

Join GitHub today

GitHub is home to over 31 million developers working together to host and review code, manage projects, and build software together.

[Sign up](#)[Dismiss](#)

Branch: master ▾ DB2 / src / test / java / index / TestBPlusTree.java

[Find file](#) [Copy path](#) dreamlegends init add all files

2c520b8 26 days ago

1 contributor

570 lines (504 sloc) 18.5 KB

[Raw](#) [Blame](#) [History](#)   

```
1 package index;
2 import static org.junit.Assert.assertEquals;
3
4 import java.io.File;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.Collections;
8 import java.util.Iterator;
9 import java.util.List;
10 import java.util.Optional;
11 import java.util.Random;
12
13 import org.junit.Before;
14 import org.junit.Rule;
15 import org.junit.Test;
16 import org.junit.rules.DisableOnDebug;
17 import org.junit.rules.TemporaryFolder;
18 import org.junit.rules.TestRule;
19 import org.junit.rules.Timeout;
20
21 import databox.DataBox;
22 import databox.IntDataBox;
23 import databox.Type;
24 import table.RecordId;
25
26 public class TestBPlusTree {
27     public static final String filename = "TestBPlusTree";
28     private File file;
29
30     @Rule
31     public TemporaryFolder tempFolder = new TemporaryFolder();
32
33     // 10 seconds max per method tested.
34     @Rule
35     public TestRule globalTimeout = new DisableOnDebug(Timeout.seconds(10));
36
37     // Helpers //////////////////////////////////////
38     @Before
39     public void initFile() throws IOException {
40         this.file = tempFolder.newFile(filename);
41     }
42
43     private BPlusTree getBPlusTree(Type keySchema, int order)
44         throws BPlusTreeException, IOException {
45         return new BPlusTree(file.getAbsolutePath(), keySchema, order);
46     }
47 }
```

```

48     }
49
50     private static <T> List<T> iteratorToList(Iterator<T> iter) {
51         List<T> xs = new ArrayList<>();
52         while (iter.hasNext()) {
53             xs.add(iter.next());
54         }
55         return xs;
56     }
57
58     // Tests //////////////////////////////////////
59     // HIDDEN
60     @Test
61     public void testEmptyTree() throws BPlusTreeException, IOException {
62         BPlusTree tree = getBPlusTree(Type.intType(), 2);
63         List<RecordId> empty = new ArrayList<>();
64
65         // Make sure that operations on an empty B+ tree doesn't throw any
66         // exceptions.
67         for (int i = 0; i < 10; ++i) {
68             tree.remove(new IntDataBox(i));
69             assertEquals(Optional.empty(), tree.get(new IntDataBox(i)));
70             Iterator<RecordId> eq = tree.scanEqual(new IntDataBox(i));
71             Iterator<RecordId> all = tree.scanAll();
72             Iterator<RecordId> ge = tree.scanGreaterEqual(new IntDataBox(i));
73             assertEquals(empty, iteratorToList(eq));
74             assertEquals(empty, iteratorToList(all));
75             assertEquals(empty, iteratorToList(ge));
76         }
77     }
78
79     // HIDDEN
80     @Test
81     public void testBPlusTreeFromDisk() throws BPlusTreeException, IOException {
82         BPlusTree tree = getBPlusTree(Type.intType(), 2);
83         for (int i = 0; i < 100; ++i) {
84             tree.put(new IntDataBox(i), new RecordId(i, (short) i));
85         }
86
87         BPlusTree fromDisk = new BPlusTree(file.getAbsolutePath());
88         for (int i = 0; i < 100; ++i) {
89             IntDataBox key = new IntDataBox(i);
90             RecordId rid = new RecordId(i, (short) i);
91             assertEquals(Optional.of(rid), fromDisk.get(key));
92         }
93     }
94
95     // HIDDEN
96     @Test
97     public void testSimpleGets() throws BPlusTreeException, IOException {
98         BPlusTree tree = getBPlusTree(Type.intType(), 2);
99         for (int i = 0; i < 100; ++i) {
100             tree.put(new IntDataBox(i), new RecordId(i, (short) i));
101         }
102
103         for (int i = 0; i < 100; ++i) {
104             IntDataBox key = new IntDataBox(i);
105             RecordId rid = new RecordId(i, (short) i);
106             assertEquals(Optional.of(rid), tree.get(key));
107         }
108
109         for (int i = 100; i < 150; ++i) {
110             assertEquals(Optional.empty(), tree.get(new IntDataBox(i)));
111         }
112     }
113
114     // HIDDEN
115     @Test
116     public void testEmptyScans() throws BPlusTreeException, IOException {

