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]

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.

5.61.3.8 virtual void Traversal::SetMaximumHops (dex::gdb::int32_t maxhops) [virtual, inherited]

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.

The documentation for this class was generated from the following file:

- TraversalBFS.h

5.62 TraversalDFS Class Reference

Depth-First Search (DFS) implementation of Traversal.

Inheritance diagram for TraversalDFS:
Collaboration diagram for TraversalDFS:

```
Traversal

TraversalDFS

Traversal
```

### Public Member Functions

- **virtual dex::gdb::oid_t** Next ()
  
  *Get the next object of the traversal.*

- **virtual dex::gdb::bool_t** HasNext ()
  
  *Gets if there are more objects to be traversed.*

- **virtual dex::gdb::int32_t** GetCurrentDepth () const
  
  *Returns the depth of the current node.*

- **virtual ~TraversalDFS ()
  
  *Destructor.*

- **TraversalDFS (dex::gdb::Session &, dex::gdb::oid_t node)
  
  *Creates a new instance.*

- **virtual void** AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  
  *Allows for traversing edges of the given type.*

- **virtual void** AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  
  *Allows for traversing all edge types of the graph.*

- **virtual void** AddNodeType (dex::gdb::type_t t)
  
  *Allows for traversing nodes of the given type.*

- **virtual void** AddAllNodeTypes ()
  
  *Allows for traversing all node types of the graph.*

- **virtual void** ExcludeNodes (dex::gdb::Objects &nodes)
  
  *Set which nodes can’t be used.*

- **virtual void** ExcludeEdges (dex::gdb::Objects &edges)
  
  *Set which edges can’t be used.*

- **virtual void** SetMaximumHops (dex::gdb::int32_t maxhops)
  
  *Sets the maximum hops restriction.*
Protected Member Functions

- void AssertAddedEdges ()
  Check that edges had been added.

- void AssertAddedNodes ()
  Check that nodes had been added.

- void AssertEdgeType (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

- void AssertNodeType (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)
  Check if the given node has an allowed type.

- dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)
  Check if the given node is forbidden.

- dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)
  Check if the given edge is forbidden.

Protected Attributes

- dex::gdb::Session * sess
  Session.

- dex::gdb::Graph * graph
  Graph.

- dex::gdb::oid_t src
  Source node of the traversal.

- std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > edgeTypes
  Allowed edge types.

- std::vector< dex::gdb::type_t > nodeTypes
  Allowed node types.

- dex::gdb::int32_t maxHops
  Maximum number of hops allowed.

- dex::gdb::Objects * excludedNodes
  The set of excluded nodes.

- dex::gdb::Objects * excludedEdges
  The set of excluded edges.
5.62 TraversalDFS Class Reference

Classes

- class NeighborsInfo

  Store neighbors information.

5.62.1 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 DEX User Manual for more details on this.

Author:

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

5.62.2 Constructor & Destructor Documentation

5.62.2.1 TraversalDFS::TraversalDFS (dex::gdb::Session & s, dex::gdb::oid_t node)

  Creates a new instance.

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

5.62.3 Member Function Documentation

5.62.3.1 virtual dex::gdb::oid_t TraversalDFS::Next () [virtual]

  Gets the next object of the traversal.

  Returns:
   - A node or edge identifier.

  Implements Traversal.

5.62.3.2 virtual dex::gdb::bool_t TraversalDFS::HasNext () [virtual]

  Gets if there are more objects to be traversed.

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

  Implements Traversal.
5.62.3.3 virtual dex::gdb::int32_t TraversalDFS::GetCurrentDepth () const  [virtual]

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.

Implements Traversal.

5.62.3.4 virtual void Traversal::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)  [virtual, inherited]

Allows for traversing edges of the given type.

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

5.62.3.5 virtual void Traversal::AddAllEdgeTypes (dex::gdb::EdgesDirection d)  [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:
    d [in] Edge direction.

5.62.3.6 virtual void Traversal::ExcludeNodes (dex::gdb::Objects & nodes)  [virtual, inherited]

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.

5.62.3.7 virtual void Traversal::ExcludeEdges (dex::gdb::Objects & edges)  [virtual, inherited]

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.
5.62.3.8 virtual void Traversal::SetMaximumHops (dex::gdb::int32_t maxhops) [virtual, inherited]

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.

The documentation for this class was generated from the following file:

  • TraversalDFS.h

5.63 Type Class Reference

Type data class.

Public Member Functions

  • type_t GetId () const
    
    Gets the Dex type identifier.

  • ObjectType GetObjectType () const
    
    Gets the object type.

  • const std::wstring & GetName () const
    
    Gets the unique type name.

  • int64_t GetNumObjects () const
    
    Gets the number of objects belonging to the type.

  • bool_t GetIsDirected () const
    
    Gets if this is a directed edge type.

  • bool_t GetIsRestricted () const
    
    Gets if this is a restricted edge type.

  • bool_t GetAreNeighborsIndexed () const
    
    Gets if this is an edge type with neighbors index.

  • type_t GetRestrictedFrom () const
    
    Gets the tail or source type identifier for restricted edge types.

  • type_t GetRestrictedTo () const
    
    Gets the head or target type identifier for restricted edge types.
5.63 Type Class Reference

Static Public Attributes

- static const type_t InvalidType
  Invalid type identifier constant.

- static const type_t GlobalType
  Global type identifier constant.

Friends

- class Graph

5.63.1 Detailed Description

Type data class.
It contains information about a node or edge type.

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

5.63.2 Member Function Documentation

5.63.2.1 type_t Type::GetId () const [inline]
Gets the Dex type identifier.

Returns:
The Dex type identifier.

5.63.2.2 ObjectType Type::GetObjectType () const [inline]
Gets the object type.

Returns:
The object type.

5.63.2.3 const std::wstring& Type::GetName () const [inline]
Gets the unique type name.

Returns:
The unique type name.
5.63.2.4 int64_t Type::GetNumObjects () const [inline]
Gets the number of objects belonging to the type.

Returns:
The number of objects belonging to the type.

5.63.2.5 bool_t Type::GetIsDirected () const [inline]
Gets if this is a directed edge type.

Returns:
TRUE for directed edge types, FALSE otherwise.

5.63.2.6 bool_t Type::GetIsRestricted () const [inline]
Gets if this is a restricted edge type.

Returns:
TRUE for restricted edge types, FALSE otherwise.

5.63.2.7 bool_t Type::GetAreNeighborsIndexed () const [inline]
Gets if this is an edge type with neighbors index.

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

5.63.2.8 type_t Type::GetRestrictedFrom () const [inline]
Gets the tail or source type identifier for restricted edge types.

Returns:
For restricted edge types, the tail or source type identifier, Type::InvalidType otherwise.

5.63.2.9 type_t Type::GetRestrictedTo () const [inline]
Gets the head or target type identifier for restricted edge types.

Returns:
For restricted edge types, the head or target type identifier, Type::InvalidType otherwise.

The documentation for this class was generated from the following file:

- Graph_data.h
5.64 TypeExporter Class Reference

Base TypeExporter class.

Inheritance diagram for TypeExporter:

```
TypeExporter
   ↓
EdgeTypeExporter
   ↓
NodeTypeExporter
```

Collaboration diagram for TypeExporter:

```
TypeExporter
  +-- RowWriter
```

Public Member Functions

- virtual ~TypeExporter ()
  Destructor.

- void Register (TypeExporterListener &tel)
  Registers a new listener.

- virtual void Run ()=0 throw (dex::gdb::IOException, dex::gdb::Error)
  Runs export process.

- void SetRowWriter (RowWriter &rw)
  Sets the output data destination.

- void SetGraph (dex::gdb::Graph &g)
  Sets the graph that will be exported.

- void SetType (dex::gdb::type_t t)
  Sets the type to be exported.

- void SetAttributes (dex::gdb::AttributeList &attrs)
  Sets the list of Attributes.

- void SetFrequency (dex::gdb::int32_t freq)
  Sets the frequency of listener notification.

- void SetHeader (dex::gdb::bool_t header)
  Sets the presence of a header row.
Protected Member Functions

- **TypeExporter ()**
  Creates a new instance.

- **TypeExporter (RowWriter &rw, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs)**
  Creates a new instance with the minimum common required arguments.

- **dex::gdb::bool_t CanRun ()**
  Checks that all the required settings are ready to run.

- **void NotifyListeners (const TypeExporterEvent &ev)**
  Notifies progress to all registered listeners.

- **void RunProcess () throw (dex::gdb::IOException, dex::gdb::Error)**
  Runs export process.

- **void SetHeadAttribute (dex::gdb::attr_t attr)**
  Sets the attribute that will be used to get the value to be dumped for the head of the edge.

- **void SetHeadPosition (dex::gdb::int32_t pos)**
  Sets the position (index column) of the head attribute in the exported data.

- **void SetTailAttribute (dex::gdb::attr_t attr)**
  Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

- **void SetTailPosition (dex::gdb::int32_t pos)**
  Sets the position (index column) of the tail attribute in the exported data.

5.64.1 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 DEX User Manual for more details on this.

Author:

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

5.64.2 Constructor & Destructor Documentation

5.64.2.1 TypeExporter::TypeExporter (RowWriter &rw, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs) [protected]

Creates a new instance with the minimum common required arguments.
Parameters:

- \( rw \) [in] Output RowWriter.
- \( g \) [in] Graph.
- \( t \) [in] Type identifier.
- \( attrs \) [in] Attribute identifiers to be exported.

5.64.3 Member Function Documentation

5.64.3.1 `dex::gdb::bool_t TypeExporter::CanRun () [protected]`
Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.64.3.2 `void TypeExporter::NotifyListeners (const TypeExporterEvent & ev) [protected]`
Notifies progress to all registered listeners.

Parameters:

- \( ev \) [in] Progress event to be notified.

5.64.3.3 `void TypeExporter::RunProcess () throw (dex::gdb::IOException, dex::gdb::Error) [protected]`
Runs export process.

Exceptions:

- \( IOException \) If bad things happen writing to the RowWriter.

5.64.3.4 `void TypeExporter::SetHeadAttribute (dex::gdb::attr_t attr) [protected]`
Sets the attribute that will be used to get the value to be dumped for the head of the edge.
This method is protected because only the Edge exporters should have it.

Parameters:

- \( attr \) [in] Head Attribute

Reimplemented in EdgeTypeExporter.

5.64.3.5 `void TypeExporter::SetHeadPosition (dex::gdb::int32_t pos) [protected]`
Sets the position (index column) of the head attribute in the exported data.
This method is protected because only the Edge exporters should have it.

Parameters:

- \( pos \) [in] Head position

Reimplemented in EdgeTypeExporter.
5.64.3.6 void TypeExporter::SetTailAttribute (dex::gdb::attr_t attr)  [protected]
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Tail Attribute

Reimplemented in EdgeTypeExporter.

