TestCSVFile

```
1  package database;
2
3  import databox.*;
4  import index.BPlusTree;
5  import index.BPlusTreeException;
6  import query.*;
7  import table.*;
8
9  import org.junit.After;
10 import org.junit.Before;
11 import org.junit.Test;
12 import org.junit.Rule;
13
14 import org.junit.rules.TemporaryFolder;
15
16 import java.io.File;
17 import java.io.IOException;
18 import java.nio.charset.Charset;
19 import java.nio.file.Files;
20 import java.nio.file.Paths;
21 import java.util.*;
22
23 import static org.junit.Assert.assertEquals;
24
25 public class TestCSVFile {
26     public static final String TestDir = "testDatabase";
27     private Database db;
28     private String filename;
29     private File file;
30     private String btree_filename = "TestBPlusTree";
31
32     @Rule
33     public TemporaryFolder tempFolder = new TemporaryFolder();
34
35     @Before
36     public void beforeEach() throws Exception {
37         File testDir = tempFolder.newFolder(TestDir);
38         this.filename = testDir.getAbsolutePath();
39         this.db = new Database(filename);
40         this.db.deleteAllTables();
41         this.file = tempFolder.newFile(btree_filename);
42     }
43
44     @After
45     public void afterEach() {
46         this.db.deleteAllTables();
47         this.db.close();
48     }
49
50     private BPlusTree getBPlusTree(Type keySchema, int order) throws
       BPlusTreeException {
51         return new BPlusTree(file.getAbsolutePath(), keySchema, order);
52     }
53
54     @Test
55     public void testCSVFileDB() throws DatabaseException, IOException {
56         List<String> names = Arrays.asList("sid", "name", "major", "gpa");
57         List<Type> types = Arrays.asList(Type.intType(), Type.stringType(20),
58                 Type.stringType(20), Type.floatType());
59         Schema s = new Schema(names, types);
60
61         // create table student
62         String tableName = "student";
63         db.createTable(s, tableName);
64
65         List<String> studentLines = Files.readAllLines(Paths.get("students.csv"),
           Charset.defaultCharset());
```

```
66
67          Database.Transaction t1 = db.beginTransaction();
68
69          // add recode for student
70          for (String line : studentLines) {
71              String[] splits = line.split(",");
72              List<DataBox> values = new ArrayList<>();
73
74              values.add(new IntDataBox(Integer.parseInt(splits[0])));
75              values.add(new StringDataBox(splits[1].trim(), 20));
76              values.add(new StringDataBox(splits[2].trim(), 20));
77              values.add(new FloatDataBox(Float.parseFloat(splits[3])));
78
79              RecordId rid = t1.addRecord(tableName, values);
80              Record rec = t1.getRecord(tableName, rid);
81              assertEquals(new Record(values), rec);
82          }
83          t1.end();
84      }
85
86
87      @Test
88      public void testCSVFileBtree() throws DatabaseException, BPlusTreeException,
        IOException{
89          List<String> names = Arrays.asList("sid", "name", "major", "gpa");
90          List<Type> types = Arrays.asList(Type.intType(), Type.stringType(20),
91                  Type.stringType(20), Type.floatType());
92          Schema s = new Schema(names, types);
93
94          BPlusTree tree = getBPlusTree(Type.intType(), 2);
95
96          // create table student
97          String tableName = "student";
98          db.createTable(s, tableName);
99
100         List<String> studentLines = Files.readAllLines(Paths.get("students.csv"),
            Charset.defaultCharset());
101
102         Database.Transaction t1 = db.beginTransaction();
103
104         // add recode for student
105         for (String line : studentLines) {
106             String[] splits = line.split(",");
107             ArrayList<DataBox> values = new ArrayList<>();
108
109             values.add(new IntDataBox(Integer.parseInt(splits[0])));
110             values.add(new StringDataBox(splits[1].trim(), 20));
111             values.add(new StringDataBox(splits[2].trim(), 20));
112             values.add(new FloatDataBox(Float.parseFloat(splits[3])));
113
114             RecordId rid = t1.addRecord(tableName, values);
115             tree.put(values.get(0), rid);
116             Record rec = t1.getRecord(tableName, rid);
117             assertEquals(new Record(values), rec);
118         }
119
120         Optional<RecordId> opt_rid = tree.get(new IntDataBox(10));
121         if (opt_rid.isPresent()){
122             RecordId rid = opt_rid.get();
123             System.out.println(rid);
124             Record rec = t1.getRecord(tableName, rid);
125             System.out.println(rec);
126         }
127
128         t1.end();
129     }
130
```

```
    @Test
    public void testINLJStudentEnrollment() throws DatabaseException,
    BPlusTreeException, IOException, QueryPlanException {
134
135
136         // create second table
137         String table1Name = "student";
138         String table2Name = "enrollment";
139
140         Database.Transaction t1 = db.beginTransaction();
141
142         BPlusTree rightBtree = loadStudent(t1);
143         loadEnrollment(t1);
144
145         SequentialScanOperator leftSCO = new SequentialScanOperator(t1, table2Name);
146         BtreeIndexScanOperator rightBTO = new BtreeIndexScanOperator(t1,
            table1Name, rightBtree);
147         INLJOperator inljOperator = new INLJOperator(leftSCO, rightBTO, "sid",
            "sid", t1);
148
149         Iterator<Record> recordIterator =  inljOperator.iterator();
150
151         while (recordIterator.hasNext()){
152             Record record = recordIterator.next();
153             System.out.println(record);
154         }
155
156
157     }
158
159     @Test