```

```

115 // Create and then empty the tree.
116 BPlusTree tree = getBPlusTree(Type.intType(), 2);
117 for (int i = 0; i < 100; ++i) {
118     tree.put(new IntDataBox(i), new RecordId(i, (short) i));
119 }
120 for (int i = 0; i < 100; ++i) {
121     tree.remove(new IntDataBox(i));
122 }
123
124 // Scan over the tree.
125 Iterator<RecordId> actual = tree.scanAll();
126 assertEquals(new ArrayList<RecordId>(), iteratorToList(actual));
127 actual = tree.scanGreaterEqual(new IntDataBox(42));
128 assertEquals(new ArrayList<RecordId>(), iteratorToList(actual));
129 actual = tree.scanGreaterEqual(new IntDataBox(100));
130 assertEquals(new ArrayList<RecordId>(), iteratorToList(actual));
131 }
132
133 // HIDDEN
134 @Test
135 public void testPartiallyEmptyScans()
136     throws BPlusTreeException, IOException {
137     // Create and then empty part of the tree.
138     BPlusTree tree = getBPlusTree(Type.intType(), 2);
139     for (int i = 0; i < 100; ++i) {
140         tree.put(new IntDataBox(i), new RecordId(i, (short) i));
141     }
142     for (int i = 25; i < 75; ++i) {
143         tree.remove(new IntDataBox(i));
144     }
145
146     // Scan over the tree.
147     Iterator<RecordId> actual = tree.scanAll();
148     List<RecordId> expected = new ArrayList<>();
149     for (int i = 0; i < 25; ++i) {
150         expected.add(new RecordId(i, (short) i));
151     }
152     for (int i = 75; i < 100; ++i) {
153         expected.add(new RecordId(i, (short) i));
154     }
155     assertEquals(expected, iteratorToList(actual));
156
157     actual = tree.scanGreaterEqual(new IntDataBox(42));
158     expected = new ArrayList<>();
159     for (int i = 75; i < 100; ++i) {
160         expected.add(new RecordId(i, (short) i));
161     }
162     assertEquals(expected, iteratorToList(actual));
163
164     actual = tree.scanGreaterEqual(new IntDataBox(99));
165     expected = new ArrayList<>();
166     expected.add(new RecordId(99, (short) 99));
167     assertEquals(expected, iteratorToList(actual));
168 }
169
170 @Test(expected = BPlusTreeException.class)
171 public void testDuplicatePut() throws BPlusTreeException, IOException {
172     BPlusTree tree = getBPlusTree(Type.intType(), 2);
173     tree.put(new IntDataBox(0), new RecordId(0, (short) 0));
174     tree.put(new IntDataBox(0), new RecordId(0, (short) 0));
175 }
176
177 // HIDDEN
178 @Test
179 public void testRandomRids() throws BPlusTreeException, IOException {
180     int d = 3;
181     BPlusTree tree = getBPlusTree(Type.intType(), d);
182
183     List<DataBox> keys = new ArrayList<DataBox>();