5.64.3.7 void TypeExporter::SetTailPosition (dex::gdb::int32_t pos)  [protected]
Sets the position (index column) of the tail attribute in the exported data.
This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Tail position

Reimplemented in EdgeTypeExporter.

5.64.3.8 void TypeExporter::Register (TypeExporterListener & tel)
Registers a new listener.

Parameters:

tel [in] TypeExporterListener to be registered.

5.64.3.9 virtual void TypeExporter::Run () throw (dex::gdb::IOException, dex::gdb::Error)  [pure virtual]
Runs export process.

Exceptions:

IOException If bad things happen writing to the RowWriter.

Implemented in EdgeTypeExporter, and NodeTypeExporter.

5.64.3.10 void TypeExporter::SetRowWriter (RowWriter & rw)
Sets the output data destination.

Parameters:

rw [in] Input RowWriter.

5.64.3.11 void TypeExporter::SetGraph (dex::gdb::Graph & g)
Sets the graph that will be exported.

Parameters:

g [in] Graph.
5.64.3.12 void TypeExporter::SetType (dex::gdb::type_t t)
Sets the type to be exported.
Parameters:
  t [in] Type identifier.

5.64.3.13 void TypeExporter::SetAttributes (dex::gdb::AttributeList & attrs)
Sets the list of Attributes.
Parameters:
  attrs [in] Attribute identifiers to be exported

5.64.3.14 void TypeExporter::SetFrequency (dex::gdb::int32_t freq)
Sets the frequency of listener notification.
freq [in] Frequency in number of rows managed to notify progress to all listeners

5.64.3.15 void TypeExporter::SetHeader (dex::gdb::bool_t header)
Sets the presence of a header row.
Parameters:
  header [in] If TRUE, a header row is dumped with the name of the attributes.

The documentation for this class was generated from the following file:
• TypeExporter.h

5.65 TypeExporterEvent Class Reference
Provides information about the progress of a TypeExproter instance.

Public Member Functions
• virtual ~TypeExporterEvent ()
  Destructor.
• dex::gdb::type_t GetTypeId () const
  Gets the type identifier.
• dex::gdb::int64_t GetCount () const
  Gets the current number of objects exported.
• dex::gdb::int64_t GetTotal () const
  Gets the total number of objects exported.
• dex::gdb::bool_t IsLast () const
  Gets if this is the last event or not.
5.65.1 Detailed Description

Provides information about the progress of an TypeExproter instance.
Check out the 'Data export' section in the DEX User Manual for more details on this.

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

5.65.2 Member Function Documentation

5.65.2.1 dex::gdb::type_t TypeExporterEvent::GetTypeId () const [inline]
Gets the type identifier.

Returns:
The type identifier.

5.65.2.2 dex::gdb::int64_t TypeExporterEvent::GetCount () const [inline]
Gets the current number of objects exported.

Returns:
The current number of objects exported.

5.65.2.3 dex::gdb::int64_t TypeExporterEvent::GetTotal () const [inline]
Gets the total number of objects exported.

Returns:
The total number of objects exported.

5.65.2.4 dex::gdb::bool_t TypeExporterEvent::IsLast () const [inline]
Gets if this is the last event or not.

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

The documentation for this class was generated from the following file:

• TypeExporter.h
5.66 TypeExporterListener Class Reference

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Public Member Functions

• virtual void NotifyEvent (const TypeExporterEvent &tee)=0
  
  Method to be notified from a TypeExporter.

• virtual ~TypeExporterListener ()
  
  Destructor.

Protected Member Functions

• TypeExporterListener ()
  
  Protected because none should instantiate a RowWriter.

5.66.1 Detailed Description

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.
Check out the ‘Data export’ section in the DEX User Manual for more details on this.

Author:

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

5.66.2 Constructor & Destructor Documentation

5.66.2.1 TypeExporterListener::TypeExporterListener () [inline, protected]

Protected because none should instantiate a RowWriter.
Just inherited classes may use this empty constructor.

5.66.3 Member Function Documentation

5.66.3.1 virtual void TypeExporterListener::NotifyEvent (const TypeExporterEvent & tee) [pure virtual]

Method to be notified from a TypeExporter.

Parameters:

  tee [in] Notified event.

The documentation for this class was generated from the following file:

• TypeExporter.h
5.67 TypeList Class Reference

Dex type identifier list.

Public Member Functions

- **int32_t Count () const**
  
  Number of elements in the list.

- **TypeListIterator * Iterator ()**
  
  Gets a new TypeListIterator.

- **TypeList ()**
  
  Constructor.

- **TypeList (const std::vector< type_t >& v)**
  
  Constructor.

- **void Add (type_t type)**
  
  Adds a Dex type identifier at the end of the list.

- **void Clear ()**
  
  Clears the list.

5.67.1 Detailed Description

Dex type identifier list.

It stores a Dex node or edge type identifier list.

Use TypeListIterator to access all elements into this collection.

Author:

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

5.67.2 Constructor & Destructor Documentation

5.67.2.1 TypeList::TypeList ()

Constructor.

This creates an empty list.

5.67.2.2 TypeList::TypeList (const std::vector< type_t >& v)

Constructor.

Parameters:

- v [in] Vector.
5.67.3 Member Function Documentation

5.67.3.1 int32_t TypeList::Count () const [inline]
Number of elements in the list.

Returns:
Number of elements in the list.

5.67.3.2 TypeListIterator TypeList::Iterator ()
Gets a new TypeListIterator.

Returns:
TypeListIterator instance.

5.67.3.3 void TypeList::Add (type_t type) [inline]
Adds a Dex type identifier at the end of the list.

Parameters:
type [in] Dex type identifier.

The documentation for this class was generated from the following file:
• Graph_data.h

5.68 TypeListIterator Class Reference

TypeList iterator class.

Public Member Functions

• type_t Next ()
  Moves to the next element.

• bool_t HasNext ()
  Gets if there are more elements.

Friends

• class TypeList
5.68.1 Detailed Description

TypeList iterator class.
Iterator to traverse all the Dex node or edge type identifiers into a TypeList instance.

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

5.68.2 Member Function Documentation

5.68.2.1 type_t TypeListIterator::Next () [inline]
Moves to the next element.

Returns:
The next element.

5.68.2.2 bool_t TypeListIterator::HasNext () [inline]
Gets if there are more elements.

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

The documentation for this class was generated from the following file:

• Graph_data.h

5.69 TypeLoader Class Reference

Base TypeLoader class.
Inheritance diagram for TypeLoader:

Collaboration diagram for TypeLoader:
Public Member Functions

- void SetLogError (const std::wstring &path) throw (dex::gdb::IOException)
  
  Sets a log error file.

- void SetLogOff ()
  
  Turns off all the error reporting.

- virtual ~TypeLoader ()
  
  Destructor.

- void Register (TypeLoaderListener &tel)
  
  Registers a new listener.

- virtual void Run ()=0 throw (dex::gdb::IOException, dex::gdb::Error)
  
  Run the loader.

- virtual void RunTwoPhases ()=0 throw (dex::gdb::IOException, dex::gdb::Error)
  
  Run the loader for two phases loading.

- virtual void RunNPhases (dex::gdb::int32_t partitions)=0 throw (dex::gdb::IOException, dex::gdb::Error)
  
  Run the loader for N phases loading.

- void SetRowReader (RowReader &rr)
  
  Sets the input data source.

- void SetGraph (dex::gdb::Graph &g)
  
  Sets the graph where the data will be loaded.

- void SetLocale (const std::wstring &localeStr)
  
  Sets the locale that will be used to read the data.

- void SetType (dex::gdb::type_t)
  
  Sets the type to be loaded.

- void SetAttributes (dex::gdb::AttributeList &attrs)
  
  Sets the list of Attributes.

- void SetAttributePositions (dex::gdb::Int32List &attrsPos)
  
  Sets the list of attribute positions.

- void SetTimestampFormat (const std::wstring &timestampFormat)
  
  Sets a specific timestamp format.

- void SetFrequency (dex::gdb::int32_t freq)
  
  Sets the frequency of listener notification.
Protected Types

- enum Mode {
    ONE_PHASE,
    TWO_PHASES,
    N_PHASES }

  Load can work in different ways.

Protected Member Functions

- dex::gdb::bool_t CanRun ()
  Checks that all the required settings are ready to run.

- void Run (Mode ph, dex::gdb::int32_t par) throw (dex::gdb::IOException, dex::gdb::Error)
  Runs load process.

- TypeLoader (RowReader &rr, dex::gdb::Graph &g, dex::gdb::type_t t, dex::gdb::AttributeList &attrs, dex::gdb::Int32List &attrsPos)
  Creates a new instance with the minimum common required arguments.

- TypeLoader ()
  Creates a new instance.

- void NotifyListeners (const TypeLoaderEvent &ev)
  Notifies progress to all registered listeners.

- void SetHeadAttribute (dex::gdb::attr_t attr)
  Sets the attribute that will be used to find the head of the edge.

- void SetHeadPosition (dex::gdb::int32_t pos)
  Sets the position of the head attribute in the source data.

- void SetTailAttribute (dex::gdb::attr_t attr)
  Sets the attribute that will be used to find the tail of the edge.

- void SetTailPosition (dex::gdb::int32_t pos)
  Sets the position of the tail attribute in the source data.

5.69.1 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 DEX User Manual for more details on this.
Author:
Sparsity Technologies http://www.sparsity-technologies.com

5.69.2 Member Enumeration Documentation

5.69.2.1 enum TypeLoader::Mode [protected]
Load can work in different ways.

Enumerator:
- **ONE_PHASE**  Performs the load in a phases.
  Load all objects an attributes at the same time.
- **TWO_PHASES** Performs the load in two phases.
  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.
- **N_PHASES** Performs the load in N phases.
  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 to the RowReader are necessary.
  Working on this mode it is necessary to build a temporary file.

5.69.3 Constructor & Destructor Documentation

5.69.3.1 TypeLoader::TypeLoader (RowReader & rr, dex::gdb::Graph & g, dex::gdb::type_t t, dex::gdb::AttributeList & attrs, dex::gdb::Int32List & attrsPos) [protected]
Creates a new instance with the minimum common required arguments.

Parameters:
- **rr** [in] Input RowReader.
- **g** [in] Graph.
- **t** [in] Type identifier.
- **attrs** [in] Attribute identifiers to be loaded
- **attrsPos** [in] Attribute positions (column index >=0)

5.69.4 Member Function Documentation

5.69.4.1 dex::gdb::bool_t TypeLoader::CanRun () [protected]
Checks that all the required settings are ready to run.