160     public void testINLJStudentEnrollmentCourses() throws DatabaseException,
        BPlusTreeException, IOException, QueryPlanException {
161
162
163         // create second table
164         String table1Name = "student";
165         String table2Name = "enrollment";
166         String table3Nmae = "course";
167
168         Database.Transaction t1 = db.beginTransaction();
169
170         BPlusTree studentBtree = loadStudent(t1);
171         loadEnrollment(t1);
172         BPlusTree courseBtree = loadCourse(t1);
173
174         SequentialScanOperator leftSCO = new SequentialScanOperator(t1, table2Name);
175         BtreeIndexScanOperator rightBTO = new BtreeIndexScanOperator(t1,
            table1Name, studentBtree);
176         INLJOperator inljOperator = new INLJOperator(leftSCO, rightBTO, "sid",
            "sid", t1);
177
178         Iterator<Record> recordIterator =  inljOperator.iterator();
179
180         List<Record> student_enrollment = new ArrayList<>();
181         while (recordIterator.hasNext()){
182             Record record = recordIterator.next();
183             student_enrollment.add(record);
184         }
185
186
187         // schema
188         List<String> names = Arrays.asList("cid", "cname", "dept");
189         List<Type> types = Arrays.asList(Type.intType(), Type.stringType(20),
            Type.stringType(20));
190         Schema s = new Schema(names, types);
```

```
191
192             TestSourceOperator sourceOperator = new
                TestSourceOperator(student_enrollment, s, student_enrollment.size());
193
194             BtreeIndexScanOperator courseBTO = new BtreeIndexScanOperator(t1,
                table3Nmae, courseBtree);
195
196             INLJOperator inljOperator2 = new INLJOperator(sourceOperator, courseBTO,
                "cid", "cid", t1);
197
198             Iterator<Record> recordIterator1 =  inljOperator2.iterator();
199
200             while (recordIterator1.hasNext()){
201                 Record record = recordIterator1.next();
202                 System.out.println(record);
203             }
204
205         }
206
207
208
209         private BPlusTree loadStudent(Database.Transaction t1) throws
            DatabaseException, BPlusTreeException, IOException{
210             List<String> names = Arrays.asList("sid", "cid", "major", "gpa");
211             List<Type> types = Arrays.asList(Type.intType(), Type.stringType(20),
212                     Type.stringType(20), Type.floatType());
213             Schema s = new Schema(names, types);
214
215             BPlusTree tree = getBPlusTree(Type.intType(), 2);
216
217             // create table student
218             String tableName = "student";
219             db.createTable(s, tableName);
220
221             List<String> studentLines = Files.readAllLines(Paths.get("students.csv"),
                Charset.defaultCharset());
222
223
224             // add recode for student
225             for (String line : studentLines) {
226                 String[] splits = line.split(",");
227                 ArrayList<DataBox> values = new ArrayList<>();
228
229                 values.add(new IntDataBox(Integer.parseInt(splits[0])));
230                 values.add(new StringDataBox(splits[1].trim(), 20));
231                 values.add(new StringDataBox(splits[2].trim(), 20));
232                 values.add(new FloatDataBox(Float.parseFloat(splits[3])));
233
234                 RecordId rid = t1.addRecord(tableName, values);
235                 tree.put(values.get(0), rid);
236             }
237             return tree;
238         }
239
240         private void loadEnrollment(Database.Transaction t1) throws  DatabaseException,
            BPlusTreeException, IOException{
241             List<String> names = Arrays.asList("sid", "cid");
242             List<Type> types = Arrays.asList(Type.intType(), Type.intType());
243             Schema s = new Schema(names, types);
244
245             // create table student
246             String tableName = "enrollment";
247             db.createTable(s, tableName);
248
249             List<String> studentLines =
                Files.readAllLines(Paths.get("enrollments.csv"), Charset.defaultCharset());
250
```

```
        // add recode for student
        for (String line : studentLines) {
            String[] splits = line.split(",");
            ArrayList<DataBox> values = new ArrayList<>();

            values.add(new IntDataBox(Integer.parseInt(splits[0])));
            values.add(new IntDataBox(Integer.parseInt(splits[1])));

            t1.addRecord(tableName, values);
        }
    }

    private BPlusTree loadCourse(Database.Transaction t1) throws
    DatabaseException, BPlusTreeException, IOException{
        List<String> names = Arrays.asList("cid", "cname", "dept");
        List<Type> types = Arrays.asList(Type.intType(), Type.stringType(20),
        Type.stringType(20));
        Schema s = new Schema(names, types);

        // create table student
        String tableName = "course";
        db.createTable(s, tableName);

        BPlusTree tree = getBPlusTree(Type.intType(), 2);


        List<String> courseLines = Files.readAllLines(Paths.get("courses.csv"),
        Charset.defaultCharset());

        // add recode for student
        for (String line : courseLines) {
            String[] splits = line.split(",");
            ArrayList<DataBox> values = new ArrayList<>();

            values.add(new IntDataBox(Integer.parseInt(splits[0])));
            values.add(new StringDataBox(splits[1].trim(), 20));
            values.add(new StringDataBox(splits[2].trim(), 20));

            RecordId rid = t1.addRecord(tableName, values);
            tree.put(values.get(0), rid);
        }

        return tree;
    }
}
```