```

```

184 List<RecordId> rids = new ArrayList<RecordId>();
185 for (int i = 0; i < 50 * d; ++i) {
186     keys.add(new IntDataBox(i));
187     rids.add(new RecordId(i, (short) i));
188 }
189 Collections.shuffle(rids, new Random(42));
190
191 for (int i = 0; i < keys.size(); ++i) {
192     tree.put(keys.get(i), rids.get(i));
193     assertEquals(Optional.of(rids.get(i)), tree.get(keys.get(i)));
194 }
195
196 for (int i = 0; i < keys.size(); ++i) {
197     assertEquals(Optional.of(rids.get(i)), tree.get(keys.get(i)));
198 }
199 }
200
201 @Test
202 public void testWhiteBoxTest() throws BPlusTreeException, IOException {
203     BPlusTree tree = getBPlusTree(Type.intType(), 1);
204     assertEquals("()", tree.toSexp());
205
206     // (4)
207     tree.put(new IntDataBox(4), new RecordId(4, (short) 4));
208     assertEquals("((4 (4 4)))", tree.toSexp());
209
210     // (4 9)
211     tree.put(new IntDataBox(9), new RecordId(9, (short) 9));
212     assertEquals("((4 (4 4)) (9 (9 9)))", tree.toSexp());
213
214     // (6)
215     // / \
216     // (4) (6 9)
217     tree.put(new IntDataBox(6), new RecordId(6, (short) 6));
218     String l = "((4 (4 4)))";
219     String r = "((6 (6 6)) (9 (9 9)))";
220     assertEquals(String.format("%s 6 %s", l, r), tree.toSexp());
221
222     // (6)
223     // / \
224     // (2 4) (6 9)
225     tree.put(new IntDataBox(2), new RecordId(2, (short) 2));
226     l = "((2 (2 2)) (4 (4 4)))";
227     r = "((6 (6 6)) (9 (9 9)))";
228     assertEquals(String.format("%s 6 %s", l, r), tree.toSexp());
229
230     // (6 7)
231     // / | \
232     // (2 4) (6) (7 9)
233     tree.put(new IntDataBox(7), new RecordId(7, (short) 7));
234     l = "((2 (2 2)) (4 (4 4)))";
235     String m = "((6 (6 6)))";
236     r = "((7 (7 7)) (9 (9 9)))";
237     assertEquals(String.format("%s 6 %s 7 %s", l, m, r), tree.toSexp());
238
239     // (7)
240     // / \
241     // (6) (8)
242     // / \ / \
243     // (2 4) (6) (7) (8 9)
244     tree.put(new IntDataBox(8), new RecordId(8, (short) 8));
245     String ll = "((2 (2 2)) (4 (4 4)))";
246     String lr = "((6 (6 6)))";
247     String rl = "((7 (7 7)))";
248     String rr = "((8 (8 8)) (9 (9 9)))";
249     l = String.format("%s 6 %s", ll, lr);
250     r = String.format("%s 8 %s", rl, rr);
251     assertEquals(String.format("%s 7 %s", l, r), tree.toSexp());
252