Returns:
Returns true if all the settings are ready.
5.69.4.2 void TypeLoader::Run (Mode \textit{ph}, dex::gdb::int32_t \textit{par}) throw (dex::gdb::IOException, dex::gdb::Error) [protected]
Runs load process.

Exceptions:

\textit{IOException} If bad things happen reading from the RowReader.

Parameters:

\textit{ph} [in] The load mode.
\textit{par} [in] Number of horizontal partitions to perform the load.

5.69.4.3 void TypeLoader::NotifyListeners (const TypeLoaderEvent & \textit{ev}) [protected]
Notifies progress to all registered listeners.

Parameters:

\textit{ev} [in] Progress event to be notified.

5.69.4.4 void TypeLoader::SetHeadAttribute (dex::gdb::attr_t \textit{attr}) [protected]
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:

\textit{attr} [in] Head Attribute
Reimplemented in EdgeTypeLoader.

5.69.4.5 void TypeLoader::SetHeadPosition (dex::gdb::int32_t \textit{pos}) [protected]
Sets the position of the head attribute in the source data.
This method is protected because only the Edge loaders should have it.

Parameters:

\textit{pos} [in] Head position
Reimplemented in EdgeTypeLoader.

5.69.4.6 void TypeLoader::SetTailAttribute (dex::gdb::attr_t \textit{attr}) [protected]
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:

\textit{attr} [in] Tail Attribute
Reimplemented in EdgeTypeLoader.
5.69.4.7  void TypeLoader::SetTailPosition (dex::gdb::int32_t pos)  [protected]
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

Reimplemented in EdgeTypeLoader.

5.69.4.8  void TypeLoader::SetLogError (const std::wstring & path) throw (dex::gdb::IOException)
Sets a log error file.
By default errors are thrown as an exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Exceptions:

  IOException  If bad things happen opening the file.

5.69.4.9  void TypeLoader::SetLogOff ()
Turns off all the error reporting.
The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.69.4.10  void TypeLoader::Register (TypeLoaderListener & tel)
Registers a new listener.

Parameters:

  ← tel  TypeLoaderListener to be registered.

5.69.4.11  virtual void TypeLoader::RunTwoPhases () throw (dex::gdb::IOException, dex::gdb::Error)  [pure virtual]
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.
Implemented in EdgeTypeLoader, and NodeTypeLoader.

5.69.4.12  virtual void TypeLoader::RunNPhases (dex::gdb::int32_t partitions) throw (dex::gdb::IOException, dex::gdb::Error)  [pure virtual]
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.

Implemented in `EdgeTypeLoader`, and `NodeTypeLoader`.

**5.69.4.13 void TypeLoader::SetRowReader (RowReader & rr)**

Sets the input data source.

**Parameters:**

- **rr** [in] Input `RowReader`.

**5.69.4.14 void TypeLoader::SetGraph (dex::gdb::Graph & g)**

Sets the graph where the data will be loaded.

**Parameters:**

- **g** [in] `Graph`.

**5.69.4.15 void TypeLoader::SetLocale (const std::wstring & 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`.

**5.69.4.16 void TypeLoader::SetType (dex::gdb::type_t t)**

Sets the type to be loaded.

**Parameters:**

- **t** [in] `Type` identifier.

**5.69.4.17 void TypeLoader::SetAttributes (dex::gdb::AttributeList & attrs)**

Sets the list of Attributes.

**Parameters:**

- **attrs** [in] `Attribute` identifiers to be loaded

**5.69.4.18 void TypeLoader::SetAttributePositions (dex::gdb::Int32List & attrsPos)**

Sets the list of attribute positions.

`attrsPos` [in] `Attribute` positions (column index $\geq 0$).
5.69.4.19 void TypeLoader::SetFrequency (dex::gdb::int32_t freq)
Sets the frequency of listener notification.
freq [in] Frequency in number of rows managed to notify progress to all listeners
The documentation for this class was generated from the following file:

• TypeLoader.h

5.70 TypeLoaderEvent Class Reference

Provides information about the progress of a TypeLoader instance.

Public Member Functions

• virtual ~TypeLoaderEvent ()
  Destructor.

• dex::gdb::type_t GetTypeID () const
  Gets the type identifier.

• dex::gdb::int64_t GetCount () const
  Gets the current number of objects created.

• dex::gdb::int32_t GetPhase () const
  Gets the current phase.

• dex::gdb::int32_t GetTotalPhases () const
  Gets the total number of phases.

• dex::gdb::int32_t GetPartition () const
  Gets the current partition.

• dex::gdb::int32_t GetTotalPartitions () const
  Gets the total number of partitions.

• dex::gdb::int32_t GetTotalPartitionSteps () const
  Gets the total number of steps in the current partition.

• dex::gdb::bool_t IsLast () const
  Gets if this is the last event or not.

Friends

• class TypeLoader
5.70.1 Detailed Description

Provides information about the progress of a TypeLoader instance.
Check out the 'Data import' section in the DEX User Manual for more details on this.

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

5.70.2 Member Function Documentation

5.70.2.1 dex::gdb::type_t TypeLoaderEvent::GetTypeId () const [inline]
Gets the type identifier.

Returns:
The type identifier.

5.70.2.2 dex::gdb::int64_t TypeLoaderEvent::GetCount () const [inline]
Gets the current number of objects created.

Returns:
The current number of objects created.

5.70.2.3 dex::gdb::int32_t TypeLoaderEvent::GetPhase () const [inline]
Gets the current phase.

Returns:
The current phase.

5.70.2.4 dex::gdb::int32_t TypeLoaderEvent::GetTotalPhases () const [inline]
Gets the total number of phases.

Returns:
The total number of phases.

5.70.2.5 dex::gdb::int32_t TypeLoaderEvent::GetPartition () const [inline]
Gets the current partition.

Returns:
The current partition.
5.71 TypeLoaderListener Class Reference

5.70.2.6 \texttt{dex::gdb::int32\_t TypeLoaderEvent::GetTotalPartitions () const} \hfill [inline]

Gets the total number of partitions.

\textbf{Returns:}

The total number of partitions.

5.70.2.7 \texttt{dex::gdb::int32\_t TypeLoaderEvent::GetTotalPartitionSteps () const} \hfill [inline]

Gets the total number of steps in the current partition.

\textbf{Returns:}

The total number steps in the current partition.

5.70.2.8 \texttt{dex::gdb::bool\_t TypeLoaderEvent::IsLast () const} \hfill [inline]

Gets if this is the last event or not.

\textbf{Returns:}

TRUE if this is the last event, FALSE otherwise.

The documentation for this class was generated from the following file:

- TypeLoader.h

5.71 TypeLoaderListener Class Reference

Interface to be implemented to receive \texttt{TypeLoaderEvent} events from a \texttt{TypeLoader}.

\textbf{Public Member Functions}

- virtual void \texttt{NotifyEvent (const TypeLoaderEvent \&ev)=0}

  \textit{Method to receive events from a \texttt{Loader}.}

- virtual \texttt{\~TypeLoaderListener ()}

  \textit{Destructor.}

\textbf{Protected Member Functions}

- \texttt{TypeLoaderListener ()}

  \textit{Protected because none should instantiate a \texttt{RowWriter}.}
5.71.1 Detailed Description

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader. Check out the ‘Data import’ section in the DEX User Manual for more details on this.

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

5.71.2 Constructor & Destructor Documentation

5.71.2.1 TypeLoaderListener::TypeLoaderListener () [inline, protected]
Protected because none should instantiate a RowWriter. Just inherited classes may use this empty constructor.

5.71.3 Member Function Documentation

5.71.3.1 virtual void TypeLoaderListener::NotifyEvent (const TypeLoaderEvent & ev) [pure virtual]
Method to receive events from a Loader.

Parameters:

   
   * ev  Loader::LoaderEvent with information from a running Loader.

The documentation for this class was generated from the following file:

   
   * TypeLoader.h

5.72 UnsupportedOperationError Class Reference

Unsupported operation error class.

Inheritance diagram for UnsupportedOperationError:

```
Exception

Error

AppError

UnsupportedOperationError
```

Generated on Tue Jul 16 12:10:09 2013 for Dex by Doxygen
Public Member Functions

- **UnsupportedOperationError ()**
  
  *Creates a new instance.*

- **UnsupportedOperationError (const std::string &mess)**
  
  *Creates a new instance.*

- **virtual ~UnsupportedOperationError ()**
  
  *Destructor.*

- **const std::string & Message () const**
  
  *Gets the message of the exception.*

- **void SetMessage (const std::string &mess)**
  
  *Sets the message of the exception.*

Static Public Member Functions

- **static Error NewError (int32_t coreErrorCode)**
  
  *Creates a new Error instance from a dexcore error code.*

Protected Attributes

- **std::string message**
  
  *Message of the exception.*

5.72.1 Detailed Description

Unsupported operation error class.

Author:

Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)
5.72.2 Constructor & Destructor Documentation

5.72.2.1 UnsupportedOperationError::UnsupportedOperationError (const std::string & mess)
Creates a new instance.

Parameters:
  
mess [in] Message of the exception.

5.72.3 Member Function Documentation

5.72.3.1 static Error Error::NewError (int32_t coreErrorCode) [static, inherited]
Creates a new Error instance from a dexcore error code.

Parameters:
  

Returns:
  
Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.

5.72.3.2 const std::string& Exception::Message () const [inherited]
Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:
  
The message of the exception.

5.72.3.3 void Exception::SetMessage (const std::string & mess) [inherited]
Sets the message of the exception.

Parameters:
  

The documentation for this class was generated from the following file:

• Exception.h

5.73 Value Class Reference

Value class.
Inheritance diagram for Value:

Collaboration diagram for Value:

Public Member Functions

- **Value ()**
  *Creates a new instance.*

- **Value (const Value &v)**
  *Copy constructor.*

- **virtual ~Value ()**
  *Destructor.*

- **Value & operator= (const Value &v)**
  *Assignment operator.*

- **bool_t IsNull () const**
  *Gets if this is a NULL Value.*

- **void SetNullVoid ()**
  *Sets the Value to NULL.*

- **Value & SetNull ()**
  *Sets the Value to NULL.*
• **DataType** GetDataType () const
  
  Gets the **DataType**.

• **bool_t** GetBoolean () const
  
  Gets **Boolean Value**.

• **int32_t** GetInteger () const
  
  Gets **Integer Value**.

• **int64_t** GetLong () const
  
  Gets **Long Value**.

• **double64_t** GetDouble () const
  
  Gets **Double Value**.

• **int64_t** GetTimestamp () const
  
  Gets **Timestamp Value**.

• **const std::wstring &** GetString () const
  
  Gets **String Value**.

• **oid_t** GetOID () const
  
  Gets **OID Value**.

