TEST JOIN

```
1  package query;
2
3  import database.Database;
4  import org.junit.Rule;
5  import org.junit.Test;
6
7  import java.io.File;
8  import java.io.IOException;
9  import java.util.*;
10
11 import database.DatabaseException;
12 import testutil.TestUtils;
13 import databox.BoolDataBox;
14 import databox.DataBox;
15 import databox.FloatDataBox;
16 import databox.IntDataBox;
17 import databox.StringDataBox;
18 import databox.Type;
19 import table.Record;
20 import table.Schema;
21
22 import org.junit.rules.TemporaryFolder;
23
24 import static org.junit.Assert.*;
25
26 public class TestJoinOperator {
27
28   @Rule
29   public TemporaryFolder tempFolder = new TemporaryFolder();
30
31   @Test(timeout=5000)
32   public void testOperatorSchema() throws QueryPlanException, DatabaseException,
     IOException {
33     TestSourceOperator sourceOperator = new TestSourceOperator();
34     File tempDir = tempFolder.newFolder("joinTest");
35     Database.Transaction transaction = new
       Database(tempDir.getAbsolutePath()).beginTransaction();
36     JoinOperator joinOperator = new SNLJOperator(sourceOperator, sourceOperator,
       "int", "int", transaction);
37
38     List<String> expectedSchemaNames = new ArrayList<String>();
39     expectedSchemaNames.add("bool");
40     expectedSchemaNames.add("int");
41     expectedSchemaNames.add("string");
42     expectedSchemaNames.add("float");
43     expectedSchemaNames.add("bool");
44     expectedSchemaNames.add("int");
45     expectedSchemaNames.add("string");
46     expectedSchemaNames.add("float");
47
48     List<Type> expectedSchemaTypes = new ArrayList<Type>();
49     expectedSchemaTypes.add(Type.boolType());
50     expectedSchemaTypes.add(Type.intType());
51     expectedSchemaTypes.add(Type.stringType(5));
52     expectedSchemaTypes.add(Type.floatType());
53     expectedSchemaTypes.add(Type.boolType());
54     expectedSchemaTypes.add(Type.intType());
55     expectedSchemaTypes.add(Type.stringType(5));
56     expectedSchemaTypes.add(Type.floatType());
57
58     Schema expectedSchema = new Schema(expectedSchemaNames, expectedSchemaTypes);
59
60     assertEquals(expectedSchema, joinOperator.getOutputSchema());
61   }
62
63   @Test(timeout=5000)
64   public void testSimpleJoin() throws QueryPlanException, DatabaseException,
```

```
    IOException {
65    TestSourceOperator sourceOperator = new TestSourceOperator();
66    File tempDir = tempFolder.newFolder("joinTest");
67    Database.Transaction transaction = new
      Database(tempDir.getAbsolutePath()).beginTransaction();
68    JoinOperator joinOperator = new SNLJOperator(sourceOperator, sourceOperator,
      "int", "int", transaction);
69
70    Iterator<Record> outputIterator = joinOperator.iterator();
71    int numRecords = 0;
72
73    List<DataBox> expectedRecordValues = new ArrayList<DataBox>();
74    expectedRecordValues.add(new BoolDataBox(true));
75    expectedRecordValues.add(new IntDataBox(1));
76    expectedRecordValues.add(new StringDataBox("abcde", 5));
77    expectedRecordValues.add(new FloatDataBox(1.2f));
78    expectedRecordValues.add(new BoolDataBox(true));
79    expectedRecordValues.add(new IntDataBox(1));
80    expectedRecordValues.add(new StringDataBox("abcde", 5));
81    expectedRecordValues.add(new FloatDataBox(1.2f));
82    Record expectedRecord = new Record(expectedRecordValues);
83
84
85    while (outputIterator.hasNext()) {
86      assertEquals(expectedRecord, outputIterator.next());
87      numRecords++;
88    }
89
90    assertEquals(100*100, numRecords);
91  }
92
93  @Test(timeout=5000)
94  public void testEmptyJoin() throws QueryPlanException, DatabaseException,
    IOException {
95    TestSourceOperator leftSourceOperator = new TestSourceOperator();
96
97    List<Integer> values = new ArrayList<Integer>();
98    TestSourceOperator rightSourceOperator =
      TestUtils.createTestSourceOperatorWithInts(values);
99    File tempDir = tempFolder.newFolder("joinTest");
100   Database.Transaction transaction = new
      Database(tempDir.getAbsolutePath()).beginTransaction();
101   JoinOperator joinOperator = new SNLJOperator(leftSourceOperator,
      rightSourceOperator, "int", "int", transaction);
102   Iterator<Record> outputIterator = joinOperator.iterator();
103
104   assertFalse(outputIterator.hasNext());
105 }
106
107 }
108
```