```

```

2 //          (7)
3 //          / \
4 //      (3 6)      (8)
5 //      / | \ / \
6 // (2) (3 4) (6) (7) (8 9)
7 tree.put(new IntDataBox(3), new RecordId(3, (short) 3));
8 ll = "((2 (2 2)))";
9 String lm = "((3 (3 3)) (4 (4 4)))";
10 lr = "((6 (6 6)))";
11 rl = "((7 (7 7)))";
12 rr = "((8 (8 8)) (9 (9 9)))";
13 l = String.format("(%s 3 %s 6 %s)", ll, lm, lr);
14 r = String.format("(%s 8 %s)", rl, rr);
15 assertEquals(String.format("(%s 7 %s)", l, r), tree.toSexp());
16
17 //          (4 7)
18 //          / | \
19 //      (3)      (6)      (8)
20 //      / \ / \ / \
21 // (2) (3) (4 5) (6) (7) (8 9)
22 tree.put(new IntDataBox(5), new RecordId(5, (short) 5));
23 ll = "((2 (2 2)))";
24 lr = "((3 (3 3)))";
25 String ml = "((4 (4 4)) (5 (5 5)))";
26 String mr = "((6 (6 6)))";
27 rl = "((7 (7 7)))";
28 rr = "((8 (8 8)) (9 (9 9)))";
29 l = String.format("(%s 3 %s)", ll, lr);
30 m = String.format("(%s 6 %s)", ml, mr);
31 r = String.format("(%s 8 %s)", rl, rr);
32 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
33
34 //          (4 7)
35 //          / | \
36 //      (3)      (6)      (8)
37 //      / \ / \ / \
38 // (1 2) (3) (4 5) (6) (7) (8 9)
39 tree.put(new IntDataBox(1), new RecordId(1, (short) 1));
40 ll = "((1 (1 1)) (2 (2 2)))";
41 lr = "((3 (3 3)))";
42 ml = "((4 (4 4)) (5 (5 5)))";
43 mr = "((6 (6 6)))";
44 rl = "((7 (7 7)))";
45 rr = "((8 (8 8)) (9 (9 9)))";
46 l = String.format("(%s 3 %s)", ll, lr);
47 m = String.format("(%s 6 %s)", ml, mr);
48 r = String.format("(%s 8 %s)", rl, rr);
49 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
50
51 //          (4 7)
52 //          / | \
53 //      (3)      (6)      (8)
54 //      / \ / \ / \
55 // ( 2) (3) (4 5) (6) (7) (8 9)
56 tree.remove(new IntDataBox(1));
57 ll = "((2 (2 2)))";
58 lr = "((3 (3 3)))";
59 ml = "((4 (4 4)) (5 (5 5)))";
60 mr = "((6 (6 6)))";
61 rl = "((7 (7 7)))";
62 rr = "((8 (8 8)) (9 (9 9)))";
63 l = String.format("(%s 3 %s)", ll, lr);
64 m = String.format("(%s 6 %s)", ml, mr);
65 r = String.format("(%s 8 %s)", rl, rr);
66 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
67
68 //          (4 7)
69 //          / | \
70 //      (3)      (6)      (8)

```

```

322 // / \ / \ / \
323 // ( 2) (3) (4 5) (6) (7) (8 )
324 tree.remove(new IntDataBox(9));
325 ll = "((2 (2 2)))";
326 lr = "((3 (3 3)))";
327 ml = "((4 (4 4)) (5 (5 5)))";
328 mr = "((6 (6 6)))";
329 rl = "((7 (7 7)))";
330 rr = "((8 (8 8)))";
331 l = String.format("(%s 3 %s)", ll, lr);
332 m = String.format("(%s 6 %s)", ml, mr);
333 r = String.format("(%s 8 %s)", rl, rr);
334 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
335
336 //          (4 7)
337 //          / | \
338 //   (3)      (6)      (8)
339 // / \ / \ / \ / \
340 // ( 2) (3) (4 5) ( ) (7) (8 )
341 tree.remove(new IntDataBox(6));
342 ll = "((2 (2 2)))";
343 lr = "((3 (3 3)))";
344 ml = "((4 (4 4)) (5 (5 5)))";
345 mr = "()";
346 rl = "((7 (7 7)))";
347 rr = "((8 (8 8)))";
348 l = String.format("(%s 3 %s)", ll, lr);
349 m = String.format("(%s 6 %s)", ml, mr);
350 r = String.format("(%s 8 %s)", rl, rr);
351 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
352
353 //          (4 7)
354 //          / | \
355 //   (3)      (6)      (8)
356 // / \ / \ / \ / \
357 // ( 2) (3) ( 5) ( ) (7) (8 )
358 tree.remove(new IntDataBox(4));
359 ll = "((2 (2 2)))";
360 lr = "((3 (3 3)))";
361 ml = "((5 (5 5)))";
362 mr = "()";
363 rl = "((7 (7 7)))";
364 rr = "((8 (8 8)))";
365 l = String.format("(%s 3 %s)", ll, lr);
366 m = String.format("(%s 6 %s)", ml, mr);
367 r = String.format("(%s 8 %s)", rl, rr);
368 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
369
370 //          (4 7)
371 //          / | \
372 //   (3)      (6)      (8)
373 // / \ / \ / \ / \
374 // ( ) (3) ( 5) ( ) (7) (8 )
375 tree.remove(new IntDataBox(2));
376 ll = "()";
377 lr = "((3 (3 3)))";
378 ml = "((5 (5 5)))";
379 mr = "()";
380 rl = "((7 (7 7)))";
381 rr = "((8 (8 8)))";
382 l = String.format("(%s 3 %s)", ll, lr);
383 m = String.format("(%s 6 %s)", ml, mr);
384 r = String.format("(%s 8 %s)", rl, rr);
385 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
386
387 //          (4 7)
388 //          / | \
389 //   (3)      (6)      (8)
390 // / \ / \ / \ / \