• void SetBooleanVoid (bool_t v)
  
  Sets the **Value**.

• **Value &** SetBoolean (bool_t v)
  
  Sets the **Value**.

• void SetIntegerVoid (int32_t v)
  
  Sets the **Value**.

• **Value &** SetInteger (int32_t v)
  
  Sets the **Value**.

• void SetLongVoid (int64_t v)
  
  Sets the **Value**.

• **Value &** SetLong (int64_t v)
  
  Sets the **Value**.

• void SetDoubleVoid (double64_t v)
  
  Sets the **Value**.

• **Value &** SetDouble (double64_t v)
  
  Sets the **Value**.

• void SetTimestampVoid (int64_t v)
  
  Sets the **Value**.
• void SetTimestampVoid (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t millisecs)
  Sets the Value.

• Value & SetTimestamp (int64_t v)
  Sets the Value.

• Value & SetTimestamp (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t millisecs)
  Sets the Value.

• void SetStringVoid (const std::wstring &v)
  Sets the Value.

• Value & SetString (const std::wstring &v)
  Sets the Value.

• void SetOIDVoid (oid_t v)
  Sets the OID Value.

• Value & SetOID (oid_t v)
  Sets the Value.

• void SetVoid (Value &v)
  Sets the Value.

• Value & Set (Value &v)
  Sets the Value.

• int32_t Compare (const Value &v) const
  Compares with the given Value.

• bool_t Equals (const Value &v) const
  Compares with the given Value.

• std::wstring & ToString (std::wstring &str) const
  Gets a string representation of the Value.

### Static Public Attributes

• static const int32_t MaxLengthString
  Maximum number of characters allowed for a String.

### Friends

• class Graph
• class ValuesIterator
5.73 Value Class Reference

5.73.1 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

5.73.2 Constructor & Destructor Documentation

5.73.2.1 Value::Value ()
Creates a new instance.
It creates a NULL Value.

5.73.2.2 Value::Value (const Value & v)
Copy constructor.

Parameters:

v [in] Value to be copied.

5.73.3 Member Function Documentation

5.73.3.1 Value& Value::operator= (const Value & v)
Assignment operator.

Parameters:

v [in] Value to be copied.

5.73.3.2 bool_t Value::IsNull () const
Gets if this is a NULL Value.

Returns:

TRUE if this is a NULL Value, FALSE otherwise.

Reimplemented from Handler< dexcore::Value >.

5.73.3.3 Value& Value::SetNull () [inline]
Sets the Value to NULL.

Returns:

The calling instance.
5.73.3.4   **DataType** Value::GetDataType () const
Gets the DataType.
**Value** cannot be NULL.

**Returns:**
   The DataType.

5.73.3.5   **bool_t** Value::GetBoolean () const
Gets Boolean **Value**.
This must be a non-NULL Boolean **Value**.

**Returns:**
   The Boolean **Value**.

5.73.3.6   **int32_t** Value::GetInteger () const
Gets Integer **Value**.
This must be a non-NULL Integer **Value**.

**Returns:**
   The Integer **Value**.

5.73.3.7   **int64_t** Value::GetLong () const
Gets Long **Value**.
This must be a non-NULL Long **Value**.

**Returns:**
   The Long **Value**.

5.73.3.8   **double64_t** Value::GetDouble () const
Gets Double **Value**.
This must be a non-NULL Double **Value**.

**Returns:**
   The Double **Value**.

5.73.3.9   **int64_t** Value::GetTimestamp () const
Gets Timestamp **Value**.
This must be a non-NULL Timestamp **Value**.

**Returns:**
   The Timestamp **Value**.
5.73.3.10 const std::wstring& Value::GetString () const
Gets String Value.
This must be a non-NULL String Value.

Returns:
The String Value.

5.73.3.11 oid_t Value::GetOID () const
Gets OID Value.
This must be an non-NULL OID Value.

Returns:
The OID Value.

5.73.3.12 void Value::SetBooleanVoid (bool_t v)
Sets the Value.
Parameters:
  v [in] New Boolean value.

5.73.3.13 Value& Value::SetBoolean (bool_t v) [inline]
Sets the Value.
Parameters:
  v [in] New Boolean value.

Returns:
The calling instance.

5.73.3.14 void Value::SetIntegerVoid (int32_t v)
Sets the Value.
Parameters:
  v [in] New Integer value.

5.73.3.15 Value& Value::SetInteger (int32_t v) [inline]
Sets the Value.
Parameters:
  v [in] New Integer value.

Returns:
The calling instance.
5.73.3.16  void Value::SetLongVoid (int64_t v)
Sets the Value.

Parameters:
   v [in] New Long value.

5.73.3.17  Value& Value::SetLong (int64_t v) [inline]
Sets the Value.

Parameters:
   v [in] New Long value.

Returns:
   The calling instance.

5.73.3.18  void Value::SetDoubleVoid (double64_t v)
Sets the Value.

Parameters:
   v [in] New Double value.

5.73.3.19  Value& Value::SetDouble (double64_t v) [inline]
Sets the Value.

Parameters:
   v [in] New Double value.

Returns:
   The calling instance.

5.73.3.20  void Value::SetTimestampVoid (int64_t v)
Sets the Value.

Parameters:
5.73.3.21 void Value::SetTimestampVoid (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t 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]).

5.73.3.22 Value& Value::SetTimestamp (int64_t v) [inline]
Sets the Value.

Parameters:

- **v** [in] New Timestamp value.

Returns:

The calling instance.

5.73.3.23 Value& Value::SetTimestamp (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t millisecs) [inline]
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]).

Returns:

The calling instance.

5.73.3.24 void Value::SetStringValue (const std::wstring & v)
Sets the Value.

Parameters:

- **v** [in] New String value.
5.73.3.25  `Value& Value::SetString (const std::wstring & v)` [inline]

Sets the `Value`.

**Parameters:**

- `v` [in] New String value.

**Returns:**

The calling instance.

5.73.3.26  `void Value::SetOIDVoid (oid_t v)`

Sets the OID `Value`.

**Parameters:**

- `v` [in] New OID value.

5.73.3.27  `Value& Value::SetOID (oid_t v)` [inline]

Sets the `Value`.

**Parameters:**

- `v` [in] New OID `Value`.

**Returns:**

The calling instance.

5.73.3.28  `void Value::SetVoid (Value & v)` [inline]

Sets the `Value`.

**Parameters:**

- `v` [in] New value.

5.73.3.29  `Value& Value::Set (Value & v)` [inline]

Sets the `Value`.

**Parameters:**

- `v` [in] New value.

**Returns:**

The calling instance.
5.73.3.30  int32_t Value::Compare (const Value & v) const
Compares with the given Value.
It does not work for Text or if given Value objects does not have the same DataType.

Parameters:
   v  Given value to compare to.

Returns:
   0 if this Value is equal to the given one; a value less than 0 if this Value is less than the given one; and a value greater than 0 if this Value is greater than the given one.

5.73.3.31  bool_t Value::Equals (const Value & v) const
Compares with the given Value.
It does not work for Text or if given Value objects does not have the same DataType.

Parameters:
   v  Given value to compare to.

Returns:
   TRUE if this Value is equal to the given one; FALSE otherwise.

5.73.3.32  std::wstring& Value::ToString (std::wstring & str) const
Gets a string representation of the Value.
It does not work for Text.

Parameters:
   str  String to be used. It is cleared and set with the string representation of the Value.

Returns:
   The given string which has been updated.

The documentation for this class was generated from the following file:

   • Value.h

5.74  Values Class Reference

Value set class.
Inheritance diagram for Values:

Collaboration diagram for Values:

**Public Member Functions**

- virtual ~Values ()
  
  Destructor.

- int64_t Count ()
  
  Gets the number of elements into the collection.
5.75 ValuesIterator Class Reference

- `ValuesIterator * Iterator (Order order)`
  
  Gets a `ValuesIterator`.

Friends

- class `Graph`
- class `ValuesIterator`

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

5.74.2 Member Function Documentation

5.74.2.1 `int64_t Values::Count ()`

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

5.74.2.2 `ValuesIterator* Values::Iterator (Order order)`

Gets a `ValuesIterator`.

Returns:

`ValuesIterator` instance.

Parameters:

`order [in]` Ascending or descending order.

The documentation for this class was generated from the following file:

- `Values.h`

5.75 ValuesIterator Class Reference

`Values` iterator class.
Inheritance diagram for ValuesIterator:
Collaboration diagram for ValuesIterator:

```
T
  ^
  | handler
  |
  Handler< T >

< dexcore::GraphPool >  < dexcore::DEX >
< dexcore::Session >
Handler< dexcore::DEX >

< dexcore::GraphPool >
Handler< dexcore::GraphPool >

< dexcore::Values >
Handler< dexcore::Values >
Handler< dexcore::Session >
< dexcore::DbGraph >
Handler< dexcore::DbGraph >

< dexcore::DbGraph >

Dex
dex

Database
db

Session
sess

Graph
graph

Values
values

ValuesIterator
```

**Public Member Functions**

- **virtual ~ValuesIterator ()**
  
  Destructor.

- **bool_t HasNext ()**
  
  Gets if there are more elements to traverse.

- **Value * Next ()**
  
  Gets the next element to traverse.
Friends

- class Values

5.75.1 Detailed Description

Values iterator class.
It allows for traversing all the elements into a Values instance.

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

5.75.2 Member Function Documentation

5.75.2.1 bool_t ValuesIterator::HasNext ()
Gets if there are more elements to traverse.

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

5.75.2.2 Value* ValuesIterator::Next ()
Gets the next element to traverse.

Returns:
The next element.

The documentation for this class was generated from the following file:

- ValuesIterator.h

5.76 WeakConnectivity Class Reference

WeakConnectivity class.
Inheritance diagram for WeakConnectivity:
Collaboration diagram for WeakConnectivity:

```
+-------------------+
| ConnectedComponents |
|                   |
|       ↑           |
|       ↓           |
|     Connectivity |
|                   |
|                   |
|     WeakConnectivity |
```

Public Member Functions

- virtual ~WeakConnectivity ()
  Destructor.

- virtual void AddEdgeType (dex::gdb::type_t t)
  Allows connectivity through edges of the given type.

- virtual void AddAllEdgeTypes ()
  Allows connectivity through all edge types of the graph.

- virtual void AddNodeType (dex::gdb::type_t t)
  Allows connectivity through nodes of the given type.

- virtual void AddAllNodeTypes ()
  Allows connectivity through all node types of the graph.

- virtual void ExcludeNodes (dex::gdb::Objects &nodes)
  Set which nodes can’t be used.

- virtual void ExcludeEdges (dex::gdb::Objects &edges)
  Set which edges can’t be used.

- ConnectedComponents * GetConnectedComponents ()
  Returns the results generated by the execution of the algorithm.

