lab5

521120910245 严冬

1 structure

1. `BTreeFile`: This class implements a database file that stores a B+ tree index. It provides methods for inserting, deleting, and searching tuples in the B+ tree. The key methods are:

   - `insertTuple`: Inserts a tuple into the B+ tree, potentially causing pages to split if they become full.
   - `deleteTuple`: Deletes a tuple from the B+ tree, merging or redistributing pages if they become less than half full.
   - `findLeafPage`: Finds and locks the leaf page corresponding to a given key field.
   - `splitLeafPage` and `splitInternalPage`: Methods for splitting leaf and internal pages when they become full.
   - `mergeLeafPages` and `mergeInternalPages`: Methods for merging leaf and internal pages when they become less than half full.

2. `BufferPool`: This class manages the reading and writing of pages into memory from disk. It is responsible for locking pages, evicting pages when the buffer is full, and ensuring the ACID properties of transactions. The key methods are:

   - `getPage`: Retrieves a page from the buffer pool, acquiring the necessary locks and potentially evicting other pages if the buffer is full.
   - `releasePage` and `transactionComplete`: Methods for releasing locks and committing or aborting transactions.
   - `insertTuple` and `deleteTuple`: Methods for inserting and deleting tuples, handling dirty pages and updating the buffer pool accordingly.
   - `flushAllPages` and `discardPage`: Methods for flushing pages to disk and removing pages from the buffer pool.

3. `HeapFile`: This class implements a database file that stores tuples in a heap file (an unordered collection of pages). It provides methods for reading, writing, inserting, and deleting tuples in the heap file.

1. **B+ Tree Operations**: Implementing various operations on a B+ tree index, such as insertion, deletion, splitting, and merging of pages.
2. **Buffer Pool Management**: Managing the buffer pool, which caches pages in memory, handling page eviction, and ensuring the ACID properties of transactions through locking and logging mechanisms.
3. **Heap File Operations**: Performing basic operations on a heap file, such as reading, writing, inserting, and deleting tuples.

2 test

```
Terminal  Local
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.274 sec
    [junit]
    [junit] Testcase: attemptTransactionTwice took 0.217 sec
    [junit] Testcase: commitTransaction took 0.023 sec
    [junit] Testcase: abortTransaction took 0.025 sec
    [junit] Running simpledb.TupleDescTest
    [junit] Testsuite: simpledb.TupleDescTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.02 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.02 sec
    [junit]
    [junit] Testcase: combine took 0.006 sec
    [junit] Testcase: getType took 0 sec
    [junit] Testcase: getSize took 0 sec
    [junit] Testcase: testEquals took 0 sec
    [junit] Testcase: numFields took 0.001 sec
    [junit] Testcase: nameToId took 0.006 sec
    [junit] Running simpledb.TupleTest
    [junit] Testsuite: simpledb.TupleTest
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec
    [junit]
    [junit] Testcase: modifyFields took 0.005 sec
    [junit] Testcase: getTupleDesc took 0 sec
    [junit] Testcase: modifyRecordId took 0.001 sec

BUILD SUCCESSFUL
Total time: 22 seconds
PS C:\Users\Lenovo\Desktop\lab5>
```

```
Terminal  Local
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.996 sec
    [junit]
    [junit] Testcase: testSingleThread took 0.158 sec
    [junit] Testcase: testTwoThreads took 0.025 sec
    [junit] Testcase: testFiveThreads took 0.19 sec
    [junit] Testcase: testTenThreads took 2.593 sec
    [junit] Testcase: testAllDirtyFails took 0.023 sec

BUILD SUCCESSFUL
Total time: 31 seconds
PS C:\Users\Lenovo\Desktop\lab5>
```