```

```

391 // ( ) (3) ( ) ( ) (7) (8 )
392 tree.remove(new IntDataBox(5));
393 ll = "()";
394 lr = "((3 (3 3)))";
395 ml = "()";
396 mr = "()";
397 rl = "((7 (7 7)))";
398 rr = "((8 (8 8)))";
399 l = String.format("(%s 3 %s)", ll, lr);
400 m = String.format("(%s 6 %s)", ml, mr);
401 r = String.format("(%s 8 %s)", rl, rr);
402 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
403
404 //          (4 7)
405 //        / | \
406 //   (3)   (6)   (8)
407 // / \   / \   / \
408 // ( ) (3) ( ) ( ) ( ) (8 )
409 tree.remove(new IntDataBox(7));
410 ll = "()";
411 lr = "((3 (3 3)))";
412 ml = "()";
413 mr = "()";
414 rl = "()";
415 rr = "((8 (8 8)))";
416 l = String.format("(%s 3 %s)", ll, lr);
417 m = String.format("(%s 6 %s)", ml, mr);
418 r = String.format("(%s 8 %s)", rl, rr);
419 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
420
421 //          (4 7)
422 //        / | \
423 //   (3)   (6)   (8)
424 // / \   / \   / \
425 // ( ) ( ) ( ) ( ) ( ) (8 )
426 tree.remove(new IntDataBox(3));
427 ll = "()";
428 lr = "()";
429 ml = "()";
430 mr = "()";
431 rl = "()";
432 rr = "((8 (8 8)))";
433 l = String.format("(%s 3 %s)", ll, lr);
434 m = String.format("(%s 6 %s)", ml, mr);
435 r = String.format("(%s 8 %s)", rl, rr);
436 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
437
438 //          (4 7)
439 //        / | \
440 //   (3)   (6)   (8)
441 // / \   / \   / \
442 // ( ) ( ) ( ) ( ) ( ) ( )
443 tree.remove(new IntDataBox(8));
444 ll = "()";
445 lr = "()";
446 ml = "()";
447 mr = "()";
448 rl = "()";
449 rr = "()";
450 l = String.format("(%s 3 %s)", ll, lr);
451 m = String.format("(%s 6 %s)", ml, mr);
452 r = String.format("(%s 8 %s)", rl, rr);
453 assertEquals(String.format("(%s 4 %s 7 %s)", l, m, r), tree.toSexp());
454 }
455
456 @Test
457 public void testRandomPuts() throws BPlusTreeException, IOException {
458     List<DataBox> keys = new ArrayList<>();
459     List<RecordId> rids = new ArrayList<>();