- virtual void Run ()=0
  Runs the algorithm in order to find the connected components.

- void SetMaterializedAttribute (const std::wstring &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.

Protected Types

- typedef std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > EdgeTypes_t
  A type definition to store allowed edge types.
typedef std::vector< dex::gdb::type_t > NodeTypes_t
A type definition to store allowed node types.

Protected Member Functions

- **WeakConnectivity** (dex::gdb::Session &s)
  Creates a new instance of WeakConnectivity.

- **void AddEdgeType** (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  Allows connectivity through edges of the given type.

- **void AddAllEdgeTypes** (dex::gdb::EdgesDirection d)
  Allows connectivity through all edge types of the graph.

- **void AssertAddedEdges**()
  Check that edges had been added.

- **void AssertAddedNodes**()
  Check that nodes had been added.

- **void AssertNotComputed**()
  Check that the connectivity had not been calculated yet.

- **void SetConnectedComponent** (dex::gdb::oid_t idNode)
  Assigns the current component to the given node.

- **void SetNodesNotVisited**()
  Set all the selected nodes in nodesNotVisited.

- **void AssertNotComponentAttribute** (const std::wstring &attributeName)
  Check that the given attribute name is not already in use.

- **void AssertComputed**()
  Check that the connectivity had been calculated.

- **void AssertEdgeType** (dex::gdb::type_t edgetype)
  Check that the given edge type is valid.

- **void AssertNodeType** (dex::gdb::type_t nodetype)
  Check that the given node type is valid.

- **void CreateGlobalPersistentAttribute** (const std::wstring &attributeName)
  Set a new persistent global attribute to store the connectivity information.

- **void CreateGlobalTransientAttribute**()
  Set a new temporary global attribute to store the connectivity information.

- **void RemoveGlobalAttribute**()
Removes the global attribute where the connectivity information is stored.

- `dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)`
  Check if the given node has an allowed type.

- `dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)`
  Check if the given node is forbidden.

- `dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)`
  Check if the given edge is forbidden.

Protected Attributes

- `dex::gdb::Session * sess`
  Session.

- `dex::gdb::Graph * graph`
  Graph.

- `EdgeTypes_t edgeTypes`
  Allowed edge types.

- `std::vector< dex::gdb::type_t > nodeTypes`
  Allowed node types.

- `dex::gdb::attr_t attrComponent`
  Common attribute where the connected component information is stored.

- `std::wstring attrComponentName`
  Name of the common attribute where the connected component information is stored.

- `dex::gdb::int64_t actualComponent`
  Current component identifier.

- `dex::gdb::Objects * nodesNotVisited`
  Identifiers of the nodes not visited.

- `dex::gdb::bool_t matResults`
  Materialized results.

- `dex::gdb::bool_t computed`
  True if the connectivity has been calculated.

- `dex::gdb::Objects * excludedNodes`
  The set of excluded nodes.

- `dex::gdb::Objects * excludedEdges`
  The set of excluded edges.
5.76 WeakConnectivity Class Reference

- ConnectedComponents * ccs
  The calculated connectivity information.

5.76.1 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 ConnectedComponents class using the getConnectedComponents method.

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

Author:

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

5.76.2 Constructor & Destructor Documentation

5.76.2.1 WeakConnectivity::WeakConnectivity (dex::gdb::Session & s)  [protected]

Creates a new instance of WeakConnectivity.

Parameters:

- \(s\) [in] Session to get the graph from and calculate the connectivity

5.76.3 Member Function Documentation

5.76.3.1 virtual void WeakConnectivity::AddEdgeType (dex::gdb::type_t t)  [virtual]

Allows connectivity through edges of the given type.

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

Parameters:

- \(t\) [in] Edge type.
- \(d\) [in] Edge direction.

5.76.3.2 virtual void WeakConnectivity::AddAllEdgeTypes ()  [virtual]

Allows connectivity through all edge types of the graph.

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

Parameters:

- \(d\) [in] Edge direction.
5.76.3.3 virtual void Connectivity::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]

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.

5.76.3.4 virtual void Connectivity::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]

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.

5.76.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

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.

5.76.3.6 virtual void Connectivity::Run () [pure virtual, inherited]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in StrongConnectivityGabow, and WeakConnectivityDFS.

5.76.3.7 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]

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.

5.76.3.8 void Connectivity::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [protected, inherited]
Allows connectivity through edges of the given type.

Parameters:

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

Reimplemented in StrongConnectivity.

5.76.3.9 void Connectivity::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [protected, inherited]
Allows connectivity through all edge types of the graph.

Parameters:

  d [in] Edge direction.

Reimplemented in StrongConnectivity.

5.76.3.10 void Connectivity::SetNodesNotVisited () [protected, inherited]
Set all the selected nodes in nodesNotVisited.
That’s all the nodes of the allowed node types but not the excluded ones.
The documentation for this class was generated from the following file:

- WeakConnectivity.h

5.77 WeakConnectivityDFS Class Reference

WeakConnectivityDFS class.
Inheritance diagram for WeakConnectivityDFS:
Collaboration diagram for WeakConnectivityDFS:

```
Collaboration diagram for WeakConnectivityDFS:
```

Public Member Functions

- **WeakConnectivityDFS** (dex::gdb::Session &s)
  
  Creates a new instance of WeakConnectivityDFS.

- **virtual ~WeakConnectivityDFS** ()
  
  Destructor.

- **void Run** ()
  
  Executes the algorithm.

- **virtual void AddEdgeType** (dex::gdb::type_t t)
  
  Allows connectivity through edges of the given type.

- **virtual void AddAllEdgeTypes** ()
  
  Allows connectivity through all edge types of the graph.

- **virtual void AddNodeType** (dex::gdb::type_t t)
  
  Allows connectivity through nodes of the given type.

- **virtual void AddAllNodeTypes** ()
  
  Allows connectivity through all node types of the graph.

- **virtual void ExcludeNodes** (dex::gdb::Objects &nodes)
  
  Set which nodes can’t be used.

- **virtual void ExcludeEdges** (dex::gdb::Objects &edges)
  
  Set which edges can’t be used.

- **ConnectedComponents * GetConnectedComponents** ()
  
  Returns the results generated by the execution of the algorithm.

- **void SetMaterializedAttribute** (const std::wstring &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.
**Protected Types**

- typedef std::map< dex::gdb::type_t, dex::gdb::EdgesDirection > EdgeTypes_t
  
  A type definition to store allowed edge types.

- typedef std::vector< dex::gdb::type_t > NodeTypes_t
  
  A type definition to store allowed node types.

**Protected Member Functions**

- void AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d)
  
  Allows connectivity through edges of the given type.

- void AddAllEdgeTypes (dex::gdb::EdgesDirection d)
  
  Allows connectivity through all edge types of the graph.

- void AssertAddedEdges ()
  
  Check that edges had been added.

- void AssertAddedNodes ()
  
  Check that nodes had been added.

- void AssertNotComputed ()
  
  Check that the connectivity had not been calculated yet.

- void SetConnectedComponent (dex::gdb::oid_t idNode)
  
  Assigns the current component to the given node.

- void SetNodesNotVisited ()
  
  Set all the selected nodes in nodesNotVisited.

- void AssertNotComponentAttribute (const std::wstring &attributeName)
  
  Check that the given attribute name is not already in use.

- void AssertComputed ()
  
  Check that the connectivity had been calculated.

- void AssertEdgeType (dex::gdb::type_t edgetype)
  
  Check that the given edge type is valid.

- void AssertNodeType (dex::gdb::type_t nodetype)
  
  Check that the given node type is valid.

- void CreateGlobalPersistentAttribute (const std::wstring &attributeName)
  
  Set a new persistent global attribute to store the connectivity information.

- void CreateGlobalTransientAttribute ()
  
  Set a new temporary global attribute to store the connectivity information.
• **void RemoveGlobalAttribute ()**
  Removes the global attribute where the connectivity information is stored.

• **dex::gdb::bool_t IsNodeTypeAllowed (dex::gdb::oid_t nodeId)**
  Check if the given node has an allowed type.

• **dex::gdb::bool_t IsNodeExcluded (dex::gdb::oid_t node)**
  Check if the given node is forbidden.

• **dex::gdb::bool_t IsEdgeExcluded (dex::gdb::oid_t edge)**
  Check if the given edge is forbidden.

**Protected Attributes**

• **dex::gdb::Session * sess**
  Session.

• **dex::gdb::Graph * graph**
  Graph.

• **EdgeTypes_t edgeTypes**
  Allowed edge types.

• **std::vector< dex::gdb::type_t > nodeTypes**
  Allowed node types.

• **dex::gdb::attr_t attrComponent**
  Common attribute where the connected component information is stored.

• **std::wstring attrComponentName**
  Name of the common attribute where the connected component information is stored.

• **dex::gdb::int64_t actualComponent**
  Current component identifier.

• **dex::gdb::Objects * nodesNotVisited**
  Identifiers of the nodes not visited.

• **dex::gdb::bool_t matResults**
  Materialized results.

• **dex::gdb::bool_t computed**
  True if the connectivity has been calculated.

• **dex::gdb::Objects * excludedNodes**
  The set of excluded nodes.

• **dex::gdb::Objects * excludedEdges**
The set of excluded edges.

- **ConnectedComponents** * ccs
  
  The calculated connectivity information.

### 5.77.1 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 DEX User Manual for more details on this.

**Author:**

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

### 5.77.2 Constructor & Destructor Documentation

#### 5.77.2.1 WeakConnectivityDFS::WeakConnectivityDFS (dex::gdb::Session & s)

Creates a new instance of **WeakConnectivityDFS**.

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

**Parameters:**

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

### 5.77.3 Member Function Documentation

#### 5.77.3.1 virtual void WeakConnectivity::AddEdgeType (dex::gdb::type_t t) [virtual, inherited]

Allows connectivity through edges of the given type.

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

**Parameters:**

- **t** [in] Edge type.
- **d** [in] Edge direction.
5.77.3.2 void Connectivity::AddEdgeType (dex::gdb::type_t t, dex::gdb::EdgesDirection d) [protected, inherited]
Allows connectivity through edges of the given type.

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

Reimplemented in StrongConnectivity.

5.77.3.3 virtual void WeakConnectivity::AddAllEdgeTypes () [virtual, inherited]
Allows connectivity through all edge types of the graph.
In a weak connectivity the edges can be used in Any direction.

Parameters:
- d [in] Edge direction.

5.77.3.4 void Connectivity::AddAllEdgeTypes (dex::gdb::EdgesDirection d) [protected, inherited]
Allows connectivity through all edge types of the graph.

Parameters:
- d [in] Edge direction.

Reimplemented in StrongConnectivity.

5.77.3.5 virtual void Connectivity::ExcludeNodes (dex::gdb::Objects & nodes) [virtual, inherited]
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.

5.77.3.6 virtual void Connectivity::ExcludeEdges (dex::gdb::Objects & edges) [virtual, inherited]
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.
5.77.3.7 ConnectedComponents\texttt{ Connectivity::GetConnectedComponents ()} [inherited]

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.

5.77.3.8 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]

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:**

\texttt{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.

5.77.3.9 void Connectivity::SetNodesNotVisited () [protected, inherited]

Set all the selected nodes in nodesNotVisited.

That’s all the nodes of the allowed node types but not the excluded ones.

The documentation for this class was generated from the following file:

- WeakConnectivityDFS.h

5.78 WrongArgumentError Class Reference

Wrong argument error class.
Inheritance diagram for WrongArgumentError:

Collaboration diagram for WrongArgumentError:

Public Member Functions

- **WrongArgumentError ()**
  
  *Creates a new instance.*

- **WrongArgumentError (const std::string &mess)**
  
  *Creates a new instance.*

- virtual **~WrongArgumentError ()**
  
  *Destructor.*

- const std::string & **Message ()** const
  
  *Gets the message of the exception.*

- void **setMessage (const std::string &mess)**
  
  *Sets the message of the exception.*

Static Public Member Functions

- static **Error NewError (int32_t coreErrorCode)**
5.78  WrongArgumentError Class Reference

Creates a new Error instance from a dexcore error code.

Protected Attributes

- std::string message
  Message of the exception.

5.78.1 Detailed Description

Wrong argument error class.

Author:

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

5.78.2 Constructor & Destructor Documentation

5.78.2.1 WrongArgumentError::WrongArgumentError (const std::string & mess)

Creates a new instance.

Parameters:

  mess  [in] Message of the exception.

5.78.3 Member Function Documentation

5.78.3.1 static Error Error::NewError (int32_t coreErrorCode)  [static, inherited]

Creates a new Error instance from a dexcore error code.

Parameters:


Returns:

  Depending on the given dexcore error, this may return an Error instance or an specific Error subclass instance.

5.78.3.2 const std::string& Exception::Message () const  [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

  The message of the exception.
5.78.3.3  void Exception::setMessage (const std::string & mess)  [inherited]
Sets the message of the exception.

Parameters:


The documentation for this class was generated from the following file:

  • Exception.h
Index

~Handler
gdb, 20

Add
AttributeList, 31
BooleanList, 38
Int32List, 128
Objects, 156
OIDList, 164
StringList, 201
TypeList, 243

AddAllEdgeTypes
Connectivity, 47
Context, 49
ShortestPath, 179
SinglePairShortestPath, 185
SinglePairShortestPathBFS, 190
SinglePairShortestPathDijkstra, 197
StrongConnectivity, 206
StrongConnectivityGabow, 212
Traversal, 221
TraversalBFS, 226
TraversalDFS, 231
WeakConnectivity, 275, 277
WeakConnectivityDFS, 282

AddEdgeType
Connectivity, 46
Context, 49
ShortestPath, 179
SinglePairShortestPath, 185
SinglePairShortestPathBFS, 190
SinglePairShortestPathDijkstra, 196
StrongConnectivity, 206
StrongConnectivityGabow, 212
Traversal, 221
TraversalBFS, 226
TraversalDFS, 231
WeakConnectivity, 275, 277
WeakConnectivityDFS, 281

AddWeightedEdgeType
SinglePairShortestPathDijkstra, 196

Algorithms, 24
Any
gdb, 17
Objects, 157

AppError, 26
AppError, 27
Message, 27
NewError, 27
SetMessage, 28

Ascendent
gdb, 19
AsDirected
EdgeExport, 84
Attribute, 28
GetCount, 30
GetDataType, 29
GetId, 29
GetKind, 30
GetName, 29
GetSize, 29
GetTypeId, 29
IsSessionAttribute, 30

AttributeKind
gdb, 15
AttributeList, 30
Add, 31
AttributeList, 31
Count, 31
Iterator, 31

AttributeListIterator, 32
HasNext, 32
Next, 32

AttributeStatistics, 33
GetAvgLengthString, 35
GetDistinct, 35
GetMax, 35
GetMaxLengthString, 35
GetMean, 36
GetMedian, 36
GetMin, 35
GetMinLengthString, 35
GetMode, 36
GetModeCount, 36
GetNull, 34
GetTotal, 34
GetVariance, 36

Backup
Graph, 124

Basic
gdb, 15

Between
gdb, 16

Boolean
gdb, 17

BooleanList, 37
Add, 38
BooleanList, 38
Count, 38
Iterator, 38

BooleanListIterator, 38
HasNext, 39
Next, 39
Box
gdb, 18
CanRun
  EdgeTypeExporter, 89
  EdgeTypeLoader, 94
  NodeTypeExporter, 143
  NodeTypeLoader, 148
  TypeExporter, 237
  TypeLoader, 247
CheckOnlyExistence
  SinglePairShortestPathBFS, 190
Close
  CSVReader, 54
  CSVWriter, 57
  RowReader, 169
  RowWriter, 171
  TextStream, 218
ColorRGB
gdb, 15
CombineDifference
  Objects, 159
CombineIntersection
  Objects, 159
CombineUnion
  Objects, 158
Compare
  Value, 265
Compute
  Context, 50
Condition
gdb, 15
Config
gdb, 18
ConnectedComponents, 39
  ConnectedComponents, 40
  GetConnectedComponent, 41
  GetCount, 41
  GetNodes, 41
  GetSize, 41
Connectivity, 42
  AddAllEdgeTypes, 47
  AddEdgeType, 46
  Connectivity, 45
  ExcludeEdges, 45
  ExcludeNodes, 45
  GetConnectedComponents, 46
  Run, 46
  SetMaterializedAttribute, 46
  SetNodesNotVisited, 47
Contains
  Objects, 158
Context, 47
  AddAllEdgeTypes, 49
  AddEdgeType, 49
  Compute, 50
  Context, 49
  ExcludeEdges, 49
  ExcludeNodes, 49
  SetMaximumHops, 50
Copy
  Objects, 156, 159
Count
  AttributeList, 31
  BooleanList, 38
  Int32List, 128
  Objects, 156
  OIDList, 163
  StringList, 201
  TypeList, 243
  Values, 268
CountEdges
  Graph, 112
CountNodes
  Graph, 112
Create
  Dex, 66
CSVReader, 51
  Close, 54
  GetRow, 54
  Open, 53
  Read, 54
  Reset, 54
  SetLocale, 53
  SetMultilines, 53
  SetNumLines, 53
  SetQuotes, 53
  SetSeparator, 53
  SetStartLine, 53
CSVWriter, 55
  Close, 57
  Open, 57
  SetAutoQuotes, 56
  SetForcedQuotes, 57
  SetLocale, 57
  SetQuotes, 56
  SetSeparator, 56
  Write, 57
Database, 58
  GetAlias, 59
  GetPath, 59
  GetStatistics, 59
DatabaseStatistics, 60
  GetCache, 61
  GetData, 61
INDEX

Edges
Graph, 121

EdgesDirection
gdb, 17

EdgeTypeExporter, 86
CanRun, 89
EdgeTypeExporter, 88
NotifyListeners, 89
Register, 89
RunProcess, 89
SetAttributes, 90
SetFrequency, 90
SetGraph, 89
SetHeadAttribute, 88
SetHeader, 90
SetHeadPosition, 88
SetRowWriter, 89
SetTailAttribute, 88
SetTailPosition, 88
SetType, 90

EdgeTypeLoader, 90
CanRun, 94
EdgeTypeLoader, 93
Mode, 93
N_PHASES, 93
NotifyListeners, 95
ONE_PHASE, 93
Register, 95
Run, 94
SetAttributePositions, 96
SetAttributes, 96
SetFrequency, 96
SetGraph, 96
SetHeadAttribute, 94
SetHeadPosition, 94
SetLocale, 96
SetLogError, 95
SetLogOff, 95
SetRowReader, 95
SetTailAttribute, 94
SetTailPosition, 94
SetType, 96
TWO_PHASES, 93

EnableType
DefaultExport, 64
ExportManager, 103

Equal
gdb, 16

Equals
Objects, 158
Value, 266

Error
Error, 97
Message, 98

NewError, 98
SetMessage, 99

Exception, 99
Exception, 100
Message, 100
SetMessage, 100

ExcludeEdges
Connectivity, 45
Context, 49
ShortestPath, 180
SinglePairShortestPath, 185
SinglePairShortestPathBFS, 191
SinglePairShortestPathDijkstra, 197
StrongConnectivity, 207
StrongConnectivityGabow, 212
Traversal, 222
TraversalBFS, 227
TraversalDFS, 231
WeakConnectivity, 276
WeakConnectivityDFS, 282

ExcludeNodes
Connectivity, 45
Context, 49
ShortestPath, 180
SinglePairShortestPath, 185
SinglePairShortestPathBFS, 191
SinglePairShortestPathDijkstra, 197
StrongConnectivity, 206
StrongConnectivityGabow, 212
Traversal, 221
TraversalBFS, 226
TraversalDFS, 231
WeakConnectivity, 275
WeakConnectivityDFS, 282

Exists
Objects, 157

Explode
Graph, 120

Export
Graph, 124

ExportManager, 101
EnableType, 103
GetEdge, 103
GetEdgeType, 102
GetGraph, 102
GetNode, 103
GetNodeType, 102
Prepare, 102
Release, 102