```

```

360 List<RecordId> sortedRids = new ArrayList<>();
361 for (int i = 0; i < 1000; ++i) {
362     keys.add(new IntDataBox(i));
363     rids.add(new RecordId(i, (short) i));
364     sortedRids.add(new RecordId(i, (short) i));
365 }
366
367 // Try trees with different orders.
368 for (int d = 2; d < 5; ++d) {
369     // Try trees with different insertion orders.
370     for (int n = 0; n < 2; ++n) {
371         Collections.shuffle(keys, new Random(42));
372         Collections.shuffle(rids, new Random(42));
373
374         // Insert all the keys.
375         BPlusTree tree = getBPlusTree(Type.intType(), d);
376         for (int i = 0; i < keys.size(); ++i) {
377             tree.put(keys.get(i), rids.get(i));
378         }
379
380         // Test get.
381         for (int i = 0; i < keys.size(); ++i) {
382             assertEquals(Optional.of(rids.get(i)), tree.get(keys.get(i)));
383         }
384
385         // Test scanAll.
386         assertEquals(sortedRids, iteratorToList(tree.scanAll()));
387
388         // Test scanGreaterEqual.
389         for (int i = 0; i < keys.size(); i += 100) {
390             Iterator<RecordId> actual = tree.scanGreaterEqual(new IntDataBox(i));
391             List<RecordId> expected = sortedRids.subList(i, sortedRids.size());
392             assertEquals(expected, iteratorToList(actual));
393         }
394
395         // Load the tree from disk.
396         BPlusTree fromDisk = new BPlusTree(file.getAbsolutePath());
397         assertEquals(sortedRids, iteratorToList(fromDisk.scanAll()));
398
399         // Test remove.
400         Collections.shuffle(keys, new Random(42));
401         Collections.shuffle(rids, new Random(42));
402         for (DataBox key : keys) {
403             fromDisk.remove(key);
404             assertEquals(Optional.empty(), fromDisk.get(key));
405         }
406     }
407 }
408
409 // HIDDEN
410 @Test
411 public void testRepeatedInsertsAndRemoves()
412     throws BPlusTreeException, IOException {
413     BPlusTree tree = getBPlusTree(Type.intType(), 4);
414
415     // Insert [0, 200).
416     for (int i = 0; i < 200; ++i) {
417         tree.put(new IntDataBox(i), new RecordId(i, (short) i));
418     }
419
420     // Delete [100, 200).
421     for (int i = 100; i < 200; ++i) {
422         tree.remove(new IntDataBox(i));
423     }
424
425     // Insert [150, 300).
426     for (int i = 150; i < 300; ++i) {
427         tree.put(new IntDataBox(i), new RecordId(i, (short) i));
428     }

```



```

528     }
529
530     // Delete [250, 300].
531     for (int i = 250; i < 300; ++i) {
532         tree.remove(new IntDataBox(i));
533     }
534
535     // Add [100, 150]
536     for (int i = 100; i < 150; ++i) {
537         tree.put(new IntDataBox(i), new RecordId(i, (short) i));
538     }
539
540     // Add [250, 300]
541     for (int i = 250; i < 300; ++i) {
542         tree.put(new IntDataBox(i), new RecordId(i, (short) i));
543     }
544
545     // Range [0, 300) should be full.
546     List<RecordId> rids = new ArrayList<>();
547     for (int i = 0; i < 300; ++i) {
548         rids.add(new RecordId(i, (short) i));
549     }
550     assertEquals(rids, iteratorToList(tree.scanAll()));
551 }
552
553 @Test
554 public void testMaxOrder() {
555     // Note that this white box test depend critically on the implementation
556     // of toBytes and includes a lot of magic numbers that won't make sense
557     // unless you read toBytes.
558     assertEquals(4, Type.intType().getSizeInBytes());
559     assertEquals(6, RecordId.getSizeInBytes());
560     int pageSizeInBytes = 100;
561     Type keySchema = Type.intType();
562     assertEquals(4, LeafNode.maxOrder(pageSizeInBytes, keySchema));
563     assertEquals(5, InnerNode.maxOrder(pageSizeInBytes, keySchema));
564     assertEquals(4, BPlusTree.maxOrder(pageSizeInBytes, keySchema));
565 }
566
567 }

```