ExportType
gdb, 17

FileNotFoundException, 104
FileNotFoundException, 105
LessEqual, 16
LessThan, 16
Like, 16
LikeNoCase, 16
LogLevel, 18
Long, 17
Node, 19
NodeShape, 18
NotEqual, 16
ObjectType, 18
Off, 18
OID, 17
operator<<, 20
Order, 19
Outgoing, 17
RegExp, 16
Round, 18
SetHandler, 20
Severe, 18
String, 17
Text, 17
Timestamp, 17
Unique, 15
Warning, 18
YGraphML, 18
GenerateSchemaScript
ScriptParser, 173
Get
DexProperties, 80
GetAlias
Database, 59
GetAreNeighborsIndexed
Type, 234
GetAttribute
Graph, 115, 118
GetAttributeIntervalCount
Graph, 117
GetAttributes
Graph, 123
GetAttributeStatistics
Graph, 116
GetAttributeText
Graph, 116
GetAvailableMem
PlatformStatistics, 167
GetAvgLengthString
AttributeStatistics, 35
GetBoolean
DexProperties, 80
Value, 261
GetCache
DatabaseStatistics, 61
GetCacheMaxSize
DexConfig, 74
GetCacheStatisticsEnabled
  DexConfig, 75
GetCacheStatisticsFile
  DexConfig, 75
GetCacheStatisticsSnapshotTime
  DexConfig, 75
GetColorRGB
  EdgeExport, 84
  NodeExport, 138
GetConnectedComponent
  ConnectedComponents, 41
GetConnectedComponents
  Connectivity, 46
  StrongConnectivity, 207
  StrongConnectivityGabow, 213
  WeakConnectivity, 276
  WeakConnectivityDFS, 282
GetCost
  SinglePairShortestPath, 184
  SinglePairShortestPathBFS, 190
  SinglePairShortestPathDijkstra, 196
GetCount
  Attribute, 30
  ConnectedComponents, 41
  TypeExporterEvent, 240
  TypeLoaderEvent, 252
GetCurrentDepth
  Traversal, 222
  TraversalBFS, 226
  TraversalDFS, 230
GetData
  DatabaseStatistics, 61
GetDataType
  Attribute, 29
  Value, 260
GetDistinct
  AttributeStatistics, 35
GetDouble
  Value, 261
GetEdge
  DefaultExport, 64
  EdgeData, 82
  ExportManager, 103
GetEdgeData
  Graph, 112
GetEdgePeer
  Graph, 113
GetEdgeType
  DefaultExport, 64
  ExportManager, 102
GetExtentPages
  DexConfig, 72
GetExtentSize
  DexConfig, 71
GetFontSize
  EdgeExport, 85
  NodeExport, 140
GetGraph
  DefaultExport, 63
  ExportManager, 102
  Session, 176
GetHandler
  gdb, 19
GetHead
  EdgeData, 82
GetHeight
  NodeExport, 139
GetHighAvailabilityCoordinators
  DexConfig, 78
GetHighAvailabilityEnabled
  DexConfig, 77
GetHighAvailabilityIP
  DexConfig, 77
GetHighAvailabilityMasterHistory
  DexConfig, 78
GetHighAvailabilitySynchronization
  DexConfig, 78
GetId
  Attribute, 29
  Type, 233
GetInteger
  DexProperties, 80
  Value, 261
GetIsDirected
  Type, 234
GetIsRestricted
  Type, 234
GetKind
  Attribute, 30
GetLabel
  EdgeExport, 84
  GraphExport, 125
  NodeExport, 138
GetLabelColorRGB
  EdgeExport, 84
  NodeExport, 139
GetLicense
  DexConfig, 74
GetLogFile
  DexConfig, 74
GetLogLevel
  DexConfig, 74
GetLog
  Value, 261
GetMax
  AttributeStatistics, 35
GetMaxLengthString
  AttributeStatistics, 35
GetMean
AttributeStatistics, 36
GetMedian
AttributeStatistics, 36
GetMin
AttributeStatistics, 35
GetMinLengthString
AttributeStatistics, 35
GetMode
AttributeStatistics, 36
GetModeCount
AttributeStatistics, 36
GetName
Attribute, 29
Type, 233
GetNode
DefaultExport, 64
ExportManager, 103
GetNodes
ConnectedComponents, 41
GetNodeType
DefaultExport, 63
ExportManager, 102
GetNull
AttributeStatistics, 34
GetNumCPUs
PlatformStatistics, 167
GetNumObjects
Type, 233
GetObjectType
Graph, 113
Type, 233
GetOID
Value, 262
GetPartition
TypeLoaderEvent, 252
GetPath
Database, 59
GetPathAsEdges
SinglePairShortestPath, 184
SinglePairShortestPathBFS, 189
SinglePairShortestPathDijkstra, 195
GetPathAsNodes
SinglePairShortestPath, 184
SinglePairShortestPathBFS, 189
SinglePairShortestPathDijkstra, 195
GetPhase
TypeLoaderEvent, 252
GetPoolFrameSize
DexConfig, 72
GetPoolPersistentMaxSize
DexConfig, 73
GetPoolPersistentMinSize
DexConfig, 73
GetPoolTemporaryMaxSize
DexConfig, 73
GetPoolTemporaryMinSize
DexConfig, 73
GetRead
DatabaseStatistics, 61
GetRealTime
PlatformStatistics, 167
GetRecoveryCacheMaxSize
DexConfig, 76
GetRecoveryCheckpointTime
DexConfig, 77
GetRecoveryEnabled
DexConfig, 76
GetRecoveryLogFile
DexConfig, 76
GetRestrictedFrom
Type, 234
GetRestrictedTo
Type, 234
GetRow
CSVReader, 54
RowReader, 169
GetSessions
DatabaseStatistics, 61
GetShape
NodeExport, 138
GetSize
Attribute, 29
ConnectedComponents, 41
GetStatistics
Database, 59
Platform, 166
GetString
Value, 261
GetSystemTime
PlatformStatistics, 167
GetTail
Edgedata, 82
GetTemp
DatabaseStatistics, 61
GetTimestamp
Value, 261
GetTimeUnit
DexProperties, 80
GetTotal
AttributeStatistics, 34
TypeExporterEvent, 240
GetTotalMem
PlatformStatistics, 167
GetTotalPartitions
TypeLoaderEvent, 252
GetTotalPartitionSteps
TypeLoaderEvent, 253
GetTotalPhases
GetTotalPhases.

TypeLoaderEvent, 252

GetType
GetType

Graph, 117

GetTypeId
GetTypeId

Attribute, 29
Attribute, 29

TypeExporterEvent, 240
TypeExporterEvent, 240

TypeLoaderEvent, 252
TypeLoaderEvent, 252

GetUserTime
GetUserTime

PlatformStatistics, 167
PlatformStatistics, 167

GetValues
GetValues

Graph, 123
Graph, 123

GetVariances
GetVariances

AttributeStatistics, 36
AttributeStatistics, 36

GetWidth
GetWidth

EdgeExport, 85
EdgeExport, 85

NodeExport, 139
NodeExport, 139

GetWrite
GetWrite

DatabaseStatistics, 61
DatabaseStatistics, 61

Graph, 106
Graph, 106

Backup, 124
Backup, 124

CountEdges, 112
CountEdges, 112

CountNodes, 112
CountNodes, 112

Degree, 120
Degree, 120

Drop, 113
Drop, 113

DumpData, 123
DumpData, 123

DumpStorage, 124
DumpStorage, 124

Edges, 121
Edges, 121

Explode, 120
Explode, 120

Export, 124
Export, 124

FindAttribute, 118
FindAttribute, 118

FindAttributes, 123
FindAttributes, 123

FindEdge, 121
FindEdge, 121

FindEdgeTypes, 122
FindEdgeTypes, 122

FindNodeTypes, 122
FindNodeTypes, 122

FindObject, 118
FindObject, 118

FindType, 117
FindType, 117

FindTypes, 123
FindTypes, 123

GetAttribute, 115, 118
GetAttribute, 115, 118

GetAttributeIntervalCount, 117
GetAttributeIntervalCount, 117

GetAttributes, 123
GetAttributes, 123

GetAttributeStatistics, 116
GetAttributeStatistics, 116

GetAttributeText, 116
GetAttributeText, 116

GetEdgeData, 112
GetEdgeData, 112

GetEdgePeer, 113
GetEdgePeer, 113

GetObjectType, 113
GetObjectType, 113

GetType, 117
GetType, 117

GetValues, 123
GetValues, 123

Heads, 122
Heads, 122

IndexAttribute, 115
IndexAttribute, 115

Indexed
Indexed

gdb, 15

gdb, 15

Info
Info

gdb, 18

gdb, 18

Ingoing
Ingoing

gdb, 17

gdb, 17

Int32List, 127
Int32List, 127

Add, 128
Add, 128

Count, 128
Count, 128

Int32List, 128
Int32List, 128

Iterator, 128
Iterator, 128

NewNode, 111
NewNode, 111

NewNodeType, 111
NewNodeType, 111

NewRestrictedEdgeType, 111
NewRestrictedEdgeType, 111

NewSessionAttribute, 114
NewSessionAttribute, 114

RemoveAttribute, 118
RemoveAttribute, 118

RemoveType, 118
RemoveType, 118

Select, 119
Select, 119

SetAttribute, 116
SetAttribute, 116

SetAttributeDefaultValue, 115
SetAttributeDefaultValue, 115

SetAttributeText, 116
SetAttributeText, 116

Tails, 122
Tails, 122

TailsAndHeads, 122
TailsAndHeads, 122

GraphExport, 125
GraphExport, 125

GetLabel, 125
GetLabel, 125

SetLabel, 125
SetLabel, 125

GraphML
gdb, 17

gdb, 17

GreaterEqual
gdb, 16

gdb, 16

GreaterThan
gdb, 16

gdb, 16

Handler, 126
Handler, 126

gdb, 19

gdb, 19

HasNext
HasNext

AttributeListIterator, 32
AttributeListIterator, 32

BooleanListIterator, 39
BooleanListIterator, 39

Int32ListIterator, 129
Int32ListIterator, 129

ObjectsIterator, 162
ObjectsIterator, 162

OIDListIterator, 165
OIDListIterator, 165

StringListIterator, 202
StringListIterator, 202

Traversal
Traversal

BFS, 222
BFS, 222

DFS, 230
DFS, 230

TypeListIterator, 244
TypeListIterator, 244

ValuesIterator, 271
ValuesIterator, 271

Heads
Heads

Graph, 122
Graph, 122

IndexAttribute
IndexAttribute

Graph, 115
Graph, 115

Indexed
Indexed

gdb, 15

gdb, 15

Info
Info

gdb, 18

gdb, 18

Ingoing
Ingoing

gdb, 17

gdb, 17

Int32List, 127
Int32List, 127

Add, 128
Add, 128

Count, 128
Count, 128

Int32List, 128
Int32List, 128

Iterator, 128
Iterator, 128
Intersection, 157
Iterator, 160
IteratorFromElement, 160
IteratorFromIndex, 160
Remove, 157
Sample, 160
Union, 157
ObjectsIterator, 161
HasNext, 162
Next, 162
ObjectType
gdb, 18
Off
gdb, 18
OID
gdb, 17
OIDList, 162
Add, 164
Count, 163
Iterator, 164
OIDList, 163
Set, 164
OIDListIterator, 164
HasNext, 165
Next, 165
ONE_PHASE
EdgeTypeLoader, 93
NodeTypeLoader, 148
TypeLoader, 247
Open
CSVReader, 53
CSVWriter, 57
Dex, 66
operator<<
gdb, 20
script, 23
operator=
Value, 260
Order
gdb, 19
Outgoing
gdb, 17
Parse
ScriptParser, 173
pathAsEdges
SinglePairShortestPath, 186
SinglePairShortestPathBFS, 191
SinglePairShortestPathDijkstra, 197
Platform, 165
GetStatistics, 166
PlatformStatistics, 166
GetAvailableMem, 167
GetNumCPUs, 167
GetRealTime, 167
GetSystemTime, 167
GetTotalMem, 167
GetUserTime, 167
Prepare
ExportManager, 102
Read
CSVReader, 54
RowReader, 169
TextStream, 218
RegExp
gdb, 16
Register
EdgeTypeExporter, 89
EdgeTypeLoader, 95
NodeTypeExporter, 144
NodeTypeLoader, 150
TypeExporter, 238
TypeLoader, 249
Release
ExportManager, 102
Remove
Objects, 157
RemoveAttribute
Graph, 118
RemoveType
Graph, 118
Reset
CSVReader, 54
RowReader, 169
Restore
Dex, 67
Round
gdb, 18
RowReader, 168
Close, 169
GetRow, 169
Read, 169
Reset, 169
RowReader, 169
RowWriter, 170
Close, 171
RowWriter, 171
Write, 171
Run
Connectivity, 46
EdgeTypeLoader, 94
NodeTypeLoader, 148
ShortestPath, 180
SinglePairShortestPath, 185
StrongConnectivity, 207
TypeExporter, 238
TypeLoader, 247
WeakConnectivity, 276

RunNPhases
  TypeLoader, 249

RunProcess
  EdgeTypeExporter, 89
  NodeTypeExporter, 143
  TypeExporter, 237

RunTwoPhases
  TypeLoader, 249

Sample
  Objects, 160

Script
  <, 23
  operator <<=, 23

ScriptParser, 171
  GenerateSchemaScript, 173
  Parse, 173
  SetErrorLog, 173
  SetOutputLog, 172

Select
  Graph, 119
  Session, 174
  GetGraph, 176
  NewObjects, 176

Set
  OIDList, 164
  Value, 265

SetAsDirected
  EdgeExport, 84

SetAttribute
  Graph, 116

SetAttributeDefaultValue
  Graph, 115

SetAttributePositions
  EdgeTypeLoader, 96
  NodeTypeLoader, 151
  TypeLoader, 250

SetAttributes
  EdgeTypeExporter, 90
  EdgeTypeLoader, 96
  NodeTypeExporter, 144
  NodeTypeLoader, 151
  TypeExporter, 239
  TypeLoader, 250

SetAttributeText
  Graph, 116

SetAutoQuotes
  CSVWriter, 56

SetBoolean
  Value, 262

SetBooleanVoid
  Value, 262

SetCacheMaxSize
  DexConfig, 74
  SetCacheStatisticsEnabled
    DexConfig, 75
  SetCacheStatisticsFile
    DexConfig, 75
  SetCacheStatisticsSnapshotTime
    DexConfig, 76
  SetColorRGB
    EdgeExport, 84
    NodeExport, 138
  SetDouble
    Value, 263
  SetDoubleVoid
    Value, 263
  SetErrorLog
    ScriptParser, 173
  SetExtentPages
    DexConfig, 72
  SetExtentSize
    DexConfig, 71
  SetFit
    NodeExport, 140
  SetFontSize
    EdgeExport, 85
    NodeExport, 140
  SetForcedQuotes
    CSVWriter, 57
  SetFrequency
    EdgeTypeExporter, 90
    EdgeTypeLoader, 96
    NodeTypeExporter, 145
    NodeTypeLoader, 151
    TypeExporter, 239
    TypeLoader, 250

SetGraph
  EdgeTypeExporter, 89
  EdgeTypeLoader, 96
  NodeTypeExporter, 144
  NodeTypeLoader, 150
  TypeExporter, 238
  TypeLoader, 250

SetHandler
gdb, 20

SetHeadAttribute
  EdgeTypeExporter, 88
  EdgeTypeLoader, 94
  NodeTypeExporter, 143
  NodeTypeLoader, 149
  TypeExporter, 237
  TypeLoader, 248

SetHeader
  EdgeTypeExporter, 90
  NodeTypeExporter, 145
  TypeExporter, 239
SetHeadPosition
  EdgeTypeExporter, 88
  EdgeTypeLoader, 94
  NodeTypeExporter, 143
  NodeTypeLoader, 149
  TypeExporter, 237
  TypeLoader, 248

SetHeight
  NodeExport, 139

SetHighAvailabilityCoordinators
  DexConfig, 78

SetHighAvailabilityEnabled
  DexConfig, 77

SetHighAvailabilityIP
  DexConfig, 78

SetHighAvailabilityMasterHistory
  DexConfig, 79

SetHighAvailabilitySynchronization
  DexConfig, 78

SetInteger
  Value, 262

SetIntegerVoid
  Value, 262

SetLabel
  EdgeExport, 84
  GraphExport, 125
  NodeExport, 138

SetLabelColorRGB
  EdgeExport, 85
  NodeExport, 139

SetLicense
  DexConfig, 74

SetLocale
  CSVReader, 53
  CSWriter, 57
  EdgeTypeLoader, 96
  NodeTypeLoader, 150
  TypeLoader, 250

SetLogError
  EdgeTypeLoader, 95
  NodeTypeLoader, 150
  TypeLoader, 249

SetLogFile
  DexConfig, 74

SetLogLevel
  DexConfig, 75

SetLogOff
  EdgeTypeLoader, 95
  NodeTypeLoader, 150
  TypeLoader, 249

SetLong
  Value, 263

SetLongVoid
  Value, 262

SetMaterializedAttribute
  Connectivity, 46
  StrongConnectivity, 207
  StrongConnectivityGabow, 213
  WeakConnectivity, 276
  WeakConnectivityDFS, 283

SetMaximumHops
  Context, 50
  ShortestPath, 179
  SinglePairShortestPath, 184
  SinglePairShortestPathBFS, 190
  SinglePairShortestPathDijkstra, 196
  Traversal, 222
  TraversalBFS, 227
  TraversalDFS, 231

SetMessage
  AppError, 28
  Error, 99
  Exception, 100
  FileNotFoundException, 105
  IOException, 133
  LicenseError, 136
  NoSuchElementException, 153
  SQLException, 199
  SystemError, 215
  UnsupportedOperationError, 256
  WrongArgumentError, 285

SetMultilines
  CSVReader, 53

SetNodesNotVisited
  Connectivity, 47
  StrongConnectivity, 208
  StrongConnectivityGabow, 213
  WeakConnectivity, 277
  WeakConnectivityDFS, 283

SetNull
  Value, 260

SetNumLines
  CSVReader, 53

SetOID
  Value, 265

SetOIDVoid
  Value, 265

SetOutputLog
  ScriptParser, 172

SetPoolFrameSize
  DexConfig, 72

SetPoolPersistentMaxSize
  DexConfig, 73

SetPoolPersistentMinSize
  DexConfig, 72

SetPoolTemporaryMaxSize
  DexConfig, 73
SetPoolTemporaryMinSize
DexConfig, 73

SetQuotes
CSVReader, 53
CSVWriter, 56

SetRecoveryCacheMaxSize
DexConfig, 77

SetRecoveryCheckpointTime
DexConfig, 77

SetRecoveryEnabled
DexConfig, 76

SetRecoveryLogFile
DexConfig, 76

SetRowReader
EdgeTypeLoader, 95
NodeTypeLoader, 150
TypeLoader, 250

SetRowWriter
EdgeTypeExporter, 89
NodeTypeExporter, 144
TypeExporter, 238

SetSeparator
CSVReader, 53
CSVWriter, 56

SetShape
NodeExport, 138

SetStartLine
CSVReader, 53

SetString
Value, 264

SetStringVoid
Value, 264

SetTailAttribute
EdgeTypeExporter, 88
EdgeTypeLoader, 94
NodeTypeExporter, 143
NodeTypeLoader, 149
TypeExporter, 237
TypeLoader, 248

SetTailPosition
EdgeTypeExporter, 88
EdgeTypeLoader, 94
NodeTypeExporter, 144
NodeTypeLoader, 149
TypeExporter, 238
TypeLoader, 248

SetTimestamp
Value, 264

SetTimestampVoid
Value, 263

SetType
EdgeTypeExporter, 90
EdgeTypeLoader, 96
NodeTypeExporter, 144
NodeTypeLoader, 151
TypeExporter, 238
TypeLoader, 250

SetUnweightedEdgeCost
SinglePairShortestPathDijkstra, 196

SetVoid
Value, 265

SetWidth
EdgeExport, 85
NodeTypeExport, 139

Severe
gdb, 18

ShortestPath, 177
AddAllEdgeTypes, 179
AddEdgeType, 179
ExcludeEdges, 180
ExcludeNodes, 180
maxHops, 180
Run, 180
SetMaximumHops, 179

ShortestPath, 179
AddAllEdgeTypes, 185
AddEdgeType, 185
ExcludeEdges, 185
ExcludeNodes, 185
GetCost, 184
GetPathAsEdges, 184
GetPathAsNodes, 184
maxHops, 186
pathAsEdges, 186
Run, 185
SetMaximumHops, 184

ShortestPath, 184
SetMaximumHops, 184

SinglePairShortestPath, 184

SinglePairShortestPathBFS, 186
AddAllEdgeTypes, 190
AddEdgeType, 190
CheckOnlyExistence, 190
ExcludeEdges, 191
ExcludeNodes, 191
GetCost, 190
GetPathAsEdges, 189
GetPathAsNodes, 189
maxHops, 191
pathAsEdges, 191
SetMaximumHops, 190

SinglePairShortestPathBFS, 189

SinglePairShortestPathDijkstra, 191
AddAllEdgeTypes, 197
AddEdgeType, 196
AddWeightedEdgeType, 196
ExcludeEdges, 197
ExcludeNodes, 197
GetCost, 196
SetVoid, 265
ToString, 266
Value, 260
Values, 266
Count, 268
Iterator, 268
ValuesIterator, 268
HasNext, 271
Next, 271

Warning
    gdb, 18
WeakConnectivity, 271
    AddAllEdgeTypes, 275, 277
    AddEdgeType, 275, 277
    ExcludeEdges, 276
    ExcludeNodes, 275
    GetConnectedComponents, 276
    Run, 276
    SetMaterializedAttribute, 276
    SetNodesNotVisited, 277
    WeakConnectivity, 275
WeakConnectivityDFS, 277
    AddAllEdgeTypes, 282
    AddEdgeType, 281
    ExcludeEdges, 282
    ExcludeNodes, 282
    GetConnectedComponents, 282
    SetMaterializedAttribute, 283
    SetNodesNotVisited, 283
    WeakConnectivityDFS, 281
Write
    CSVWriter, 57
    RowWriter, 171
    TextStream, 218
WrongArgumentError, 283
    Message, 285
    NewError, 285
    SetMessage, 285
    WrongArgumentError, 285

YGraphML
    gdb, 18